A New Framework for Combining Global and Local Methods in Black Box Optimization

Yibo Ji[*] and Sujin Kim[†]

National University of Singapore

Wendy Lu Xu[‡]

ExxonMobil Research and Engineering

July 29, 2013

Abstract

We propose a new framework for the optimization of computationally expensive black box problems, where neither closed-form expressions nor derivatives of the objective functions are available. The proposed framework consists of two procedures. The first constructs a global metamodel to approximate the underlying black box function and explores an unvisited area to search for a global solution; the other identifies a promising local region and conducts a local search to ensure local optimality. To improve the global metamodel, we propose a new method of generating sampling points for a wide class of metamodels, such as kriging and Radial Basis Function models. We also develop a criterion for switching between the global and local search procedures, a key factor affecting practical performance. Under a set of mild regularity conditions, the algorithm converges to the global optimum. Numerical experiments are conducted on a wide variety of test problems from the literature, demonstrating that our method is competitive against existing approaches.

Keywords: Global optimization; simulation optimization; metamodel-based optimization; kriging; radial basis function;

1 Introduction

Computer simulation models are widely used by practitioners to gain insights into the behavior of real-world complex systems. For example, in the oil and gas industry, numerical reservoir simulation is used to simulate fluid flow through hydrocarbon-rich, subsurface reservoir rock, attached wells and surface facilities, and to help reservoir engineers make better decisions about, for example, well location and production to minimize operating costs. These simulations are usually computationally intensive, and derivative information about the objective function is either unavailable or unreliable. In the optimization of a system that can only be evaluated by expensive simulation, the goal is often to find a set of system parameter values that give overall good performance within a limited computational budget.

[*]Email: jiyibo@nus.edu.sg

[†]Email: iseks@nus.edu.sg; corresponding author

[‡]Email: lu.xu@exxonmobil.com

In this paper, we study the problem of finding the global minimum of a real-valued deterministic function $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ defined by box constraints, and f is assumed to be lower-bounded and continuous. With these conditions, $f(\boldsymbol{x})$ is guaranteed to attain its global minimum value within $\mathcal{X}$. We assume that the closed mathematical form of $f(\boldsymbol{x})$ is not available, that is, the objective function is a black box, and can only be evaluated via simulation. Although we focus on problems with unknown structure, we make continuity assumptions in order to apply common metamodeling techniques.

For this type of problem, gradient-based methods are not applicable as numerical computation of derivatives, such as finite differences, is either computationally prohibitive or unreliable. The absence of derivatives naturally motivates the use of derivative-free optimization methods (Conn et al., 2009; Kolda et al., 2003). One of the earliest derivative-free approaches is the Nelder-Mead simplex method (Nelder and Mead (1965)), which uses the simplex structure of sample points to search for a new solution. Another class of methods is direct search methods that rely on a set of directions defined by a positive spanning basis. Examples of this class include pattern search algorithms (Torczon, 1997) and its extension, the Mesh Adaptive Direct Search (MADS) algorithm (Audet and Dennis, 2006). However, since they only depend on the function values without fully exploiting the inherent smoothness of the objective function, their convergence to the optimum can be slow (Conn et al., 1997).

One common approach for computationally intensive black box optimization problems is to use metamodels (also known as response surfaces or surrogate models). Metamodels are inexpensive models that approximate the underlying expensive objective function. Many metamodeling techniques have been developed by various communities to capture the global properties of black box optimization problems. Myers et al. (2009) defines Response Surface Methodology (RSM) as a collection of statistical and mathematical techniques useful for developing, improving, and optimizing processes. The main idea of RSM is to sequentially estimate the location of the optimum with response surfaces and indicate points that provide additional information to improve the estimation. The Polynomial Response Surface Model (PRSM) is the classic and most widely applied metamodel in engineering design (Box and Draper, 1987). Another type of regression model is Multivariate Adaptive Regression Splines (MARS) introduced by Friedman (1991). It adaptively selects basis functions to model interactions among variables in an iterative manner. More recent metamodels include kriging (also known as Gaussian Process Regression in statistics and machine learning (Bishop, 2006)), and Radial Basis Functions (RBF) (Buhmann, 2003; Powell, 1992). Empirical results show that kriging and RBF outperform PRSM and MARS under multiple modeling criteria (Jin et al., 2001). Kleijnen et al. (2005) and Forrester and Keane (2009) provide excellent reviews on recent advances in metamodel-based methods.

Kriging has been one of the most popular metamodeling methods for global optimization since the late 1990s. For a review of existing kriging-based approaches for global optimization, readers are referred to Jones (2001). Most existing work in this area is oriented toward efficiently utilizing the metamodel prediction and error estimates provided by kriging models. One of the most well-known methods is Efficient Global Optimization (EGO) introduced by Jones et al. (1998). EGO defines a statistical function called Expected Improvement (EI) as a sampling criterion. In kriging, the true function value at a new candidate point is modeled as a random variable following a normal distribution whose mean is the unbiased prediction and whose variance is the error estimate. EGO solves an optimization problem that determines the candidate point with the maximum EI relative to the current best solution. Typically, the overhead spent in generating candidate points can be significant as it increases with the number of sample points contained in the metamodel (Jones et al. (1998)). Since the introduction of EGO, many variants have followed, such as generalized expected improvement (Sasena, 2002) and weighted expected improvement (Sóbester et al., 2005).

RBF models have been extensively studied in the optimization community and proved to be successful in handling black box optimization problems. Gutmann (2001) assumes that the underlying objective function is smooth and has only a small number of local solutions. He introduces a measure of "bumpiness" for RBF models and proposes a method to select sample points in a way that minimizes the bumpiness of RBF models. Other RBF-based methods use the distance to the nearest evaluated point as an estimated measure of uncertainty. For example, Regis and Shoemaker (2007) and Regis (2011) randomly generate a set of candidate points in the feasible region and select the one with the best score computed based on two criteria, the metamodel function value and the minimum distance to previously evaluated points. High weights on the distance criterion promote exploration of the feasible region, while high weights on the metamodel function values drive the algorithm for local search. Numerical experiments suggest that their methods are efficient and promising for global optimization of expensive functions. Similarly Jakobsson et al. (2010) introduced a measure of total uncertainty, defined by the distance to the closest evaluated point weighted against the metamodel function value.

Most of the above mentioned optimization methods are based on either global or local search algorithms. Global search algorithms provide theoretical guarantee of global optimum at the expense of relatively large computational effort. Local search algorithms focus only on moving towards a better point within a local region. One key issue in black box optimization that has been extensively investigated in literature is the allocation of computational budget between local and global search, known as balancing exploration and exploitation. For example, Andradóttir and Prudius (2009) focuses on the matter, and proposes parameterized switch criteria which they call distance and improvement thresholds. But as there is no easy way to determine these key performance thresholds in advance, the parameter values for the switch criteria are pre-specified in a somehow arbitrary way, rather than being determined adaptively. Other literatures, such as Regis and Shoemaker (2005), Regis and Shoemaker (2007), and Jakobsson et al. (2010), also have the same problem.

In this paper, we introduce a new framework for global optimization of computationally expensive functions. Our framework combines a global search procedure and a local search procedure, and allows adaptive switching between the two. The global search procedure constructs a global metamodel to approximate the true objective function, and explores unvisited promising areas for good solutions. And the local search procedure exploits a promising local region for optimality. The adaptive switching between the two procedures is one of the central contributions of our research. To enable adaptive switching, the algorithm framework constructs and maintains a global metamodel at each iteration, which includes all information collected with sample points that have been evaluated so far. A set of candidate sample points is then generated for global and local search procedures respectively, and the potential performance of each candidate point is computed using this global metamodel. The algorithm determines which procedure, global search or local search, to execute next depending on which set of candidate points shows more potential. In this manner, we take full advantage of the available information. To evaluate the potential of candidate sample points using global metamodel, we need some measure based on the metamodel. For kriging, we use EI function as the performance measure. For other types of metamodels such as RBF, however, there is no known method for estimating the prediction error, and hence the EI defined for kriging does not work. In this research we propose a new measure called the Modified Expected Improvement (MEI), which extends EI to general metamodels. The MEI measure is composed of two pieces of information, the estimated objective function value from the metamodel and the minimum distance from previously evaluated points. We construct a function of the minimum distance that can be interpreted as an estimate of prediction error. With the introduction of MEI, our adaptive switching works for any type of metamodel, and therefore our algorithm framework achieves the flexibility of incorporating any type of global search procedures.

To speed up the convergence to the global optimum, our algorithm framework interrupts iterations that exhibit small improvement during local search, and restarts with a newly generated set of sample points. This restarting technique is commonly seen in literature, where people build the restarting sample set far way from the set of evaluated sample points, and as spread out as possible. One way to build the new sample set is to find points that maximize the minimum distance from the set of evaluated samples. Our framework adopts this basic idea, but tries to leverage on the available information again through the global metamodel. We find the new samples by maximizing the minimum weighted distance from the set of evaluated samples, instead of the absolute distance. The weight is calculated based on the objective function value predicted by the global metamodel. What this weight does is to guide exploration by focusing more on the unvisited areas with higher potential of containing better solutions. The effectiveness of this improvement is demonstrated in the numerical study. Under a set of mild assumptions, our proposed algorithm converges to the global optimum, provided that the algorithm is allowed to run infinitely. The performance of the algorithm is proved to be competitive in our numerical study on a wide range of test problems from the literature.

The remainder of the paper is organized as follows. In Section 2, we present our framework for combining global and local methods for black box optimization. Section 2.1 provides an overview of the proposed algorithm. The global search and local search methods are discussed in Sections 2.2 and 2.3, respectively. Section 2.4 presents the reward function that determines the search type in each iteration. In Section 2.5, we present a sampling scheme that helps explore promising regions and also guarantees convergence to the global optimum. Section 3 presents numerical experiments and Section 4 concludes this paper .

2 Framework

In this section, we present details of a general framework that combines global and local methods for global optimization on black box problems. An overview of the algorithm structure is given in Section 2.1 and its key components are further discussed in the subsequent sections.

2.1 General Structure

The framework consists of eight building blocks shown in flowchart form in Figure 1. Detailed algorithmic descriptions for each block are provided in Algorithm 1.

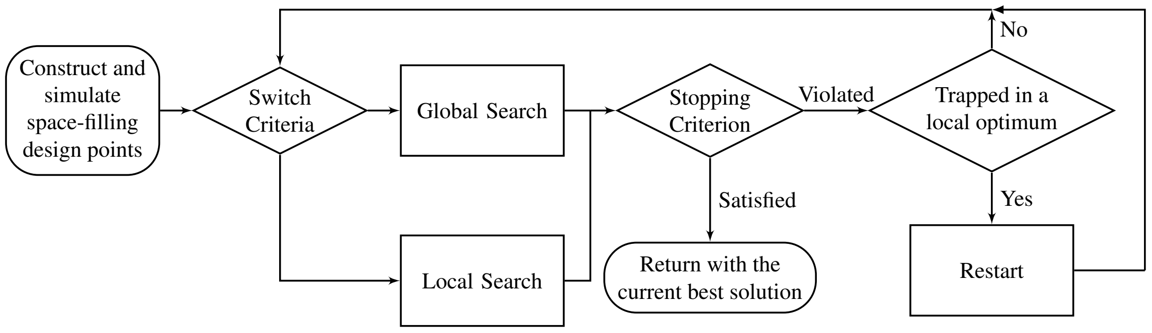
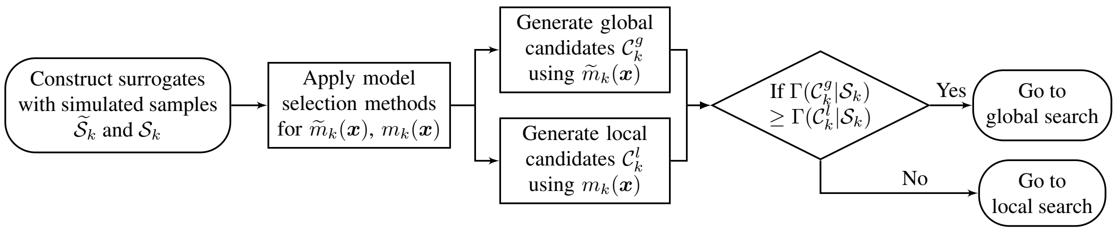

Construct and simulate space-filling design points
Switch Criteria
Global Search
Local Search
Stopping Criterion
Violated
Satisfied
Return with the current best solution
Trapped in a local optimum
No
Yes
Restart

Figure 1: General structure of the proposed framework.

Here we briefly describe the main steps of the algorithm and introduce related notation. The algorithm starts by constructing a set $\mathcal{X}_1 = \{\boldsymbol{x}_i\}_{i=1}^{N_1}$ of N_1 space-filling design points, and running simulations to evaluate their function values $f(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}_1$. This is a common initial step in metamodel-based optimization to gather initial information about the underlying problem. To obtain more informative design points, one can select points to be spread over the design space of interest. Let N_k be the total number of function evaluations performed until the end of iteration $k-1$. At the beginning of iteration k, the algorithm keeps track of evaluated points with two sets: full information $\mathcal{S}_k = \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{X}_k\}$ and partial information $\widetilde{\mathcal{S}}_k = \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in \widetilde{\mathcal{X}}_k\}$, where $\mathcal{X}_k = \{\boldsymbol{x}_i\}_{i=1}^{N_k}$ is the set of all evaluated design points so far and $\widetilde{\mathcal{X}}_k$ is the set of evaluated points since the most recent *Restart*. Note that $\mathcal{X}_k = \bigcup_{i=1}^{k} \widetilde{\mathcal{X}}_i$. $\widetilde{\mathcal{S}}_k$ only records points evaluated between two consecutive restarts, and hence $\widetilde{\mathcal{S}}_k = \mathcal{S}_k$ until the first restart and $\widetilde{\mathcal{S}}_k \subsetneq \mathcal{S}_k$ after the algorithm restarts. By discarding old sample points and restarting the algorithm, one can prevent subsequent trajectories from being distracted by the old sample points clustered around local solutions.

At the beginning of each iteration k, the algorithm enters *Switch Criteria* with the solution $\widetilde{\boldsymbol{x}}_k^\star = \operatorname{argmin}_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_k} f(\boldsymbol{x})$. The main idea of this step is summarized in Figure 2.



Construct surrogates with simulated samples $\widetilde{\mathcal{S}}_k$ and $\mathcal{S}_k$
Apply model selection methods for $\widetilde{m}_k(\boldsymbol{x})$, $m_k(\boldsymbol{x})$
Generate global candidates $\mathcal{C}_k^g$ using $\widetilde{m}_k(\boldsymbol{x})$
Generate local candidates $\mathcal{C}_k^l$ using $m_k(\boldsymbol{x})$
If $\Gamma(\mathcal{C}_k^g|\mathcal{S}_k) \geq \Gamma(\mathcal{C}_k^l|\mathcal{S}_k)$
Yes
Go to global search
No
Go to local search

Figure 2: Steps in switch criteria.

For given $\mathcal{S}_k$ and $\widetilde{\mathcal{S}}_k$, candidate points to be evaluated by each search step are determined. More specifically, the algorithm first constructs two global metamodels $m_k(\boldsymbol{x})$ with full information $\mathcal{S}_k$ and $\widetilde{m}_k(\boldsymbol{x})$ with partial information $\widetilde{\mathcal{S}}_k$. The two metamodels are used for different purposes. Since $m_k(\boldsymbol{x})$ is constructed with more information, it is generally more accurate than $\widetilde{m}_k(\boldsymbol{x})$. Thus, we use $m_k(\boldsymbol{x})$ in switch criteria (Step II). However, we use the model $\widetilde{m}_k(\boldsymbol{x})$ with less information to generate candidate points $\mathcal{C}_k^g$ for global search. As $\widetilde{\mathcal{S}}_k$ does not include old sample points, the global step is likely to generate sample points away from regions that were exhaustively explored before the restart. Meanwhile, local search methods are applied to generate a set of candidate points $\mathcal{C}_k^l$ within a local region $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$, which is the set of all points in $\mathcal{X}$ centered around $\widetilde{\boldsymbol{x}}_k^\star$ within distance of $\widetilde{\rho}_k$.

