INFORMS Journal on Computing

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

A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions

Rommel G. Regis, Christine A. Shoemaker,

To cite this article:
Rommel G. Regis, Christine A. Shoemaker, (2007) A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. INFORMS Journal on Computing 19(4):497-509. https://doi.org/10.1287/ijoc.1060.0182

INFORMS Journal on Computing
Vol. 19, No. 4, Fall 2007, pp. 497–509
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1904 | 0497

DOI 10.1287/ijoc.1060.0182

A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions

Rommel G. Regis
Cornell Theory Center and School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853, rgr6@cornell.edu

Christine A. Shoemaker
School of Civil and Environmental Engineering and School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853, cas12@cornell.edu

We introduce a new framework for the global optimization of computationally expensive multimodal functions when derivatives are unavailable. The proposed *Stochastic Response Surface* (*SRS*) *Method* iteratively utilizes a response surface model to approximate the expensive function and identifies a promising point for function evaluation from a set of randomly generated points, called *candidate points*. Assuming some mild technical conditions, SRS converges to the global minimum in a probabilistic sense. We also propose *Metric SRS* (MSRS), which is a special case of SRS where the function evaluation point in each iteration is chosen to be the best candidate point according to two criteria: the estimated function value obtained from the response surface model, and the minimum distance from previously evaluated points. We develop a global optimization version and a multistart local optimization version of MSRS. In the numerical experiments, we used a radial basis function (RBF) model for MSRS and the resulting algorithms, *Global MSRBF* and *Multistart Local MSRBF*, were compared to 6 alternative global optimization methods, including a multistart derivative-based local optimization method. Multiple trials of all algorithms were compared on 17 multimodal test problems and on a 12-dimensional groundwater bioremediation application involving partial differential equations. The results indicate that Multistart Local MSRBF is the best on most of the higher dimensional problems, including the groundwater problem. It is also at least as good as the other algorithms on most of the lower dimensional problems. Global MSRBF is competitive with the other alternatives on most of the lower dimensional test problems and also on the groundwater problem. These results suggest that MSRBF is a promising approach for the global optimization of expensive functions.

Key words: global optimization; radial basis function; response surface model; function approximation; surrogate model; expensive function
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received October 2003; revised December 2004, September 2005; accepted March 2006. Published online in *Articles in Advance* July 20, 2007.

1. Introduction and Motivation

In this paper, we focus on global optimization problems where the objective function is continuous, multimodal, and expensive to evaluate. There are many engineering optimization problems in which a single objective function evaluation requires a computationally expensive simulation, which may take anywhere from a few minutes to many hours. Examples of such simulations are those that involve numerical solutions to partial differential equations (PDEs). For such problems, only a very limited number of function evaluations is feasible. Hence, our goal is to develop global optimization algorithms that produce reasonably good solutions with only a relatively small number of function evaluations.

For a precise description of our problem, let $\mathcal{D}$ be a compact set in $\mathbb{R}^d$ and let $f: \mathcal{D} \to \mathbb{R}$ be a deterministic continuous function. The *global optimization problem* (GOP) is to find $x^* \in \mathcal{D}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{D}$. Since f is continuous over the compact set $\mathcal{D}$, it follows that f attains its global minimum value on $\mathcal{D}$. A good introduction to the GOP can be found in Horst et al. (1995) and Torn and Zilinskas (1989). In this investigation, we focus on GOPs where f is treated as a black box that results from an expensive simulation and the derivatives of f are not available. For simplicity, we assume that the domain $\mathcal{D}$ is a compact hypercube in $\mathbb{R}^d$. Since f is expensive to evaluate, we wish to find $\tilde{x} \in \mathcal{D}$ such that $f(\tilde{x})$ is close to $\inf_{x \in \mathcal{D}} f(x)$ using only a very limited number of function evaluations.

Several algorithms have been used for expensive black box functions, including heuristics such as scatter search (Glover 1998, Laguna and Marti 2003), which is implemented in the OptQuest software (Laguna and Marti 2002), simulated annealing, and genetic algorithms. Other examples are multistart frameworks for local optimization such as OQNLP (Ugray et al. 2006), which uses scatter search to select starting points for any gradient-based solver. Note

that in the absence of derivatives, gradient-based methods can use finite differencing, or sometimes, automatic differentiation to obtain derivatives.

However, many of the standard optimization methods may not be appropriate for expensive black box functions. Heuristic methods like evolutionary algorithms typically require a very large number of function evaluations to obtain adequately good solutions. Finite differencing can be too expensive or it might be sensitive to noise. Automatic differentiation techniques, which generate codes that compute exact derivatives for a black box function, cannot be applied currently in cases where the source code for the objective function is not available. Moreover, truncation error in functions involving the solutions of PDEs or the presence of branching in the code for the black box function may cause an automatic differentiation software to produce inaccurate derivatives (Nocedal and Wright 1999). The alternative is to use *derivative-free optimization methods* such as the simplex reflection algorithm by Nelder and Mead (1965), the Parallel Direct Search algorithm by Dennis and Torczon (1991), or the more general class of pattern search algorithms (Torczon 1997). However, these methods generally require a large number of function evaluations since they do not take advantage of the inherent smoothness of some objective functions (Conn et al. 1997).

A suitable optimization method for expensive continuous functions is one that is based on a *response surface model* (also known as a *surrogate model* or *metamodel*). By a *response surface model*, we simply mean an inexpensive approximate model of the underlying expensive function. The response surface model is used to identify promising points for function evaluation. Examples of these optimization methods are those that rely on kriging models (Booker et al. 1999, Jones 2001, Jones et al. 1998, Simpson et al. 1998) and radial basis functions (RBFs) (Björkman and Holmström 2000, Gutmann 2001, Ishikawa et al. 1999). A popular example is traditional response surface methodology (Box and Draper 1987, Myers and Montgomery 1995), which generally involves low-order polynomial regression. Moore and Schneider (1996) and Moore et al. (1998) also developed a variety of methods that utilize approximation models to optimize functions that are not only costly but also noisy. Finally, the derivative-free methods by Conn et al. (1997) and Powell (2002, 2003) utilize multivariate polynomial interpolation models within a trust-region framework. Trust-region methods are meant for unconstrained local optimization but they can be modified to deal with constraints and they can be used for global optimization via a multistart approach.

Our focus is on global optimization methods that utilize *global response surface models*. By *global model*, we mean that all the points for which the function values are known are used to build the response surface model for the expensive function. In contrast, derivative-free trust-region methods approximate the expensive function *locally* and utilize only a subset of the previously evaluated points when building a response surface model.

In this paper, we introduce the *Stochastic Response Surface* (*SRS*) *Method*, which is a novel stochastic method for the global optimization of expensive functions that utilizes a response surface model. The new strategy is iterative and, in each iteration, the response surface model is updated and exactly one point is selected for function evaluation from a set of randomly generated points, called *candidate points*. Assuming that a continuous function f has a unique global minimizer x^* over the domain $\mathcal{D}$ and that the random candidate points and the probability distributions that generate them satisfy some suitable technical conditions, then SRS applied to f on $\mathcal{D}$ converges almost surely to x^*.

We also propose *Metric SRS* (MSRS), which is a special case of SRS where the function evaluation point in each iteration is chosen to be the candidate point with the best weighted score from two criteria: estimated function value obtained from the response surface model, and minimum distance from previously evaluated points. Moreover, we develop a global optimization version (*Global MSRS*) and a multistart local optimization version (*Multistart Local MSRS*) of MSRS, which differ in the way the random candidate points are generated and also in how the weights for the two criteria are selected.

In the numerical experiments, an RBF model was used for MSRS and the resulting algorithms, *Global MSRBF* and *Multistart Local MSRBF* were compared to 6 alternative methods on 17 multimodal test problems with dimensions ranging from 2 to 14 and on a 12-dimensional groundwater bioremediation application involving PDEs. The results indicate that Multistart Local MSRBF is the best among the different algorithms on most of the higher dimensional problems, including the groundwater application. It is also at least as good as the other algorithms on most of the lower dimensional problems. Global MSRBF is competitive with the other alternatives on most of the problems with dimensions $d \leq 10$ and also on the groundwater problem. These results suggest that the MSRBF approach is a promising method for the global optimization of expensive multimodal functions.

2. Stochastic Response Surface Method

2.1. General Framework

We now present the general framework for the Stochastic Response Surface (SRS) Method for the global optimization of expensive multimodal functions with

box constraints. As with any optimization method that utilizes a response surface model, we begin by evaluating the expensive function at an initial set of points, typically points from a space-filling experimental design. Then, in each iteration, the function evaluation point is obtained from a set of random points in $\mathcal{D}$, which we shall refer to as *candidate points*. The meaning of the word *random* in this context will be made clear in Section 2.2.

Below is the framework for SRS. Here, n is the number of previously evaluated points, $\mathcal{A}_n$ is the set of previously evaluated points, and $s_n(x)$ is the response surface model after n function evaluations.

