Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization

Rommel G. Regis [a] & Christine A. Shoemaker [b]

[a] Department of Mathematics, Saint Joseph's University, Philadelphia, PA, 19131, USA

[b] School of Civil & Environmental Engineering and Center for Applied Mathematics, Cornell University, Ithaca, NY, 14853, USA

Published online: 26 Jun 2012.

To cite this article: Rommel G. Regis & Christine A. Shoemaker (2013) Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization, Engineering Optimization, 45:5, 529-555, DOI: 10.1080/0305215X.2012.687731

To link to this article: http://dx.doi.org/10.1080/0305215X.2012.687731

Engineering Optimization, 2013
Vol. 45, No. 5, 529–555, http://dx.doi.org/10.1080/0305215X.2012.687731

Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization

Rommel G. Regis[a]* and Christine A. Shoemaker[b]

[a]*Department of Mathematics, Saint Joseph's University, Philadelphia, PA 19131, USA;* [b]*School of Civil & Environmental Engineering and Center for Applied Mathematics, Cornell University, Ithaca, NY 14853, USA*

(*Received 30 June 2011; final version received 2 April 2012*)

This article presents the DYCORS (DYnamic COordinate search using Response Surface models) framework for surrogate-based optimization of HEB (High-dimensional, Expensive, and Black-box) functions that incorporates an idea from the DDS (Dynamically Dimensioned Search) algorithm. The iterate is selected from random trial solutions obtained by perturbing only a subset of the coordinates of the current best solution. Moreover, the probability of perturbing a coordinate decreases as the algorithm reaches the computational budget. Two DYCORS algorithms that use RBF (Radial Basis Function) surrogates are developed: DYCORS-LMSRBF is a modification of the LMSRBF algorithm while DYCORS-DDSRBF is an RBF-assisted DDS. Numerical results on a 14-D watershed calibration problem and on eleven 30-D and 200-D test problems show that DYCORS algorithms are generally better than EGO, DDS, LMSRBF, MADS with kriging, SQP, an RBF-assisted evolution strategy, and a genetic algorithm. Hence, DYCORS is a promising approach for watershed calibration and for HEB optimization.

Keywords: expensive black-box optimization; high-dimensional optimization; radial basis functions; coordinate search; watershed calibration

1. Introduction

1.1. *Motivation and problem statement*

Many real-world optimization problems involve high-dimensional black-box functions that are outputs of computationally expensive simulations. Shan and Wang (2010) used the acronym HEB (High-dimensional, Expensive, and Black-box) to refer to these optimization problems. Unfortunately, finding the global optimum of an HEB problem is unrealistic because this generally requires a large number of function evaluations especially when there are multiple local minima. Hence, the goal of this article is simply to develop new optimization approaches for HEB problems that can make quick progress given a very limited computational budget.

More precisely, let $\mathcal{D} = [a, b]$ be a closed hypercube in $\mathbb{R}^d$ and let $f : \mathcal{D} \to \mathbb{R}$ be a deterministic black-box objective function. Note that if $\mathcal{D}$ is a box in $\mathbb{R}^d$, then it can be scaled into the unit hypercube $[0, 1]^d$. Ideally, one hopes to obtain a global minimum point for f on $\mathcal{D}$ using only

*Corresponding author. Email: rregis@sju.edu

a relatively small number of function evaluations. However, it usually takes a large number of function evaluations to guarantee that the solution obtained is even approximately optimal. Hence, most practitioners are satisfied with simply getting a reasonably good solution given a severely limited computational budget. Many global optimization methods have been developed (*e.g.*, Törn and Žilinskas 1989, Pintér 1996, Horst *et al.* 2000), including surrogate-based methods for expensive black-box functions. However, many of these surrogate-based methods are only suitable for low-dimensional problems because finding good iterates also require substantial computational effort that increases with the dimension. In this article, the problem dimension d is assumed to be relatively high ($d > 10$). Also, the gradient ∇f is assumed to be unavailable, which is the case in many practical applications.

1.2. *Related work*

One approach for finding the global minimum of f over $\mathcal{D} = [a, b]$ is to run a local optimization algorithm with a multistart procedure. If f is smooth and not riddled with local minima, and ∇f can be obtained relatively cheaply, then it is usually best to run a gradient-based local solver from multiple starting points. When ∇f is not provided, it may be obtained by automatic differentiation or finite-differencing. However, automatic differentiation is not always applicable and finite-differencing may be unreliable when f is non-smooth. Hence, many practitioners use *derivative-free* (local) optimization methods such as Pattern Search (Torczon 1997), Mesh Adaptive Direct Search (MADS) (Abramson and Audet 2006, Audet and Dennis 2006), and derivative-free trust-region methods (Powell 2006, Conn *et al.* 2009). Another approach for global optimization is to use derivative-free heuristic methods such as simulated annealing, evolutionary algorithms (*e.g.*, Genetic Algorithms (GAs), Evolution Strategies and Evolutionary Programming), scatter search (Glover 1998, Laguna and Marti 2003), and particle swarm optimization.

A natural approach for the derivative-free optimization of expensive black-box functions is to use *response surface models* (also known as *surrogate models* or *metamodels*) for the expensive function. Examples of response surface models include polynomials, which are used in traditional response surface methodology (Myers and Montgomery 1995), radial basis functions (RBFs) (Powell 1992, Buhmann 2003), kriging models (Sacks *et al.* 1989, Cressie 1993) and neural networks.

Surrogate-based optimization methods were proposed a long time ago (*e.g.*, Kushner 1962). However, they became popular only relatively recently. Below are some of these methods, but this list is not meant to be exhaustive. For example, Giunta *et al.* (1997) and Simpson *et al.* (2001) used polynomial and kriging models for aerospace design. Wang *et al.* (2001) developed the Adaptive Response Surface Method, which uses quadratic models for the expensive function. Derivative-free trust-region methods for unconstrained optimization (*e.g.*, Powell 2006, Conn *et al.* 2009) typically use local quadratic interpolation models of the objective function. Wang and Simpson (2004) used fuzzy clustering and a hierarchical metamodelling strategy involving quadratic or kriging models. Yahyaie and Filizadeh (2011) used an adaptive mesh to identify potential locations of local minima and then used local quadratic and Gaussian surrogates in these regions. Jones *et al.* (1998) developed the kriging-based Efficient Global Optimization (EGO) method where the next iterate is obtained by maximizing an expected improvement function. Sasena *et al.* (2002) explored various infill sampling criteria for EGO and applied them to constrained optimization problems, including the design of a hybrid electric car. Huang *et al.* (2006) developed a variant of EGO for stochastic black-box systems and applied it to an inventory problem. Aleman *et al.* (2009) also used a variant of EGO to optimize beam orientation in IMRT treatment planning. Viana *et al.* (2010) developed an extension of EGO that uses multiple surrogates to generate multiple function evaluation points per cycle. Villemonteix *et al.* (2009) also developed a kriging-based method that

uses a minimizer entropy criterion. Kriging has been used with pattern search (Booker *et al.* 1999, Marsden *et al.* 2004) and also with scatter search (Egea *et al.* 2009).

RBFs have also been used in expensive black-box optimization. For example, Gutmann (2001) developed an RBF method where the next iterate is obtained by minimizing a bumpiness function and showed that this method is closely related to the P-algorithm by Žilinskas (1985). Variants of this RBF method have been developed by Björkman and Holmström (2000), Regis and Shoemaker (2007b), and Holmström (2008). Jakobsson *et al.* (2010) also used RBFs for expensive, noisy and multiobjective black-box optimization. Won and Ray (2005) provided a framework for a surrogate-assisted GA and used Gaussian RBFs, kriging and co-kriging surrogates. Moreover, Regis and Shoemaker (2004) used RBFs to assist an evolution strategy.

Previous surrogate-based optimization methods have mostly been tested on low-dimensional problems with $d < 15$ (*e.g.*, Gutmann 2001, Wang *et al.* 2001, Sasena *et al.* 2002, Regis and Shoemaker 2004, Wang and Simpson 2004, Won and Ray 2005, Huang *et al.* 2006, Regis and Shoemaker 2007a,b, Holmström 2008, Aleman *et al.* 2009, Egea *et al.* 2009, Viana *et al.* 2010, Yahyaie and Filizadeh 2011). This is because given a limited computational budget, one cannot hope to find the global minimum of a high-dimensional problem unless one can take advantage of some special problem structure. Relatively few surrogate-based methods have been developed for HEB problems. For example, Regis (2011) developed ConstrLMSRBF and successfully applied it to a 124-D automotive problem. Another approach for HEB problems is to use dimension reduction techniques such as sequential bifurcation (Bettonvil and Kleijnen 1997) with a surrogate-based method. Shan and Wang (2010) provided a survey of other approaches for HEB problems.

1.3. *Main contributions*

This article develops the DYCORS (DYnamic COordinate search using Response Surface models) framework for bound constrained HEB optimization. DYCORS builds and maintains a surrogate model of the objective function in each iteration and uses a dynamic coordinate search strategy for generating trial solutions that is similar to the one used in the DDS algorithm (Tolson and Shoemaker 2007). The iterate is selected from a set of random trial solutions obtained by perturbing only a subset of the coordinates of the current best solution as was done in ConstrLMSRBF-BCS (Regis 2011). However, ConstrLMSRBF-BCS uses a fixed probability of perturbing a coordinate of the current best solution while DYCORS uses decreasing perturbation probabilities. As a result, the number of coordinates perturbed tends to decrease as the algorithm progresses. Moreover, DYCORS is a more general method than ConstrLMSRBF-BCS in that it allows other ways of selecting the iterate from the set of trial solutions and it also allows other schemes for adjusting the step size. Two DYCORS algorithms that use RBF surrogates are developed: DYCORS-LMSRBF is a modification of the LMSRBF algorithm (Regis and Shoemaker 2007a) while DYCORS-DDSRBF is an RBF-assisted modification of DDS.

DYCORS-LMSRBF and DYCORS-DDSRBF are compared with seven alternative optimization methods on 12 problems: a 14-D model calibration problem of the Town Brook watershed in upstate New York (Shoemaker *et al.* 2007, Tolson and Shoemaker 2007) and eleven 30-D and 200-D test problems with large numbers of local minima. Few surrogate-based methods have been tested on problems with at least 200 decision variables so this article helps push the frontier in surrogate-based optimization. The alternative methods include LMSRBF, DDS, EGO, a MADS algorithm that uses kriging surrogates, an RBF-assisted evolution strategy (ESGRBF) (Regis and Shoemaker 2004), a Sequential Quadratic Programming (SQP) algorithm where the derivatives are obtained by finite differencing, and a GA. In addition, the two DYCORS algorithms are compared with a nonlinear least squares algorithm on the watershed problem.

The numerical results indicate that DYCORS-LMSRBF and DYCORS-DDSRBF are both promising for HEB optimization. They are the best algorithms on the Town Brook watershed

calibration problem and they are generally much better than the alternatives on most of the test problems. Previous work (Shoemaker *et al.* 2007, Tolson and Shoemaker 2007) showed that DDS and ESGRBF are the most promising algorithms for the Town Brook problem. The two DYCORS algorithms are both better than DDS, ESGRBF and the other alternatives on this problem. Furthermore, DYCORS-LMSRBF is an improvement over both LMSRBF and DDS on 11 of the 12 problems while DYCORS-DDSRBF is better than or competitive with DDS on 10 of the 12 problems. Hence, DYCORS-LMSRBF is an effective combination of LMSRBF and DDS while DYCORS-DDSRBF is an effective RBF-assisted modification of DDS. In addition, both DYCORS algorithms consistently outperformed an implementation of EGO on all the problems.

2. Response surface models and dynamic coordinate search in expensive black-box optimization

2.1. *General framework*

This section presents a general framework for bound constrained expensive black-box optimization called DYnamic COordinate search using Response Surface models (DYCORS). This method does not require derivatives and is suitable for HEB problems. It is a modification of the Local Metric Stochastic Response Surface (LMSRS) method by Regis and Shoemaker (2007a) that incorporates an idea from the DDS method by Tolson and Shoemaker (2007). Two algorithms that follow the DYCORS framework are developed, namely DYCORS-LMSRBF and DYCORS-DDSRBF. The former is a modification of the LMSRBF algorithm (Regis and Shoemaker 2007a) that incorporates the DDS strategy while the latter is an RBF-assisted modification of DDS.