The potential performance of global and local steps is measured by reward functions. The explicit form of the reward functions, $\Gamma(\mathcal{C}_k^g|\mathcal{S}_k)$ for global search and $\Gamma(\mathcal{C}_k^l|\mathcal{S}_k)$ for local search, will be given in Section 2.4. The algorithm switches to the step with larger reward. After either the global or local step is performed, the algorithm checks the stopping criterion. If the total number of function evaluations N_k exceeds a prefixed number $N_{\max}$, the algorithm terminates with the best solution found so far, given by $\boldsymbol{x}_k^\star = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}_k} f(\boldsymbol{x})$. Otherwise, it moves to the next step to check if the algorithm is trapped in a local optimum. The restarting criteria are presented in Section 2.5. If it is verified that the algorithm is trapped in a local optimum, old samples are discarded and a new set $\mathcal{C}_k^r$ of space-filling design points is generated to restart the algorithm. The full algorithmic description is presented in Algorithm 1.

In Step I, we use $2(d+1)$ initial sample points, smaller than the $10d$ "rules of thumb" proposed by Jones et al. (1998). We choose to start with a small number of initial sample points and save more computing budget for later iterations (Step II-(iii), Step II-(iv), and Step VI). The purpose is to collect more information about the true function value adaptively.

Step II-(iii) and Step II-(iv) generate candidate points for global and local search, respectively. In the simulation step (Step III), the algorithm evaluates points in either $\mathcal{C}_k^g$ or $\mathcal{C}_k^l$. One practical issue is that the size of the local candidate set $\mathcal{C}_k^l$ can be prohibitively large for a high-dimensional problem, as the positive spanning set used in local search is at least of size $d + 1$. One way of dealing with this issue in implementation is to first sort the candidate points in $\mathcal{C}_k^l$ by descending potential improvement, and then sequentially evaluate these candidate points until a better solution is found. Another issue is that the computing effort needed for Step III can be such that the algorithm uses up the computational budget before reaching the *Stopping Criterion* (Step IV). In our implementation, we check the stopping criterion whenever the algorithm evaluates a new point and terminate the algorithm right after the criterion is met.

To achieve faster convergence to a global optimal point, the algorithm restarts with a new set of samples $\mathcal{C}_k^r$ when trapped in a local optimum, and discards all previous sample points. In Section 2.5, we proposed

Algorithm 1 A framework for combining global and local methods

```
Inputs: Maximum allowed number of function evaluations $N_{\max}$; Initial sample size $N_1$; Global metamodel,
    e.g., kriging or radial basis functions method; Local method, e.g., pattern search or local perturbations;
    Initial radius of local region $\rho_0$.
Outputs: The best solution $\boldsymbol{x}_k^\star$ and its function value $f(\boldsymbol{x}_k^\star)$ obtained by the algorithm.
Step I (Initialization and Space-Filling Designs): Set iteration counter $k = 1$. Construct the set of
    space-filling design points $\mathcal{X}_1 = \{\boldsymbol{x}_i\}_{i=1}^{N_1}$, and set $\widetilde{\mathcal{X}}_1 = \mathcal{X}_1$. Evaluate the function values of design points
    and define $\mathcal{S}_1$ and $\widetilde{\mathcal{S}}_1$.
Step II (Switch Criteria): Compute $\widetilde{\boldsymbol{x}}_k^\star = \operatorname{argmin}_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_k} f(\boldsymbol{x})$ and determine the local region $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$
    by Algorithm 2 in Section 2.3.1. Set $N_k = |\mathcal{X}_k|$.
    (i) (Build Global Metamodels): Build two sets of global metamodels $\widetilde{m}_k(\boldsymbol{x})$ and $m_k(\boldsymbol{x})$ based on $\widetilde{\mathcal{S}}_k$
        and $\mathcal{S}_k$, respectively.
    (ii) (Model Selection): Apply model selection techniques to identify the most accurate model in each
        category.
    (iii) (Generate Global Candidates): Generate a set of global candidate points $\mathcal{C}_k^g \subset \mathcal{X}$ based on
        $\widetilde{m}_k(\boldsymbol{x})$ with the procedure discussed in Section 2.2.4.
    (iv) (Generate Local Candidates): Generate a set of local candidate points $\mathcal{C}_k^l \subset \mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$ based
        on $m_k(\boldsymbol{x})$ with the procedure discussed in Section 2.3.2.
    (v) (Compute Rewards): Compute the rewards $\Gamma(\mathcal{C}_k^g|\mathcal{S}_k)$ and $\Gamma(\mathcal{C}_k^l|\mathcal{S}_k)$.
Step III (Global or Local Search): If $\Gamma(\mathcal{C}_k^g|\mathcal{S}_k) > \Gamma(\mathcal{C}_k^l|\mathcal{S}_k)$, evaluate points in $\mathcal{C}_k^g$ and set $\mathcal{C}^{new} = \mathcal{C}_k^g$;
    otherwise, perform one iteration of the local search and set $\mathcal{C}^{new} = \mathcal{C}_k^l$.
Step IV (Stopping Criterion): Solve $\min_{\boldsymbol{x}\in\mathcal{X}_k\cup\mathcal{C}^{new}} f(\boldsymbol{x})$ and find the optimal solution $(\boldsymbol{x}_k^\star, f(\boldsymbol{x}_k^\star))$. If
    $N_k + |\mathcal{C}^{new}| \geq N_{\max}$, terminate the algorithm and return $\boldsymbol{x}_k^\star$ as the global solution; otherwise go to Step V.
Step V (Trapped in a Local Optimum): Determine the local region $\mathcal{L}_k(\boldsymbol{x}_k^\star, \rho_k)$ by Algorithm 2. Ex-
    amine whether the algorithm is trapped in a local optimum using the criteria presented in Section 2.5.
Step VI (Restart): If trapped in a local optimum, generate and evaluate a new set of samples $\mathcal{C}_k^r$. Set
    $\mathcal{X}_{k+1} = \mathcal{X}_k \cup \mathcal{C}^{new} \cup \mathcal{C}_k^r$, and discard old samples by setting $\widetilde{\mathcal{X}}_{k+1} = \mathcal{C}_k^r$. Otherwise, set $\mathcal{X}_{k+1} = \mathcal{X}_k \cup \mathcal{C}^{new}$,
    and $\widetilde{\mathcal{X}}_{k+1} = \widetilde{\mathcal{X}}_k \cup \mathcal{C}^{new}$. Update $\widetilde{\mathcal{S}}_{k+1}$ and $\mathcal{S}_{k+1}$. Set $k := k + 1$ and go to Step IV.
```

a new method for generating $\mathcal{C}_k^r$. The new set of space-filling design points is selected so that the points are not only far away from all previous sample points, but also show high potential to have good solutions.

2.2 Global Search

Global search uses metamodel to identify regions within feasible space that have the potential of containing good solutions. The metamodel is designed to capture the feature of the objective function over the entire feasible region. In this section, we first give a brief review of two popular metamodeling techniques used in this paper, kriging and RBF models, followed by several techniques to improve the quality of metamodel prediction. We then present several criteria of generating candidate sample points for both metamodels.

2.2.1 Review of Global Metamodels

Kriging model was first introduced to the simulation community by Sacks et al. (1989). At iteration k, given the information about a set of sample points $\mathcal{S}_k = \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{X}_k\}$, we want to make a prediction of the objective function value at an unknown point $\check{\boldsymbol{x}}$. The function value at $\check{\boldsymbol{x}}$ is "uncertain" unless we perform expensive simulations at $\check{\boldsymbol{x}}$. Kriging assumes that this uncertainty is characterized by a random process $Y(\check{\boldsymbol{x}})$, defined as follows:

$$Y(\check{\boldsymbol{x}}) = \mu(\check{\boldsymbol{x}}) + Z(\check{\boldsymbol{x}}),$$

where in general the mean of the random process $\mu(\check{\boldsymbol{x}})$ is assumed to be a constant μ_0, and the systematic departure $Z(\check{\boldsymbol{x}})$ from the mean is assumed to be Gaussian white noise with variance σ_z^2. The true function value $f(\check{\boldsymbol{x}})$ at any point $\check{\boldsymbol{x}}$ is considered as one realization of the stochastic process $Y(\check{\boldsymbol{x}})$.

In kriging the relationship between $f(\check{\boldsymbol{x}})$ and the function value of simulated samples $\mathcal{Y}_k = \{f(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}_k\}$ is characterized by correlations. Consider two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, whose function values $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ are assumed to be one realization of $Y(\boldsymbol{x}_i)$ and $Y(\boldsymbol{x}_j)$, respectively. By the continuity of the objective function, $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ tend to be similar if $\boldsymbol{x}_i$ is close to $\boldsymbol{x}_j$. The degree of closeness can be statistically measured by the correlation between $Y(\boldsymbol{x}_i)$ and $Y(\boldsymbol{x}_j)$. One of the most widely used correlation functions in the context of kriging method is a generalized exponential correlation function, which is defined as follows:

$$\mathrm{Corr}[Y(\boldsymbol{x}_i), Y(\boldsymbol{x}_j)] = \exp\left(-\sum_{s=1}^{d} \theta_s \left\|\boldsymbol{x}_{is} - \boldsymbol{x}_{js}\right\|^{p_s}\right), \tag{1}$$

where $\boldsymbol{x}_i = (\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{id})$, $\boldsymbol{x}_j = (\boldsymbol{x}_{j1}, \ldots, \boldsymbol{x}_{jd})$, $\theta_s \geq 0$ and $0 < p_s \leq 2$, for $s = 1, \ldots, d$. There are many other types of correlation functions, such as linear, cubic and spline. Kriging models with Gaussian functions ($p_s = 2$, for all $s \in \{1, \ldots, d\}$) can fit smooth and continuous functions well, but may suffer from computational instability.

Suppose that we do not have information on $\mathcal{Y}_k$, and consider a kriging model with a generalized exponential correlation function. Then, the uncertainties of the function values at points in $\mathcal{X}_k$ can be represented by a random vector, $\boldsymbol{Y} = (Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_{N_k}))$, whose mean is equal to $\mathbf{1}\mu_0$ and covariance matrix $\mathrm{Cov}[\boldsymbol{Y}]$ is equal to $\sigma_z^2\boldsymbol{R}$, where $\mathbf{1}$ is an N_k-dimensional vector of ones and the (i,j)th element of the matrix $\boldsymbol{R}$ is defined by equation (1). The likelihood of sample points $\mathcal{Y}_k$ is defined by,

$$\mathbf{L}(\mathcal{Y}_k|\boldsymbol{\theta}, \mu_0, \sigma_z^2) = \frac{1}{(2\pi\sigma_z^2)^{N_k/2}|\boldsymbol{R}|^{1/2}} \exp\left\{-\frac{(\mathcal{Y}_k - \mathbf{1}\mu_0)^\mathsf{T} \cdot \boldsymbol{R}^{-1} \cdot (\mathcal{Y}_k - \mathbf{1}\mu_0))}{2\sigma_z^2}\right\}, \tag{2}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ is the vector of correlation function parameters. The parameters $\boldsymbol{\theta}$, μ_0 and σ_z^2 are chosen to maximize the likelihood of the sampled data. In general, it is difficult to find the optimal values of $(\boldsymbol{\theta}, \mu_0, \sigma_z^2)$ due to the complex structure of the matrix $\boldsymbol{R}$. However, it is relatively easy to find the maximum likelihood estimators (MLE) of μ_0 and σ_z^2 for any fixed $\boldsymbol{\theta}$, that is,

$$\widehat{\mu}_0 = \frac{\mathbf{1}^\mathsf{T} \cdot \boldsymbol{R}^{-1} \cdot \mathcal{Y}_k}{\mathbf{1}^\mathsf{T}\boldsymbol{R}^{-1}\mathbf{1}}, \qquad \text{and} \qquad \widehat{\sigma}_z^2 = \frac{(\mathcal{Y}_k - \mathbf{1}\widehat{\mu}_0))^\mathsf{T} \cdot \boldsymbol{R}^{-1} \cdot (\mathcal{Y}_k - \mathbf{1}\widehat{\mu}_0))}{N_k}.$$

By plugging these expressions into (2), one can obtain the concentrated log-likelihood function $\ln(\mathbf{L}(\mathcal{Y}_k))$, which can be approximated with the following expression:

$$\ln(\mathbf{L}(\mathcal{Y}_k|\boldsymbol{\theta}, \mu_0, \sigma_z^2)) \approx -\frac{N_k}{2}\ln(\widehat{\sigma}_z^2) - \frac{1}{2}\ln(|\boldsymbol{R}|).$$

Note that the concentrated log-likelihood function only depends on $\boldsymbol{R}$, i.e., the parameter vector $\boldsymbol{\theta}$. Therefore, maximizing this function yields the MLE of $\boldsymbol{\theta}$. Once all parameters are estimated, the kriging predictor that minimizes the mean squared prediction error is given by

$$m_k(\check{\boldsymbol{x}}) = \widehat{\mu}_0 + \boldsymbol{r}^\mathsf{T}\boldsymbol{R}^{-1}(\mathcal{Y}_k - \mathbf{1}\widehat{\mu}_0), \tag{3}$$

where $\boldsymbol{r}$ is a N_k-dimensional correlation vector between $Y(\check{\boldsymbol{x}})$ and $Y(\boldsymbol{x}_i)$, $i = 1, \ldots, N_k$.

A distinctive feature of kriging among all metamodeling techniques is that it provides the probability distribution of the error associated with the prediction $m_k(\check{\boldsymbol{x}})$, which is a normal distribution with mean zero and variance

$$\sigma_{\check{\boldsymbol{x}}}^2 = \widehat{\sigma}_z^2\left[1 - \boldsymbol{r}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r} + \frac{(1 - \boldsymbol{r}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r})^2}{\mathbf{1}^\mathsf{T}\boldsymbol{R}^{-1}\mathbf{1}}\right]. \tag{4}$$

The variance $\sigma^2_{\check{\boldsymbol{x}}}$ is zero at any observed point $\check{\boldsymbol{x}} \in \mathcal{X}_k$ and increases as the point $\check{\boldsymbol{x}}$ moves away from the set $\mathcal{X}_k$. Many criteria for generating promising solutions have been derived based on the distribution of the prediction error, such as expected improvement and probability of improvement (Jones, 2001). However, the additional computational effort required to estimate the parameters $(\boldsymbol{\theta}, \mu_0, \sigma_z^2)$ becomes significant as the number of sample points increases.

The RBF method was originally developed by Hardy (1971) for scattered multivariate data interpolation. The underlying objective function is emulated by a weighted sum of radial basis functions. Given the set $(\mathcal{X}_k, \mathcal{Y}_k)$ of evaluated points, the augmented RBF model has the form

$$m_k(\check{\boldsymbol{x}}) = \sum_{i=1}^{N_k} w_i\, \psi\left(\|\check{\boldsymbol{x}} - \boldsymbol{x}_i\|\right) + \sum_{l=1}^{\hat{\mathsf{m}}} \beta_l\, \mathsf{p}_l(\check{\boldsymbol{x}}), \quad \check{\boldsymbol{x}} \in \mathbb{R}^d, \tag{5}$$

where $\boldsymbol{w} = \{w_1, \ldots, w_{N_k}\}$ denotes a vector of weights, $\psi\left(\|\cdot\|\right) : \mathbb{R}^d \to \mathbb{R}$ is a radial basis function, $\mathsf{p}_1(\boldsymbol{x}), \ldots, \mathsf{p}_{\hat{\mathsf{m}}}(\boldsymbol{x})$ is a basis of the space $\mathsf{P}^d_{\mathsf{m}}$ of polynomials with degree not exceeding m in $\mathbb{R}^d$, and $\hat{\mathsf{m}} = \binom{\mathsf{m}+d}{d}$ is the dimension of $\mathsf{P}^d_{\mathsf{m}}$. The response of each individual basis function monotonically decreases (or increases) with the distance from the closest sample point. Most commonly used forms of the radial basis function $\psi\left(\|\cdot\|\right)$ include $\psi\left(r\right) = \sqrt{c^2 + r^2}$ (multiquadrics), $\psi\left(r\right) = r^2 \ln(r)$ (thin-plate splines), $\psi\left(r\right) = 1/\sqrt{c^2 + r^2}$ (inverse multiquadrics), and $\psi\left(r\right) = \exp(-cr^2)$ (Gaussians). Usually m is quite small and often equal to 1. In this case, $\hat{\mathsf{m}} = d + 1$.