Inputs:

1. A continuous real-valued function f defined on a compact hypercube $\mathcal{D} = [a, b] \subseteq \mathbb{R}^d$.
2. A particular response surface model, e.g., radial basis functions or neural networks.
3. A set of initial evaluation points $\mathcal{I} = \{x_1, \ldots, x_{n_0}\} \subseteq \mathcal{D}$, e.g., a space-filling experimental design.
4. The number of candidate points in each iteration, denoted by t. We require $t = O(d)$.
5. The maximum number of function evaluations allowed denoted by $N_{\max}$.

Output: The best point encountered by the algorithm.

Step 1 (*Do Costly Function Evaluation*). Evaluate the function f at each point in $\mathcal{I}$. Set $n = n_0$ and set $\mathcal{A}_n = \mathcal{I}$. Let x_n^* be the point in $\mathcal{A}_n$ with the best function value.

Step 2. While $(n < N_{\max})$

Step 2.1 (*Fit/Update Response Surface Model*). Fit/update the response surface model $s_n(x)$ using the data points $\mathcal{B}_n = \{(x_i, f(x_i)): i = 1, \ldots, n\}$.

Step 2.2 (*Randomly Generate Candidate Points*). Randomly generate t points $\Omega_n = \{y_{n,1}, \ldots, y_{n,t}\}$ in $\mathbb{R}^d$ (see Section 2.2). For each $j = 1, \ldots, t$, if $y_{n,j} \notin \mathcal{D}$, then replace $y_{n,j}$ by the nearest point in $\mathcal{D}$. We refer to the points in Ω_n as *candidate points*.

Step 2.3 (*Select the Next Function Evaluation Point*). Use the information from the response surface model $s_n(x)$ and the data points $\mathcal{B}_n = \{(x_i, f(x_i)): i = 1, \ldots, n\}$ to select the evaluation point x_{n+1} deterministically from the t candidate points in Ω_n.

Step 2.4 (*Do Costly Function Evaluation*). Evaluate the function f at x_{n+1}.

Step 2.5 (*Update Information*). $\mathcal{A}_{n+1} := \mathcal{A}_n \cup \{x_{n+1}\}$; $\mathcal{B}_{n+1} := \mathcal{B}_n \cup \{(x_{n+1}, f(x_{n+1}))\}$. Let x_{n+1}^* be the point in $\mathcal{A}_{n+1}$ with the best function value. Reset $n := n + 1$.

Step 3 (*Return the Best Solution Found*). Return $x_{N_{\max}}^*$.

A few remarks are in order. First, the initial evaluation points typically come from a space-filling experimental design. Morris and Mitchell (1995) and Koehler and Owen (1996) describe various experimental design techniques. Second, the response surface model can be any type of function approximation model such as RBFs (Powell 1992, 1999; Buhmann 2003), kriging (Cressie 1993, Sacks et al. 1989), regression splines (Friedman 1991), or neural networks. Third, in Section 2.2, we shall specify some technical conditions for Step 2.2 that guarantee the convergence of the algorithm. Moreover, in Step 2.2, we needed to ensure that the random candidate points are all in $\mathcal{D}$ since some commonly used probability distributions, such as the normal, are not guaranteed to yield points in $\mathcal{D}$ all the time.

In Step 2.3 we did not specify the procedure for the selection of the function evaluation point. Hence, any deterministic selection procedure is covered by the general SRS framework. In Section 4, we present one particular selection procedure where the function evaluation point is selected to be the point in Ω_n with the best weighted score from two criteria: estimated function value obtained from the response surface model, and minimum distance from previously evaluated points. Other reasonable selection criteria are possible. For instance, if a measure of error is available for the response surface model, then we can use an *error criterion* to select function evaluation points since evaluating points where the error is high results in a more accurate response surface model in the next iteration.

Finally, the SRS method extends to nonlinearly constrained problems (i.e., $\mathcal{D}$ is still compact but defined by nonlinear constraints). The only adjustment that needs to be done is to restrict the initial evaluation points and the candidate points in every iteration to belong to $\mathcal{D}$. This makes the method useful for a wider class of problems.

2.2. Generation of Random Candidate Points

We now provide some details on the generation of random candidate points in Step 2.2 of the SRS method. In the notation below, we will use uppercase letters to denote random vectors and distinguish them from ordinary vectors in $\mathbb{R}^d$. Assume $\mathcal{D} = [a, b] \subseteq \mathbb{R}^d$. For any random vector X whose realizations are in $\mathbb{R}^d$, we define the random vector $X_{\mathcal{D}}$, whose realizations are always in $\mathcal{D}$, as follows: For any sample point ω, $X_{\mathcal{D}}(\omega) = X(\omega)$ if $X(\omega) \in \mathcal{D}$ while $X_{\mathcal{D}}(\omega) = \min(\max(a, X(\omega)), b)$ if $X(\omega) \notin \mathcal{D}$. Here, min and max are taken componentwise. Note that when $X(\omega) \notin \mathcal{D}$, $X_{\mathcal{D}}(\omega)$ is simply the point in $\mathcal{D}$ that is nearest to $X(\omega)$.

In the notation below, X_n is the random vector representing the nth function evaluation point x_n and $Y_{n,j}$ is the random vector representing the random candidate point $y_{n,j}$ before it was forced to be in $\mathcal{D}$. Note that for each $n \geq n_0$, the value of X_{n+1} is selected deterministically from the values of the random vectors $(Y_{n,1})_{\mathcal{D}}, (Y_{n,2})_{\mathcal{D}}, \ldots, (Y_{n,t})_{\mathcal{D}}$.

For each $n \geq n_0$, let $\mathcal{E}_n := \{X_1, \ldots, X_{n_0}, Y_{n_0,1}, \ldots, Y_{n_0,t}, \ldots, Y_{n,1}, \ldots, Y_{n,t}\}$. For convenience, we also de-

fine $\mathscr{E}_{n_0-1} := \{X_1, \ldots, X_{n_0}\}$. After the nth function evaluation, where $n \geq n_0$, observe that the entire history of the algorithm is completely determined by the random vectors in $\mathscr{E}_{n-1}$. Now we shall impose some technical conditions on the $Y_{n,j}$'s so that the convergence of the resulting SRS method is guaranteed.

For Step 2.2 in Section 2.1, we shall require the following conditions:

[C1] For each $n \geq n_0$, $Y_{n,1}, Y_{n,2}, \ldots, Y_{n,t}$ are conditionally independent given the random vectors in $\mathscr{E}_{n-1}$.

[C2] For any $j = 1, \ldots, t$, $x \in \mathscr{D}$ and $\delta > 0$, there exists $\nu_j(x, \delta) > 0$ such that

$$P[Y_{n,j} \in B(x, \delta) \cap \mathscr{D} \mid \sigma(\mathscr{E}_{n-1})] \geq \nu_j(x, \delta)$$

for all $n \geq n_0$. Here, $B(x, \delta)$ is the open ball of radius δ centered at x and $\sigma(\mathscr{E}_{n-1})$ is the σ-field generated by the random vectors in $\mathscr{E}_{n-1}$.

Condition [C2] is a property of the collection of probability distributions that generate the random candidate points. It says that there must be positive probability that each random candidate point $Y_{n,j}$ will fall within any δ-neighborhood of any point of $\mathscr{D}$. Moreover, these probabilities must be bounded away from zero by a strictly positive function of x and δ that depends on j but is independent of n. Hence, this condition ensures that no regions of $\mathscr{D}$ will be neglected by the algorithm. Note that we are not requiring that $Y_{n,1}, Y_{n,2}, \ldots, Y_{n,t}$ be identically distributed. Moreover, for each $n > n_0$, each random vector $Y_{n,j}$ possibly depends on the random vectors $\{Y_{i,1}, Y_{i,2}, \ldots, Y_{i,t}\}_{n_0 \leq i \leq n-1}$ from the previous iterations.

2.3. Uniform and Normal Random Candidate Points

In this investigation, we consider two types of random candidate points in the compact hypercube $\mathscr{D} = [a, b] \subseteq \mathbb{R}^d$: (1) points generated uniformly at random throughout $\mathscr{D}$ (to which we refer as *type U candidate points*), and (2) points near the vicinity of the current best solution x_n^* obtained by adding random perturbations to x_n^* that are normally distributed with zero mean and covariance matrix $\sigma_n^2 I_d$, where $\inf_{n \geq n_0} \sigma_n > 0$ (*type N candidate points*). Below, we verify that these types of candidate points satisfy Condition [C2] above.