DDS is a simple, but efficient, heuristic method for box-constrained optimization that scales the search for good global solutions to the user-specified maximum number of function evaluations. It is based on the idea that not all decision variables have the same effect on the objective function. In each iteration, a trial solution is generated by adding random perturbations to *some* or all of the coordinates of the current best solution x_{best}. The set of coordinates perturbed is determined probabilistically and the random perturbations on the coordinates are normally distributed with mean zero and some fixed standard deviation. DDS starts with global search that becomes more local as the number of function evaluations approaches the computational budget. At the beginning, all coordinates of the current x_{best} are perturbed when generating the trial solution. As the number of function evaluations increases, the probability of perturbing a given coordinate decreases, and so the coordinates of x_{best} that are perturbed tend to become fewer. This idea was motivated by the manual calibration of watershed models where early in the search perturbing all coordinates is helpful in improving an initially poor solution, but as the solution improves it becomes necessary to perturb only a few coordinates simultaneously so that the current improvement is not lost.

DYCORS follows the same general structure as many other global optimization methods that are based on response surface models. It begins by evaluating the expensive objective function at the points of a space-filling experimental design. Then, it fits a response surface model at the beginning of each iteration using all points where the function values are known, and uses this model to select the next iterate. After obtaining the function value at the selected point, the process iterates.

In both LMSRS and DYCORS, the iterate is obtained from a set of random trial points called *candidate points* in Regis and Shoemaker (2007a). However, DYCORS differs from LMSRS in how these trial points are generated. In LMSRS, a trial point is generated by applying normally distributed random perturbations on *all* coordinates of the current x_{best}. In DYCORS, a trial point is also generated by applying normally distributed perturbations but only on *some* of the coordinates of x_{best}. Moreover, as in DDS, the coordinates that are perturbed in DYCORS are

randomly selected and the number of coordinates perturbed tends to become fewer as the number of function evaluations gets closer to the computational budget. Another difference between LMSRS and DYCORS is that the former uses a response surface criterion and a distance criterion to select the iterate from the set of trial solutions while the latter allows other selection criteria.

In the framework below, n_0 is the number of space-filling design points, n is the number of previously evaluated points, $\mathcal{A}_n = \{x_1, \ldots, x_n\}$ is the set of previously evaluated points, and $s_n(x)$ is the response surface model built using the points in $\mathcal{A}_n$.

DYnamic COordinate search using Response Surface models (DYCORS)

Inputs:

(1) A real-valued black-box function f defined on a closed hypercube $\mathcal{D} = [a, b] \subseteq \mathbb{R}^d$.
(2) The maximum number of function evaluations allowed, denoted by $N_{\max}$.
(3) A strictly decreasing function $\varphi(n)$ defined for all positive integers $n_0 \leq n \leq N_{\max} - 1$ whose values are in $[0, 1]$.
(4) The initial step size σ_{init} and the minimum step size $\sigma_{\min}$.
(5) The number of trial points in each iteration, denoted by k.
(6) A response surface model.
(7) Initial points $\mathcal{I} = \{x_1, \ldots, x_{n_0}\} \subseteq \mathcal{D}$ from a space-filling experimental design.
(8) The tolerance for the number of consecutive failed iterations $\mathcal{T}_{\text{fail}}$ and the threshold for the number of consecutive successful iterations $\mathcal{T}_{\text{success}}$ (optional).

Output: The best point encountered by the algorithm.

Step 1 – Evaluate initial points. Compute $f(x_i)$ for each $i = 1, \ldots, n_0$. Let x_{best} be the best point found so far and let $f_{\text{best}} := f(x_{\text{best}})$. Set $n := n_0$ and $\mathcal{A}_n := \mathcal{I}$.
Step 2 – Initialize step size and counters. Set $\sigma_n := \sigma_{\text{init}}$, $C_{\text{fail}} := 0$ and $C_{\text{success}} := 0$.
Step 3. While the termination condition is not satisfied (*e.g.*, while $n < N_{\max}$) do

Step 3.1 – Fit/update response surface model. Fit or update a response surface model $s_n(x)$ using the data points $\mathcal{B}_n = \{(x, f(x)) : x \in \mathcal{A}_n\} = \{(x_i, f(x_i)) : i = 1, \ldots, n\}$.
Step 3.2 – Determine probability of perturbing a coordinate. Calculate $p_{\text{select}} = \varphi(n)$.
Step 3.3 – Generate multiple trial points. Generate $\Omega_n := \{y_{n,1}, \ldots, y_{n,k}\}$ as follows. For $j = 1, \ldots, k$, obtain $y_{n,j}$ by performing the following steps.
(a) Select coordinates to perturb. Generate d uniform random numbers $\omega_1, \ldots, \omega_d$ in [0,1]. Let $I_{\text{perturb}} := \{i : \omega_i < p_{\text{select}}\}$. If $I_{\text{perturb}} = \emptyset$, then select j uniformly at random from $\{1, \ldots, d\}$ and set $I_{\text{perturb}} = \{j\}$.
(b) Generate trial point. Generate $y_{n,j}$ by $y_{n,j} = x_{\text{best}} + z$, where $z^{(i)} = 0$ for all $i \notin I_{\text{perturb}}$ and $z^{(i)}$ is a realization of a normal random variable with mean 0 and standard deviation σ_n for all $i \in I_{\text{perturb}}$.
(c) Ensure trial point is in domain. If $y_{n,j} \notin \mathcal{D}$, then replace it by a point in $\mathcal{D}$ obtained by performing successive reflection of $y_{n,j}$ about the closest point on the boundary of $\mathcal{D}$.
End.
Step 3.4 – Select next iterate. $x_{n+1} :=$ **Select_Evaluation_Point**$(\Omega_n, \mathcal{B}_n, s_n(x))$.
Step 3.5 – Perform function evaluation. Compute $f(x_{n+1})$.
Step 3.6 – Update counters. If $f(x_{n+1}) < f_{\text{best}}$, then reset $C_{\text{success}} := C_{\text{success}} + 1$ and $C_{\text{fail}} := 0$. Otherwise, reset $C_{\text{fail}} := C_{\text{fail}} + 1$ and $C_{\text{success}} := 0$.
Step 3.7 – Adjust step size.
$[\sigma_{n+1}, C_{\text{success}}, C_{\text{fail}}] =$ **Adjust_Step_Size**$(\sigma_n, C_{\text{success}}, \mathcal{T}_{\text{success}}, C_{\text{fail}}, \mathcal{T}_{\text{fail}})$.

Step 3.8 – Update best solution. If $f(x_{n+1}) < f_{\text{best}}$, then $x_{\text{best}} = x_{n+1}$ and $f_{\text{best}} = f(x_{n+1})$.
Step 3.9 – Update information. Set $\mathcal{A}_{n+1} := \mathcal{A}_n \cup \{x_{n+1}\}$, and reset $n := n + 1$.

End.

Step 4 – Return the best solution found. Return x_{best} and f_{best}.

In Step 1 of DYCORS, the objective function f is evaluated at the n_0 points of the space-filling experimental design. In Step 2, the current step size is set to the initial step size and the counters for the number of consecutive failed iterations and the number of consecutive successful iterations are both set to zero. In Step 3, the method goes through the iterations that involve fitting or updating the response surface model, generating random trial points, selecting one of the trial points, and evaluating f at the selected point. Finally, in Step 4, the best solution found is returned. DYCORS differs from LMSRS mainly in the generation of trial points (Steps 3.2–3.3), the selection of the iterate (Step 3.4), and the adjustment of the step size (Step 3.7).

The response surface model in Step 3.1 can be any type of function approximation model such as RBFs (Powell 1992, Buhmann 2003), kriging (Sacks *et al.* 1989, Cressie 1993), or neural networks. In Step 3.2, p_{select} is the probability of perturbing a coordinate of the current x_{best} when generating trial points. Note that, for a given iteration, the number of coordinates perturbed is a random variable that has a Binomial(d, p_{select}) distribution, and so the mean number of coordinates perturbed is $d \times p_{\text{select}}$. The value of p_{select} is given by $\varphi(n)$, which is chosen to be a strictly decreasing function defined for all positive integers $n_0 \leq n \leq N_{\max} - 1$. Since $\varphi(n)$ is a probability, its values are in [0,1]. Also, since $\varphi(n)$ is strictly decreasing, p_{select} decreases as the iterations progress.

In Step 3.3, the set of trial points Ω_n is generated by adding random perturbations to some or all of the coordinates of the current x_{best}. These random perturbations are normally distributed with mean 0 and standard deviation σ_n (called the *step size*). I_{perturb} is the set of coordinates that are perturbed in the current x_{best}. To ensure the diversity of the trial points within a given iteration, I_{perturb} changes every time a new trial point is about to be generated. Moreover, whenever a trial point falls outside of $\mathcal{D}$, it is replaced by a point in $\mathcal{D}$ as described in Step 3.3(c).

The use of a dynamic coordinate search strategy for generating trial points in Step 3.3 is expected to be helpful for problems with many decision variables. By perturbing a progressively smaller fraction of the coordinates, the coordinates of the current x_{best} that are already in good settings are less likely to be disturbed. Moreover, the chances of generating improved iterates are increased because the resulting trial points are closer to the current x_{best}. That is, for a fixed value of σ_n, the trial points obtained by perturbing *all* coordinates of the current x_{best} will tend to be farther from x_{best} than trial points obtained by perturbing only a fraction of the coordinates. If the objective function is continuous and the current x_{best} is not a local minimum, then an improving solution can always be found within a small enough neighbourhood of x_{best}. Furthermore, although only a fraction of the coordinates are perturbed when generating trial points, the dynamic coordinate search strategy actually helps in considering a very diverse set of search directions in each iteration. Each trial point is generated by perturbing the current x_{best} at a few coordinates but the set of coordinates perturbed varies for different trial points.

In Step 3.4, the iterate is selected to be a point in Ω_n using the information from the response surface model and all the previously evaluated points. The DYCORS framework allows any selection procedure. In particular, the two DYCORS algorithms that are developed (described in Sections 2.3 and 2.4) differ in this step.

In Step 3.5, f is evaluated at the selected trial point. When f is computationally expensive, the total running time for this step and Step 1 dominates the running time of the algorithm. In Step 3.6, the progress of the algorithm is monitored by recording the number of *consecutive*

successful iterations (*i.e.* iterations that improved x_{best}), denoted by C_{success}, and also the number of *consecutive* failed iterations, denoted by C_{fail}. In Step 3.7, the step sizes are allowed to be adjusted according to the values of C_{success}, C_{fail}, the optional threshold parameter $\mathcal{T}_{\text{success}}$, and the optional tolerance parameter $\mathcal{T}_{\text{fail}}$. The two DYCORS algorithms also differ in Step 3.7.

By requiring some technical conditions on the algorithm parameters, LMSRS converges to the global minimum in a probabilistic sense if it is allowed to run indefinitely (Regis and Shoemaker 2007a). Likewise, for some suitable choices of the parameters of DYCORS, including the φ function that controls the probability of perturbing a coordinate of the current x_{best}, it might be possible to prove the convergence of the algorithm.

2.2. *Radial basis function interpolation*

The radial basis function (RBF) interpolation model given below, which is described in Powell (1992), is used in the DYCORS algorithms and the other RBF algorithms in this study. This RBF model is equivalent to *dual kriging* (*e.g.*, see Cressie 1993).