One of the advantages of RBF over kriging or other metamodeling techniques is that RBF models are relatively easy to build. Define matrices $\mathsf{P} \in \mathbb{R}^{n \times \hat{\mathsf{m}}}$ and $\Psi \in \mathbb{R}^{N_k \times N_k}$ as follows:

$$\mathsf{P}_{ij} = \mathsf{p}_j(\boldsymbol{x}_i), \text{ and } \Psi_{i,j} = \psi\left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right).$$

Then, the RBF model (5) interpolating $(\mathcal{X}_k, \mathcal{Y}_k)$ is obtained by solving the following linear system:

$$\begin{pmatrix} \Psi & \mathsf{P} \\ \mathsf{P}^\mathsf{T} & \mathbf{0}_{\hat{\mathsf{m}} \times \hat{\mathsf{m}}} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathcal{Y}_k \\ \mathbf{0}_{\hat{\mathsf{m}}} \end{pmatrix}.$$

If $\operatorname{rank}(\mathsf{P}) = \hat{\mathsf{m}}$, the coefficient matrix is nonsingular and the linear system has a unique solution (Powell, 1992). In general, the above linear system can be efficiently solved by exploiting the structure of the problem.

2.2.2 Data Transformation

When there is a large amount of variation in the function values of the sample points, the associated global metamodel often severely oscillates. Several approaches to alleviate this effect have been developed in the context of RBF-based global optimization. Gutmann (2001) proposes to replace large function values by the median of all available function values. The resulting RBF model is smoother, and usually more effective in identifying good candidate solutions. The most recent work addressing this issue is by Regis and Shoemaker (2012), where a function-stabilizing transformation is applied to the objective function values in order to reduce the influence of extreme function values.

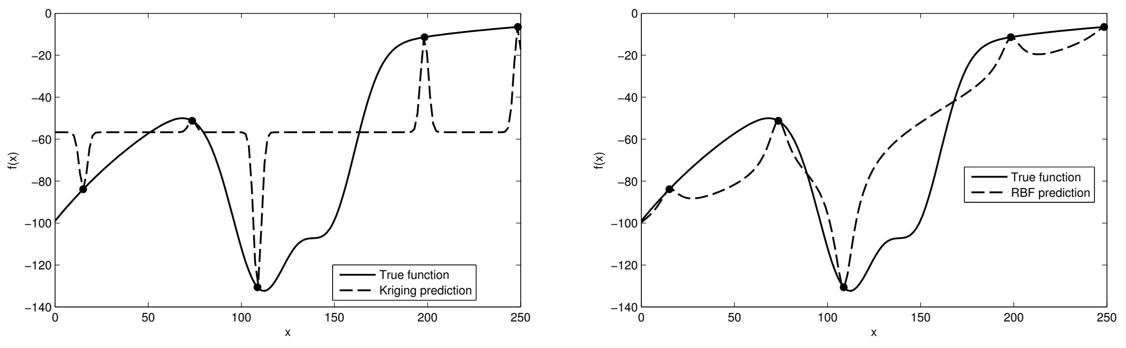
When the curvature of the objective function varies widely, the performance of kriging models deteriorates since the same covariance structure is used over the entire feasible space. Also, the normality assumption can be violated if there are extreme function values in $\mathcal{S}_k$. In this regard, employing a method for detecting outliers can be useful for enhancing the stability of kriging models. In our numerical experiments, we apply a method based on interquartile range. Let $Q_1(\mathcal{Y}_k)$ and $Q_3(\mathcal{Y}_k)$ be the lower and upper quartiles of $\mathcal{Y}_k$, respectively. Any sample point with a function value outside the interval

$$[Q_1(\mathcal{Y}_k) - \kappa(Q_3(\mathcal{Y}_k) - Q_1(\mathcal{Y}_k)),\, Q_1(\mathcal{Y}_k) + \kappa(Q_3(\mathcal{Y}_k) - Q_1(\mathcal{Y}_k))] \tag{6}$$

is considered to be an outlier. Here the parameter κ is typically set to be $\kappa = 1.5$ (Upton and Cook, 1996), and in this case the range (6) is considered as the 99.3% confidence interval of the objective function value. When we construct kriging models with $\mathcal{S}_k$, we replace the function values of outlier points with the closest boundary value of the above interval.

2.2.3 Model Validation and Selection

In Step II-(i) of the algorithm, we can construct multiple global metamodels using different methods (or the same method with different correlation function forms, or basis functions). Some metamodels may not be a good approximation of the true function. Figure 3 shows such examples in kriging and RBF models, respectively. To select the best metamodel, one of the most well-known techniques is cross validation (Hastie et al., 2008). This method partitions the data set into a training set and a test set. A metamodel is constructed with the training set and prediction errors are computed over the test set. Among all metamodels, the one with the lowest prediction errors is selected. In our numerical experiments, we use the tenfold cross validation method.



(a) Kriging model (b) RBF model

Figure 3: Inappropriate metamodel examples.

2.2.4 Global Candidates

This section details Step II-(iii), generating a set $\mathcal{C}_k^g$ of potentially good candidate points using a global metamodel $\widetilde{m}_k(\boldsymbol{x})$. The most straightforward candidate point is the global minimizer of the metamodel. However, finding the global optimum of the problem $\min_{\boldsymbol{x}\in\mathcal{X}} \widetilde{m}_k(\boldsymbol{x})$ itself can be computationally expensive, particularly when a kriging model is used. In practice, one can apply local search methods with multiple starting points instead.

Since a metamodel is merely an approximation of the true objective function, the global minimizer of the metamodel may not be the best candidate for the true global optimum. A number of criteria to identify promising candidate points in the context of kriging can be found in the literature (Jones, 2001; Jones et al., 1998; Sasena, 2002; Sóbester et al., 2005). We use Probability of Improvement (PI) in our numerical experiments (Kushner, 1964). The PI at x is defined as the probability that $f(x)$ is below a given target value.We select the point that maximizes the PI function as a candidate solution for the true global optimum. Suppose that at the kth iteration of the algorithm, a kriging model based on $\widetilde{\mathcal{S}}_k$ is constructed and a target value τ_k is given. Then, $f(\boldsymbol{x})$ can be viewed as a Gaussian random variable with mean $\widetilde{m}_k(\boldsymbol{x})$ and variance $\widetilde{\sigma}_{\boldsymbol{x}}^2$. The PI criterion identifies a candidate point with the maximum probability of improvement $\tau_k - f(\boldsymbol{x})$. In our numerical study, we set τ_k to be $f(\boldsymbol{x}_{k-1}^{\star})$, the best function value found so far. Mathematically, the candidate point $\boldsymbol{x}_k^g$ is determined by

$$\begin{aligned}\boldsymbol{x}_k^g &= \mathrm{argmax}_{\boldsymbol{x}\in\mathcal{X}} \ \mathbf{P}\left(f(\boldsymbol{x}) \le \tau_k\right) \\ &= \mathrm{argmax}_{\boldsymbol{x}\in\mathcal{X}} \ \mathbf{P}\left(\frac{f(\boldsymbol{x}) - \widetilde{m}_k(\boldsymbol{x})}{\widetilde{\sigma}_{\boldsymbol{x}}} \le \frac{\tau_k - \widetilde{m}_k(\boldsymbol{x})}{\widetilde{\sigma}_{\boldsymbol{x}}}\right) \\ &= \mathrm{argmax}_{\boldsymbol{x}\in\mathcal{X}} \ \Phi\left(\frac{\tau_k - \widetilde{m}_k(\boldsymbol{x})}{\widetilde{\sigma}_{\boldsymbol{x}}}\right),\end{aligned}$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function (CDF) of a normal random variable. Since the CDF is monotonically increasing, the above optimization problem is equivalent to

$$\boldsymbol{x}_k^g = \mathrm{argmax}_{\boldsymbol{x}\in\mathcal{X}} \quad \frac{\tau_k - \widetilde{m}_k(\boldsymbol{x})}{\widetilde{\sigma}_{\boldsymbol{x}}}. \tag{7}$$

Remark 1. One of the most well-known kriging-based criteria for selecting candidate points is the EI function, which is defined in 2.4. In our algorithm, we do not use EI function to generate candidate points as the EI function can be very hard to optimize, especially when the number of sample points used to generate the metamodel is large. Instead, we use EI as a reward function for switching in Step II.

Statistical inference on prediction error is not readily available for RBF, as is in kriging. We propose a heuristic criterion for selecting candidate points in RBF which is inspired by equation (7). Note that the standard deviation $\widetilde{\sigma}_{\boldsymbol{x}}$ increases with the distance to the nearest sample point. This motivates us to define the uncertainty in the prediction $\widetilde{m}_k(\check{\boldsymbol{x}})$ at point $\check{\boldsymbol{x}}$ for non-kriging metamodels as

$$U_{\widetilde{\mathcal{X}}_k}(\check{\boldsymbol{x}}) = u\left(\min_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_k} \|\check{\boldsymbol{x}} - \boldsymbol{x}\|\right)^{\alpha}, \tag{8}$$

where $\|\cdot\|$ is an L^p-norm with $p = 2$ or ∞, and u and α are positive numbers. Replacing the standard deviation $\widetilde{\sigma}_{\boldsymbol{x}}$ in (7) by this newly-defined uncertainty function gives us the candidate point

$$\boldsymbol{x}_k^g = \mathrm{argmax}_{\boldsymbol{x}\in\mathcal{X}} \quad \frac{\tau_k - \widetilde{m}_k(\boldsymbol{x})}{U_{\widetilde{\mathcal{X}}_k}(\boldsymbol{x})}. \tag{9}$$

The performance of the prediction uncertainty function (8) will be discussed in Section 2.4 with a numerical example. In our numerical study, $\mathcal{C}_k^g$ consists of two points, the global minimizer of the metamodel and $\boldsymbol{x}_k^g$ determined by either (7) or (9) depending on the type of metamodel.

2.3 Local Search

This section presents the details of Step II-(iv) in Algorithm 1. We first propose an adaptive procedure to define the local region $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$ around $\widetilde{\boldsymbol{x}}_k^\star$, and then describe how to generate candidates using conventional direct search methods. It is worth mentioning that our algorithm structure and convergence property does not depend on specific local search algorithms, and any other local methods can be adopted.

2.3.1 Adaptive Local Region

At the kth iteration of a conventional direct search method, for a given direction set $\mathcal{D}_k$ and a step size Δ_k, the algorithm selects a set of candidate points $x_k + \Delta_k \boldsymbol{t}$, $\boldsymbol{t} \in \mathcal{D}_k$ around the current solution x_k. If a better solution is found, the step size remains the same or expands in the next iteration. Otherwise, the solution does not change and the step size shrinks. However, there are some difficulties in applying the step size updating scheme used in conventional direct search methods to the local search step in our framework. Unlike in the original direct search method, the local search step in our algorithm may not be conducted consecutively. As a result, current solution $\widetilde{\boldsymbol{x}}_k^\star$ can be far away from the previous solution $\widetilde{\boldsymbol{x}}_{k-1}^\star$ due to the intermediate global search steps. Therefore, the step size associated with $\widetilde{\boldsymbol{x}}_{k-1}^\star$ may not be suitable for identifying a new local region around $\widetilde{\boldsymbol{x}}_k^\star$. For example, if we follow the step size updating scheme in conventional direct search method, after our algorithm executes global search steps a number of times and identifies a cluster of points around a local minimizer, the step size in the local search step can remain unchanged and still be the large initial step size, which is clearly not what we want.

In our algorithm, we use a procedure to adaptively determine a local region for the local search step. The key condition for local convergence of direct search methods is that $\lim_{k\to\infty} \Delta_k = 0$ (Kolda et al., 2003). Note that in direct search algorithms the step size Δ_k shirks whenever the iteration does not produce any improvement for any directions in a positive spanning direction set. In our case, we already have a set of points around $\widetilde{\boldsymbol{x}}_k^\star$ such that their function values are higher than $f(\widetilde{\boldsymbol{x}}_k^\star)$. By thinking backwards, our goal is to develop a procedure to adaptively construct a local region based on a set of vectors that positively spans $\mathbb{R}^d$ and then find an appropriate step size Δ_k so that Δ_k converges to zero as $k \to \infty$. The key idea is as follows: Given $\widetilde{\mathcal{X}}_k$ and $\widetilde{\boldsymbol{x}}_k^\star$, we choose $d+1$ closest sample points to $\widetilde{\boldsymbol{x}}_k^\star$ and gradually include more points until we find a positive spanning set, denoted by $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_h\}$, where $\boldsymbol{v}_i = \boldsymbol{x}^{(i)} - \widetilde{\boldsymbol{x}}_k^\star$ and $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(h)}\} \subset \widetilde{\mathcal{X}}_k$. The positive span of the set $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_h\}$ is the convex cone

$$\operatorname{cone}(V) = \left\{ \boldsymbol{v} \in \mathbb{R}^d : \ \boldsymbol{v} = \sum_{i=1}^{h} w_i \boldsymbol{v}_i, \ w_i \geq 0 \right\}.$$

The question is how to check whether V positively spans $\mathbb{R}^d$ or not. Readers can refer to Ji et al. (2012) for the details of a procedure to check this condition. The full description of the procedure to identify an adaptive local region is given in Algorithm 2. In Step 5 of Algorithm 2, the choice of L^p-norm $\|\cdot\|$ affects the geometry of the local region. The choice of $p = \infty$ is considered to be suitable for box constraints, since local candidate points can be conveniently generated with proper scaling of the feasible region $\mathcal{X}$.

Algorithm 2 Adaptive Local Region

```
Inputs: A set of evaluated points $\widetilde{\mathcal{X}}_k$, the current best point $\widetilde{\boldsymbol{x}}_k^\star$, and the initial local region size $\rho_0$.
Outputs: An adaptive local region $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$ and a positive spanning set $V$ in $\mathbb{R}^d$.
Step 1: Set $h = d + 1$, which is the number of vectors in a minimal positive basis of $\mathbb{R}^d$.
Step 2: If $h \leq N_k$, go to Step 3; otherwise terminate with failure and set $\widetilde{\rho}_k = \rho_0$.
Step 3: Find $h$ nearest sample points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(h)}\} \subset \widetilde{\mathcal{X}}_k$ to $\widetilde{\boldsymbol{x}}_k^\star$.
Step 4: Determine whether $V = \{\boldsymbol{x}^{(1)} - \widetilde{\boldsymbol{x}}_k^\star, \ldots, \boldsymbol{x}^{(h)} - \widetilde{\boldsymbol{x}}_k^\star\}$ positively spans $\mathbb{R}^d$ or not.
Step 5: If $V$ is a positive spanning set for $\mathbb{R}^d$, then $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k) = \{\check{\boldsymbol{x}} \in \mathcal{X} : \|\widetilde{\boldsymbol{x}}_k^\star - \check{\boldsymbol{x}}\| \leq \widetilde{\rho}_k\}$, where
    $\widetilde{\rho}_k = \max_{i=1,\ldots,h} \|\boldsymbol{x}^{(i)} - \widetilde{\boldsymbol{x}}_k^\star\|$, and terminate the procedure; otherwise set $h = h + 1$ and go to Step 2.
```

2.3.2 Local Candidates

There is a wide variety of methods available to generate candidate points in a local region for black-box optimization problems, and in this subsection we discuss two methods. One is a random search method and the other is a deterministic method based on a positive spanning direction set.

The random search method generates local candidate points by perturbing $\widetilde{\boldsymbol{x}}_k^\star$ with a d-dimensional random variable (Spall, 2003). For box-constrained local regions, it is relatively easy to generate uniformly distributed candidates. We illustrate our random search procedure for two-dimensional case in Figure 4a. A similar procedure can be applied to higher-dimensional cases. First, we determine $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$ with neighboring points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(5)}\}$ by Algorithm 2. We then generate random vectors from a uniform distribution $\mathcal{U}_{[0,1]^2}$, and then map these vectors onto the feasible local region $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k) \cap \mathcal{X}$ in gray with a proper scaling to identify the set $\mathcal{C}_k^l$ of candidates.