Define $\psi_{\mathscr{D}}(\delta) := \inf_{x \in \mathscr{D}} \mu(B(x, \delta) \cap \mathscr{D})$, where μ is the Lebesgue measure on $\mathbb{R}^d$. Observe that for the compact hypercube $\mathscr{D}$, we have $\psi_{\mathscr{D}}(\delta) > 0$ for any $\delta > 0$. Fix $1 \leq j \leq n$, $x \in \mathscr{D}$ and $\delta > 0$. For type U candidate points, we have

$$\begin{aligned} &P[Y_{n,j} \in B(x, \delta) \cap \mathscr{D} \mid \sigma(\mathscr{E}_{n-1})] \\ &\quad = \mu(B(x, \delta) \cap \mathscr{D})/\mu(\mathscr{D}) \geq \psi_{\mathscr{D}}(\delta)/\mu(\mathscr{D}) > 0, \end{aligned}$$

for any $n \geq n_0$, and so, Condition [C2] holds. Next, we consider type N candidate points. Since $\inf_{n \geq n_0} \sigma_n > 0$, it follows that $Y_{n,j}$ has a conditional density given $\sigma(\mathscr{E}_{n-1})$ for each $n \geq n_0$ and this is given by

$$g_{n,j}(y \mid \sigma(\mathscr{E}_{n-1})) = (2\pi\sigma_n^2)^{-d/2} \exp\left\{\frac{-\|y - x_n^*\|^2}{2\sigma_n^2}\right\}, \ y \in \mathbb{R}^d.$$

It is easy to check that

$$\begin{aligned} &g_{n,j}(y \mid \sigma(\mathscr{E}_{n-1})) \\ &\quad \geq \left(2\pi\left(\sup_{n \geq n_0} \sigma_n\right)^2\right)^{-d/2} \exp\left\{\frac{-\|b - a\|^2}{2(\inf_{n \geq n_0} \sigma_n)^2}\right\} =: C > 0, \end{aligned}$$

for all $y \in \mathscr{D}$. Hence,

$$\begin{aligned} &P[Y_{n,j} \in B(x, \delta) \cap \mathscr{D} \mid \sigma(\mathscr{E}_{n-1})] \\ &\quad = \int_{B(x,\delta) \cap \mathscr{D}} g_{n,j}(y \mid \sigma(\mathscr{E}_{n-1}))\, dy \\ &\quad \geq C\mu(B(x, \delta) \cap \mathscr{D}) \geq C\psi_{\mathscr{D}}(\delta) > 0 \end{aligned}$$

for any $n \geq n_0$, and so, Condition [C2] also holds.

3. Convergence

We now prove the convergence of the SRS Method in a probabilistic sense. The proof of the following theorem uses a similar argument to the one used by Spall (2003) in proving a theorem on random search:

THEOREM 1. *Let f be a function defined on $\mathscr{D} \subseteq \mathbb{R}^d$ and suppose that x^* is the unique global minimizer of f on $\mathscr{D}$ in the sense that $f(x^*) = \inf_{x \in \mathscr{D}} f(x) > -\infty$ and $\inf_{x \in \mathscr{D}, \|x - x^*\| \geq \eta} f(x) > f(x^*)$ for all $\eta > 0$. Suppose further that the SRS method generates the random vectors $\{X_n\}_{n \geq 1}$ and $\{Y_{n,1}, \ldots, Y_{n,t}\}_{n \geq n_0}$ satisfying Conditions* [C1] *and* [C2] *in* Section 2.2. *Define the sequence of random vectors $\{X_n^*\}_{n \geq 1}$ as follows: $X_1^* = X_1$ and $X_n^* = X_n$ if $f(X_n) < f(X_{n-1}^*)$ while $X_n^* = X_{n-1}^*$ otherwise. Then $X_n^* \to x^*$ almost surely.*

PROOF. Fix $\epsilon > 0$ and $n \geq n_0 + 1$. Clearly, the event $[X_n \in \mathscr{D}: f(X_n) < f(x^*) + \epsilon] = [X_n \in \mathscr{D}: |f(X_n) - f(x^*)| < \epsilon]$. Since f is continuous on x^*, there exists $\delta(\epsilon) > 0$ such that $|f(x) - f(x^*)| < \epsilon$ whenever $\|x - x^*\| < \delta(\epsilon)$. Hence, $[X_n \in \mathscr{D}: |f(X_n) - f(x^*)| < \epsilon] \supseteq [X_n \in \mathscr{D}: \|X_n - x^*\| < \delta(\epsilon)]$, and so,

$$\begin{aligned} &P[X_n \in \mathscr{D}: |f(X_n) - f(x^*)| < \epsilon \mid \sigma(\mathscr{E}_{n-2})] \\ &\quad \geq P[X_n \in \mathscr{D}: \|X_n - x^*\| < \delta(\epsilon) \mid \sigma(\mathscr{E}_{n-2})] \\ &\quad = P[X_n \in B(x^*, \delta(\epsilon)) \cap \mathscr{D} \mid \sigma(\mathscr{E}_{n-2})]. \end{aligned} \tag{1}$$

Observe that if $(Y_{n-1,j})_{\mathscr{D}} \in B(x^*, \delta(\epsilon)) \cap \mathscr{D}$ for each $j = 1, \ldots, t$, then the evaluation point $X_n \in B(x^*, \delta(\epsilon)) \cap \mathscr{D}$. Hence,

$$\begin{aligned} &P[X_n \in B(x^*, \delta(\epsilon)) \cap \mathscr{D} \mid \sigma(\mathscr{E}_{n-2})] \\ &\quad \geq P[(Y_{n-1,j})_{\mathscr{D}} \in B(x^*, \delta(\epsilon)) \cap \mathscr{D},\ j = 1, \ldots, t \mid \sigma(\mathscr{E}_{n-2})] \\ &\quad \geq P[Y_{n-1,j} \in B(x^*, \delta(\epsilon)) \cap \mathscr{D},\ j = 1, \ldots, t \mid \sigma(\mathscr{E}_{n-2})] \end{aligned}$$

$$= \prod_{j=1}^{t} P[Y_{n-1,j} \in B(x^*, \delta(\epsilon)) \cap \mathcal{D} \mid \sigma(\mathcal{E}_{n-2})]$$
$$\geq \prod_{j=1}^{t} \nu_j(x^*, \delta(\epsilon)) =: L(\epsilon) > 0, \quad (2)$$

where the equality and inequality involving the product sign follow from Conditions [C1] and [C2] in Section 2.2, respectively. Thus, from (1) and (2), we have $P[X_n \in \mathcal{D}: f(X_n) < f(x^*) + \epsilon \mid \sigma(\mathcal{E}_{n-2})] \geq L(\epsilon)$. By following the same argument as in the proof of the theorem in p. 40 of Spall (2003), we obtain $X_n^* \to x^*$ almost surely. □

The above theorem states that any algorithm that follows the SRS framework in Section 2.1 and satisfies Conditions [C1] and [C2] in Section 2.2 is guaranteed to converge to the global minimum almost surely provided that the algorithm is allowed to run indefinitely. The above convergence result is very general in the sense that it does not depend on the initial evaluation points, the response surface model, and the selection procedure for the function evaluation point. This allows an optimization practitioner to design a broad range of algorithms that are guaranteed to converge to the global minimum in a probabilistic sense.

4. Metric Stochastic Response Surface Method

4.1. A Distance Criterion for the Selection of Candidate Points

The *Metric SRS* (*MSRS*) *Method* is a special case of the SRS Method where the function evaluation point is selected from a set of randomly generated candidate points, as the one with the best weighted score from two criteria: (1) estimated function value obtained from the response surface model (*response surface criterion*), and (2) minimum distance from previously evaluated points (*distance criterion*). More precisely, each candidate point will be given a score between 0 and 1 on each of the two criteria. In any criterion, a more desirable point is given a score closer to 0. Ideally, a good candidate point for function evaluation should have low estimated function value (since the goal is to minimize) and should be far away from previously evaluated points (since this promotes a more global search). The precise details on how the scores are computed are given below. The two criteria are often in conflict, and hence, we pursue a trade-off by considering a weighted score of the two criteria for each candidate point. The next evaluation point is chosen to be the one with the lowest weighted score among all candidate points. In the literature, Jones (2001) used a similar idea of weighing the values of a response surface model and its predicted error to select the evaluation point in each iteration of a kriging-based response surface method.

The two criteria mentioned above are similar to the two filters used in the OQNLP multistart method (Ugray et al. 2006). In particular, the merit filter in OQNLP is used to select starting points of good quality in the sense that their penalty value is less than some threshold while the response surface criterion in MSRS gives preference to good candidate points in the sense that they have low response surface value. Moreover, the distance filter in OQNLP ensures that the starting points for the NLP solver are not too close to any previously found local minima while the distance criterion in MSRS gives preference to candidate points that are far away from previously evaluated points.

To implement the MSRS Method, we need to specify a distance metric D on $\mathbb{R}^d$ and a set of nonnegative weights $\{(w_n^R, w_n^D): n = n_0, n_0+1, \ldots\}$ such that $w_n^R + w_n^D = 1$ for all $n \geq n_0$ for the response surface and the distance criteria. Moreover, we replace Step 2.3 in the SRS framework by the following steps:

Step 2.3(a) (*Estimate the Function Value of Candidate Points*). For each $x \in \Omega_n$, compute $s_n(x)$. Also, compute $s_n^{\max} = \max\{s_n(x): x \in \Omega_n\}$ and $s_n^{\min} = \min\{s_n(x): x \in \Omega_n\}$.

Step 2.3(b) (*Determine the Minimum Distance from Previously Evaluated Points*). For each $x \in \Omega_n$, compute $\Delta_n(x) = \min_{1 \leq i \leq n} \mathsf{D}(x, x_i)$. Also, compute $\Delta_n^{\max} = \max\{\Delta_n(x): x \in \Omega_n\}$ and $\Delta_n^{\min} = \min\{\Delta_n(x): x \in \Omega_n\}$.