Given n distinct points $x_1, \ldots, x_n \in \mathbb{R}^d$ where the function values $f(x_1), \ldots, f(x_n)$ are known, the RBF algorithms in this article use an interpolant of the form

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

where $\|\cdot\|$ is the Euclidean norm, $\lambda_i \in \mathbb{R}$ for $i = 1, \ldots, n$, $p(x)$ is a linear polynomial in d variables, and ϕ has the *cubic* form: $\phi(r) = r^3$. Other forms of ϕ may be used, including the thin plate spline ($\phi(r) = r^2 \ln r$) and the Gaussian ($\phi(r) = \exp\{-\gamma r^2\}$). The cubic form is used because previous RBF methods (*e.g.*, Björkman and Holmström 2000, Regis 2011) had success with this form.

Define the matrix $\Phi \in \mathbb{R}^{n \times n}$ by: $\Phi_{ij} := \phi(\|x_i - x_j\|)$, $i, j = 1, \ldots, n$. Also, define the matrix $P \in \mathbb{R}^{n \times (d+1)}$ so that its ith row is $[1, x_i^{\mathrm{T}}]$. Now, the cubic RBF model that interpolates the points $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ is obtained by solving the system

$$\begin{pmatrix} \Phi & P \\ P^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0_{d+1} \end{pmatrix}, \tag{1}$$

where $F = (f(x_1), \ldots, f(x_n))^{\mathrm{T}}$, $\lambda = (\lambda_1, \ldots, \lambda_n)^{\mathrm{T}} \in \mathbb{R}^n$ and $c = (c_1, \ldots, c_{d+1})^{\mathrm{T}} \in \mathbb{R}^{d+1}$ consists of the coefficients for the linear polynomial $p(x)$. The coefficient matrix in Equation (1) is invertible if and only if $\text{rank}(P) = d + 1$ (Powell 1992).

2.3. *Combining dynamic coordinate search with local metric stochastic RBF*

This section develops DYCORS-LMSRBF, which a modification of the LMSRBF algorithm (Regis and Shoemaker 2007a) that follows the DYCORS framework. In this algorithm, $\varphi(n) = \varphi_0 \times [1 - \ln(n - n_0 + 1)/\ln(N_{\max} - n_0)]$, for all $n_0 \le n \le N_{\max} - 1$, where $\varphi_0 = \min(20/d, 1)$. This $\varphi(n)$ is the same as the function used in DDS except for the additional parameter $\varphi_0 = \varphi(n_0)$, which is the initial value of p_{select}. Here, φ_0 is set so that for problems of dimension $d \le 20$ the algorithm begins with $p_{\text{select}} = 1$. Moreover, for problems with $d > 20$, the average number of variables perturbed initially is $d \times (20/d) = 20$. The justification for this is that, for HEB problems with complex surfaces, it is desirable to perturb only a relatively small number of variables, even at the start, in order to increase the chances of obtaining an improving solution quickly.

As in LMSRBF, the iterate in DYCORS-LMSRBF is selected to be the random trial point with the best weighted score from two criteria: estimated function value from the RBF surrogate (the

RBF criterion); and minimum distance from previously evaluated points (the *distance criterion*). A weighted score of these two criteria is used because it is desirable to seek a trial point with a low RBF surrogate value and that is also far from previously evaluated points in order to improve the current RBF model. Moreover, a balance between global search and local search is maintained by cycling through a set of weights for the RBF and distance criteria. Hence, as in LMSRBF, another set of input parameters is needed for DYCORS-LMSRBF: the *weight pattern for the RBF criterion*, denoted by $\Upsilon = \langle \upsilon_1, \ldots, \upsilon_\kappa \rangle$, where $0 \leq \upsilon_1 \leq \cdots \leq \upsilon_\kappa \leq 1$. If n is the number of previously evaluated points, the weights $\{(w_n^R, w_n^D) : n = n_0, n_0 + 1, \ldots\}$ for the RBF and the distance criteria are set as follows: for all $n \geq n_0$, set

$$w_n^R = \begin{cases} \upsilon_{\mathrm{mod}(n-n_0+1,\kappa)} & \text{if} \quad \mathrm{mod}\ (n - n_0 + 1, \kappa) \neq 0 \\ \upsilon_\kappa & \text{otherwise} \end{cases} \qquad \text{and} \qquad w_n^D = 1 - w_n^R.$$

In the numerical experiments, $\Upsilon = \langle 0.3, 0.5, 0.8, 0.95 \rangle$ is used as the default weight pattern for the RBF criterion.

Next, the selection of the next iterate and the adjustment of the step size are done as follows.

Function x_{n+1} = Select_Evaluation_Point($\Omega_n, \mathcal{B}_n, s_n(x)$)

(a) Estimate function value of trial points. For each $x \in \Omega_n$, compute the RBF value $s_n(x)$. Also, compute $s_n^{\max} = \max\{s_n(x) : x \in \Omega_n\}$ and $s_n^{\min} = \min\{s_n(x) : x \in \Omega_n\}$.
(b) Compute score between 0 and 1 for the RBF 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.
(c) Determine minimum distance from previously evaluated points. For each $x \in \Omega_n$, compute $\Delta_n(x) = \min_{1 \leq i \leq n} \|x - x_i\|$. Also, compute $\Delta_n^{\max} = \max\{\Delta_n(x) : x \in \Omega_n\}$ and $\Delta_n^{\min} = \min\{\Delta_n(x) : x \in \Omega_n\}$. (Here, $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^d$.)
(d) Compute score between 0 and 1 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.
(e) Compute weighted score and select next evaluation point. For each $x \in \Omega_n$, compute $\mathcal{W}_n(x) = w_n^R V_n^R(x) + w_n^D V_n^D(x)$. Let x_{n+1} be the point in Ω_n that minimizes $\mathcal{W}_n$.

Function $[\sigma_{n+1}, C_{\text{success}}, C_{\text{fail}}]$ = Adjust_Step_Size($\sigma_n, C_{\text{success}}, \mathcal{T}_{\text{success}}, C_{\text{fail}}, \mathcal{T}_{\text{fail}}$)

(a) If $C_{\text{success}} \geq \mathcal{T}_{\text{success}}$, then
  $\sigma_{n+1} = 2\sigma_n$, and
  $C_{\text{success}} = 0$.
End.
(b) If $C_{\text{fail}} \geq \mathcal{T}_{\text{fail}}$, then
  $\sigma_{n+1} = \max(\sigma_n/2, \sigma_{\min})$, and
  $C_{\text{fail}} = 0$.
End.

Note that in DYCORS-LMSRBF, the current step size σ_n is reduced by half and C_{fail} is reset to 0 whenever C_{fail} reaches $\mathcal{T}_{\text{fail}}$. The step size is shrunk to facilitate convergence. A minimum step size $\sigma_{\min}$ is set to prevent the trial points from getting too close to the current x_{best}. Moreover, the current step size σ_n is doubled and C_{success} is reset to 0 whenever C_{success} reaches $\mathcal{T}_{\text{success}}$.

In Regis (2011), the RBF method ConstrLMSRBF-BCS also uses a coordinate search strategy called BCS that is similar to the one used in DYCORS-LMSRBF. The main difference between these two methods is that ConstrLMSRBF-BCS uses a fixed value of p_{select} while DYCORS-LMSRBF uses progressively decreasing values of p_{select} given by $\varphi(n)$. In numerical experiments

on 200-D problems, the simpler BCS strategy was not as effective as the dynamic coordinate search strategy in DYCORS-LMSRBF.

This article also differs from Regis (2011) in that the weight pattern for the RBF criterion in DYCORS-LMSRBF is set to $\Upsilon = \langle 0.3, 0.5, 0.8, 0.95 \rangle$ while the weight pattern used for ConstrLMSRBF-BCS was $\Upsilon = \langle 0.95 \rangle$. Numerical results suggest that the more balanced weight pattern in DYCORS-LMSRBF was more effective than the more local weight pattern used in ConstrLMSRBF-BCS. In addition, ConstrLMSRBF-BCS was tested on only one 124-D problem and it used only simple parameter settings. In this article, DYCORS-LMSRBF was shown to be effective on a watershed application and on several problems with up to 200 dimensions. Finally, some sensitivity analysis was also performed on DYCORS-LMSRBF by varying some of its parameters and observing the effect on performance.

2.4. *RBF-assisted dynamically dimensioned search*

This section develops DYCORS-DDSRBF, which is an RBF-assisted modification of DDS (Tolson and Shoemaker 2007) that follows the DYCORS framework. The iterate in DDS is generated by adding random perturbations to some or all of the coordinates of the current x_{best} as described in Section 2.1. These random perturbations are normally distributed with mean zero and some fixed standard deviation σ. Tolson and Shoemaker (2007) suggested using $\sigma = 0.2\ell(\mathcal{D})$, where $\ell(\mathcal{D})$ is the length of one side of the hypercube $\mathcal{D}$. Moreover, in DYCORS-DDSRBF, $\varphi(n) = 1 - [\ln(n - n_0 + 1)/\ln(N_{\max} - n_0)]$, for all $n_0 \le n \le N_{\max} - 1$, which is exactly the same as in DDS. The selection of the iterate does not use any information from previously evaluated points so the performance of DDS on expensive problems might be substantially improved by using a surrogate. Hence, while DDS generates only one trial point and chooses this to be the iterate, DYCORS-DDSRBF generates a large number of trial points in each iteration and the iterate is selected to be the most promising trial point in terms of predicted objective function value according to the RBF model. Note that DYCORS-DDSRBF follows the DYCORS framework where the selection of the iterate is given below.

Function x_{n+1} = Select_Evaluation_Point($\Omega_n, \mathcal{B}_n, s_n(x)$)

(a) For each $x \in \Omega_n$, compute the RBF value $s_n(x)$.
(b) Let x_{n+1} be the point in Ω_n that minimizes $s_n(x)$.

As in DDS, the step size in DYCORS-DDSRBF is fixed throughout the entire optimization run. Hence, in the **Adjust_Step_Size** function, $\sigma_{n+1} := \sigma_n$ and the values of the counters (C_{success} and C_{fail}), the threshold $\mathcal{T}_{\text{success}}$, and failure tolerance $\mathcal{T}_{\text{fail}}$ are never used to adjust the step size.

The idea of using surrogates to assist other optimization methods is not new. For example, Regis and Shoemaker (2004) used RBFs and quadratic polynomials to assist evolution strategies for bound constrained optimization problems. Annicchiarico (2007) used multidimensional Gaussian random field models to assist distributed real GAs for structural shape optimization problems. Shahrokhi and Jahangirian (2010) used neural networks to assist GAs for aerodynamic design optimization. Karakasis and Giannakoglou (2006) used RBF networks to assist evolutionary algorithms for multi-objective optimization. Ray and Smith (2006) also used RBF networks to assist a parallel multi-objective evolutionary algorithm for robust design optimization. In addition, Booker *et al.* (1999) used kriging to assist a pattern search algorithm while Egea *et al.* (2009) used kriging to assist a scatter search algorithm. DYCORS-DDSRBF is simply a surrogate-assisted DDS that is meant for single objective, bound constrained HEB global optimization. In all these methods, the surrogate is typically used to approximate the fitness values or objective function

values of trial solutions and then the expensive function is evaluated only on the most promising trial solutions to reduce computational cost. Although many surrogate-assisted methods have been developed, a surrogate-assisted DDS is still worthy of serious consideration since DDS is a relatively new heuristic that, by itself, shows promise on HEB problems (Tolson and Shoemaker 2007). In fact, in the numerical experiments, DDS outperformed the kriging-based EGO method, a standard GA, and an RBF-assisted evolutionary algorithm on 30-D and 200-D problems. Hence, using a surrogate to assist DDS is expected to yield an even better algorithm for HEB problems.

DYCORS-DDSRBF is simpler than DYCORS-LMSRBF and one main difference between them is that DYCORS-DDSRBF does not use any distance criterion so it is somewhat more greedy when it selects the iterate. Another difference is that DYCORS-DDSRBF (like DDS) uses a fixed step size for the generation of trial points whereas DYCORS-LMSRBF adjusts its step size depending on performance. Finally, the number of trial points generated by DYCORS-DDSRBF is much less than the number of trial points for DYCORS-LMSRBF. In particular, for the numerical experiments, $k = \min(100d, 5000)$ trial points are used for DYCORS-LMSRBF while only $k = \max(\lceil 0.5d \rceil, 2)$ trial points are used for DYCORS-DDSRBF.