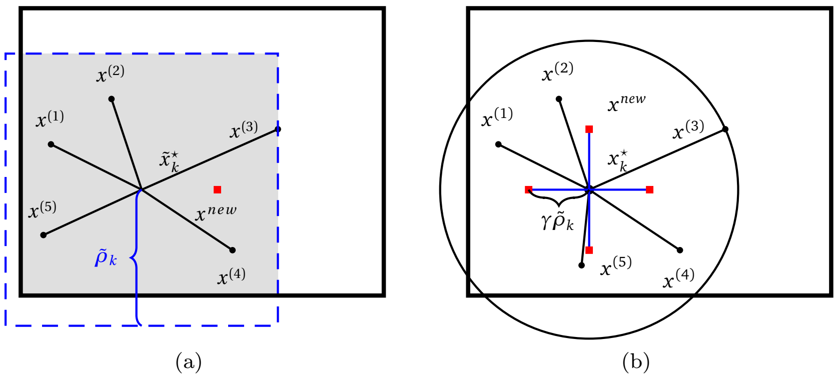


Figure 4: Generate candidate points in a local region. (a) Random search in a cubical local region. (b) Direct search in a spherical local region.

In the deterministic method, the step size Δ_k is set to be $\Delta_k = \gamma\widetilde{\rho}_k$, $0 < \gamma < 1$. Then, for a positive spanning set $\mathcal{D}_k$, a set of local candidate points is defined by $\mathcal{C}_k^l = \{\check{\boldsymbol{x}} : \check{\boldsymbol{x}} = \widetilde{\boldsymbol{x}}_k^\star + \Delta_k\, \boldsymbol{t}, \boldsymbol{t} \in \mathcal{D}_k\}$. Figure 4b illustrates the deterministic method. The candidate points are marked as red squares.

In the actual simulation in Step III, candidate points in $\mathcal{C}_k^l$ are sorted by the estimated function value in ascending order. To conserve the computational budget, the search step simulates each point in $\mathcal{C}_k^l$ until a point better than $\widetilde{\boldsymbol{x}}_k^\star$ is found. Based on the points evaluated, the algorithm updates N_k, $\widetilde{\mathcal{S}}_k$, and $\mathcal{S}_k$ accordingly.

2.4 Switch Criteria

In this section, we present the key element of our algorithm, the *Switch Criteria* (Step II). First, with information $\mathcal{S}_k$ and $\widetilde{\mathcal{S}}_k$, we construct multiple global metamodels and choose the best models m_k and $\tilde{m}_k$ using model selection techniques such as cross validation. We then determine "switching" based on the information obtained from the global metamodels. In kriging, we estimate the potential performance of each set of global and local candidate solutions using EI, and choose the set with higher potential. The EI function is defined based on the estimation of prediction error. However, as discussed in Section 2.2.4, information about estimation error is not readily available for other types of global metamodels. To make the EI function more general so that it can be applied to other metamodels such as RBF, we extend the EI function and propose a new measure to estimate potential improvement of candidate solutions, which is called Modified Expected Improvement (MEI).

In kriging, given the predictor $m_k(\check{\boldsymbol{x}})$ and its variance $\sigma_{\check{\boldsymbol{x}}}$ at point $\check{\boldsymbol{x}}$, the likelihood of achieving a positive improvement over the target value τ_k is given by

$$\frac{1}{\sigma_{\check{\boldsymbol{x}}}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\tau_k - m_k(\check{\boldsymbol{x}}) - a_{\check{\boldsymbol{x}}}}{\sigma_{\check{\boldsymbol{x}}}}\right)^2\right),$$

where $a_{\check{\boldsymbol{x}}} = \max(\tau_k - f(\check{\boldsymbol{x}}), 0)$. The expected improvement of $f(\check{\boldsymbol{x}})$ is then given by integrating all possible values of $a_{\check{\boldsymbol{x}}}$,

$$\begin{aligned} \mathrm{EI}(\check{\boldsymbol{x}}|\mathcal{S}_k) &= \int_0^\infty a_{\check{\boldsymbol{x}}} \frac{1}{\sigma_{\check{\boldsymbol{x}}}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\tau_k - m_k(\check{\boldsymbol{x}}) - a_{\check{\boldsymbol{x}}}}{\sigma_{\check{\boldsymbol{x}}}}\right)^2\right) \mathrm{d}\, a_{\check{\boldsymbol{x}}} \\ &= (\tau_k - m_k(\check{\boldsymbol{x}})) \cdot \Phi\left(\frac{\tau_k - m_k(\check{\boldsymbol{x}})}{\sigma_{\check{\boldsymbol{x}}}}\right) + \sigma_{\check{\boldsymbol{x}}} \cdot \phi\left(\frac{\tau_k - m_k(\check{\boldsymbol{x}})}{\sigma_{\check{\boldsymbol{x}}}}\right) \quad \text{if } \sigma_{\check{\boldsymbol{x}}} > 0, \end{aligned} \tag{10}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution and probability density functions of the standard normal distribution. The target value τ_k is typically set to be $f(\boldsymbol{x}_k^\star)$, the best objective function value so far. We use the full information set $\mathcal{S}_k$ in equation (10) instead of $\widetilde{\mathcal{S}}_k$, since the predictor $m_k(\breve{\boldsymbol{x}})$ based on $\mathcal{S}_k$ tends to be more accurate as it contains more information.

The reward function $\Gamma(\mathcal{C}|\mathcal{S}_k)$ is defined to be the average expected improvement,

$$\Gamma(\mathcal{C}|\mathcal{S}_k) = \frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{x}\in\mathcal{C}} \mathrm{EI}(\boldsymbol{x}|\mathcal{S}_k), \quad \text{where } \mathcal{C} = \mathcal{C}_k^g \text{ or } \mathcal{C}_k^l. \tag{11}$$

The average expected improvement is suitable when the algorithm evaluates all candidate points or evaluates them sequentially until a better point is found. In cases where only the candidate point with the largest expected improvement is evaluated, the reward function is given by

$$\Gamma(\mathcal{C}|\mathcal{S}_k) = \max_{\boldsymbol{x}\in\mathcal{C}} \mathrm{EI}(\boldsymbol{x}|\mathcal{S}_k).$$

For non-kriging metamodels, the same form of function used to predict the metamodel prediction uncertainty in the global search (8) is reused to replace the prediction error $\sigma_{\breve{\boldsymbol{x}}}$ in the EI function. The uncertainty in the prediction $m_k(\breve{\boldsymbol{x}})$ at point $\breve{\boldsymbol{x}}$ is defined by

$$U_{\mathcal{X}_k}(\breve{\boldsymbol{x}}) = u\left(\min_{\boldsymbol{x}\in\mathcal{X}_k} \|\breve{\boldsymbol{x}} - \boldsymbol{x}\|\right)^{\alpha},$$

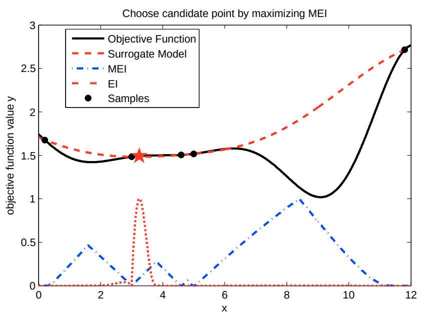
where u and α are positive numbers. As a candidate point is closer to the available samples, its prediction becomes more accurate. For a point $\breve{\boldsymbol{x}}$ with $U_{\mathcal{X}_k}(\breve{\boldsymbol{x}}) > 0$, the MEI function is given by

$$\mathrm{MEI}(\breve{\boldsymbol{x}}|\mathcal{S}_k) = (\tau_k - m_k(\breve{\boldsymbol{x}}))\Phi\left(\frac{\tau_k - m_k(\breve{\boldsymbol{x}})}{U_{\mathcal{X}_k}(\breve{\boldsymbol{x}})}\right) + U_{\mathcal{X}_k}(\breve{\boldsymbol{x}})\phi\left(\frac{\tau_k - m_k(\breve{\boldsymbol{x}})}{U_{\mathcal{X}_k}(\breve{\boldsymbol{x}})}\right). \tag{12}$$

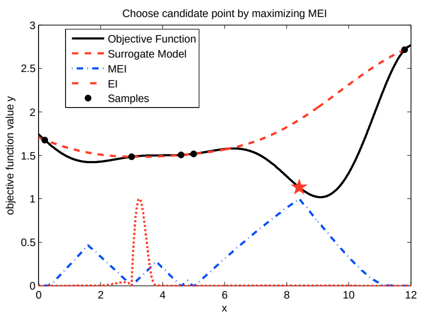
To illustrate the performance of MEI compared with EI, we approximate a one-dimensional function using kriging. Six sample points have been evaluated to obtain an initial model. We intentionally choose unevenly distributed sample points over the feasible region since such a scenario often occurs in high-dimensional problems, and also sample points are often clustered around a local region as the algorithm progresses. We set $u = 1$ and $\alpha = 1$.

In Figures 5a and 6a, a kriging model is constructed based on the six samples marked as circles. The MEI and EI functions have been normalized for better illustration. This will not affect performance since the normalized function retains the relative relationship between points. The short dashed line represents the EI function, and the dash-dot line represents the MEI function based on the distance measure. Figures 5 and 6 show how candidate points are selected based on the maximum EI and MEI function values, respectively. In Figures 5a and 6a, the global maximum of EI and MEI are located at $x = 3.25$ and $x = 8.4$. After the candidate points are evaluated, the kriging models are updated as shown in Figures 5b and 6b. After running four iterations, both solutions obtained by EI and MEI are close to the true global minimizer.

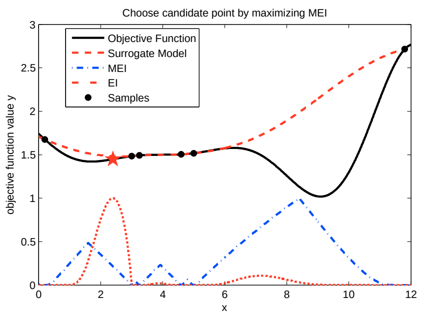
In general, MEI is easy to compute compared to EI, as the computation of prediction errors involves computing the inverse of the correlation matrix. Particularly, the computation of MEI for RBF models is even faster due to its simple and efficient construction process. The MEI function, however, may overestimate the prediction uncertainty, as can be seen in Figure 6d. This can be alleviated by choosing proper values of parameters u and α that can provide better approximations of prediction errors. In our numerical experiments, the results with $u = 1$ and $\alpha = 1$ show robust performance in all test problems.



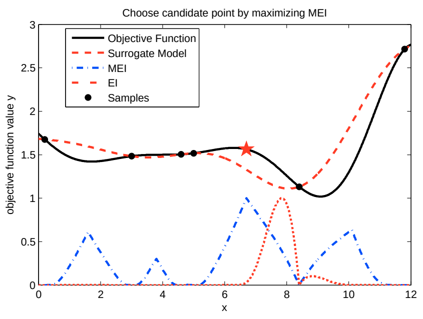
(a) Select $x = 3.25$ as the candidate.



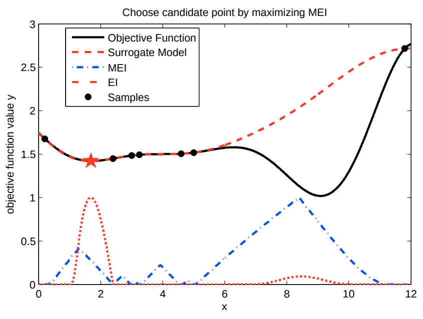
(a) Select $x = 8.4$ as the candidate.



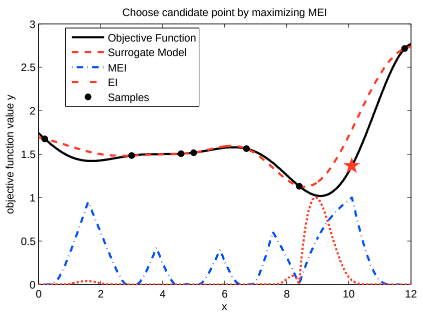
(b) Select $x = 2.4$ as the candidate.



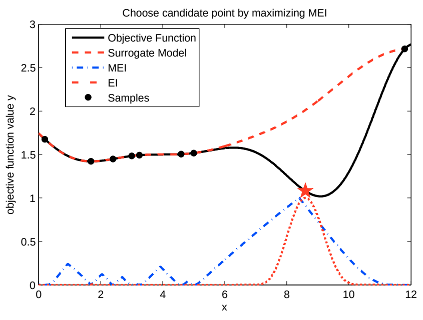
(b) Select $x = 6.7$ as the candidate.



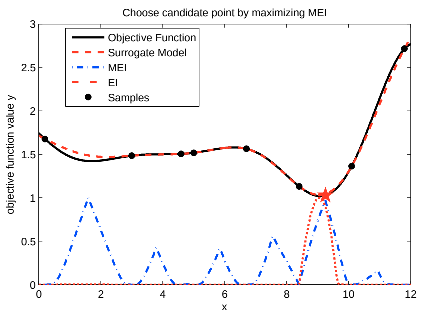
(c) Select $x = 1.69$ as the candidate.



(c) Select $x = 10.1$ as the candidate.



(d) Select $x = 8.6$ as the candidate.



(d) Select $x = 9.25$ as the candidate.

Figure 5: Candidates selected based on EI.

Figure 6: Candidates selected based on MEI.

2.5 Restart

One of the most straightforward ways to apply a local search algorithm to global optimization problems is to run it with multiple starting points. Similarly, Algorithm 1 restarts with new sample points when the current iterate is in a neighborhood of a local minimizer or a fully explored area. To check if it is time to initiate a restart, we monitor two types of local regions. One is $\mathcal{L}_k(\widetilde{\boldsymbol{x}}_k^\star, \widetilde{\rho}_k)$ constructed based on $\widetilde{\mathcal{S}}_k$, and the other one is $\mathcal{L}_k(\boldsymbol{x}_k^\star, \rho_k)$ constructed based on $\mathcal{S}_k$. The algorithm restarts whenever either $\widetilde{\rho}_k$ or ρ_k is smaller than a pre-specified parameter ϵ_ρ. A small $\widetilde{\rho}_k$ suggests that the algorithm is about to converge to a local optimal solution, while small ρ_k indicates that the algorithm is likely to converge to a visited area. Moreover, to further improve empirical performance, the algorithm also restarts when no progress has been made after a pre-specified number N_f of function evaluations.

When the algorithm restarts, a new set of design points will be generated to construct a metamodel $\widetilde{m}_k(\boldsymbol{x})$ in the next iteration. In most work found in the literature, a new set of random space-filling design points are used. In our algorithm, we choose samples with a greater distance from the full information set $\mathcal{S}_k$ in order to identify unexplored and promising regions. We first present our sampling scheme and then prove that it generates an everywhere dense set of candidate points, which guarantees the convergence of the algorithm to the global optimal solution.

We sample a set of new design points based on a weighted distance measure, where points with better predicted function values have higher weights. In this manner, we can expect better performance in the subsequent iteration by exploring a promising region. In the meantime, points in unexplored areas will be also generated with positive probability and eventually the full information set $\mathcal{S}_k$ will become a dense subset of the feasible region $\mathcal{X}$. This form of weights is also used in simulated annealing as the acceptance criterion for a feasible solution. Our weighting function is defined by

$$w_{\check{\boldsymbol{x}}} = \exp\left(-c\frac{m_k(\check{\boldsymbol{x}}) - \min_{\boldsymbol{x}\in\mathcal{X}} m_k(\boldsymbol{x})}{\max_{\boldsymbol{x}\in\mathcal{X}} m_k(\boldsymbol{x}) - \min_{\boldsymbol{x}\in\mathcal{X}} m_k(\boldsymbol{x})}\right),$$

where $0 \le c < +\infty$. Note that the weight at any point $\check{\boldsymbol{x}}$ is strictly positive, and thus no areas are completely neglected. Given the weight parameter c, a candidate point is generated by maximizing the minimum weighted distance,

$$\max_{\check{\boldsymbol{x}}\in\mathcal{X}} \min_{\boldsymbol{x}\in\mathcal{X}_k} w_{\check{\boldsymbol{x}}} \left\|\check{\boldsymbol{x}} - \boldsymbol{x}\right\|. \tag{13}$$

The weight parameter c controls the distribution of candidate points over $\mathcal{X}$. If $c = 0$, then $w_{\check{\boldsymbol{x}}} = 1$ for all $\check{\boldsymbol{x}} \in \mathcal{X}$. In this case, the algorithm simply generates candidates by maximizing the minimum distance with existing samples $\mathcal{X}_k$. With a larger value of c, samples with low predicted function values have much higher weights than samples with high predicted function values, and thus a sample point near the minimizer of the global metamodel will be generated. In the implementation of the algorithm, one can generate a set of samples $\mathcal{C}_k^r$ by changing the parameter c that satisfies $0 \le c_1 \le c_2 \le \cdots \le c_{|\mathcal{C}_k^r|} < +\infty$. In the next section, we prove the convergence of the algorithm under this sampling scheme.