Step 2.3(c) (*Compute the Score for the Response Surface Criterion*). For each $x \in \Omega_n$, compute $V_n^R(x) = (s_n(x) - s_n^{\min})/(s_n^{\max} - s_n^{\min})$ if $s_n^{\max} \neq s_n^{\min}$ and $V_n^R(x) = 1$ otherwise.

Step 2.3(d) (*Compute the Score for the Distance Criterion*). For each $x \in \Omega_n$, compute $V_n^D(x) = (\Delta_n^{\max} - \Delta_n(x))/(\Delta_n^{\max} - \Delta_n^{\min})$ if $\Delta_n^{\max} \neq \Delta_n^{\min}$ and $V_n^D(x) = 1$ otherwise.

Step 2.3(e) (*Compute the Weighted Score*). For each $x \in \Omega_n$, compute $\mathcal{W}_n(x) = w_n^R V_n^R(x) + w_n^D V_n^D(x)$.

Step 2.3(f) (*Select the Next Evaluation Point*). Let x_{n+1} be the point in Ω_n that minimizes $\mathcal{W}_n$.

MSRS is easy to implement compared to alternative optimization methods for expensive functions like the kriging-based EGO method (Jones et al. 1998) and the RBF method by Gutmann (2001) since it avoids solving any difficult optimization subproblems. In Step 2.3(f), we simply find the point from the set Ω_n of $t = O(d)$ points that minimizes $\mathcal{W}_n(x)$. Moreover, assuming that the time complexity to evaluate $s_n(x)$ for each $x \in \Omega_n$ is $O(nd)$, which is true for the RBF model to be used later, and assuming that D is the Euclidean metric, then the selection of the next evaluation point (i.e., Steps 2.3(a)–2.3(f) above) can be accomplished in $O(nd^2)$ time. In contrast, EGO and the RBF method by Gutmann need to find the global minimum of a generally nonconvex utility function over the entire domain in every iteration. The worst-case complexity of nonconvex global optimization is

exponential in the dimension d (Vavasis 1991). Hence, we expect the time complexity of one iteration of MSRS to be better than either EGO or the RBF method of Gutmann.

In Sections 4.2 and 4.3, we describe the global optimization and local optimization versions of MSRS, to which we refer as *Global MSRS* and *Local MSRS*, respectively. Local MSRS can be used for global optimization by restarting it using a different experimental design whenever it appears to have converged to a local minimum. Global MSRS and Local MSRS differ mainly in the generation of candidate points in Step 2.2 of the SRS framework and also in the selection of the weights for the response surface and distance criteria.

4.2. Global Metric Stochastic Response Surface Method

In the *Global Metric Stochastic Response Surface Method* (Global MSRS), the set Ω_n of candidate points are generated uniformly at random throughout $\mathcal{D}$ (i.e., type U candidate points). To achieve a balance between global and local search, which is necessary for effective global optimization, the weights for the response surface and distance criteria are allowed to cycle through the iterations, starting from a high weight for the distance criterion (global search) and ending with a low weight for the distance criterion (local search). A high weight for the distance criterion, drives the algorithm towards unexplored regions of the domain. On the other hand, a low weight for the distance criterion, which is equivalent to a high weight for the response surface criterion, forces the algorithm to consider candidate points with potentially low function values. The guiding principle behind this method is that the selection of points for function evaluation has the dual goals of (a) finding new points that have a low objective function value, and (b) improving the future approximation models by sampling regions of $\mathcal{D}$ for which little information exists.

More precisely, the parameters (w_n^R, w_n^D) are set by performing cycles of $N+1$ iterations where each cycle employs a range of values for w_n^R, starting with a low value close to 0 (global search) and ending with $w_n^R = 1$ (local search). That is, $w_n^R = w_{n+N+1}^R \ \forall n \geq n_0$ and $0 \leq w_{n_0}^R \leq w_{n_0+1}^R \leq \cdots \leq w_{n_0+N}^R = 1$. (Recall that n_0 is the number of initial evaluation points.) We refer to N as the *cycle length* and we refer to the sequence $\langle w_{n_0}^R, w_{n_0+1}^R, \ldots, w_{n_0+N}^R = 1\rangle$ as the *RS-weight pattern* or simply the *weight pattern*.

In Global MSRS, we have stipulated that the end of a weight pattern be 1. When $w_n^R = 1$, Global MSRS selects the candidate point with the best response surface value. The purpose of this requirement is to ensure that we are doing the most natural thing of finding the global minimum of the approximation model every $N+1$ iterations. Moreover, when $w_n^R = 1$, we could replace Steps 2.3(a)–2.3(f) by a more accurate global minimization of $s_n(x)$ than the one provided by these steps, which is essentially a Monte Carlo approach for approximating the global minimum of $s_n(x)$.

4.3. Local Metric Stochastic Response Surface Method

In the *Local Metric Stochastic Response Surface Method* (Local MSRS), the set Ω_n of candidate points will be generated by adding random perturbations to the current best solution x_n^* that are normally distributed with zero mean and covariance matrix $\sigma_n^2 I_d$, where $\sigma_n = \rho_n l(\mathcal{D})$ (i.e., type N candidate points). Here, $l(\mathcal{D})$ is the length of one side of the hypercube $\mathcal{D}$, and we require $0 < \rho_n < 1/4$ for all $n \geq n_0$ and $\inf_{n \geq n_0} \rho_n > 0$. We shall refer to σ_n as the *step size*.

The convergence theorem still holds when the step sizes are relatively small. However, in this case, Local MSRS is essentially a local search algorithm since, for all practical purposes, it explores only the region near the current best solution in every iteration. Moreover, a large weight for distance criterion w_n^D will not force the algorithm towards unexplored regions since the candidate points are all close to the current best solution. Hence, we will simply set w_n^R to some fixed value close to 1 for all $n \geq n_0$ since this allows the algorithm to obtain improved function values quickly.

Since Local MSRS is essentially a heuristic local minimization algorithm, global search can be achieved by restarting the algorithm whenever it appears to have converged to a local minimum. To do this, we monitor the progress of the algorithm by recording the number of *consecutive* failed iterations (i.e., iterations that did not improve the best function value encountered so far), which we denote by C_{fail}. Whenever C_{fail} exceeds some pre-specified tolerance parameter $f_{\max}$, we reduce the current step size σ_n by half, reset C_{fail} to zero, and continue running the algorithm. The purpose of shrinking the step size is to facilitate convergence of the algorithm. We will set a maximum number of times that the step size will be reduced and denote this parameter by $r_{\max}$. Now the algorithm will be restarted when the number of times that the step size was reduced exceeds $r_{\max}$. We shall refer to the modified algorithm as *Multistart Local MSRS*.

When the algorithm gets restarted, it restarts completely from scratch using a new experimental design. While it may seem better to retain all the information from previous Local MSRS runs in the next run, our computational experience suggests otherwise. Suppose that a Local MSRS run converged to some local minimum that is not global. Preliminary experiments suggest that using the trajectory of this local minimization run in building the response surface model in the next run will bias the selection of candidate points towards regions containing the previously

found local minimizer. That is, the previously found local minimizer will attract the new trajectory. Hence, it seems that, in order to increase the chance of obtaining a new local minimizer, we need to ignore the previous local minimization trajectories and restart completely from scratch.

5. Radial Basis Function Model

We now present the radial basis function (RBF) interpolation model that was used in our implementation of the two MSRS algorithms and other RBF methods for global optimization. This RBF model was extensively studied by Powell (1992, 1999) and by Buhmann (2003) and was used as the basis of the RBF method by Gutmann (2001).

Assume that we are given n distinct points $x_1, \ldots, x_n \in \mathbb{R}^d$ where the function values are known. We use an interpolant of the form

$$s_n(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) + p(x), \quad x \in \mathbb{R}^d, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm, $\lambda_i \in \mathbb{R}$ for $i = 1, \ldots, n$, $p \in \Pi_m^d$ (the linear space of polynomials in d variables of degree less than or equal to m), and ϕ is one of the following forms: (1) *surface splines*: $\phi(r) = r^\kappa$, $\kappa \in \mathbb{N}$, κ odd, or $\phi(r) = r^\kappa \log r$, $\kappa \in \mathbb{N}$, κ even; (2) *multiquadrics*: $\phi(r) = (r^2 + \gamma^2)^\kappa$, $\kappa > 0$, $\kappa \notin \mathbb{N}$; (3) *inverse multiquadrics*: $\phi(r) = (r^2 + \gamma^2)^\kappa$, $\kappa < 0$; (4) *Gaussians*: $\phi(r) = e^{-\gamma r^2}$; where $r \geq 0$ and $\gamma > 0$. Special cases lead to the *cubic splines* ($\phi(r) = r^3$) and the *thin-plate splines* ($\phi(r) = r^2 \log r$).