3. Calibration of a watershed simulation model

The proposed DYCORS algorithms and alternative optimization methods are applied to a calibration (*i.e.* parameter estimation) problem for a simulation model of the Town Brook watershed (37 km^2), which is inside the larger Cannonsville (1200 km^2) watershed. The Cannonsville Reservoir in upstate New York supplies drinking water to New York City (NYC) and it is necessary to monitor phosphorous loads from the watershed into the reservoir because of concerns about *eutrophication*, which is a type of pollution that can cause severe water quality problems. Phosphorous promotes the growth of algae, which can then clog the water supply. NYC does not have a filtration plant for the drinking water from its reservoirs in upstate New York. If phosphorous levels become too high, NYC would either have to abandon the water supply or build an US$8 billion filtration plant. Hence, it is more effective to control the phosphorous at the watershed level than to build a plant. However, to do this, an accurate model to assess the impact of changes in management practices on phosphorous loads is needed. Part of this analysis requires modelling the flow of water in response to measured rainfall.

This investigation considers the following global optimization problem where the goal is to calibrate the watershed model for flow against real measured flow data:

$$\min \text{SSE}(x) = \sum_{t=1}^{T} (Q_t^{\text{meas}} - Q_t^{\text{sim}}(x))^2,$$

$$\text{s.t.} \quad x_i^{\min} \le x_i \le x_i^{\max}, \ i = 1, \ldots, d.$$

Here, x is the vector of $d = 14$ model parameters, Q_t^{meas} and Q_t^{sim} are the measured and simulated flows on day t, respectively, and $T = 1096$ is the total number of days in the calibration period. For a complete description of these 14 model parameters, see Shoemaker *et al.* (2007). Note that this problem is a nonlinear least squares problem with 1096 residual functions. Hence, in the computational experiments below, DYCORS-LMSRBF and DYCORS-DDSRBF are also compared with a derivative-based algorithm that has been tailored for such problems.

The simulation times for watershed calibration problems can take anywhere from a few seconds to several minutes depending on the size of the modelled region and the length of time in the calibration period. This example covers a relatively small region inside the Cannonsville Reservoir and the simulation time is only about 1.3 seconds on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz

desktop. However, this example is representative of the type of function used in more complex watershed calibration problems. This 14-D optimization problem is referred to as TownBrook14.

4. Computational experiments

4.1. *Alternative optimization methods*

Below are alternative optimization methods for expensive functions that are compared with the proposed DYCORS-LMSRBF and DYCORS-DDSRBF algorithms. The alternative methods include the LMSRBF and DDS algorithms, which are discussed in Section 2, and the EGO method (Jones *et al.* 1998) as implemented by Viana (2010). The comparison with EGO is very important since EGO is one of the most widely cited methods in surrogate model-based global optimization. The RBF method by Gutmann (2001), which is another popular method, is not included in the comparisons because previous work by Regis and Shoemaker (2007a) showed that a multistart version of LMSRBF performed much better than this RBF method on a wide variety of test problems, especially on the relatively high-dimensional ones.

4.1.1. *Evolutionary algorithms*

A popular approach for solving global optimization problems is to use evolutionary algorithms. Hence, the proposed algorithms will be compared with the GA routine from the MATLAB Genetic Algorithm and Direct Search Toolbox (MathWorks® 2009a) and with ESGRBF (Regis and Shoemaker 2004, Shoemaker *et al.* 2007), which is an Evolution Strategy (ES) that uses RBF surrogates.

A (μ, λ)-Evolution Strategy (or simply (μ, λ)-ES) (Bäck 1996) is an evolutionary algorithm that allows self-adaptation of the algorithm parameters. In each generation of this algorithm, there are μ parents that produce λ offspring via crossover and mutation. Typical parameters for an ES are the standard deviations of the normal random mutations that are applied to the components of an offspring solution from a crossover operation. An (m, μ, λ, ν)-ESGRBF is essentially a (μ, λ)-ES that uses an RBF model to estimate the objective function values of the λ offspring and evaluates the objective function only on the ν offspring with the best RBF values. The μ parent solutions for the next generation are then selected to be the best μ solutions from the ν offspring that are evaluated in the current generation. The parameter m is the size of the symmetric Latin hypercube design (SLHD) (Ye *et al.* 2000) used to initialize the RBF model.

4.1.2. *Local optimization methods with restarts*

The DYCORS algorithms are also compared with local optimization methods run in conjunction with a multistart approach. However, because the problems in this article are high-dimensional and there is a relatively limited computational budget, a local minimization algorithm will hardly be able to restart before the budget is exhausted. Hence, one might argue that local minimization algorithms (with few restarts) should not really be compared to global optimization algorithms since they have little chance of finding the global minimum of highly multimodal problems. However, as pointed out earlier, finding the global minimum is not a realistic goal for multimodal HEB problems. Instead, the aim is simply to make quick progress given the relatively limited computational budget, so local minimization methods should not be ruled out. In fact, in the numerical experiments, some of the local solvers performed even better than some algorithms for global optimization in terms of making substantial improvements within a relatively limited number of function evaluations. This suggests that for HEB problems it might be better to invest in local search (from a good starting point) than in global search.

The DYCORS algorithms are compared with two local minimization solvers. One is the NOMADm software (Abramson 2007), which is a MATLAB implementation of the MADS algorithm (Abramson and Audet 2006, Audet and Dennis 2006). MADS is an extension of pattern search that can solve constrained optimization problems. NOMADm is run with the DACE kriging surrogate model (Lophaven *et al.* 2002) and the resulting algorithm is referred to as 'NOMADm-DACE'. Another local solver is 'Fmincon' from the MATLAB Optimization Toolbox (MathWorks® 2009b), where the derivatives are estimated by finite differencing. Fmincon implements an Active-Set SQP algorithm.

The 'Lsqnonlin' solver from the MATLAB Optimization Toolbox (MathWorks® 2009b) is also applied to the Town Brook watershed model calibration problem. Lsqnonlin is a derivative-based algorithm for solving nonlinear least squares problems. It is a subspace trust region method that is based on the interior-reflective Newton method. As in Fmincon, the derivatives needed by Lsqnonlin are estimated by finite differencing.

The starting point for the local minimization solvers is chosen to be the best point from a space-filling experimental design. Moreover, the restart point is chosen to be far from all previous local minimization trajectories. There was no need for a more sophisticated multistart approach since the local minimization solvers performed very few restarts, if any, because of the very limited computational budgets used in this study.

4.2. *Test problems*

The DYCORS algorithms are compared with alternative methods on seven 30-D and four 200-D test functions. These test functions are the well-known Ackley, Rastrigin, Griewank, Levy, Keane and Michalewicz functions, and one 30-D test problem from Schoen (1993) that has the form

$$f(x) = \frac{\sum_{i=1}^{k} f_i \prod_{j \neq i} \|x - z_j\|^2}{\sum_{i=1}^{k} \prod_{j \neq i} \|x - z_j\|^2},$$

where $k \geq 1$, $z_j \in [0,1]^d$ for all $j = 1, \ldots, k$, and $f_i \in R$ for all $i = 1, \ldots, k$. These test problems all have a large number of local minima. Note that 30 and 200 dimensions are considered high-dimensional in surrogate-based optimization since previous articles in this area have mostly dealt with problems of dimension $d < 15$. Table 1 summarizes the characteristics of the test problems in this study.

Since the above test problems are not computationally expensive to evaluate, the different algorithms are compared by pretending that these functions are expensive. This is done by keeping track of the best function values obtained by the different algorithms as the number of function evaluations increases. The relative performance of algorithms on the test problems are expected to be similar to their relative performance on truly expensive functions that have the same general shape as these test problems.

4.3. *Experimental setup*

All computations are performed in MATLAB 7.11 on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop. Eight algorithms are compared: DYCORS-LMSRBF, DYCORS-DDSRBF, LMSRBF, DDS, NOMADm-DACE, EGO, ESGRBF, and the GA and Fmincon solvers from MATLAB. Most of these algorithms are applied to the 14-D Town Brook watershed calibration

Table 1. Test problems for the computational experiments.

Test function	Domain	Global minimum value
Ackley30	$[-15, 20]^{30}$	$-20 - e$
Rastrigin30	$[-4, 5]^{30}$	-30
Griewank30	$[-500, 700]^{30}$	0
Levy30	$[-5, 5]^{30}$	< -11
Keane30	$[1, 10]^{30}$	< -0.39
Michalewicz30	$[0, \pi]^{30}$	< -23
Schoen30	$[0, 1]^{30}$	-991.3269
Ackley200	$[-15, 20]^{200}$	$-20 - e$
Rastrigin200	$[-4, 5]^{200}$	-200
Griewank200	$[-500, 700]^{200}$	0
Keane200	$[1, 10]^{200}$	< -0.21

problem and to the seven 30-D and four 200-D test problems. In addition, the Lsqnonlin solver from MATLAB is also run on the Town Brook problem. There are no results on the 200-D problems for EGO and NOMADm-DACE because these algorithms ran out of memory. Thirty trials of all algorithms are performed on all problems except only five trials of DYCORS-LMSRBF, DYCORS-DDSRBF and LMSRBF are performed on the 200-D problems because the overheads of these algorithms are computationally prohibitive. Each run of each algorithm is given a computational budget of 500 function evaluations for the 14-D and 30-D problems and 1000 function evaluations for the 200-D problems.

All RBF algorithms used the cubic RBF model with a linear tail and they are all initialized using symmetric Latin hypercube designs (SLHDs) (Ye *et al.* 2000) of size $2(d + 1)$ for the 14-D and 30-D problems as was done in Regis and Shoemaker (2007a). However, non-symmetric Latin hypercube designs (LHDs) of size $d + 1$ (the minimum number of initial points required) are used for the 200-D problems to speed up the initialization of the RBF methods and make them more competitive with the non-surrogate-based methods. Note that an SLHD of size $d + 1$ cannot be used because the design points will be affinely dependent thus violating the condition required for cubic RBF interpolation (see Section 2.2). In addition, the initial parent solutions in ESGRBF are the best points from the space-filling experimental design. Each run of an RBF-based algorithm uses a different random seed and a different randomly generated approximately optimal experimental design for the initial evaluation points. The experimental design is chosen to be approximately optimal in the sense that the distances between design points are as large as possible. However, to ensure fair comparison among the algorithms, the same experimental design is used for all RBF-based algorithms on a given run.

In Regis and Shoemaker (2007a), LMSRBF was implemented with restarts. However, in this study, the problems are high-dimensional ($d = 14, 30$ and 200) and the maximum number of function evaluations was relatively limited ($N_{\max} = 500$ or 1000) so restarts would not have occurred for LMSRBF or DYCORS-LMSRBF even if they were allowed. Hence, restarts were not implemented for these algorithms.

Table 2 summarizes the values of the parameters for DYCORS-LMSRBF and LMSRBF that were used in the numerical experiments. Here, $\ell(\mathcal{D})$ represents the length of one side of the hypercube $\mathcal{D}$. The parameters in Table 2 may be adjusted depending on whatever prior information is known about the problem and they are probably not optimal for each problem in this study. However, these parameters of DYCORS-LMSRBF and LMSRBF are fixed for all the problems to ensure fair comparison with alternative methods.

In preliminary investigations, DDS was also run where the initial solution is the best point from an experimental design (SLHD or LHD). The resulting algorithms are referred to as DDS-SLHD or DDS-LHD. The numerical experiments indicate that there is not much difference in performance

Table 2. Parameter values for LMSRBF and DYCORS-LMSRBF.