2.6 Convergence Analysis

The following well-known theorem provides the foundation of the proof of convergence of our proposed algorithm to a global minimizer.

Theorem 1 (Törn and Pilinskas (1989))**.** *Let the minimization region $\mathcal{X}$ be compact. Then an global optimization algorithm converges to the global minimum of any continuous function if and only if the sequence generated by the algorithm is everywhere dense in $\mathcal{X}$.*

Now, we present our global convergence result.

Theorem 2. *For a given real-valued deterministic continuous function $f(\boldsymbol{x})$ defined on a compact subset $\mathcal{X}$ of $\mathbb{R}^d$. Suppose that we run Algorithm 1 infinitely many times. Then, the set of sample points generated by Algorithm 1 is dense, and thus convergence to the global minimizer of $f(\boldsymbol{x})$ is guaranteed.*

Proof. Proof of Theorem 2

Let $\mathcal{X}_{k_r} = \{\boldsymbol{x}_i\}_{i=1}^{N_{k_r}}$, $k_r \geq 1$, $r \geq 1$, be a sequence of sample points generated by Algorithm 1, where the subscript k_r represents the iteration where the algorithm restarts for the rth time. The candidate point $\boldsymbol{x}'$ generated by (13) satisfies,

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\boldsymbol{x}'} \|\boldsymbol{x}' - \boldsymbol{x}\| &= \max_{\breve{\boldsymbol{x}}\in\mathcal{X}} \min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\breve{\boldsymbol{x}}} \|\breve{\boldsymbol{x}} - \boldsymbol{x}\| \\ &\geq \min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\breve{\boldsymbol{x}}} \|\breve{\boldsymbol{x}} - \boldsymbol{x}\|, \quad \text{for any } \breve{\boldsymbol{x}} \in \mathcal{X}. \end{aligned} \tag{14}$$

Assume that $\mathcal{X}_{k_r}$ is not a dense set in $\mathcal{X}$. Then there exists a $\boldsymbol{x}_0 \in \mathcal{X}$ and $\epsilon > 0$ such that the open neighborhood $\mathcal{L}(\boldsymbol{x}_0, \epsilon) = \{\breve{\boldsymbol{x}} \in \mathcal{X} : \|\breve{\boldsymbol{x}} - \boldsymbol{x}_0\| < \epsilon\}$ does not contain any points in $\mathcal{X}_{k_r}$, that is,

$$\min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\boldsymbol{x}_0} \|\boldsymbol{x}_0 - \boldsymbol{x}\| \geq \epsilon w_{\boldsymbol{x}_0}. \tag{15}$$

Combining (14) with (15) yields

$$\min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\boldsymbol{x}'} \|\boldsymbol{x}' - \boldsymbol{x}\| \geq \min_{\boldsymbol{x}\in\mathcal{X}_{k_r}} w_{\boldsymbol{x}_0} \|\boldsymbol{x}_0 - \boldsymbol{x}\| \geq \epsilon w_{\boldsymbol{x}_0}.$$

Therefore,

$$\|\boldsymbol{x}' - \boldsymbol{x}\| \geq \frac{\epsilon w_{\boldsymbol{x}_0}}{w_{\boldsymbol{x}'}}, \quad \text{for any } \boldsymbol{x} \in \mathcal{X}_{k_r}. \tag{16}$$

This implies that given $\mathcal{X}_{k_r}$, the candidate point $\boldsymbol{x}'$ always keeps a distance of $\frac{\epsilon w_{\boldsymbol{x}_0}}{w_{\boldsymbol{x}'}} > 0$ from $\boldsymbol{x} \in \mathcal{X}_{k_r}$.

Since the algorithm restarts at iterations $k_1, \ldots, k_r$, there exists a strictly increasing sequence of integers $\{i_1, \ldots, i_r\}$ such that $\{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_r}\} \subset \mathcal{X}_{k_r}$ is the set of points generated by (13). By (16), we have

$$\|\boldsymbol{x}_p - \boldsymbol{x}_q\| \geq \delta, \quad \text{for any } p > q \text{ and } \boldsymbol{x}_p\,,\, \boldsymbol{x}_q \in \{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_r}\},$$

where $\delta = \min_p \frac{\epsilon w_{\boldsymbol{x}_0}}{w_{\boldsymbol{x}_p}} > 0$. Consider a collection of balls $\{\mathcal{L}(\boldsymbol{x}_{i_l}, \delta), l = 1, \ldots, r\}$, where $\mathcal{L}(\boldsymbol{x}_{i_l}, \delta) = \{\breve{\boldsymbol{x}} \in \mathcal{X} : \|\breve{\boldsymbol{x}} - \boldsymbol{x}_{i_l}\| < \delta\}$. Then, the collection is pairwise disjoint and the volume of each ball is $\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}\delta^d$, where $\Gamma(\cdot)$ is the gamma function. Let the volume of a set A be denoted by $\text{Vol}(A)$. Then,

$$\text{Vol}(\cup_{l=1}^r \mathcal{L}(\boldsymbol{x}_{i_l}, \delta)) = r\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}\delta^d \leq \text{Vol}(\mathcal{X}) < \infty.$$

We run the algorithm infinitely many times, and $r \to \infty$ as the algorithm progresses. Therefore, we have

$$\lim_{r\to\infty} \delta^d \leq \lim_{r\to\infty} \text{Vol}(\mathcal{X})\frac{\Gamma(\frac{d}{2}+1)}{\pi^{\frac{d}{2}}}\frac{1}{r} = 0,$$

which is a contradiction to $\delta > 0$. Thus, the set of points $\mathcal{X}_{k_r}$ generated by Algorithm 1 is dense in $\mathcal{X}$, and convergence to the global optimal point is achieved.

□

3 Computational Experiments

In this section, we conduct numerical studies to compare the performance of Algorithm 1 with a group of direct search/metamodel-based global optimization algorithms. We consider two variants of Algorithm 1, one based on kriging and the other based on RBF. We first describe our test problems and the alternative algorithms, and then present the comparisons.

3.1 Description of Test Problems

All global optimization methods are compared on 24 test problems from Dixon and Szegö (1978); Hock and Schittkowski (1980) and Moré et al. (1981). The dimension of the problems ranges from 2 to 12. The analytical form of the objective function is available for all test problems and the exact global optimum is known. Here we provide the details of eight representative test problems. The other test problems are discussed in Appendix A. Although the objective functions of these problems are not truly expensive to evaluate, the numerical results still provide meaningful insights on the performance of different algorithms.

Table 1: Properties of eight representative test problems.

Problem	Dim	Box constraints	Global	Local	$f^\star_{\text{true}}$
Ackley5	5	$[-20, 40]^5$	1	lots	1
Goldstein-Price	2	$[-2, 2]^2$	1	4	1
Hartman6	6	$[0, 1]^6$	1	4	1
Powell12	12	$[-5, 4]^{12}$	1	0	1
Rastrigin2	2	$[-4, 6]^2$	1	lots	1
Rosen5	5	$[-5, 5]^5$	1	0	1
Schwefel	2	$[-500, 500]^2$	1	lots	1
Shekel10	4	$[0, 10]^4$	1	10	1
SP10	10	$[-80, 120]^{10}$	1	0	1

Note: Constant values are added to each test problem in order to set the optimal function value $f^\star_{\text{true}}$ to be 1. This facilities the computation of relative errors and comparisons among algorithms.

The objective functions of most of these problems are multimodal. In particular, Ackley5, Rastrigin2, and Schwefel have a number of local optima all over the feasible region, and the global optimal solution is located off-center. The high-dimensional Rosenbrock function Rosen5 has a unique global minimum lying in a narrow valley. This feature causes many algorithms to converge slowly or even stop prematurely as they keep changing the search directions and shrinking the step size. The function values of Goldstein-Price, Rosen5 and Powell12 range over several orders of magnitude, which makes it difficult for metamodels to capture the underlying trends of the objective functions. Some problems like Powell12 and SP10 have large feasible regions. Shekel10 contains 10 local optima in a steep valley distributed over a flat terrain, and hence algorithms without an exploration procedure will easily be trapped in one local minimum. In addition, the extreme values of local optima give rise to near-singularities in the parameter estimations of metamodels.

3.2 Alternative Algorithms

We conduct numerical experiments on two variants of Algorithm 1.

1. KR-PB uses a kriging metamodel in the global search. Local candidate points within the adaptive local region determined by Algorithm 2 are generated by random search described in Section 2.3.2, and then the one with the largest EI-based reward (equation (10)) is included in $\mathcal{C}^l_k$.
2. RBF-PB uses a RBF model in the global search. Local candidate points are generated similarly as KR-PB, except that the reward is computed by the MEI function (equation (11)).

We compare these two variants of Algorithm 1 with five direct search/metamodel-based global optimization methods. The first algorithm is MADS (Audet and Dennis, 2006), a direct search method that consists of two steps, a Poll step and a Search step. The Poll step is a local search that is conducted by evaluating points in a local region, and guarantees convergence to a local solution. The Search step incorporates a metamodel-based search algorithm in order to search for a global solution. In our numerical experiments, MADS conducts the Search step at the first iteration using random sampling and performs only the Poll step afterward.

The next two alternative algorithms use kriging metamodels. MADS-DACE (Design and Analysis of Computer Experiments) is a variant of MADS, where kriging is used in the Search step. At each iteration, MADS-DACE conducts the Search step by minimizing a kriging model. After that, it performs the Poll step if the Search step fails and skips the Poll step otherwise. We implement MADS and MADS-DACE using the optimization software NOMADm (www.gerad.ca/NOMAD/Abramson/nomadm.html). EGO (Jones et al., 1998) is a sequential global optimization method based on kriging. At each iteration of the algorithm, the candidate point is obtained by globally maximizing the EI function. For implementation, we use EGO from TOMLAB Optimization Environment.

Lastly, we considered two RBF-based algorithms; MLMSRBF (Multistart Local Metric Stochastic Radial Basis Function Method, Regis and Shoemaker (2007)) and *rbfSolve* (Björkman and Holmström, 2000). MLMSRBF selects candidate solutions based on a weighted score of two criteria, the metamodel function value and the minimum distance from previously evaluated points. We use a Matlab version of MLMSRBF (www.sju.edu/∼rregis/pages/software.html) in our implementation. The *rbfSolve* method selects candidate solutions based on the *bumpiness* measure developed by Gutmann (2001). The algorithm first sets a target value as an approximation of the optimal function value, and then choose a point that produces a metamodel with the lowest bumpiness measure. We use *rbfSolve* implemented in TOMLAB Optimization Environment.

3.3 Experimental Setup

All numerical experiments are conducted in Matlab on a Windows machine with Intel(R) Xeon(R) 2.80GHz processor. For each test problem, we run each algorithm 30 times with a set of $2(d+1)$ random initial points, which are generated by Latin Hypercube Sampling (LHS) (McKay et al., 2000) with maximin distance design (Johnson et al., 1990). LHS generates design points that are uniformly projected onto each dimension, and works well with nonparametric metamodels such as kriging and RBF.

In MADS, the one-time search step generates $2(d+1)$ LHS design points and the initial mesh size in the Poll step is $0.1 * \min\{x_i^{ub} - x_i^{lb} : i = 1, \dots, d\}$, where $x^{lb} = (x_1^{lb}, \dots, x_d^{lb})$ and $x^{ub} = (x_1^{ub}, \dots, x_d^{ub})$ are the lower and upper bounds of the feasible region $\mathcal{X}$. The mesh refinement factor is set to be 0.5 and the mesh coarsening factor is 2. All the other parameters have default values. MADS-DACE uses kriging metamodels implemented in the DACE toolbox (Lophaven et al., 2002). We choose a constant regression model and a Gaussian correlation function. The initial values of parameters in the correlation functions are all 10, and the lower and upper bounds are 10^{-2} and 20. We leave all parameters in EGO, *rbfSolve* and MLMSRBF at default values in our implementations. In particular, we use the default value of 200 as the maximum computational budget for EGO since the method requires a considerable amount of time to generate candidate points as the sample size increases.

Table 2 summarizes the values of parameters for KR-PB and RBF-PB used in our numerical experiments. The parameter values are kept the same for all test problems.

3.4 Experimental Results

In this section, we present comparison results of the seven algorithms on the 8 representative problems. The results on the other 16 test problems can be found in Appendix B. To give an overview of performance for

Table 2: Parameter values for KR-PB and RBF-PB.

Parameter	Value
κ (Parameter for removing extreme function values)	1.5
ρ_0 (Initial mesh size)	$0.1 * \min\{x_i^{ub} - x_i^{lb} : i = 1, \ldots, d\}$
τ_k (Target value for finding global candidates)	$f(\boldsymbol{x}_k^\star)$
$\mathrm{Corr}[Y(\boldsymbol{x}_i), Y(\boldsymbol{x}_j)]$ (Correlation function in kriging)	$\exp\left(-\sum_{l=1}^d \theta_l \left\|\boldsymbol{x}_{is} - \boldsymbol{x}_{js}\right\|^{\theta_{d+1}}\right)$
$\boldsymbol{\theta}$ (Initial parameters in kriging)	$\theta_s = 1,\ s = 1, \ldots, d,\ \theta_{d+1} = 1.99$
Lower bound of $\boldsymbol{\theta}$ in kriging	$\theta_s = 10^{-3},\ s = 1, \ldots, d,\ \theta_{d+1} = 1$
Upper bound of $\boldsymbol{\theta}$ in kriging	$\theta_s = 10^{3},\ s = 1, \ldots, d,\ \theta_{d+1} = 2$
N_1 (Number of initial sample points)	$2(d+1)$
$N_{\max}$ (Maximum number of function evaluations)	800
c (Parameter in maximizing the weighted minimum distance)	$(0, 2, 4, 6)$ (cycle over these values)
$\lvert\mathcal{C}_k^r\rvert$ (Number of points newly generated while restarting)	$2(d+1)$
p (Order of L^p-norm)	∞
N_f (Maximum number of function evaluations while restarting)	$\min(4d, 0.02N_{\max})$
ϵ_ρ (Tolerance parameter for restarting)	10^{-3}

all test problems, we present results using the *performance profile* method proposed by Dolan and Moré (2002).

A common performance measure in black box optimization is the best objective function value found under a given computational budget, which is typically defined by a total number of function evaluations. As the true optimum for each test problem is known, we can measure the absolute error between the best objective function value found by the algorithm and the true optimal value. Given a number of replicated runs, we can obtain the average and the variance of the absolute error. The average absolute error represents the capability of finding an accurate solution, whereas the variance indicates the reliability of the algorithms in consistently producing accurate solutions.

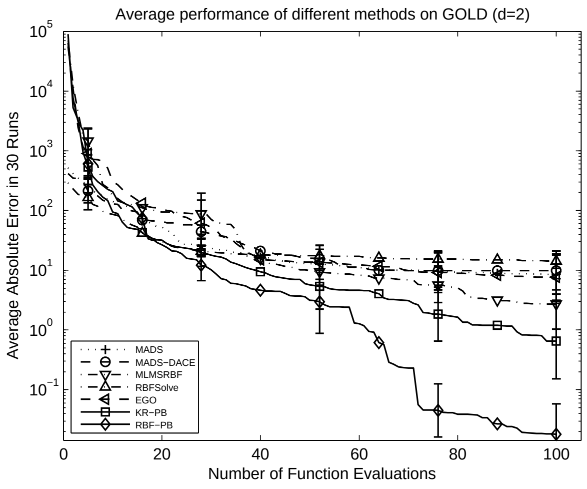
3.4.1 Results and Analysis on Individual Problems

Figures 7 and 8 present the statistics of absolute errors of the current best solution found by global optimization methods in 30 runs against the total number of function evaluations. Error bars represent 95% confidence intervals, which are calculated with the t-statistic. The length of one side of the error bar is about $\frac{t_{0.025,29}}{\sqrt{30}} \approx 0.3734$ times the sample standard deviation of the mean.