Select a particular ϕ. Define the matrix $\Phi \in \mathbb{R}^{n \times n}$ by $\Phi_{ij} := \phi(\|x_i - x_j\|)$, $i, j = 1, \ldots, n$. Moreover, define m_ϕ to be -1 if ϕ is Gaussian or the inverse multiquadric, $\lfloor \kappa/2 \rfloor$ if ϕ is a surface spline, and $\lfloor \kappa \rfloor$ if ϕ is a multiquadric. Select $m \geq m_\phi$ and let $\hat{m}$ be the dimension of the linear space Π_m^d. (Note that $\hat{m} = \binom{m+d}{d}$.) Also, let $p_1, \ldots, p_{\hat{m}}$ be a basis of Π_m^d, and define the matrix $P \in R^{n \times \hat{m}}$ as follows: $P_{ij} := p_j(x_i)$, $i = 1, \ldots, n$; $j = 1, \ldots, \hat{m}$. The RBF model that interpolates the points $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ is obtained by solving the system

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0_{\hat{m}} \end{pmatrix}, \tag{4}$$

where $F = (f(x_1), \ldots, f(x_n))^T$, $\lambda = (\lambda_1, \ldots, \lambda_n)^T \in \mathbb{R}^n$, and $c = (c_1, \ldots, c_{\hat{m}})^T \in \mathbb{R}^{\hat{m}}$. Powell (1992) showed that the coefficient matrix in (4) is invertible if and only if $\text{rank}(P) = \hat{m}$, where P is the matrix defined above, and $\hat{m} = \dim(\Pi_m^d)$.

6. Computational Experiments

6.1. Alternative Global Optimization Methods

We will implement Global MSRS and Multistart Local MSRS using the RBF model described in Section 5. This gives us the *Global MSRBF* and *Multistart Local MSRBF* methods. To assess the significance of these proposed methods, we compare them with the following alternative global optimization methods for expensive functions: (1) a greedy RBF method; (2) the RBF method by Gutmann (2001); (3) a multistart derivative-based local optimization method; (4) a multistart derivative-free trust-region method; (5) a multistart pattern search algorithm; and (6) simulated annealing. The particular multistart approach we used was *multi level single linkage* (*MLSL*) (Rinnooy Kan and Timmer 1987). For the derivative-based method, we used *Sequential Quadratic Programming* (*SQP*), which is implemented in the FMINCON solver from the Matlab Optimization Toolbox (The MathWorks 2004b), and Implicit Filtering (Gilmore and Kelley 1995), which is implemented in Matlab by Kelley (1999). For the derivative-free trust region method, we used *UOBYQA* (Powell 2002), which is implemented in Matlab by Vanden Berghen and Bersini (2005). Finally, we used the pattern search algorithm (Torczon 1997) that is implemented in the Matlab Genetic Algorithm and Direct Search Toolbox (The MathWorks 2004a). The descriptions of these global optimization algorithms are provided in Appendix A, which can be found in the Online Supplement to this paper on the journal's website.

6.2. Test Problems

The different global optimization methods were compared on 17 multimodal test problems with dimensions ranging from 2 to 14 and on a 12-dimensional groundwater bioremediation application (GWB12) (Yoon and Shoemaker 1999) involving partial differential equations. The test problems include 6 of the Dixon-Szegö (1978) functions, 10 Schoen (1993) functions, and the 10-dimensional Griewank (1981) function. Details on the test problems are given in Appendix B, which can be found in the Online Supplement.

6.3. Experimental Setup

Each global optimization method was run for 30 trials on each of the problems. For each trial of each algorithm, we recorded the best function value encountered by the algorithm after every function evaluation.

To ensure a fair comparison among these methods, parameters and initial conditions were fixed for all of them. For instance, in all multistart methods, except Multistart Local MSRBF, we used the same random seed in each trial so that exactly the same starting points were used by the different local minimization solvers. Moreover, in all RBF methods, we used a particular RBF model of the form (3) (see Section 5) where ϕ is a thin-plate spline and $p(x)$ is a linear polynomial. The initial evaluation points, which are needed to fit the initial RBF model, were chosen to be the points of a randomly generated symmetric Latin

hypercube design (SLHD) (Ye et al. 2000) with $n_0 = 2(d+1)$ points. The justification for selecting $2(d+1)$ initial evaluation points is that this is twice the minimum number of points required to fit a thin-plate spline RBF model with a linear polynomial tail in $\mathbb{R}^d$. Of course, these methods can be used with a larger (or smaller) number of experimental design points. Each trial of an RBF method uses a different SLHD. However, exactly the same SLHD was used for all these algorithms on a particular trial.

For Global MSRBF (or simply G-MSRBF) and Multistart Local MSRBF (or simply ML-MSRBF), we used the Euclidean distance metric and we generated the random candidate points Ω_n (as described in Sections 4.2 and 4.3) of size $1000d$ in each iteration, where d is the dimension of the problem. For G-MSRBF, we used a weight pattern of $\langle 0.2, 0.4, 0.6, 0.9, 0.95, 1\rangle$. For ML-MSRBF, we set the initial step size to $\sigma_n = 0.1l(\mathcal{D})$ and we set $w_n^R = 0.95$ and $w_n^D = 0.05$ in every iteration. The tolerance parameter for deciding when to reduce the step size was set to $f_{\max} = \max(5, d)$ while the maximum number of times we can reduce the step size was set to $r_{\max} = 5$.

The global optimization subproblems in Greedy-RBF and Gutmann-RBF (see Appendix A in the Online Supplement) were solved by using the multi level single linkage (MLSL) multistart method by Rinnooy Kan and Timmer (1987). The local minimization runs for MLSL were carried out using the derivative-based solver FMINCON from the Matlab Optimization Toolbox (The MathWorks 2004b). To make sure that the algorithm does not spend an unreasonable amount of time in generating the next evaluation point, we allowed MLSL to run only a maximum of 30 local minimizations in solving each global optimization subproblem.

In the original implementation of Gutmann-RBF (Gutmann 2001) on the Dixon-Szegö test functions, the initial evaluation points were chosen to be the corners of the hypercube $\mathcal{D}$. Such a procedure is practical only for very low dimensional problems since the number of corners is exponential in the dimension. For higher dimensional problems, our choice of $2(d+1)$ SLHD points is more reasonable.

In all RBF methods, we adopted a strategy used by Gutmann (2001) and by Björkman and Holmström (2000) of replacing large function values by the median of all available function values. The purpose of this transformation is to prevent oscillations in the RBF interpolant that are due to the large differences in function values. Hence, the RBF model we are using is not really an approximate global model of the unknown function since it does not interpolate data points with large function values. However, this is not a problem since our goal is to minimize the unknown function and the model still is a good approximation for data points with low function values.

As noted in Section 6.1, we implemented the multistart approach known as multi level single linkage (MLSL) for the local minimization methods. In each iteration of MLSL, we generated a random sample of 10 points where the objective function was evaluated and the best half of the collection of all sample points (including those from previous iterations) were subjected to a critical distance criterion (see Section A.1.3 in the Online Supplement) to determine the starting points for the local minimization runs. For each of the local minimization methods mentioned in Section 6.1, we also implemented a multistart approach that uses the best point in an SLHD of size $2(d+1)$ as the starting point for the initial local minimization run. For comparison purposes, we used the same SLHD that was used by the other RBF methods in a given trial.

We also implemented two versions of simulated annealing depending on how the initial evaluation points are selected. In the first version, $N_{\text{init}} = \min(2d, 10)$ initial evaluation points were generated uniformly at random on $\mathcal{D}$. In the second version, the initial evaluation points correspond to the same SLHD of size $2(d+1)$ that was used by all RBF methods in a given trial. In both versions, the expensive function is evaluated at these initial points and the starting point for the algorithm is always chosen to be the one with the best function value.

7. Results and Discussion

7.1. Description of the Results

For each algorithm and for each test function, we calculated the average best function value (over 30 trials) obtained after every function evaluation. The results are summarized in 18 figures. Because of space limitations, here we only show 10 (Figures 1–10) of the 18 figures corresponding to the higher dimensional test problems and the 12-dimensional groundwater problem. All other results are given in Appendix C, which can be found in the Online Supplement. The error bars represent 95% confidence intervals. That is, the length of one side of the error bar is about 1.96 times the standard error of the mean. To highlight the differences in performance of the algorithms, we started the plots at $n_0 = 2(d+1)$ function evaluations, which is at the end of the evaluation of the experimental design points. Finally, in Table 2 of Appendix C, we report the average running times (over 30 trials) of the different algorithms on each of the test problems after a fixed number of function evaluations (Column 2 of the table). All algorithms, except Multistart Pattern Search (PS), were run on a 1.5 GHz Pentium 4 desktop machine. Multistart PS was run on a 2.4 GHz Pentium 4 desktop machine because of the unavailability of a software license.