Parameter	Value
$k = \lvert\Omega_n\rvert$ (number of trial points for each iteration)	$\min(100d, 5000)$
Υ (weight pattern)	$\langle 0.3, 0.5, 0.8, 0.95\rangle$
κ (number of weights in Υ)	4
σ_{init} (initial step size)	$0.2\ell(\mathcal{D})$
σ_{min} (minimum step size)	$(0.2)(1/2)^6\ell(\mathcal{D})$
$\mathcal{T}_{\text{success}}$ (threshold parameter for deciding when to increase the step size)	3
$\mathcal{T}_{\text{fail}}$ (tolerance parameter for deciding when to reduce the step size)	$\max(d, 5)$

between DDS and DDS-SLHD on the 14-D and 30-D problems. Moreover, DDS-LHD was worse than DDS on the 200-D problems because the initialization in DDS-LHD consumed too many function evaluations that were not really needed by the algorithm. To avoid clutter in the presentation of the results, DDS-SLHD and DDS-LHD were removed from the plots.

NOMADm is run with the option that uses a kriging surrogate implementation from the DACE toolbox by Lophaven *et al.* (2002) and the resulting algorithm is referred to as NOMADm-DACE. A kriging model with a Gaussian correlation function is used where the parameters are obtained by maximum likelihood estimation. The initial mesh size is set to $0.05\ell([a, b])$, the mesh refinement factor is set to 0.5 and the mesh coarsening factor is set to 2. For the other parameters of NOMADm-DACE, the default values are used. Moreover, NOMADm-DACE uses the same initial experimental design that is used in all RBF-based algorithms.

For the (m, μ, λ, ν)-ESGRBF algorithm, $\mu = 8$, $\lambda = 50$ and $\nu = 20$ as in Shoemaker *et al.* (2007). Moreover, for each trial, ESGRBF used the same experimental design used by the other RBF algorithms. Finally, the mutation rate is $0.05\ell(\mathcal{D})$.

For the MATLAB solvers Fmincon and Lsqnonlin, the minimum step size for calculating the finite difference derivatives is $10^{-8}\ell(\mathcal{D})$ for the test problems and $10^{-4}\ell(\mathcal{D})$ for the watershed problem. The larger step size for the watershed problem is meant to improve the performance of the solvers. For MATLAB's GA, the population size is set to d. Moreover, for each trial of GA, the best point from an experimental design is used as a starting point for generating the initial population. For all other parameters of Fmincon, Lsqnonlin and GA, the default values are used.

When function evaluations are expensive, the computational overhead of an optimization algorithm is small compared to the time spent on function evaluations. Hence, the running time of the algorithm is practically determined by the number of function evaluations. In this case, the performance of the algorithm on a particular problem is measured by keeping track of the best function value obtained after every function evaluation. Since there are multiple trials for each problem, the *average best function value* for the algorithm after every function evaluation is obtained and the results are summarized in a plot called an *average progress curve*. For expensive functions, it makes sense to present the results in this way since one is mostly interested in how the different algorithms compare after a fixed computational budget. For example, one can easily determine the relative performance of the different algorithms after 50, 100 or 200 function evaluations by looking at the values indicated by the corresponding average progress curves.

5. Results and discussion

5.1. *Explanation of the results*

Figures 1–3 show the average progress curves when the various optimization algorithms are applied to the 14-D watershed calibration problem and to the 30-D and 200-D test problems. To

get an idea of the variability in the results, error bars that represent 95% t-confidence intervals for the mean are included. In this case, each side of the error bar has length equal to 2.045 (for 30 trials) or 2.776 (for 5 trials) times the standard deviation of the best function value divided by the square root of the number of trials. Here, 2.045 and 2.776 are the critical values corresponding to a 95% confidence level for a t-distribution with 29 and 4 degrees of freedom, respectively. In addition, Table 3 provides the minimum (best), maximum (worst), median, mean and standard error of the mean of the best function values over the different trials for all algorithms after 500 function evaluations for the 14-D watershed problem and the 30-D problems and after 1000 function evaluations for the 200-D problems.

In general, an algorithm A is better than algorithm B for a particular global minimization problem if the average best function value obtained by algorithm A is consistently less than the average best function value obtained by algorithm B for a wide range of computational budgets. That is, algorithm A is better than algorithm B for a particular problem if the average progress curve for algorithm A is generally below the average progress curve for algorithm B for that problem.

5.2. *Results on watershed calibration and the 30-D test problems*

Figure 1(a) and Table 3 show that DYCORS-LMSRBF and DYCORS-DDSRBF are the best algorithms on the 14-D watershed calibration problem (TownBrook14). In particular, they are much

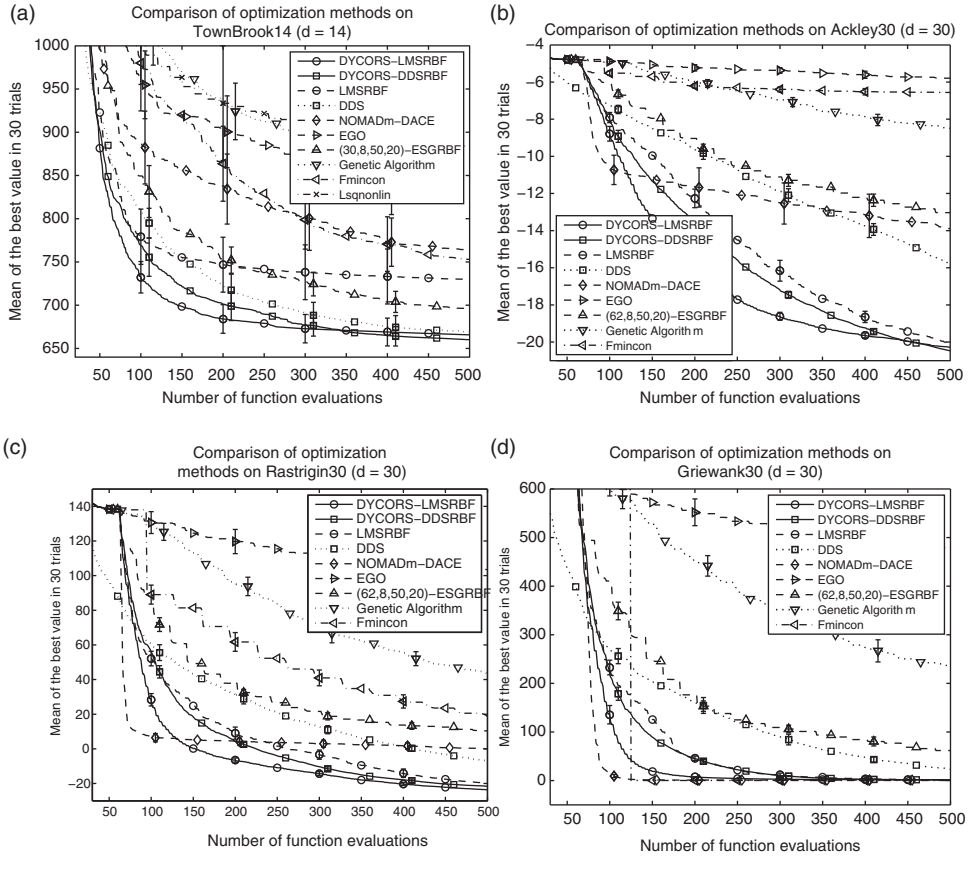

Figure 1. Average values of best solutions (over 30 trials) found by optimization methods on the test problems. Error bars represent 95% confidence intervals about the mean.

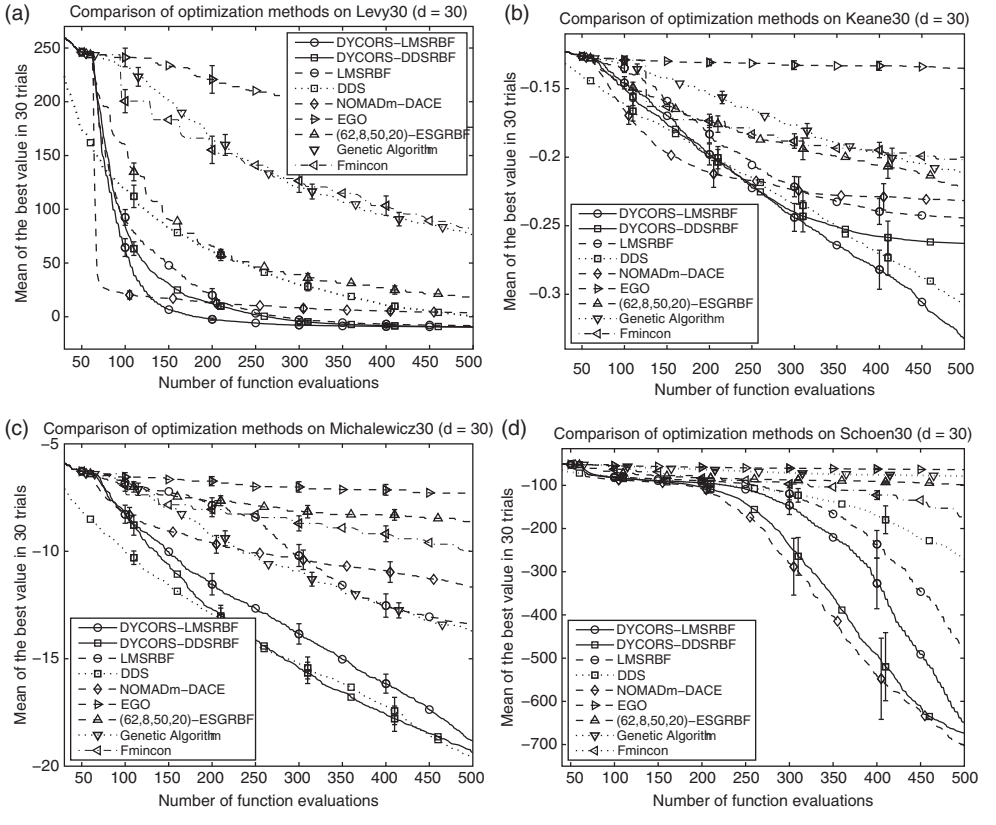

Figure 2. Average values of best solutions (over 30 trials) found by optimization methods on the test problems. Error bars represent 95% confidence intervals about the mean.

better than EGO, NOMADm-DACE, GA, Fmincon and Lsqnonlin, and they are also better than DDS and the other RBF algorithms (LMSRBF and ESGRBF). Previous studies (Shoemaker *et al.* 2007, Tolson and Shoemaker 2007) indicated that DDS and ESGRBF are among the most promising algorithms for watershed calibration. The results suggest that the DYCORS algorithms are even more promising than either of these alternatives. Further, Figure 1(a) shows that DYCORS-LMSRBF was initially better than DYCORS-DDSRBF on TownBrook14 but the latter caught up with the former after about 350 function evaluations.

Figures 1–2 and Table 3 also show that DYCORS-LMSRBF is generally the best algorithm on the 30-D test problems followed by DYCORS-DDSRBF. In particular, DYCORS-LMSRBF is consistently better than LMSRBF, ESGRBF, EGO and GA on all seven 30-D problems and it is also consistently much better than DDS on six of the 30-D problems (all except Michalewicz30). In addition, it is much better than Fmincon on six of the 30-D problems (all except Griewank30) and it is generally better than NOMADm-DACE on five of the 30-D problems (all except Griewank30 and Schoen30). Finally, DYCORS-LMSRBF is generally better than DYCORS-DDSRBF on five of the 30-D problems (all except Michalewicz30 and Schoen30).

DYCORS-DDSRBF is better than DDS on five of the 30-D problems (all except Keane30 and Michalewicz30). It is consistently better than LMSRBF, ESGRBF, EGO and GA on all 30-D problems and it is also better than Fmincon on six of the 30-D problems (all except Griewank30). Finally, DYCORS-DDSRBF is better or competitive with NOMADm-DACE on five of the 30-D problems (all except Griewank30 and Schoen30).

Table 3. Statistics on the best objective function value from 30 trials of the algorithms on the test problems after 500 function evaluations.