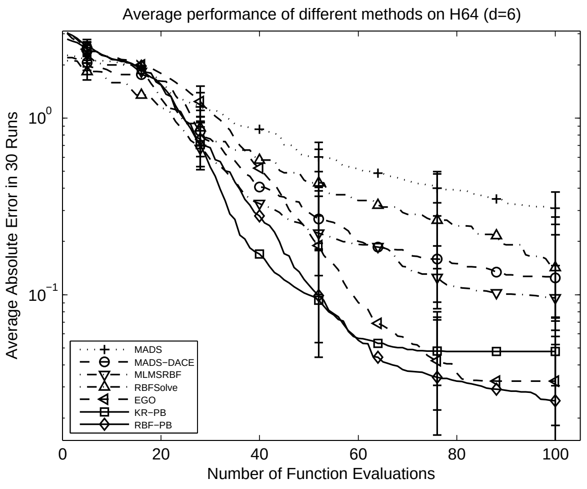
Note that, in Figures 7c and 7d, the average performance curve of EGO becomes flat when the number of function evaluations reaches around 150. This occurs when it takes a considerable amount of computational time to execute one iteration and the EGO algorithm announces termination prematurely. EGO is generally time-consuming compared with all the other methods, since it globally optimizes the EI function using a Branch-and-Bound method, which requires enumeration of all candidate solutions.

We first compared KR-PB with kriging-based methods. KR-PB outperforms MADS-DACE on all test problems. One can observe that MADS-DACE always performs better than MADS on all eight problems, which provides evidence that the global search step with kriging can significantly enhance the performance of MADS algorithms. At each iteration of MADS-DACE, both the Poll and Search steps are always conducted. On the other hand, our method performs either local or global search by comparing the potential performance of each step. Numerical results indicate that our switching scheme between local and global search steps is effective and enhances the performance of the algorithm, making a good balance between exploitation and exploration.

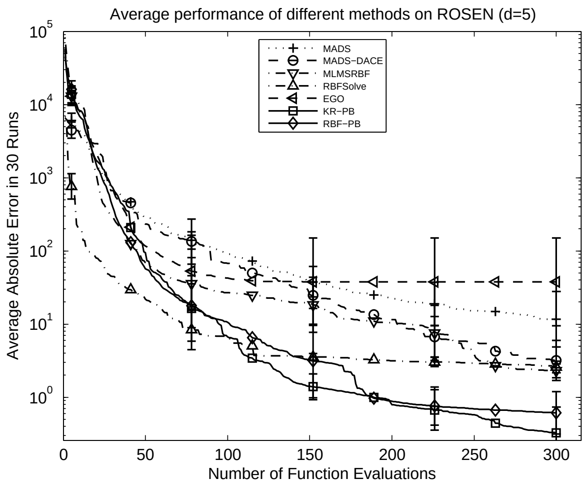
In the comparison with EGO, the performance of KR-PB is much more reliable on seven test problems, expected on Hartman6. Recall that EGO selects the point with maximum EI value as a candidate point



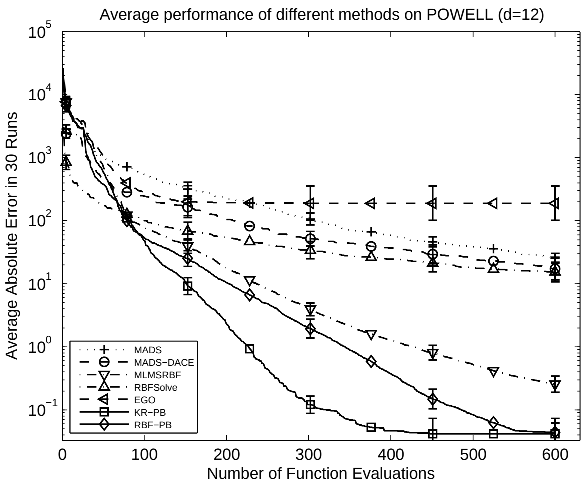
(a) Comparison of algorithms on Goldstein-Price.



(b) Comparison of algorithms on Hartman6.

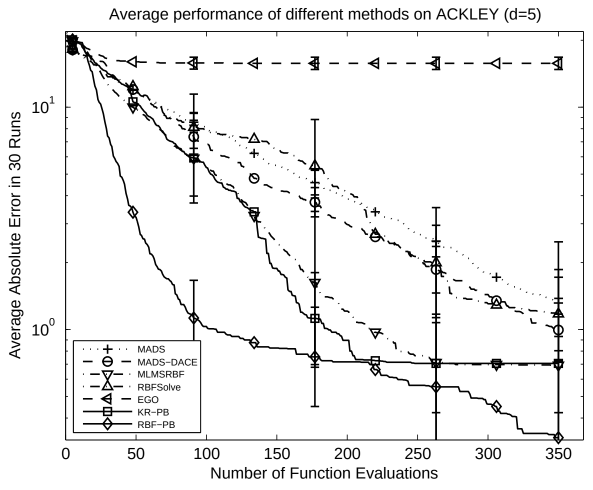


(c) Comparison of algorithms on Rosen5.

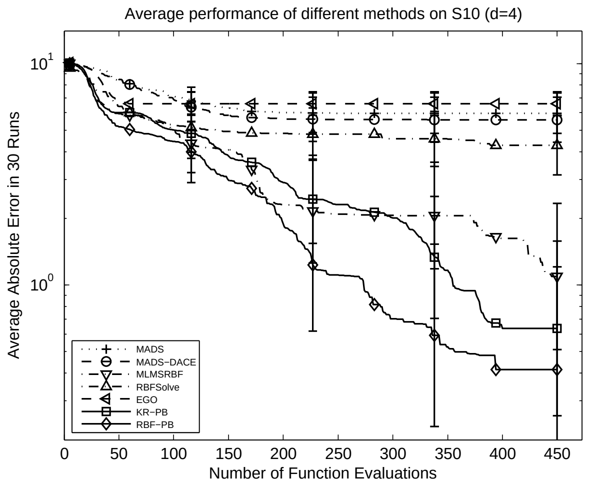


(d) Comparison of algorithms on Powell12.

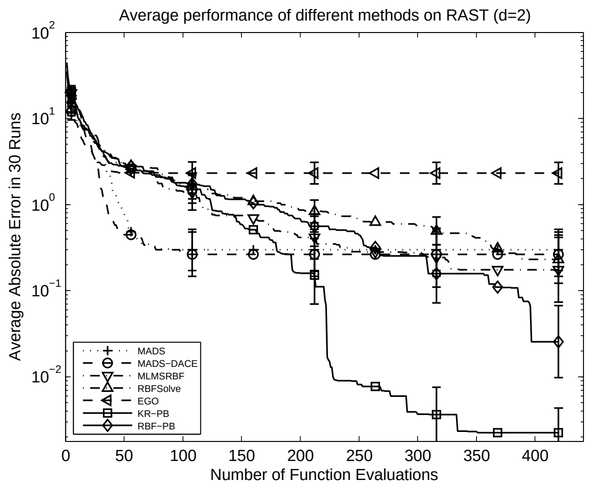
Figure 7: Experimental results on representative problems.



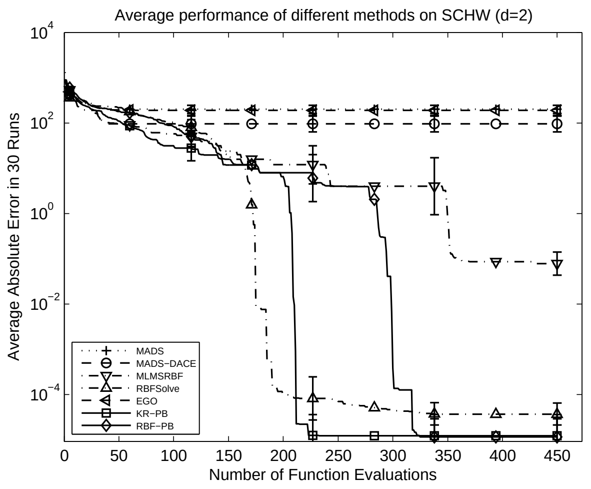
(a) Comparison of algorithms on Ackley5.



(b) Comparison of algorithms on Shekel10.



(c) Comparison of algorithms on Rastrigin2.



(d) Comparison of algorithms on Schwefel.

Figure 8: Experimental results on representative problems.

while in KR-PB candidate points are determined based on the average EI value over the entire feasible region or a local region depending on the search type. The numerical results demonstrate that the use of the EI-based reward function in determining search type and selecting candidate points is effective.

Secondly, we compare RBF-PB with RBF-based methods. RBF-PB performs best among all RBF-based methods, except for the Schwefel function. Particularly, RBF-PB substantially outperforms *rbfSolve* on high-dimensional problems, such as Shekel4, Ackley5 Hartman6, and Powell12. In the Schwefel problem, however, *rbfSolve* is able to identify the promising region faster than any other methods. The Schwefel function exhibits a number of local optima and its global optimum is located in the upper right corner. RBF-PB converges to the global solution slower than *rbfSolve* at the beginning, but eventually escapes from a local solution and catches up to *rbfSolve*. In all eight problems, MLMSRBF converges to global solutions slower than RBF-PB, but performs robustly. These results indicate that our MEI-based reward function is capable of maintaining a good balance between global and local search.

Finally, we analyze the performance of two variants of Algorithm 1, KR-PB and RBF-PB. In most test problems, both algorithms show good performance compared to other benchmark methods in terms of convergence rate and robustness in finding global solutions. KR-PB performs better than RBF-PB on Rosen5, Powell12, Rastrigin2 and Schwefel, while RBF-PB performs better on the other four test problems. For problems with extreme function values such as Ackley5 and Shekel4, RBF-PB performs better than KR-PB. This can be credited to the robustness of RBF models. In kriging, large variations in the estimated function values make it difficult to estimate model parameters.

3.4.2 Overall Performance

We now present the results of seven global optimization methods on all 24 test problems using *performance profiles* proposed by Dolan and Moré (2002). Let $\mathtt{P}$ and $\mathtt{A}$ be the set of test problems and the set of test algorithms, respectively. Algorithm $\mathtt{a} \in \mathtt{A}$ is said to solve problem $\mathtt{p} \in \mathtt{P}$ if the relative error of the best solution found by algorithm $\mathtt{a}$ within 800 number of function evaluations is less than or equal to a given value η. The performance measure $\mathtt{T}_{\mathtt{p},\mathtt{a}}$ is defined by the number of function evaluations required by algorithm $\mathtt{a}$ in solving problem $\mathtt{p}$. The larger the value of $\mathtt{T}_{\mathtt{p},\mathtt{a}}$, the worse the performance of algorithm $\mathtt{a}$. If algorithm $\mathtt{a}$ fails to solve problem $\mathtt{p}$, $\mathtt{T}_{\mathtt{p},\mathtt{a}} = \infty$. For each $(\mathtt{p}, \mathtt{a}) \in \mathtt{P} \times \mathtt{A}$, the *performance ratio* is defined as

$$\mathtt{R}_{\mathtt{p},\mathtt{a}} = \frac{\mathtt{T}_{\mathtt{p},\mathtt{a}}}{\min_{\mathtt{a}\in\mathtt{A}}\{\mathtt{T}_{\mathtt{p},\mathtt{a}}\}}.$$

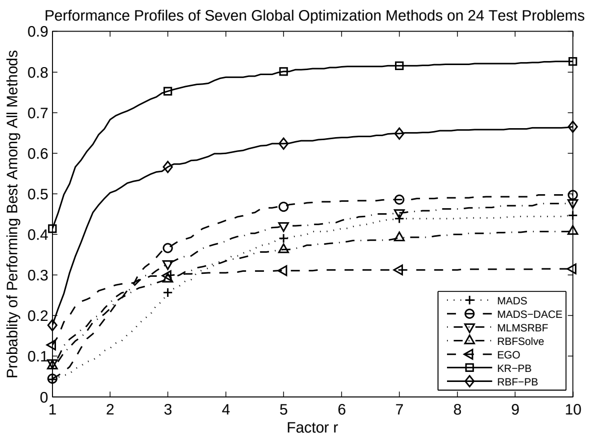
The lower bound $\mathtt{R}_{\mathtt{p},\mathtt{a}} = 1$ indicates that algorithm $\mathtt{a}$ performs best on problem $\mathtt{p}$ among all algorithms in $\mathtt{A}$. We are interested not only in the performance of algorithm $\mathtt{a}$ on a particular problem, but also in its overall performance on all test problems. We measure the overall performance by the following ratio:

$$\Pi_{\mathtt{a}}(\mathtt{r}) = \frac{|\{\mathtt{p} \in \mathtt{P} : \mathtt{R}_{\mathtt{p},\mathtt{a}} \leq \mathtt{r}\}|}{|\mathtt{P}|},$$

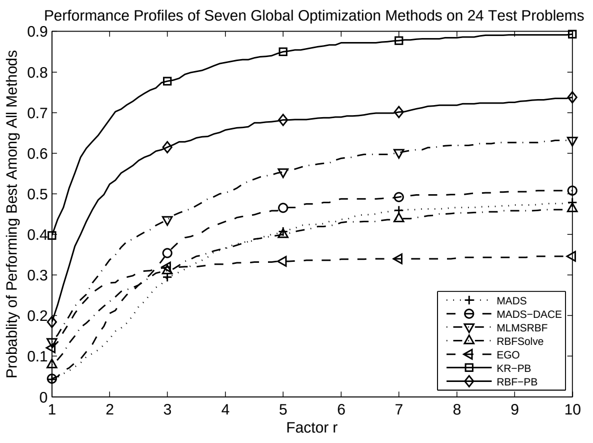
where $\mathtt{r} \in \mathbb{R}$ is a given threshold performance ratio. $\Pi_{\mathtt{a}}(\mathtt{r})$ is the probability that the performance ratio $\mathtt{R}_{\mathtt{p},\mathtt{a}}$ for any problem $\mathtt{p} \in \mathtt{P}$ is within a factor of $\mathtt{r}$. In particular, the value $\Pi_{\mathtt{a}}(1)$ is the probability that algorithm $\mathtt{a}$ performs best among all algorithms in $\mathtt{A}$. An algorithm that shows fast convergence of $\Pi_{\mathtt{a}}(\mathtt{r})$ to 1 as $\mathtt{r}$ increases is considered be efficient and robust.

Figure 9 presents the performance profiles of seven algorithms with $\eta = 1\%$ and $\eta = 5\%$. Table 3 gives the y-intercept $\Pi_{\mathtt{a}}(1)$ of each algorithm. One can see that with an accuracy level of 1%, the probability that KR-PB wins over the other algorithms is 41% (18% for RBF-PB). Similar results can be observed for an accuracy level of 5%. Figure 9 also shows superior performance of KR-PB and RBF-PB over other benchmark methods with high values of $\Pi_{\mathtt{a}}(\mathtt{r})$. Table 4 shows that with $\eta = 1\%$, KR-PB solves about 81%

of test problems and RBF-PB solves about 66% within a factor of 10 of the number of function evaluations used by the best algorithm.



(a) $\eta = 1\%$.



(b) $\eta = 5\%$.

Figure 9: Performance profile of seven algorithms: $\Pi_{\mathtt{a}}(\mathtt{r})$ vs. r

Table 3: Probability $\Pi_{\mathtt{a}}(1)$.

Accuracy	MADS	MADS-DACE	MLMSRBF	*rbfSolve*	EGO	KR-PB	RBF-PB
$\eta = 1\%$	0.0458	0.0444	0.0833	0.0736	0.1278	0.4069	0.1778
$\eta = 5\%$	0.0444	0.0431	0.1361	0.0792	0.1208	0.3972	0.1819

Table 4: Percentage $\Pi_{\mathtt{a}}(10)$.