In the discussion that follows, we will use the average best function value obtained after a fixed number

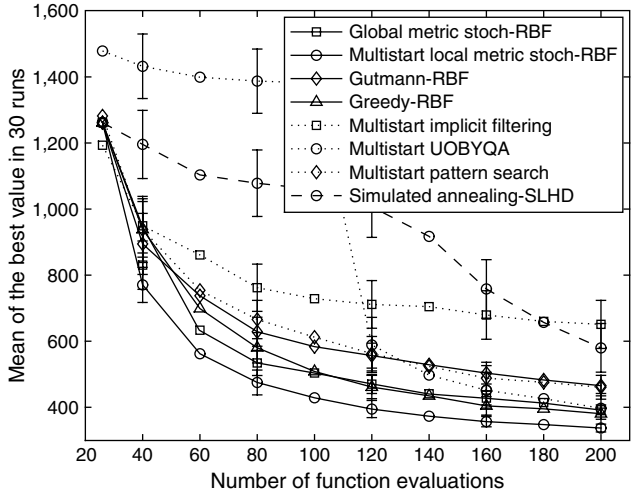

Figure 1 **Comparison of Global Optimization Methods on the GWB12 Function** ($d = 12$)

of function evaluations as our performance measure for comparing the different algorithms. For expensive functions, this is a suitable performance measure since the overall running time of an algorithm is expected to be dominated by total time spent on function evaluations. Hence, when we say that an algorithm is the best or is better than another, it should be understood that we are using the above performance measure.

The results for the two multistart approaches were quite similar for most of the problems. Hence, for clarity in the presentation, we present only the results for the MLSL approach. In the case of simulated annealing (SA), the implementation that uses an SLHD was at least as good as, and sometimes better than, the one that uses uniform random points on most of the test problems. Again, for clarity, we present only the results for SA initialized by an SLHD.

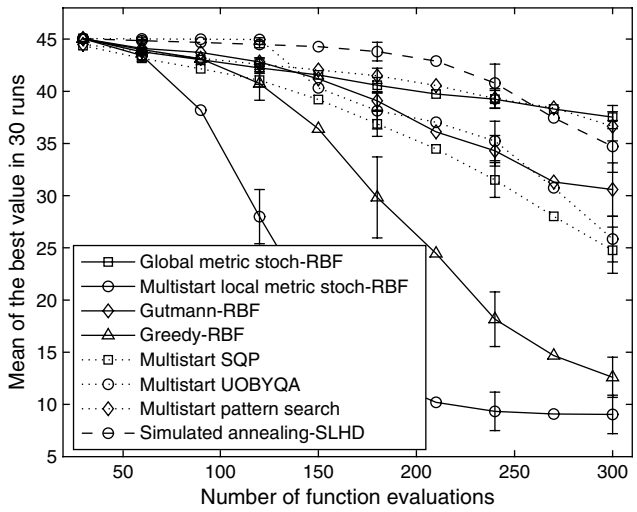

Figure 2 **Comparison of Global Optimization Methods on the Schoen14.100 Function** ($d = 14$)

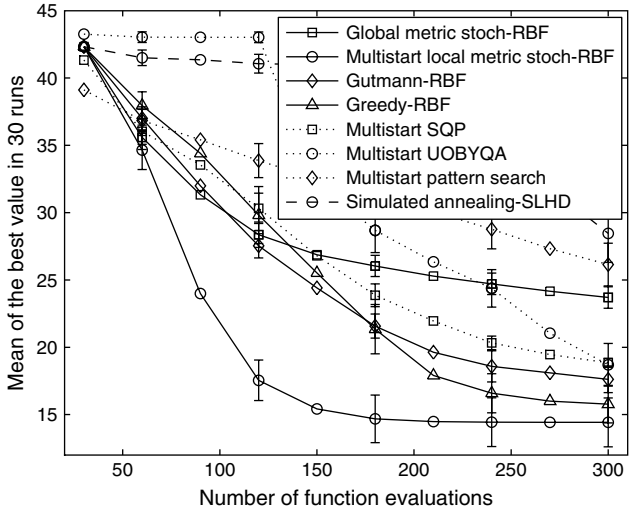

Figure 3 **Comparison of Global Optimization Methods on the Schoen14.20 Function** ($d = 14$)

7.2. Performance of the RBF Methods

The results show that ML-MSRBF is the best among the different global optimization methods we considered on 7 out of 8 test problems with dimension $d \geq 8$, including the groundwater bioremediation problem. Moreover, it is better than, or at least as good as, the other algorithms on the 10 remaining test problems, except on Goldstein-Price and Schoen5.20. In particular, ML-MSRBF is much better than the MLSL multistart local optimization methods on all problems with dimension $d \geq 8$. It is also much better than SA on all test problems. When compared to other RBF methods, ML-MSRBF is better than Greedy-RBF on all test problems with dimension $d \geq 8$ and it is much better than the more sophisticated Gutmann-RBF on all test problems with $d \geq 4$, except on Schoen5.20. Finally, it is also much better than G-MSRBF on all

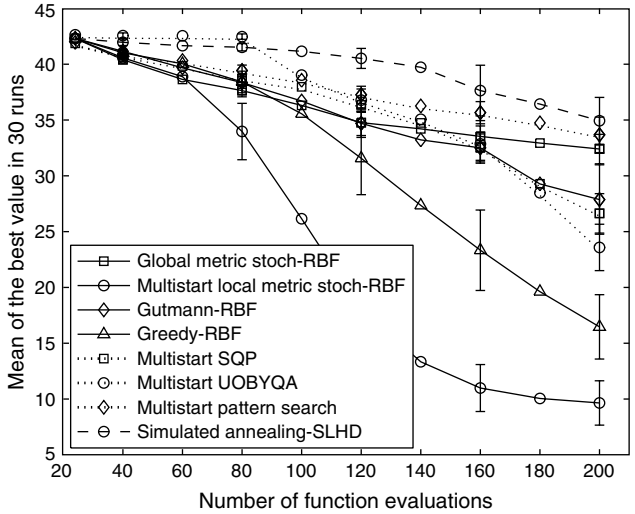

Figure 4 **Comparison of Global Optimization Methods on the Schoen11.100 Function** ($d = 11$)

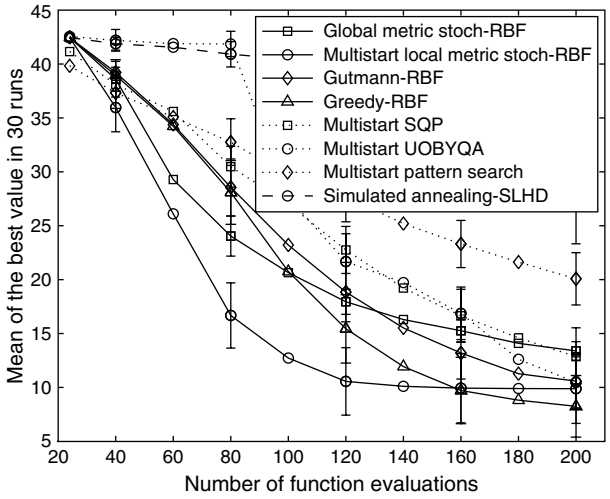

Figure 5 **Comparison of Global Optimization Methods on the Schoen11.20 Function** ($d = 11$)

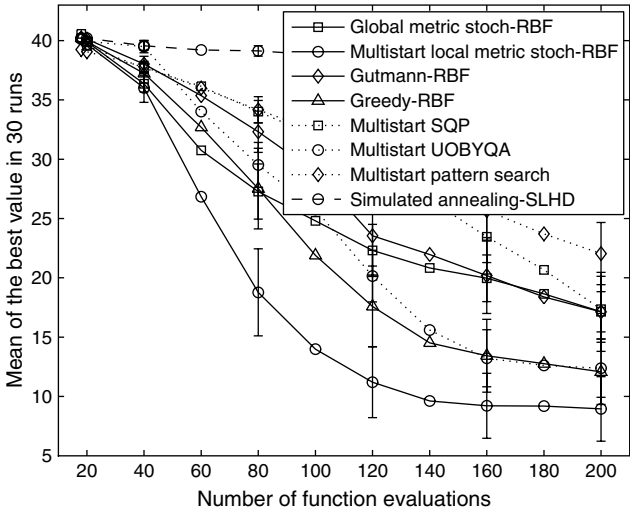

Figure 7 **Comparison of Global Optimization Methods on the Schoen8.100 Function** ($d = 8$)

test problems with $d \geq 4$, except on Schoen5.20, Hartman6, and Griewank10, where the two MSRBF algorithms are comparable.

The results also show that G-MSRBF performed relatively well compared to the other alternatives on most of the test problems with dimensions $d \leq 10$ and also on the groundwater problem. Except for a few cases, G-MSRBF is better than, or at least as good as, Multistart SQP and Gutmann-RBF on the test problems with $d \leq 10$. Moreover, G-MSRBF is much better than Multistart PS on all test problems, except on Schoen3.20, Schoen11.100, and Schoen14.100, where these two algorithms are comparable. It is also better than SA on all problems, except on Schoen14.100. Finally, on GWB12, G-MSRBF is better than Gutmann-RBF and the MLSL multistart methods, and it is comparable to Greedy-RBF.