		TownBrook14	Ackley30	Rastrigin30	Griewank30	Levy30	Keane30	Michalewicz30	Schoen30
DYCORS-LMSRBF	best	609.20	−21.56	−26.83	1.13	−11.23	−0.41	−21.73	−941.04
	worst	779.07	−19.47	−16.71	1.68	−6.82	−0.33	−15.88	−265.41
	median	653.40	−20.35	−23.29	1.35	−10.32	−0.37	−19.61	−678.75
	mean	665.73	−20.39	−23.51	1.36	−9.96	−0.37	−19.50	−673.14
	std error	7.55	0.07	0.40	0.03	0.20	0.00	0.26	38.59
DYCORS-DDSRBF	best	615.74	−21.48	−24.83	1.13	−11.33	−0.32	−22.63	−935.20
	worst	736.98	−19.61	−16.40	1.63	−2.61	−0.22	−16.03	−118.56
	median	663.97	−20.43	−22.10	1.31	−10.06	−0.26	−19.67	−653.48
	mean	660.20	−20.47	−21.58	1.33	−9.24	−0.26	−19.36	−674.39
	std error	5.58	0.08	0.40	0.02	0.40	0.01	0.30	43.47
LMSRBF	best	628.83	−21.34	−25.91	1.03	−10.00	−0.30	−18.17	−791.45
	worst	1070.02	−18.97	−10.21	4.17	−5.73	−0.21	−10.47	−214.07
	median	718.62	−20.02	−20.37	1.26	−8.23	−0.24	−13.30	−450.91
	mean	729.44	−20.07	−19.65	1.47	−8.22	−0.24	−13.42	−469.82
	std error	15.65	0.11	0.80	0.11	0.20	0.00	0.31	32.87
DDS	best	618.99	−17.67	−13.66	12.19	−4.09	−0.37	−23.09	−706.10
	worst	755.50	−13.97	2.18	44.94	6.66	−0.23	−15.68	−93.21
	median	667.40	−15.94	−7.64	21.45	−0.20	−0.30	−19.56	−258.17
	mean	669.03	−15.75	−6.97	23.16	0.69	−0.31	−19.60	−269.87
	std error	5.77	0.18	0.62	1.54	0.65	0.01	0.26	26.82
NOMADm-DACE	best	675.64	−20.74	−11.80	0.76	−5.11	−0.30	−14.90	−956.73
	worst	964.05	−9.20	7.66	0.98	22.93	−0.19	−8.76	−261.35
	median	740.85	−13.53	0.14	0.90	2.78	−0.23	−11.84	−659.90
	mean	762.95	−13.90	0.07	0.90	3.74	−0.23	−11.66	−702.77
	std error	12.55	0.61	0.85	0.01	1.08	0.00	0.25	39.56

EGO	best	677.45	−6.70	78.45	364.30	131.94	−0.17	−8.74	−72.42
	worst	1159.08	−4.74	130.83	646.99	229.37	−0.12	−6.34	−57.38
	median	805.73	−5.78	101.48	471.15	176.16	−0.13	−7.14	−63.82
	mean	835.98	−5.79	103.51	477.81	176.33	−0.14	−7.30	−63.68
	std error	20.81	0.09	2.57	13.13	4.51	0.00	0.12	0.77
ESGRBF	best	647.26	−14.60	2.65	26.53	1.38	−0.28	−9.85	−164.24
	worst	781.18	−10.62	20.34	105.88	53.04	−0.18	−6.85	−87.77
	median	694.87	−13.28	9.85	62.45	18.10	−0.22	−8.67	−91.88
	mean	695.61	−13.03	10.38	61.69	18.47	−0.22	−8.62	−98.63
	std error	5.35	0.18	0.82	3.55	1.98	0.00	0.13	2.83
GA	best	677.88	−11.33	23.27	133.65	45.59	−0.24	−16.17	−104.26
	worst	1034.12	−6.87	72.86	371.59	109.95	−0.18	−11.87	−64.08
	median	822.08	−8.60	42.18	229.76	75.58	−0.22	−13.66	−78.01
	mean	849.36	−8.49	43.32	235.20	76.55	−0.21	−13.72	−78.90
	std error	15.76	0.18	1.87	11.37	2.85	0.00	0.23	1.22
Fmincon	best	650.02	−8.63	4.93	0.00	49.94	−0.26	−12.57	−335.43
	worst	946.09	−6.01	46.81	0.00	130.85	−0.17	−7.45	−96.98
	median	742.34	−6.34	18.78	0.00	83.77	−0.20	−10.09	−159.20
	mean	753.00	−6.56	18.92	0.00	82.62	−0.20	−9.98	−174.77
	std error	12.79	0.10	1.57	0.00	3.90	0.00	0.20	10.68

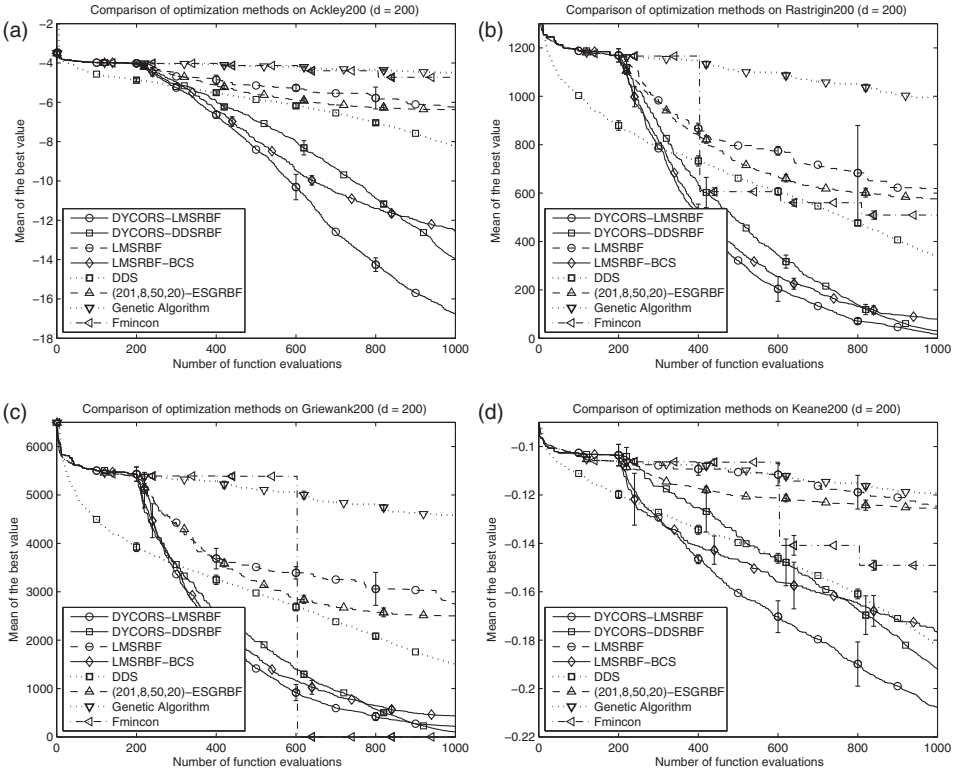

Figure 3. Average values of best solutions found by optimization methods on the 200-D test problems. Error bars represent 95% confidence intervals about the mean.

5.3. *Results on the 200-D test problems*

The two DYCORS algorithms are also applied to the 200-D versions of the Ackley, Rastrigin, Griewank and Keane test problems and compared with the alternatives, except the kriging-based EGO and NOMADm-DACE algorithms. The results are shown in Figure 3 and Table 4. EGO and NOMADm-DACE are not included because they ran out of memory during the optimization trials on the 200-D problems. The same memory issue was observed by Regis (2011) when running NOMADm-DACE on a 124-D problem. Fitting kriging models is known to be more computationally demanding than building the RBF models described in Section 2.2. Besides, even if EGO can produce results on these 200-D problems, the results on the 30-D versions suggest that these results might be relatively poor. Hence, kriging might not be suitable for HEB problems.

As mentioned earlier, the main difference between DYCORS-LMSRBF and ConstrLMSRBF-BCS (Regis 2011) is that DYCORS-LMSRBF uses a more dynamic coordinate search strategy than the one used in ConstrLMSRBF-BCS. In particular, when selecting trial points (out of which the actual iterate will be selected), the probability of perturbing a given coordinate of the current best solution (p_{select}) is fixed in ConstrLMSRBF-BCS while this probability changes in DYCORS-LMSRBF as described in Section 2.3. Figure 3 also includes the results of the LMSRBF-BCS algorithm, which is essentially ConstrLMSRBF-BCS applied to a bound constrained problem. LMSRBF-BCS is similar to DYCORS-LMSRBF except that p_{select} is fixed at 0.1, which is the same value used in ConstrLMSRBF-BCS. Note that in all four problems, DYCORS-LMSRBF is consistently better than LMSRBF-BCS, especially in later iterations. This suggests that the

Table 4. Statistics on the best objective function value from 5† or 30 trials of the algorithms on the 200-D test problems after 1000 function evaluations.

		Ackley200	Rastrigin200	Griewank200	Keane200
DYCORS-LMSRBF†	best	−17.03	6.88	106.48	−0.21
	worst	−16.44	35.07	289.40	−0.20
	median	−16.74	8.86	243.13	−0.21
	mean	−16.77	16.15	216.32	−0.21
	std error	0.11	5.51	34.71	0.00
DYCORS-DDSRBF†	best	−14.19	19.48	88.33	−0.20
	worst	−13.66	40.61	108.87	−0.18
	median	−13.90	25.87	104.88	−0.19
	mean	−13.97	29.97	102.70	−0.19
	std error	0.10	4.39	3.68	0.00
LMSRBF†	best	−7.13	384.98	1922.28	−0.14
	worst	−5.59	749.71	3237.99	−0.12
	median	−6.03	632.90	3015.94	−0.12
	mean	−6.32	618.70	2735.72	−0.12
	std error	0.29	64.85	249.90	0.00
LMSRBF-BCS†	best	−13.00	69.08	414.88	−0.19
	worst	−12.24	87.47	488.42	−0.17
	median	−12.45	80.35	418.93	−0.18
	mean	−12.54	79.30	433.24	−0.18
	std error	0.13	2.97	13.89	0.00
DDS	best	−8.85	277.49	1210.66	−0.20
	worst	−7.52	397.81	1816.00	−0.17
	median	−8.23	340.29	1480.59	−0.18
	mean	−8.22	336.72	1500.24	−0.18
	std error	0.06	5.01	27.32	0.00
ESGRBF	best	−7.32	440.64	1844.62	−0.14
	worst	−5.44	662.42	2976.87	−0.12
	median	−6.29	569.63	2527.10	−0.13
	mean	−6.37	576.32	2500.58	−0.13
	std error	0.07	10.28	52.88	0.00
GA	best	−4.82	858.52	4101.73	−0.12
	worst	−4.30	1057.37	4941.73	−0.12
	median	−4.45	1003.55	4606.89	−0.12
	mean	−4.49	995.05	4585.59	−0.12
	std error	0.02	8.03	32.23	0.00
Fmincon	best	−5.05	416.53	0.00	−0.16
	worst	−4.14	599.88	0.13	−0.14
	median	−4.82	513.49	0.00	−0.15
	mean	−4.73	508.91	0.02	−0.15
	std error	0.05	8.36	0.01	0.00

strategy of reducing p_{select} is better than simply fixing it at a particular value for all iterations. Hence, the dynamic coordinate search strategy used in DYCORS-LMSRBF is generally more effective than the BCS strategy.

Figure 3 and Table 4 also show that both DYCORS-LMSRBF and DYCORS-DDSRBF are generally much better than LMSRBF, DDS, ESGRBF, GA and Fmincon on the 200-D problems. DDS was initially better than the two DYCORS algorithms but it was soon overtaken by these algorithms. Note that these DYCORS algorithms require $d + 1 = 201$ LHD points on 200-D

problems while DDS only requires one starting point, usually chosen to be the best from a few uniform random points over the search space. These results are significant for surrogate-based optimization since few articles have dealt with HEB problems and they suggest that DYCORS is promising for these problems.