Accuracy	MADS	MADS-DACE	MLMSRBF	*rbfSolve*	EGO	KR-PB	RBF-PB
$\eta = 1\%$	0.447	0.497	0.478	0.408	0.315	0.814	0.657
$\eta = 5\%$	0.479	0.508	0.632	0.463	0.346	0.893	0.738

4 Conclusions and Future Work

In this paper, we introduce a new framework for global optimization of black box problems. It combines and balances global and local search procedures for numerical efficiency, and can easily incorporate a variety of global and local search methods. We extend the EI function of kriging and propose a new measure to assess potential performance of candidate solutions for general metamodels utilizing distance information obtained from all evaluated sample points. This measure facilitates automatic and adaptive switching between global and local search procedures, a key feature of our framework, and leads to good empirical performance. We also present a new sampling scheme to identify promising regions that uses not only the distance information, but also the prediction model by the global metamodel. Our algorithm converges to the global optimum asymptotically with mild assumptions.

In the numerical experiments, we use kriging and RBF models as global metamodels, random search as the local method, giving rise to KR-PB and RBF-PB. These two methods are compared with five alternative algorithms: two methods (MADS-DACE, EGO) utilizing kriging metamodels, two RBF-based methods (*rbfSolve*, MLMSRBF), and one direct search method (MADS). All seven algorithms are tested on 24 test problems, whose dimensions range from 2 to 12. The results indicate that the two methods based on the proposed framework are robust and can find the global optimum with fewer function evaluations than other approaches, which suggests that our framework is promising for global optimization of black box problems. Note that MADS-DACE also combines a kriging-based global search procedure with MADS, a local optimization algorithm. The superior performance of our algorithms to MADS-DACE demonstrates that our framework achieves a better balance between global and local search procedures, and suggests that it is reasonable to use EI (or MEI) as a switch criterion, where the algorithm selects a subset of more promising candidate points that are already deemed to be promising by global and local methods.

There are many aspects of the proposed algorithm that can be improved to achieve better performance. In the future work, we will investigate more on how to choose parameters adaptively based on accumulated information. These include different choices of radial basis functions in RBF models or correlation functions in kriging, the cycle parameter c in the weighted distance measure, and the tolerance parameter ϵ_ρ for deciding unpromising iterations. One major difficulty concerns the appropriate time to interrupt unpromising iterations. A better choice for ϵ_ρ would depend on the metamodel since the metamodel gives some knowledge about the true function. Besides parameter calibrations, more sophisticated error estimates for RBF models are needed for better decisions in switch criteria.

Acknowledgements

This work is supported by ExxonMobil Research & Engineering R-266-000-062-597.

A Test Problems

Table 5 summarizes the major features of all test problems. The objective functions of all problems have analytical forms and the exact global optimum is known.

Table 5: Test problems for numerical experiments.

Problem	Dim	Box constraints	Global	Local	$f^\star_{\text{true}}$
Ackley5	5	$[-20, 40]^5$	1	lots	1
Beale	2	$[-4.5, 4.5]^2$	1	2	1
Branin	2	$[-5, 10] \cdot [0, 15]$	3	0	1
Colville	4	$[-10, 10]^4$	1	0	1
Goldstein-Price	2	$[-2, 2]^2$	1	4	1
Hartman3	3	$[0, 1]^4$	1	4	1
Hartman6	6	$[0, 1]^6$	1	4	1
Camel	2	$[-3, 3] \cdot [-2, 2]$	1	4	1
Powell4	4	$[-5, 4]^4$	1	0	1
Powell8	8	$[-5, 4]^8$	1	0	1
Powell12	12	$[-5, 4]^{12}$	1	0	1
Rastrigin2	2	$[-4, 6]^2$	1	lots	1
Rosen2	2	$[-5, 5]^2$	1	0	1
Rosen5	5	$[-5, 5]^5$	1	0	1
Schoen4.20	4	$[0, 1]^6$	1	≥ 10	1
Schoen6.20	6	$[0, 1]^6$	1	≥ 10	1
Schwefel	2	$[-500, 500]^2$	1	lots	1
Shekel5	4	$[0, 10]^4$	1	5	1
Shekel7	4	$[0, 10]^4$	1	7	1
Shekel10	4	$[0, 10]^4$	1	10	1
SP10	10	$[-80, 120]^{10}$	1	0	1
Wood	4	$[-10, 10]^4$	1	0	1
Zakharov2	2	$[-5, 10]^2$	1	0	1
Zakharov5	5	$[-5, 10]^5$	1	0	1

Constant values are added to each test problem to set the optimal function value $f^\star_{\text{true}}$ to be 1. This facilities the computation of relative errors and comparisons between algorithms.

Ackley5: $f(\boldsymbol{x}) = 21 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}\boldsymbol{x}_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi\boldsymbol{x}_i)\right)$

Beale: $f(\boldsymbol{x}) = (1.5 - \boldsymbol{x}_1 + \boldsymbol{x}_1\boldsymbol{x}_2)^2 + (2.25 - \boldsymbol{x}_1 + \boldsymbol{x}_1\boldsymbol{x}_2^2)^2 + (2.625 - \boldsymbol{x}_1 + \boldsymbol{x}_1\boldsymbol{x}_2^3)^2 + 1$

Branin: $f(\boldsymbol{x}) = (\boldsymbol{x}_2 - \frac{5.1\boldsymbol{x}_1^2}{4\pi^2} + \frac{5\boldsymbol{x}_1}{\pi} - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(\boldsymbol{x}_1) + 10.6021$

Colville:

$$f(\boldsymbol{x}) = 100(\boldsymbol{x}_1^2 - \boldsymbol{x}_2)^2 + (\boldsymbol{x}_1 - 1)^2 + (\boldsymbol{x}_3 - 1)^2 + 90(\boldsymbol{x}_3^2 - \boldsymbol{x}_4)^2 + \\ 10.1((\boldsymbol{x}_2 - 1)^2 + (\boldsymbol{x}_4 - 1)^2) + 19.8(\boldsymbol{x}_2 - 1)(\boldsymbol{x}_4 - 1) + 1$$

Goldstein-Price:

$$f(\boldsymbol{x}) = (1 + (\boldsymbol{x}_1 + \boldsymbol{x}_2 + 1)^2(19 - 14\boldsymbol{x}_1 + 3\boldsymbol{x}_1^2 - 14\boldsymbol{x}_2 + 6\boldsymbol{x}_1\boldsymbol{x}_2 + 3\boldsymbol{x}_2^2))\times \\ (30 + (2\boldsymbol{x}_1 - 3\boldsymbol{x}_2)^2(18 - 32\boldsymbol{x}_1 + 12\boldsymbol{x}_1^2 + 48\boldsymbol{x}_2 - 36\boldsymbol{x}_1\boldsymbol{x}_2 + 27\boldsymbol{x}_2^2)) - 2$$

Hartman3:

$$f(\boldsymbol{x}) = -\sum_{k=1}^{4} c_k \exp\left[-\sum_{i=1}^{3} A_{ki}(\boldsymbol{x}_i - P_{ki})^2\right] + 4.862888,$$

where

$$c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = \begin{bmatrix} 0.36890 & 0.11700 & 0.26730 \\ 0.46990 & 0.43870 & 0.74700 \\ 0.10910 & 0.87320 & 0.55470 \\ 0.03815 & 0.57430 & 0.88280 \end{bmatrix}$$

Hartman6:

$$f(x) = -\sum_{k=1}^{4} c_k \exp\left[-\sum_{i=1}^{6} A_{ki}(\boldsymbol{x}_i - P_{ki})^2\right] + 4.322368,$$

where

$$c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, A = \begin{bmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{bmatrix},$$

$$P = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}.$$

Camel: $f(\boldsymbol{x}) = 4\boldsymbol{x}_1^2 - 2.1\boldsymbol{x}_1^4 + \frac{1}{3}\boldsymbol{x}_1^6 + \boldsymbol{x}_1\boldsymbol{x}_2 - 4\boldsymbol{x}_2^2 + 4\boldsymbol{x}_2^4$

Powell4, Powell8, Powell12:

$$f(\boldsymbol{x}) = \sum_{i=1}^{d/4}(\boldsymbol{x}_{4j-4} + 10\boldsymbol{x}_{4j-3})^2 + 5(\boldsymbol{x}_{4j-2} - \boldsymbol{x}_{4j-1})^2 + (\boldsymbol{x}_{4j-3} - 2\boldsymbol{x}_{4j-2})^4 + 10(\boldsymbol{x}_{4j-4} - \boldsymbol{x}_{4j-1})^4$$

Rastrigin2: $f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d}(\boldsymbol{x}_i^2 - 10\cos(2\pi\boldsymbol{x}_i))$

Rosen2, Rosen5: $f(\boldsymbol{x}) = \sum\limits_{i=1}^{d-1}(100(\boldsymbol{x}_i^2 - \boldsymbol{x}_{i+1})^2 + (\boldsymbol{x}_i - 1)^2)$

Schoen4.20, Schoen6.20: This class of test functions is proposed by Schoen (1993) and it has the form

$$f(\boldsymbol{x}) = \frac{\sum_{i=1}^{k} f_i \prod_{j\neq i} \|\boldsymbol{x} - \boldsymbol{z}_j\|_2}{\sum_{i=1}^{k} \prod_{j\neq i} \|\boldsymbol{x} - \boldsymbol{z}_j\|_2},$$

where $k \geq 1$, $\boldsymbol{x} \in [0,1]^d$, $\boldsymbol{z}_j \in [0,1]^d$, for all $j = 1, \dots, k$ and $f_i \in \mathbb{R}$ for all $i = 1, \dots, k$.
We use the same method as Regis and Shoemaker (2007) to generate two Schoen functions with $(k,d) = (20,4)$ and $(k,d) = (20,6)$, where $\{\boldsymbol{z}_j\}_{j=1}^k$ and $\{f_i\}_{i=1}^k$ are generated uniformly on $[0,1]^d$ and $[0,100]$, respectively. We here do not present the analytic forms of $\{f_i\}_{i=1}^k$ and $\{\boldsymbol{z}_j\}_{j=1}^k$ due to their complexity (Ji et al., 2012). The optimal solutions of Schoen4.20 and Schoen6.20 are, respectively,

$$\boldsymbol{x}^\star_{\text{true}} = [0.81854, 0.95594, 0.22380, 0.55149] \text{ and}$$
$$\boldsymbol{x}^\star_{\text{true}} = [0.72784, 0.68585, 0.06888, 0.71714, 0.20900, 0.37853].$$

Shekel5, Shekel7, Shekel10:

$$S(\boldsymbol{x}) = \sum_{j=1}^{m} \frac{1}{(\boldsymbol{x} - A_j)^\mathsf{T}(\boldsymbol{x} - A_j) + c_j},$$

$$\text{where,} \quad A = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}, \quad c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$$S_{4,5} = S(\boldsymbol{x}) + 11.1532, \quad S_{4,7} = S(\boldsymbol{x}) + 11.4029, \quad S_{4,10} = S(\boldsymbol{x}) + 11.5364.$$

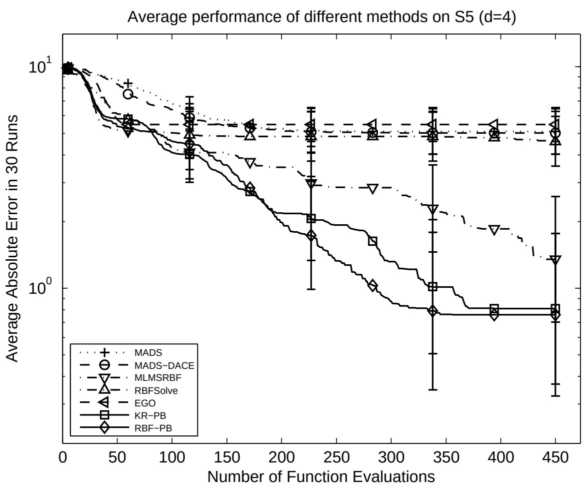
SP10: $f(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{x}$

Zakharov2, Zakharov5:

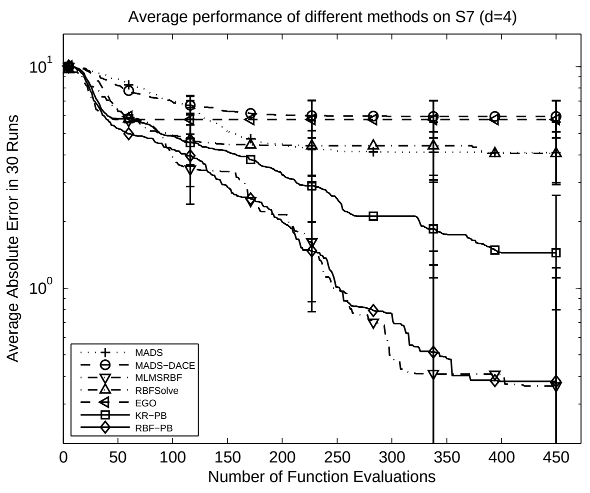
$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \boldsymbol{x}_i^2 + \left(\sum_{i=1}^{d} 0.5i\boldsymbol{x}_i\right)^2 + \left(\sum_{i=1}^{d} 0.5i\boldsymbol{x}_i\right)^4$$

B Additional Numerical Results

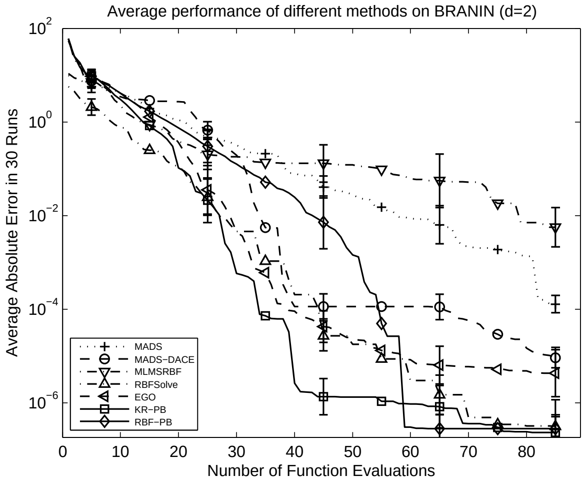
Here we present numerical results on the remaining sixteen test problems from Dixon and Szegö (1978); Hock and Schittkowski (1980) and Moré et al. (1981).



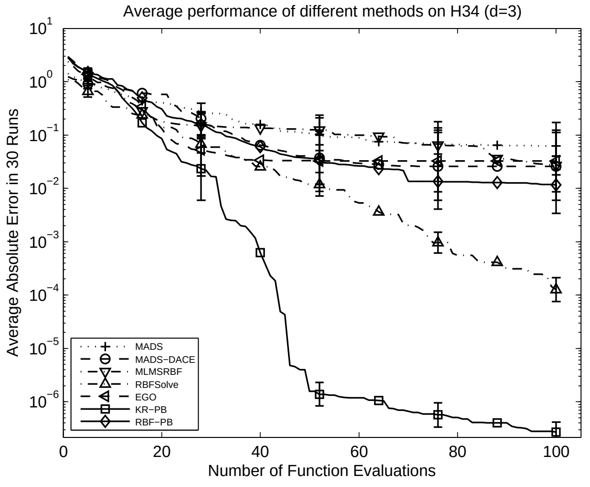
(a) Comparison of Algorithms on Shekel5.



(b) Comparison of Algorithms on Shekel5.



(c) Comparison of Algorithms on Branin.



(d) Comparison of Algorithms on Hartman3.

Figure 10: Numerical results for Shekel5, Shekel5, Branin and Hartman3.

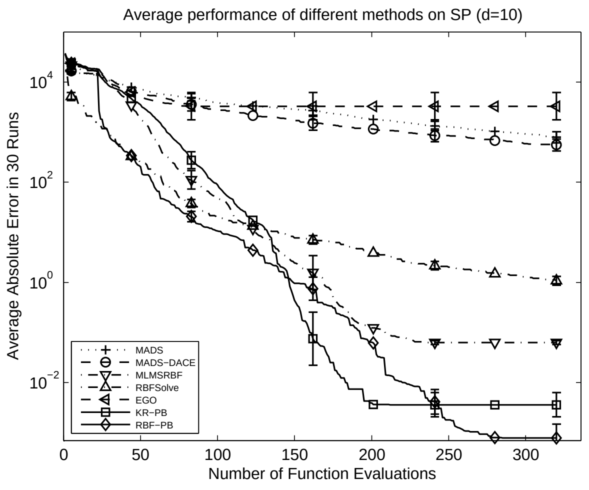
References

Andradóttir, S., A.A. Prudius. 2009. Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing* **21** 193–208.