The results indicate that ML-MSRBF is much better than G-MSRBF and Gutmann-RBF on almost all of the higher dimensional test problems. While one might expect that the algorithms with better balance between local and global search to be superior, our results suggest otherwise. Recall that in these RBF algorithms, global search is realized at the beginning, when the experimental design points are evaluated and the best one is selected as the starting point. After the evaluation of the design points, ML-MSRBF focuses mainly on local search until no further improvement is possible and then it performs global search again in the form of a restart. On the other hand, G-MSRBF and also Gutmann-RBF always maintain a good balance between local and global search. The results suggest that we should put more emphasis on local search when dealing with a very limited

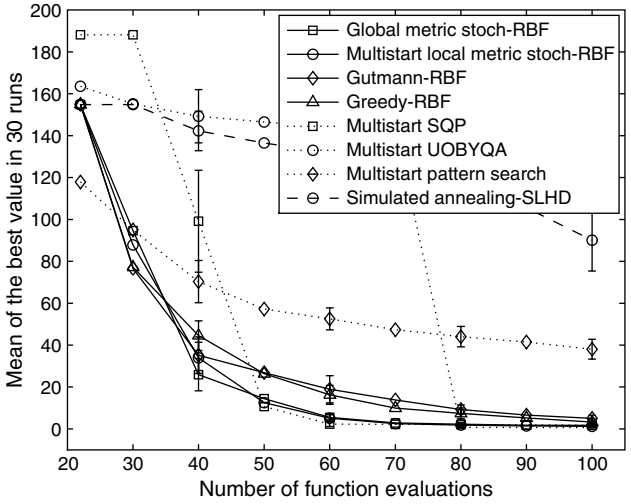

Figure 6 **Comparison of Global Optimization Methods on the Griewank10 Function** ($d = 10$)

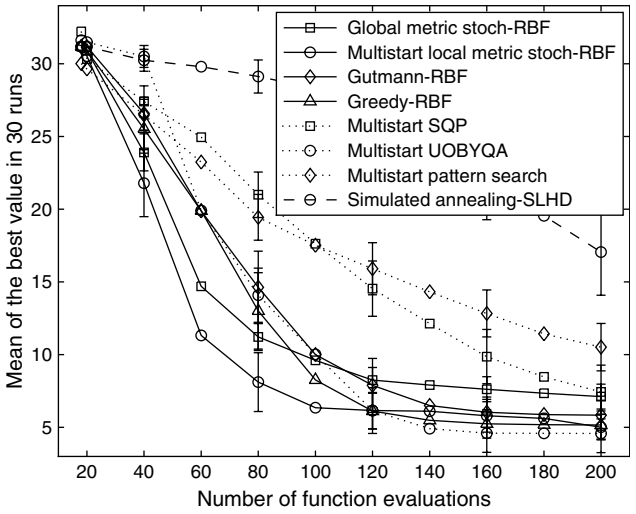

Figure 8 **Comparison of Global Optimization Methods on the Schoen8.20 Function** ($d = 8$)

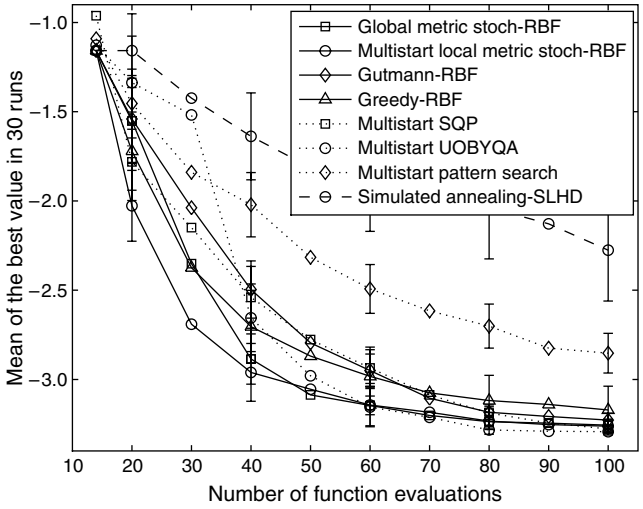

Figure 9 **Comparison of Global Optimization Methods on the Hartman6 Function** ($d = 6$)

number of function evaluations on higher dimensional problems. Because of the curse of dimensionality, global search cannot effectively search the space with a very limited number of function evaluations when the dimension is high. Hence, it appears that local search has a better chance of finding an improving solution in every iteration than global search for higher dimensional problems.

Greedy-RBF performed relatively well on the test problems. It is generally worse than ML-MSRBF but it is better than, or at least as good as, all other algorithms on most problems, including the groundwater application. In particular, it is much better than SA and it is also better than the MLSL multistart methods on the higher dimensional test problems. A possible reason for the good performance of Greedy-RBF is that it adopts a similar strategy to the one used by Local MSRBF in generating its function evaluation

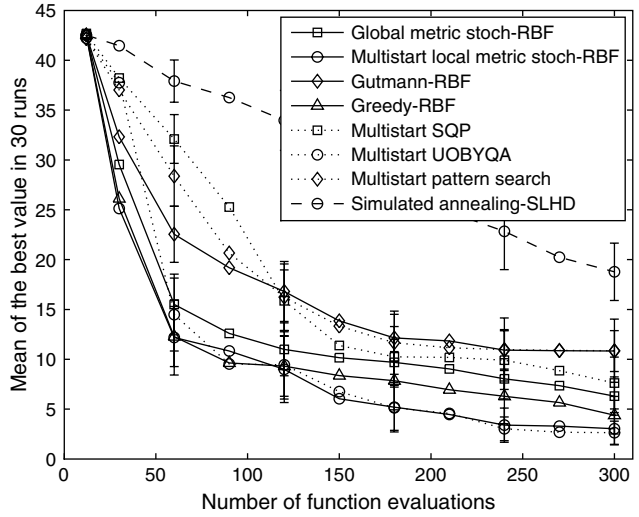

Figure 10 **Comparison of Global Optimization Methods on the Schoen5.100 Function** ($d = 5$)

point when all the local minimizers of the RBF model are previously evaluated points (see Section A.1 in the Online Supplement).

7.3. Performance of the Multistart Algorithms and Simulated Annealing

ML-MSRBF is superior to Multistart SQP (or Implicit Filtering) on the higher dimensional problems probably because finite differencing consumes d extra function evaluations to estimate the gradient, which is then used to determine the actual iterate of the algorithm. On the other hand, ML-MSRBF does not need these extra function evaluations and simply relies on previously evaluated points to select its iterate. In general, RBF methods indirectly use an estimate of the gradient of the underlying function via the RBF model when selecting its evaluation point, making them superior to, or at least competitive with, derivative-based methods on some smooth (i.e., continuously differentiable) test problems.

Multistart UOBYQA performed well compared to the other algorithms on the lower dimensional test problems. In particular, it is generally better than Multistart SQP on the test problems with dimension $d \leq 8$. However, since it builds a quadratic interpolation model in every iteration, it requires $(d+1)(d+2)/2$ points to initialize the method. Hence, its performance deteriorates as the dimension of the problem increases.

Among the global optimization methods considered, SA and Multistart PS are the worst on most of the test problems. However, this does not necessarily generalize to other optimization problems according to the No Free Lunch (NFL) theorems (Wolpert and Macready 1997). Our results are consistent with the NFL theorems because all our test problems, including the groundwater problem, possess some trends that can be approximated well locally by RBFs or quadratic models. Moreover, all the algorithms we used, except SA and Multistart PS, use some form of prior information that detects these trends on the unknown objective function and uses it to guide the search. For instance, all RBF algorithms use previously evaluated points to build an RBF model that guides the selection of new points. SQP or implicit filtering uses gradient information, which is obtained by finite differencing, to find their next iterate. UOBYQA uses a local quadratic interpolation model to guide the selection of the next iterate. On the other hand, SA and PS are somewhat blind since the choice of the next function evaluation point depends only on the current solution and ignores previously evaluated points. Multistart PS and SA are good when function evaluations are cheap since they do not incur any overhead computations to determine the next function evaluation point. However, in the context

of expensive functions, combining these algorithms with a surrogate model might be helpful in improving performance.

7.4. Running Times

Finally, we discuss the average running times of the different algorithms. Among the RBF algorithms, ML-MSRBF has the fastest running times on all the test problems. SA has the fastest running times overall but its performance is also generally the worst. The MLSL multistart algorithms have much faster running times than ML-MSRBF. However, these differences in running times become less important as the time for the function evaluation increases. For example, if the time to evaluate the Schoen14.100 function is actually 5 minutes, then from Table 2 of the Appendix in the Online Supplement, the average running time of Multistart SQP after 300 evaluations would be 25 hours while the average running time of ML-MSRBF would be 25.4 hours. However, from Figure 2, the average performance of ML-MSRBF is much better than Multistart SQP after 300 function evaluations.

8. Summary and Conclusions

We have introduced the Stochastic Response Surface (SRS) Method, which is a new framework for stochastic global optimization of expensive functions using response surface models. Assuming that a continuous function has a unique global minimizer over its feasible region and that the random candidate points are generated by probability distributions that satisfy the technical conditions in Section 2.2, then SRS converges almost surely to the global minimizer. The SRS method is easy to implement and it extends to nonlinearly constrained problems. Finally, SRS can be used with any response surface model such as RBFs, kriging, regression splines, or neural networks.