5.4. *Discussion of the comparison between algorithms*

In summary, Figures 1–3 and Tables 3 and 4 indicate that the two DYCORS algorithms are generally better than the alternatives on the 14-D watershed calibration problem and on the 30-D and 200-D test problems. In particular, DYCORS-LMSRBF is better than LMSRBF and DDS on 11 of the 12 problems (all except Michalewicz30) while DYCORS-DDSRBF is better than or competitive with DDS on 10 of the 12 problems (all except Keane30 and Michalewicz30). Recall that DYCORS-LMSRBF is a modification of LMSRBF that incorporates the dynamic coordinate search strategy of DDS. These results suggest that DYCORS-LMSRBF is a more effective and robust algorithm than either LMSRBF or DDS on the problems in this study, which are all high-dimensional. Furthermore, DYCORS-DDSRBF is also an improvement over DDS.

EGO and NOMADm-DACE ran out of memory when applied to the 200-D problems. Moreover, the two DYCORS algorithms substantially outperformed EGO and NOMADm-DACE on TownBrook14 and on most of the 30-D problems. EGO is a popular surrogate-based method while NOMADm-DACE is a state-of-the-art direct search method that also uses kriging surrogates. Jones *et al.* (1998) and Björkman and Holmström (2000) reported good results for EGO on low-dimensional problems. However, the results here suggest that it might not be effective on HEB problems. A possible explanation is that EGO focuses more on global search and has limited ability to do local search, which is necessary for quick progress. NOMADm-DACE is a good algorithm compared to EGO, ESGRBF, Fmincon and GA but it did not perform as well as the DYCORS algorithms. This might be because NOMADm-DACE is meant to converge to a local minimum while all the problems in this study have multiple local minima. Although NOMADm-DACE is allowed to restart after it has converged to a local minimum, the computational budget for each trial is not even enough for a single local minimization run. In addition, for both EGO and NOMADm-DACE, it is also possible that the kriging surrogates had trouble approximating the black-box function well enough.

LMSRBF performed relatively well on the 30-D problems. It is consistently much better than ESGRBF, EGO and GA on all seven 30-D problems. Moreover, it is better than DDS on five of the 30-D problems (all except Keane30 and Michalewicz30) and it is also better than Fmincon on six of the 30-D problems (all except Griewank30). In addition, LMSRBF outperformed NOMADm-DACE on TownBrook14 and Ackley30 and it is competitive with NOMADm-DACE on five other test problems. However, its performance on TownBrook14 is relatively poor and much worse than that of the DYCORS algorithms and DDS. After about 200 function evaluations, the average progress curve of LMSRBF on TownBrook14 is essentially flat, indicating that there is hardly any improvement from that point. In addition, LMSRBF performed quite poorly on the 200-D problems compared to all other algorithms except GA.

From Figures 1–3, the derivative-based methods Fmincon and Lsqnonlin did not perform well on most of the problems. However, Fmincon was consistently better than EGO on TownBrook14 and on all 30-D problems. Moreover, it was better than GA on TownBrook14, Rastrigin30, Griewank30, Schoen30 and the four 200-D problems. Also, Fmincon performed very well on the 30-D and 200-D Griewank functions. This is not really surprising since Locatelli (2003) showed that the global minimum of the Griewank function is hard to find in lower dimensions but is relatively easy to find in higher dimensions. Lsqnonlin is a derivative-based algorithm that takes advantage of the nonlinear least squares structure of TownBrook14. However, it was the

worst method on TownBrook14. One main reason for the relatively poor performance of Fmincon or Lsqnonlin is that the finite differencing procedure for estimating the gradient vector or the Jacobian matrix requires too many function evaluations on high-dimensional problems. The function evaluations performed to calculate the finite difference derivatives might be better spent on more promising points that can improve the objective function.

The main advantage of the DDS strategy is that, by focusing on a subset of the coordinates when perturbing the current best solution x_{best}, there is a bigger chance of getting an improved solution since many of the coordinates in the current x_{best} are preserved. This idea is similar to a traditional coordinate search except that groups of coordinates are perturbed instead of a single coordinate in each iteration. This strategy is helpful when dealing with complex, high-dimensional and highly multimodal surfaces because the trial points generated tend to be closer to the current x_{best} (see Section 2.1). Moreover, the random selection of groups of coordinates in DDS is also helpful when the objective function is not very sensitive to changes in some of the decision variables. The poor performance of LMSRBF on TownBrook14 suggests that the surface of its objective function is rather complex so that at some point (after about 100 function evaluations) the RBF model appears to be only marginally helpful in guiding the search for improved solutions. However, by implementing the DDS strategy within LMSRBF, one obtains a much better and more robust algorithm (*i.e.* DYCORS-LMSRBF) that can still effectively use an RBF model to guide the search for an improved solution. In DYCORS-LMSRBF, the RBF model appears to do a better job of identifying a promising point among the trial points since these trial points have more in common with, and are closer to, the current x_{best} compared to the trial points generated by LMSRBF.

This article also compares a surrogate-assisted evolutionary algorithm (ESGRBF) with other surrogate-based optimization algorithms such as LMSRBF and EGO. In the ESGRBF implemented here, the RBF model is used to select the $\nu = 20$ most promising points from the $\lambda = 50$ offspring that are randomly generated in each generation of the evolutionary algorithm. On the other hand, LMSRBF and DYCORS-LMSRBF use an RBF model to select only one point from a large number ($k = \min\{100d, 5000\}$) trial points in each iteration. An interesting question is: 'Would it be better to use the surrogate model to speed up a standard evolutionary algorithm or simply use it to select each function evaluation point carefully?' Intuitively, the latter approach seems better since the selection of function evaluation points in either LMSRBF or DYCORS-LMSRBF is more deliberate compared to that in ESGRBF. The results appear to support this idea since DYCORS-LMSRBF is consistently much better than ESGRBF on all problems, including TownBrook14. Moreover, LMSRBF is consistently much better than ESGRBF on all 30-D problems and it is somewhat competitive with ESGRBF on TownBrook14.

5.5. *Sensitivity of DYCORS-LMSRBF to the algorithm parameters*

A set of input parameters for DYCORS-LMSRBF is the weight pattern Υ for the RBF criterion. Weights that are close to one typically result in local search since the emphasis is on selecting a trial point with low RBF value while weights that are smaller typically result in global search since the emphasis is on selecting trial points that are far from previous iterates. A weight pattern of $\Upsilon = \langle 0.3, 0.5, 0.8, 0.95\rangle$, which provides a good balance of global and local search, is used in this study as was done in a parallel implementation of LMSRBF (Regis and Shoemaker 2009).

To examine the effect of changing the weight pattern on the performance of DYCORS-LMSRBF, this algorithm is also implemented using a more local weight pattern of $\Upsilon = \langle 0.95\rangle$, which is the same one used in LMSRBF (Regis and Shoemaker 2007a) and ConstrLMSRBF and ConstrLMSRBF-BCS (Regis 2011). This algorithm is referred to as DYCORS-LMSRBF (local) while the default implementation is simply DYCORS-LMSRBF. Figure 4 shows the comparison between these two implementations. Observe that DYCORS-LMSRBF

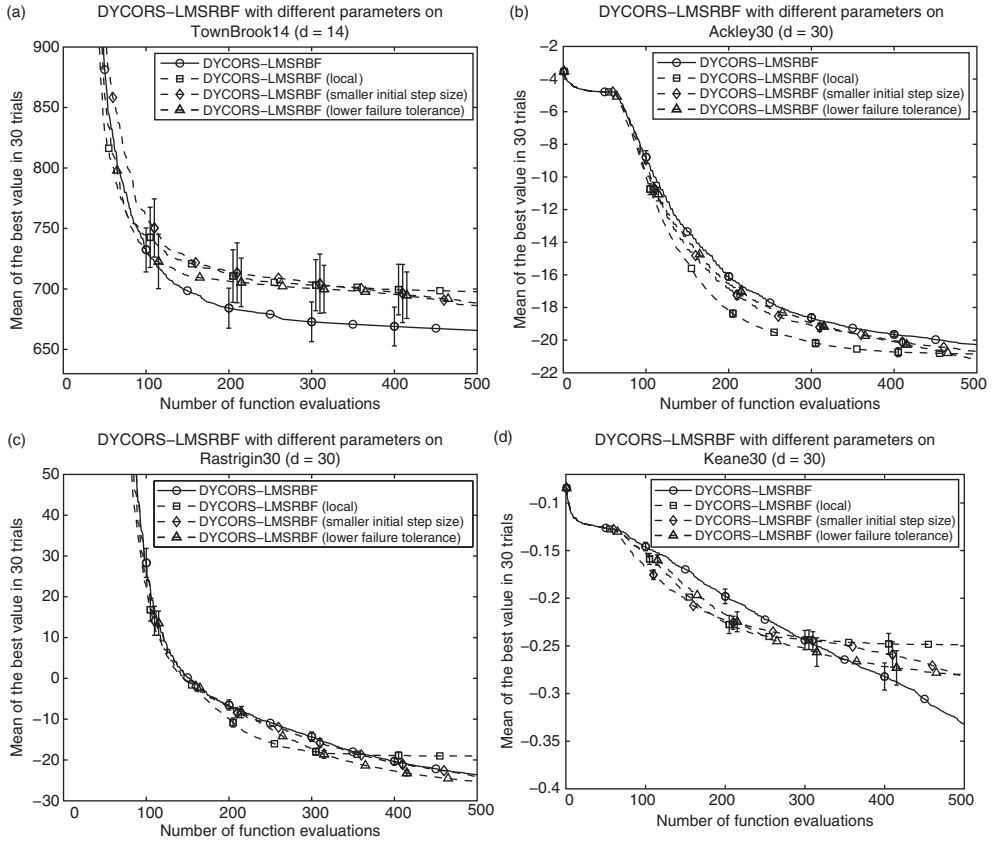

Figure 4. Average values of best solutions (over 30 trials) found by DYCORS-LMSRBF with different parameters. Error bars represent 95% confidence intervals about the mean.

is better than DYCORS-LMSRBF (local) on TownBrook14 and Keane30. On the other hand, DYCORS-LMSRBF (local) is better than DYCORS-LMSRBF on Ackley30. These results suggest that the best weight pattern varies from problem to problem. On more complicated surfaces, such as TownBrook14 and Keane30, a more balanced weight pattern might be better while on relatively easier surfaces a more local weight pattern might result in quicker progress.

Another input parameter for DYCORS-LMSRBF is the initial step size σ_{init}. Here, $\sigma_{\text{init}} = 0.2\ell(\mathcal{D})$, which is the value recommended for DDS (Tolson and Shoemaker 2007). To see the effect of changing this parameter, DYCORS-LMSRBF was also run with a smaller value of $\sigma_{\text{init}} = 0.05\ell(\mathcal{D})$. Figure 4 shows that this results in a substantial deterioration in the performance of DYCORS-LMSRBF on TownBrook14 and Keane30. However, DYCORS-LMSRBF is not sensitive to this parameter for Ackley30 and Rastrigin30. This suggests that on more complicated surfaces a relatively large initial step size might be more effective since it allows the algorithm to explore a large region initially before focusing the search on a smaller region.

Finally, the effect of changing the tolerance parameter $\mathcal{T}_{\text{fail}}$ for deciding when to reduce the step size was also examined. The original setting for this parameter was $\mathcal{T}_{\text{fail}} = \max(d, 5)$. DYCORS-LMSRBF was run with a lower value for this parameter of $\mathcal{T}_{\text{fail}} = \max(\lfloor d/2 \rfloor, 5)$. Figure 4 shows that this results in worse performance for DYCORS-LMSRBF on TownBrook14 and Keane30 while it results in a slight improvement on Rastrigin30. A smaller value for $\mathcal{T}_{\text{fail}}$ results in more frequent reduction in step sizes, which does not seem to be effective on more complicated surfaces.