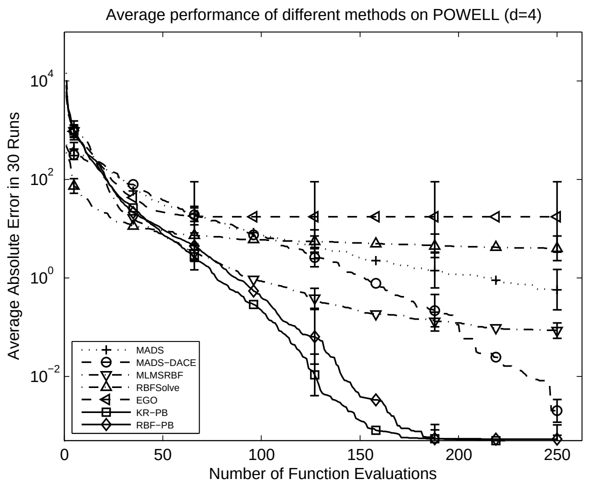
Audet, Charles, J. E. Dennis, Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization* **17** 188–217.

Bishop, C.M. 2006. *Pattern Recognition And Machine Learning*. Information Science and Statistics, Springer.

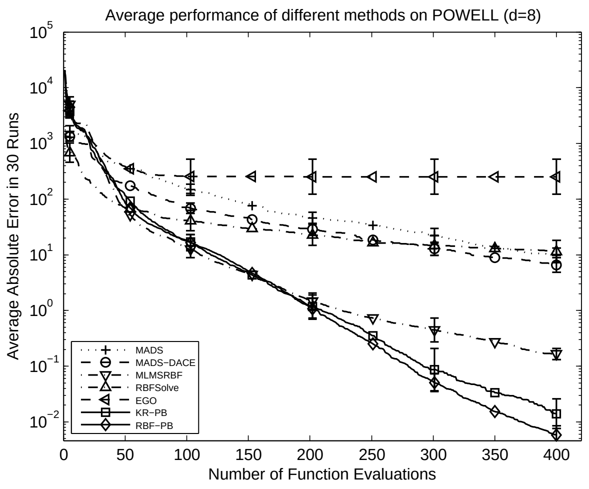
Björkman, M., K. Holmström. 2000. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering* **1** 373–397.



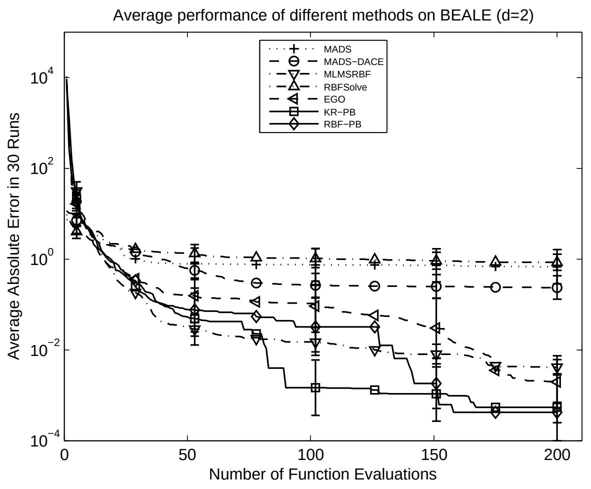
(a) Comparison of Algorithms on SP10.



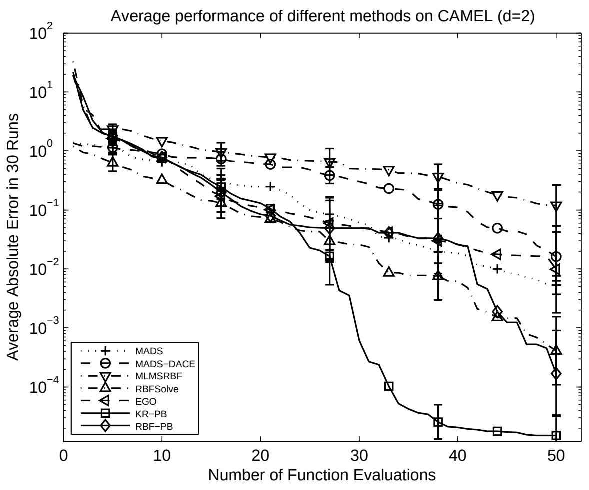
(b) Comparison of Algorithms on Powell4.



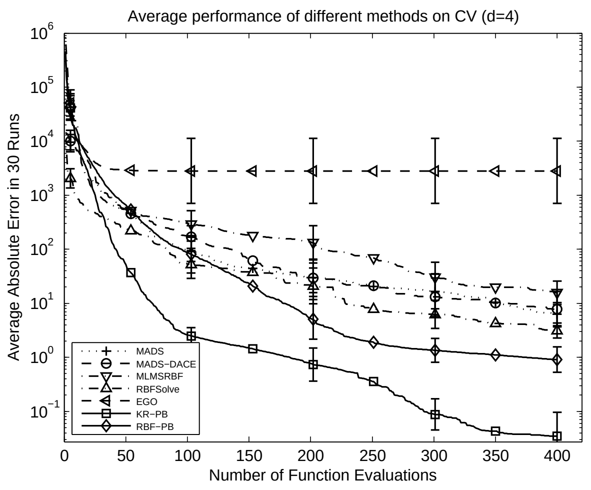
(c) Comparison of Algorithms on Powell8.



(d) Comparison of Algorithms on Beale.

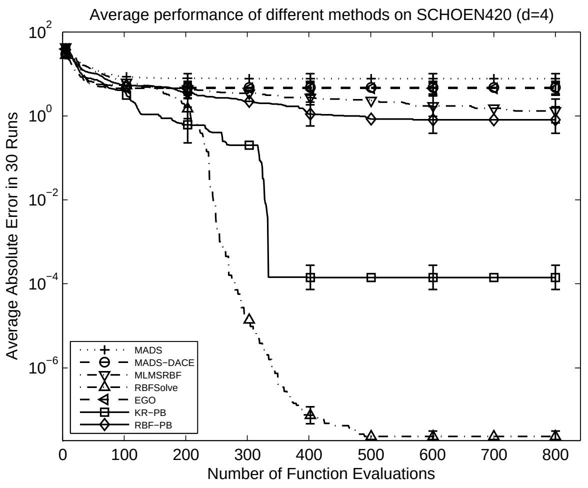


(e) Comparison of Algorithms on Camel.

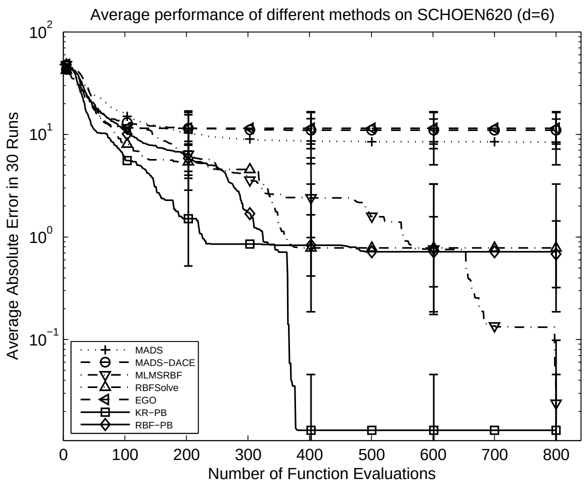


(f) Comparison of Algorithms on Colville.

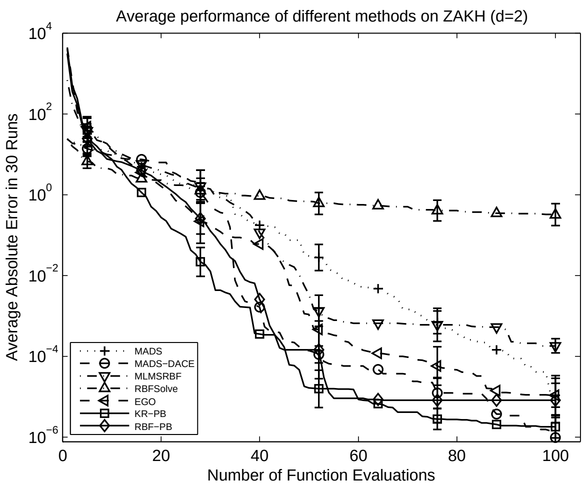
Figure 11: Numerical results for SP10, Powell4, Beale, Camel and Colville.



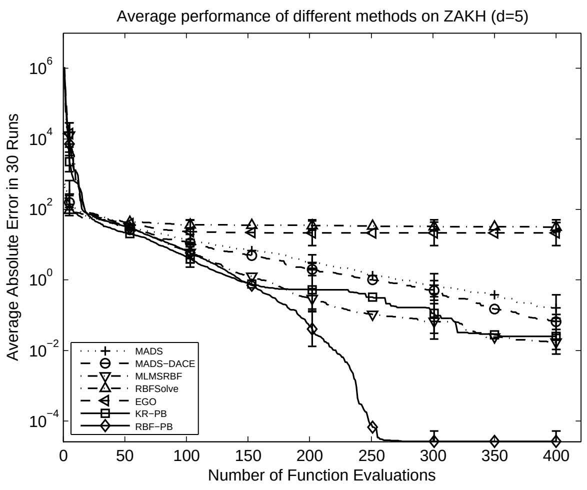
(a) Comparison of Algorithms on Schoen4.20.



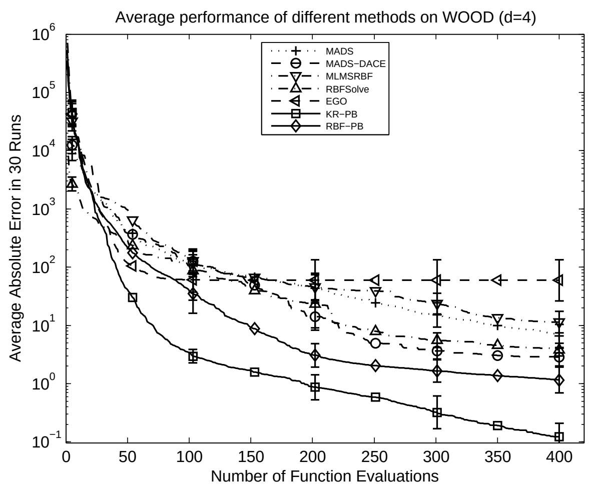
(b) Comparison of Algorithms on Schoen6.20.



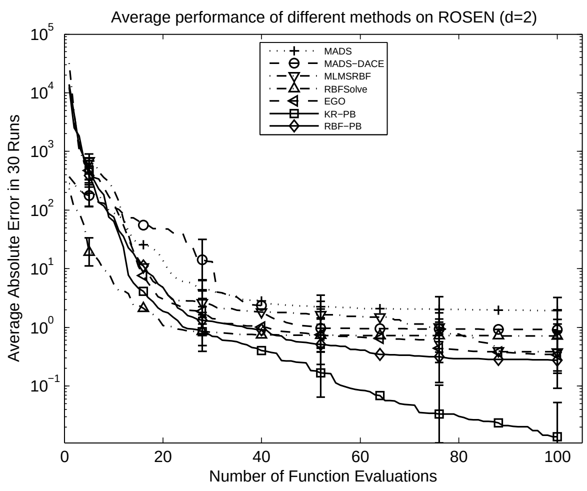
(c) Comparison of Algorithms on Zakharov2.



(d) Comparison of Algorithms on Zakharov5.



(e) Comparison of Algorithms on Wood.



(f) Comparison of Algorithms on Rosen2.

Figure 12: Numerical results for Schoen4.20, Schoen6.20, Zakharov2, Zakharov5, Wood and Rosen2.

Box, G.E.P., N.R. Draper. 1987. *Empirical model-building and response surfaces*. Wiley series in probability and mathematical statistics: Applied probability and statistics, Wiley.

Buhmann, M.D. 2003. *Radial basis functions: theory and implementations*, vol. 12. Cambridge Univ Pr.

Conn, A.R., K. Scheinberg, P.L. Toint. 1997. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming* **79** 397–414.

Conn, A.R., K. Scheinberg, L.N. Vicente. 2009. *Introduction to derivative-free optimization*. MPS-SIAM series on optimization, Society for Industrial and Applied Mathematics/Mathematical Programming Society.

Dixon, LCW, GP Szegö. 1978. The global optimization problem: an introduction. *Towards Global Optimization* **2** 1–15.

Dolan, E.D., J.J. Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical Programming* **91** 201–213.

Forrester, Alexander I J, Andy J Keane. 2009. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences* **45** 50–79.

Friedman, J.H. 1991. Multivariate adaptive regression splines. *The Annals of Statistics* 1–67.

Gutmann, H.-M. 2001. A radial basis function method for global optimization. *Journal of Global Optimization* **19** 201–227.

Hardy, R.L. 1971. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research* **76** 1905–1915.

Hastie, T., R. Tibshirani, J. H. Friedman. 2008. *The Elements of Statistical Learning*. 2nd ed. Springer-Verlag.

Hock, W., K. Schittkowski. 1980. Test examples for nonlinear programming codes. *Journal of Optimization Theory and Applications* **30** 127–129.

Jakobsson, S., M. Patriksson, J. Rudholm, A. Wojciechowski. 2010. A method for simulation based optimization using radial basis functions. *Optimization and Engineering* **11** 501–532.

Ji, Yibo, Sujin Kim, Wendy Xu. 2012. A New Framework for Combining Global and Local Methods in Black-Box Optimization. Tech. rep., NUS.

Jin, R., W. Chen, T.W. Simpson. 2001. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization* **23** 1–13.

Johnson, M E, L M Moore, D Ylvisaker. 1990. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* **26** 131–148.

Jones, D.R. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization* **21** 345–383.

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

Kleijnen, J.P.C., S.M. Sanchez, T.W. Lucas, T.M. Cioppa. 2005. A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing* **17** 263–289.

Kolda, T.G., R.M. Lewis, V. Torczon. 2003. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review* 385–482.

Kushner, H. J. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering* **86** 97–106.

Lophaven, S.N., H.B. Nielsen, J. Søndergaard. 2002. Dace: A matlab kriging toolbox. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark.

McKay, M D, R J Beckman, W J Conover. 2000. A comparision of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **42**.

Moré, J.J., B.S. Garbow, K.E. Hillstrom. 1981. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software* **7** 17–41.

Myers, R.H., D.C. Montgomery, C.M. Anderson-Cook. 2009. *Response surface methodology: process and product optimization using designed experiments*, vol. 705. John Wiley & Sons Inc.

Nelder, J A, R Mead. 1965. A simplex method for function minimization. *The Computer Journal* **7** 308–313.

Powell, MJD. 1992. The theory of radial basis function approximation. *Advances in Numerical Analysis* **2** 105–210.

Regis, R.G. 2011. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers & Operations Research* **38** 837–853.

Regis, R.G., C.A. Shoemaker. 2007. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing* **19** 457–509.

Regis, Rommel, Christine Shoemaker. 2012. A quasi-multistart framework for global optimization of expensive functions using response surface models. *Journal of Global Optimization* 1–3510.1007/s10898-012-9940-1.

Regis, Rommel G, Christine A Shoemaker. 2005. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global Optimization* **31** 153–171.

Sacks, J., W.J. Welch, T.J. Mitchell, H.P. Wynn. 1989. Design and analysis of computer experiments. *Statistical Science* **4** 409–423.

Sasena, M.J. 2002. Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations. Ph.D. thesis, University of Michigan.

Schoen, Fabio. 1993. A wide class of test functions for global optimization. *Journal of Global Optimization* **3** 133–137.

Sóbester, A., S.J. Leary, A.J. Keane. 2005. On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization* **33** 31–59.

Spall, J.C. 2003. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Wiley-Interscience series in discrete mathematics and optimization, Wiley-Interscience.

Torczon, Virginia. 1997. On the convergence of pattern search algorithms. *SIAM Journal on Optimization* **7** 1–25.

Törn, A., A. Pilinskas. 1989. *Global optimization*, vol. 350. Springer-Verlag. Berlin Heidelberg.

Upton, G., I. Cook. 1996. *Understanding Statistics*. Oxford University Press.