We have also introduced Metric SRS (MSRS), which is a special case of SRS that utilizes a distance criterion when selecting the function evaluation points. We developed a global optimization and a multistart local optimization version of MSRS. In the numerical experiments, we used an RBF model for MSRS, giving rise to G-MSRBF and ML-MSRBF, which we compared to six alternative algorithms representing a range of types of global optimization methods: multistart local optimization (derivative-based, derivative-free trust-region, pattern search), simulated annealing, and two other RBF methods (Greedy-RBF and Gutmann-RBF). The RBF methods were all initialized by symmetric Latin hypercube designs and they all used a thin-plate spline RBF model augmented by a linear polynomial. The eight algorithms were compared on 17 multimodal test functions, with dimensions ranging from 2 to 14, and on a 12-dimensional groundwater bioremediation problem.

The computational results indicate that ML-MSRBF is the best in terms of the average best function value obtained after a fixed number of function evaluations among the eight global optimization algorithms on 7 out of 8 test problems with dimension $d \geq 8$, including the groundwater application. It is also better than, or at least comparable to, the other algorithms on 8 of the 10 remaining test problems. G-MSRBF is competitive with the other alternatives on the test problems with dimensions $d \leq 10$ and also on the groundwater problem. However, ML-MSRBF is generally much better than G-MSRBF on most test problems. This suggests that more emphasis on local search is important when dealing with a very limited number of function evaluations on higher dimensional problems.

The superior performance of an RBF method over a multistart derivative-based method on a substantial collection of smooth nonconvex test problems is significant since it is widely believed that the optimization of smooth functions is best carried out by means of gradient-based methods. Our results suggest that ML-MSRBF, which possesses implicit gradient estimates via the RBF model, is a competitive alternative for gradient-based methods that utilize finite differencing. ML-MSRBF is better than the multistart derivative-based methods partly because it does not require extra function evaluations to determine gradients.

Finally, the successful performance of ML-MSRBF and G-MSRBF on the test problems as compared to alternative optimization methods, especially on the 12-dimensional groundwater application, indicates that SRS is a promising approach and demonstrates the potential of the method for computationally expensive real-world optimization problems.

Acknowledgments

This work has been supported by the CISE Directorate in NSF, Grant ACI-0305583 to the second author. The groundwater application was supported by the Engineering Directorate in NSF, Grant BES-0229176. The authors thank the Intelligent Information Systems Institute (IISI), directed by Carla Gomes, for providing GRA funding for the first author (AFOSR, Grant F49620-01-1-0076) during the early stages of this project. The authors are grateful to Shane Henderson of the School of Operations Research and Industrial Engineering (ORIE) at Cornell for suggesting that they prove convergence of the algorithm by using Spall's theorem on random search. They also thank David Ruppert of the School of ORIE at Cornell for providing them with some comments on the theoretical part of this paper. Bryan Tolson and Pradeep Mugunthan from the School of Civil and Environmental Engineering provided the authors with some feedback on the performance of the proposed algorithms on their real-world environmental problems. John Zollweg of the Cornell Theory Center provided some advice on how to write fast and efficient Matlab codes. Raju Rohde and Jae-Heung Yoon provided the Fortran simulation code for the groundwater bioremediation problem. Stefan Wild

provided the authors with a reference to the Condor software package, which implements UOBYQA. Finally, the authors thank the anonymous reviewers for their helpful comments and suggestions.

References

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

Booker, A. J., J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, M. W. Trosset. 1999. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optim.* **17** 1–13.

Box, G. E. P., N. R. Draper. 1987. *Empirical Model-Building and Response Surfaces*. John Wiley and Sons, New York.

Buhmann, M. D. 2003. *Radial Basis Functions*. Cambridge University Press, Cambridge, UK.

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

Cressie, N. 1993. *Statistics for Spatial Data*. John Wiley and Sons, New York.

Dennis, J. E., V. Torczon. 1991. Direct search methods on parallel machines. *SIAM J. Optim.* **1** 448–474.

Dixon, L. C. W., G. Szegö. 1978. The global optimization problem: An introduction. L. C. W. Dixon, G. Szegö, eds. *Towards Global Optimization 2*. North-Holland, Amsterdam, The Netherlands, 1–15.

Friedman, J. 1991. Multivariate adaptive regression splines (with discussion). *Ann. Statist.* **19** 1–141.

Gilmore, P., C. T. Kelley. 1995. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM J. Optim.* **5** 269–285.

Glover, F. 1998. A template for scatter search and path relinking. J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers, eds. *Artificial Evolution, Lecture Notes in Computer Science*, Vol. 1363. Springer Verlag, Berlin, Germany, 13–54.

Griewank, A. O. 1981. Generalized descent for global optimization. *J. Optim. Theory Appl.* **34** 11–39.

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

Horst, R., P. M. Pardalos, N. V. Thoai. 1995. *Introduction to Global Optimization*. Kluwer, Dordrecht, The Netherlands.

Ishikawa, T., Y. Tsukui, M. Matsunami. 1999. A combined method for the global optimization using radial basis function and deterministic approach. *IEEE Trans. Magnetics* **35** 1730–1733.

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

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

Kelley, C. T. 1999. *Iterative Methods for Optimization*. SIAM, Philadelphia, PA.

Koehler, J. R., A. B. Owen. 1996. Computer experiments. S. Ghosh, C. R. Rao, eds. *Handbook of Statistics, 13: Design and Analysis of Computer Experiments*. North-Holland, Amsterdam, The Netherlands, 261–308.

Laguna, M., R. Marti. 2002. The OptQuest callable library. S. Voss, D. Woodruff, eds. *Optimization Software Class Libraries*. Kluwer Academic Publishers, Boston, MA, 193–218.

Laguna, M., R. Marti. 2003. *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers, Boston, MA.

Moore, A., J. Schneider. 1996. Memory-based stochastic optimization. *Neural Inform. Processing Systems* **8** 1066–1072.

Moore, A., J. Schneider, J. Boyan, M. S. Lee. 1998. Q2: Memory-based active learning for optimizing noisy continuous functions. J. Shavlik, ed. *Proc. Fifteenth Internat. Conf. Machine Learn.* Morgan Kaufmann, San Francisco, CA, 386–394.

Morris, M. D., T. J. Mitchell. 1995. Exploratory designs for computational experiments. *J. Statist. Planning Inference* **43** 381–402.

Myers, R. H., D. C. Montgomery. 1995. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, New York.

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

Nocedal, J., S. J. Wright. 1999. *Numerical Optimization*. Springer, New York.

Powell, M. J. D. 1992. The theory of radial basis function approximation in 1990. W. Light, ed. *Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions*. Oxford University Press, Oxford, UK, 105–210.

Powell, M. J. D. 1999. Recent research at Cambridge on radial basis functions. M. Müller, M. Buhmann, D. Mache, M. Felten, eds. *New Developments in Approximation Theory, International Series of Numerical Mathematics*, Vol. 132. Birkhäuser Verlag, Basel, Switzerland, 215–232.

Powell, M. J. D. 2002. UOBYQA: Unconstrained optimization by quadratic approximation. *Math. Programming* **92** 555–582.

Powell, M. J. D. 2003. On trust region methods for unconstrained minimization without derivatives. *Math. Programming* **97** 605–623.

Rinnooy Kan, A. H. G., G. T. Timmer. 1987. Stochastic global optimization methods, Part II: Multi level methods. *Math. Programming* **39** 57–78.

Sacks, J., W. J. Welch, T. J. Mitchell, H. P. Wynn. 1989. Design and analysis of computer experiments. *Statist. Sci.* **4** 409–435.

Schoen, F. 1993. A wide class of test functions for global optimization. *J. Global Optim.* **3** 133–137.

Simpson, T. W., T. M. Mauery, J. J. Korte, F. Mistree. 1998. Comparison of response surface and kriging models for multidisciplinary design optimization. *Proc. 7th AIAA/USAF/NASA/ISSMO Sympos. Multidisciplinary Anal. Optim.*, Vol. 1. St. Louis, MO, 381–391.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization*. John Wiley and Sons, Hoboken, NJ.

The MathWorks. 2004a. *Genetic Algorithm and Direct Search Toolbox for Use with MATLAB: User's Guide, Version 1*. Natick, MA.

The MathWorks. 2004b. *Optimization Toolbox for Use with MATLAB: User's Guide, Version 3*. Natick, MA.

Torczon, V. 1997. On the convergence of pattern search algorithms. *SIAM J. Optim.* **7** 1–25.

Torn, A., A. Zilinskas. 1989. *Global Optimization, Lecture Notes in Computer Science*, Vol. 350. Springer-Verlag, Berlin, Germany.

Ugray, Z., L. Lasdon, J. Plummer, F. Glover, J. Kelley, R. Marti. 2006. Scatter search and local NLP solvers: A multistart framework for global optimization. *INFORMS J. Comput.* Forthcoming.

Vanden Berghen, F., H. Bersini. 2005. CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm. *J. Comput. Appl. Math.* **181** 157–175.

Vavasis, S. A. 1991. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York.

Wolpert, D. H., W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Comput.* **1** 67–82.

Ye, K. Q., W. Li, A. Sudjianto. 2000. Algorithmic construction of optimal symmetric latin hypercube designs. *J. Statist. Planning Inference* **90** 145–159.

Yoon, J.-H., C. A. Shoemaker. 1999. Comparison of optimization methods for ground-water bioremediation. *J. Water Resources Planning Management* **125** 54–63.