5.6. *Running times*

The running times of optimization algorithms on computationally expensive functions are dominated by the total time spent on function evaluations. However, the computational overhead of surrogate-based algorithms can also be substantial. For example, the average running times (excluding time spent on function evaluations) of DYCORS-LMSRBF and DYCORS-DDSRBF for 500 function evaluations on TownBrook14 using an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop are 61.81 and 3.42 seconds, respectively. The average running times of DYCORS-LMSRBF and DYCORS-DDSRBF for 500 function evaluations on Ackley30 using the same machine are 233.21 and 4.61 seconds, respectively. Finally, the average running times of DYCORS-LMSRBF and DYCORS-DDSRBF for 1000 function evaluations on Ackley200 using this machine are 10127.07 seconds (2.81 hours) and 277.39 seconds, respectively. Clearly, DYCORS-DDSRBF involves much less computational overhead than DYCORS-LMSRBF. This is because DYCORS-DDSRBF uses considerably fewer trial points than DYCORS-LMSRBF in every iteration. Note that if the objective function is truly expensive, say each evaluation takes 1 hour, then these computational overheads are negligible compared to the total running time of either algorithm for 500 or 1000 function evaluations.

6. Summary and conclusions

This article introduced the DYCORS framework for surrogate-based HEB optimization that uses a dynamic coordinate search strategy similar to the one used in the DDS algorithm (Tolson and Shoemaker 2007). The iterate in DYCORS is selected from a set of random trial solutions obtained by perturbing only a subset of the coordinates of the current best solution as was done in ConstrLMSRBF-BCS (Regis 2011). However, the probability of perturbing a coordinate of the current best solution is fixed in ConstrLMSRBF-BCS while this perturbation probability decreases in DYCORS. As a result, the number of coordinates perturbed tends to decrease as the algorithm reaches the computational budget. Numerical results showed that the strategy for generating trial solutions in DYCORS is more effective than the one used in ConstrLMSRBF-BCS. Moreover, DYCORS is more general and flexible than ConstrLMSRBF-BCS in that it allows other ways of selecting the iterate from the set of trial solutions and it also allows other schemes for adjusting the step size.

Two algorithms that follow the DYCORS framework, namely DYCORS-LMSRBF and DYCORS-DDSRBF, were developed. DYCORS-LMSRBF is a modification of the LMSRBF algorithm (Regis and Shoemaker 2007a) while DYCORS-DDSRBF is an RBF-assisted DDS. The two DYCORS algorithms were compared with alternative methods on the 14-D Town Brook watershed calibration problem and on eleven 30-D and 200-D test problems with a large number of local minima. The alternatives were: the kriging-based EGO algorithm; NOMADm-DACE, which is an implementation of the MADS algorithm that uses kriging surrogates; LMSRBF; DDS; ESGRBF, which is an RBF-assisted evolution strategy; an active-set SQP algorithm that uses finite-difference derivatives; and a GA. For the watershed calibration problem, DYCORS-LMSRBF and DYCORS-DDSRBF were also compared with a nonlinear least squares algorithm that uses finite-difference derivatives.

Overall, DYCORS is very promising for watershed calibration and for HEB optimization. It helps push the frontier in surrogate-based optimization since previous methods in this area have mostly been tested on low-dimensional problems while DYCORS algorithms have been successfully applied to 200-D problems. Previous studies have shown that DDS and ESGRBF are the best algorithms for the Town Brook watershed calibration problem. The numerical results indicate that DYCORS-LMSRBF is now the best algorithm on this watershed problem followed by

DYCORS-DDSRBF. Further, the two DYCORS algorithms are generally much better than the other alternatives on most of the 30-D and 200-D test problems. In particular, DYCORS-LMSRBF is an improvement over LMSRBF and DDS while DYCORS-DDSRBF is an improvement over DDS on most of the problems. In addition, the two DYCORS algorithms are consistently much better than EGO on the watershed application and on the test problems. The Town Brook problem is representative of watershed calibration problems, which constitute an important class of optimization problems in the field of water resources. Hence, these results are expected to have significant impact in water resources research in addition to the area of surrogate-based optimization, which applies to many engineering fields.

Acknowledgements

The authors would like to thank Saint Joseph's University for providing a Summer Research Grant to Rommel Regis and NSF for grant CCSF1116298 to Prof. Shoemaker. They would also like to thank Dr Bryan Tolson for providing the simulation code for the Townbrook watershed calibration model as well as for some discussions during the initial stages of this project.

References

Abramson, M.A., 2007. NOMADm version 4.6 user's guide. Unpublished manuscript.

Abramson, M.A. and Audet, C., 2006. Convergence of mesh adaptive direct search to second-order stationary points. *SIAM Journal on Optimization*, 17 (2), 606–619.

Aleman, D.M., Romeijn, H.E., and Dempsey, J.F., 2009. A response surface approach to beam orientation optimization in intensity modulated radiation therapy treatment planning. *INFORMS Journal on Computing*, 21 (1), 62–76.

Annicchiarico, W., 2007. Metamodel-assisted distributed genetic algorithms applied to structural shape optimization problems. *Engineering Optimization*, 39 (7), 757–772.

Audet, C. and Dennis, J.E., Jr, 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17 (2), 188–217.

Bäck, T., 1996. *Evolutionary algorithms in theory and practice*. New York: Oxford University Press.

Bettonvil, B. and Kleijnen, J.P.C., 1997. Searching for important factors in simulation models with many factors: sequential bifurcation. *European Journal of Operational Research*, 96 (1), 180–194.

Björkman, M. and Holmström, K., 2000. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1 (4), 373–397.

Booker, A.J., *et al.*, 1999. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17 (1), 1–13.

Buhmann, M.D., 2003. *Radial basis functions*. Cambridge, UK: Cambridge University Press.

Conn, A.R., Scheinberg, K., and Vicente, L.N., 2009. *Introduction to derivative-free optimization*. Philadelphia, PA: SIAM.

Cressie, N., 1993. *Statistics for spatial data*. New York: Wiley.

Egea, J.A., *et al.*, 2009. Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization*, 43 (2-3), 175–190.

Giunta, A.A., *et al.*, 1997. Aircraft multidisciplinary design optimisation using design of experiments theory and response surface modelling. *Aeronautical Journal*, 101 (1008), 347–356.

Glover, F., 1998. A template for scatter search and path relinking. *In*: J.-K. Hao, *et al.*, eds. *Artificial evolution*. Lecture notes in computer science 1363. Berlin: Springer-Verlag, 13–54.

Gutmann, H.-M., 2001. A radial basis function method for global optimization. *Journal of Global Optimization*, 19 (3), 201–227.

Holmström, K., 2008. An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. *Journal of Global Optimization*, 41 (3), 447–464.

Horst, R., Pardalos, P.M., and Thoai, N.V., 2000. *Introduction to global optimization*. 2nd ed. Dordrecht, The Netherlands: Kluwer.

Huang, D., *et al.*, 2006. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34 (3), 441–466.

Jakobsson, S., *et al.*, 2010. A method for simulation based optimization using radial basis functions. *Optimization and Engineering*, 11 (4), 501–532.

Jones, D.R., Schonlau, M., and Welch, W.J., 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13 (4), 455–492.

Karakasis, M.K. and Giannakoglou, K.C., 2006. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38 (8), 941–957.

Kushner, H., 1962. A versatile stochastic model of a function of unknown and time-varying form. *Journal of Mathematical Analysis and Applications*, 5, 150–167.
Laguna, M. and Marti, R., 2003. *Scatter search: methodology and implementations in C*. Boston, MA: Kluwer.
Locatelli, M., 2003. A note on the Griewank test function. *Journal of Global Optimization*, 25 (2), 169–174.
Lophaven, S.N., Nielsen, H.B., and Søndergaard, J., 2002. *DACE: a MATLAB kriging toolbox, version 2.0.* Technical Report IMM-TR-2002-12. Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.
Marsden, A.L., *et al.*, 2004. Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering*, 5 (2), 235–262.
MathWorks®, 2009a. *MATLAB genetic algorithm and direct search toolbox: user's guide, version 2.* Natick, MA: The Mathworks, Inc.
MathWorks®, 2009b. *MATLAB optimization toolbox: user's guide, version 4.* Natick, MA: The Mathworks, Inc.
Myers, R.H. and Montgomery, D.C., 1995. *Response surface methodology: process and product optimization using designed experiments*. New York: Wiley.
Pintér, J.D., 1996. *Global optimization in action*. Dordrecht, The Netherlands: Kluwer.
Powell, M.J.D., 1992. The theory of radial basis function approximation in 1990. *In*: W. Light, ed. *Advances in numerical analysis*. Vol. 2. *Wavelets, subdivision algorithms and radial basis functions*. Oxford, UK: Oxford University Press, 105–210.
Powell, M.J.D., 2006. The NEWUOA software for unconstrained optimization without derivatives. *In*: G. Di Pillo and M. Roma, eds. *Large-scale nonlinear optimization*. New York, NY: Springer-Verlag, 255–297.
Ray, T. and Smith, W., 2006. A surrogate assisted parallel multiobjective evolutionary algorithm for robust engineering design. *Engineering Optimization*, 38 (8), 997–1011.
Regis, R.G., 2011. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38 (5), 837–853.
Regis, R.G. and Shoemaker, C.A., 2004. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8 (5), 490–505.
Regis, R.G. and Shoemaker, C.A., 2007a. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19 (4), 497–509.
Regis, R.G. and Shoemaker, C.A., 2007b. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37 (1), 113–135.
Regis, R.G. and Shoemaker, C.A., 2009. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing*, 21 (3), 411–426.
Sacks, J., *et al.*, 1989. Design and analysis of computer experiments. *Statistical Science*, 4 (4), 409–435.
Sasena, M.J., Papalambros, P., and Goovaerts, P., 2002. Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34 (3), 263–278.
Schoen, F., 1993. A wide class of test functions for global optimization. *Journal of Global Optimization*, 3 (2), 133–137.
Shahrokhi, A. and Jahangirian, A., 2010. A surrogate assisted evolutionary optimization method with application to the transonic airfoil design. *Engineering Optimization*, 42 (6), 497–515.
Shan, S. and Wang, G., 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41 (2), 219–241.
Shoemaker, C.A., Regis, R.G., and Fleming, R.C., 2007. Watershed calibration using multistart local optimization and evolutionary optimization with radial basis function approximation. *Hydrological Sciences Journal*, 52 (3), 450–465.
Simpson, T.W., *et al.*, 2001. Kriging metamodels for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal*, 39 (12), 2233–2241.
Tolson, B.A. and Shoemaker, C.A., 2007. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resources Research*, 43, W01413. doi:10.1029/2005WR004723
Torczon, V., 1997. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7 (1), 1–25.
Törn, A. and Žilinskas, A., 1989. *Global optimization*. Lecture notes in computer science 350. Berlin: Springer-Verlag.
Viana, F.A.C., 2010. *SURROGATES Toolbox User's Guide, Version 2.1*. Available from: http://sites.google.com/site/felipeacviana/surrogatestoolbox [accessed 16 June 2011].
Viana, F.A.C., Haftka, R.T., and Watson, L.T., 2010. Why not run the efficient global optimization algorithm with multiple surrogates? *51st AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference*, 12–15 April 2010, Orlando, FL. Reston, VA: AIAA, 2010–3090.
Villemonteix, J., Vazquez, E., and Walter, E., 2009. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44 (4), 509–534.
Wang, G. and Simpson, T., 2004. Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization. *Engineering Optimization*, 36 (3), 313–335.
Wang, G., Dong, Z., and Aitchison, P., 2001. Adaptive response surface method – a global optimization scheme for approximation-based design problems. *Engineering Optimization*, 33 (6), 707–733.
Won, K.S. and Ray, T., 2005. A framework for design optimization using surrogates. *Engineering Optimization*, 37 (7), 685–703.

Yahyaie, F. and Filizadeh, S., 2011. A surrogate-model based multi-modal optimization algorithm. *Engineering Optimization*, 43 (7), 779–799.

Ye, K.Q., Li, W., and Sudjianto, A., 2000. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference*, 90 (1), 145–159.

Žilinskas, A., 1985. Axiomatic characterization of a global optimization algorithm and investigation of its search strategies. *Operations Research Letters*, 4 (1), 35–39.