Journal of Computational Science 5 (2014) 12–23

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Particle swarm with radial basis function surrogates for expensive black-box optimization

Rommel G. Regis*

Department of Mathematics, Saint Joseph's University, Philadelphia, PA 19131, United States

ARTICLE INFO

Article history:
Received 10 January 2013
Received in revised form 19 July 2013
Accepted 20 July 2013
Available online 14 August 2013

Keywords:
Particle swarm optimization
Surrogate model
Radial basis function
Expensive function
Groundwater bioremediation
Watershed model calibration

ABSTRACT

This paper develops the *OPUS (Optimization by Particle swarm Using Surrogates)* framework for expensive black-box optimization. In each iteration, OPUS considers multiple trial positions for each particle in the swarm and uses a surrogate model to identify the most promising trial position. Moreover, the current overall best position is refined by finding the global minimum of the surrogate in the neighborhood of that position. OPUS is implemented using an RBF surrogate and the resulting OPUS-RBF algorithm is applied to a 36-D groundwater bioremediation problem, a 14-D watershed calibration problem, and ten mostly 30-D test problems. OPUS-RBF is compared with a standard PSO, CMA-ES, two other surrogate-assisted PSO algorithms, and an RBF-assisted evolution strategy. The numerical results suggest that OPUS-RBF is promising for expensive black-box optimization.

1. Introduction

Particle swarm optimization (PSO) is among the most popular black-box optimization methods. It is a numerical algorithm that simulates the behavior of a swarm of agents or particles, such as a flock of birds or a school of fish, as they collectively attempt to find some optimal state. PSO has been shown to be effective on a wide variety of practical optimization applications and this paper extends its usefulness to computationally expensive black-box optimization problems.

The focus of this paper is on the bound constrained, computationally expensive black-box optimization problem:

$$\begin{aligned} &\min \quad f(x) \\ &\text{s.t.} \\ &x \in \mathbb{R}^d, \quad a \le x \le b, \end{aligned} \tag{1}$$

where f is a black-box function whose value at an input is typically obtained from a computationally expensive but deterministic computer simulation and $a, b \in \mathbb{R}^d$ are the bounds on the input variables. Here, *computationally expensive* means that evaluating the objective function is time consuming and completely dominates the cost of the entire optimization process. Moreover, *black-box* means that analytical expressions for the objective or constraint functions are not available. In addition, in many practical applications, the derivatives of the objective and constraint functions are not explicitly available.

When objective or constraint functions are computationally expensive, a standard PSO approach might not be suitable because it uses many function evaluations. This paper develops a new surrogate-based approach, called *OPUS (Optimization by Particle swarm Using Surrogates)*, that can accelerate the convergence of PSO and reduce the number of function evaluations in the optimization process. The main idea in OPUS is to consider multiple trial velocities resulting in multiple trial positions for each particle that are then screened by a surrogate model to identify which trial position is most promising for each particle. Moreover, the current overall best position visited by any particle is refined by finding the global minimum of the surrogate within the neighborhood of that position. To maximize the use of information from previously evaluated positions, the surrogate model is always updated at the end of each iteration. In the numerical experiments, the proposed OPUS method is implemented using a cubic radial basis function (RBF) surrogate augmented by a linear polynomial tail. The resulting OPUS-RBF algorithm is applied to a 36-D groundwater bioremediation problem, a 14-D watershed calibration problem, and ten test problems with at least 30 dimensions. It is compared with alternative optimization methods, including a standard PSO method, an evolution strategy with covariance matrix adaptation (CMA-ES) [10], two other surrogate-assisted PSO algorithms developed by Parno et al. [19] and by Praveen and Duvigneau [22], and an RBF-assisted evolution strategy [25]. The results show that OPUS-RBF either outperforms or compares favorably with these alternative methods on the problems considered.

* Tel.: +1 6106601514.
E-mail address: rregis@sju.edu

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.07.004

The rest of this paper is organized as follows. Section 2 provides a brief review of literature. Section 3 gives a brief description of a standard particle swarm optimization algorithm while Section 4 provides a detailed description of a surrogate-assisted particle swarm optimization method. Sections 5 and 6 cover the numerical experiments and results and discussion. Finally, Section 7 provides a summary of the paper.

2. Review of literature

Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart [15] and it is now among the most popular methods for black-box optimization. Many variants of PSO have been proposed and applied to a wide variety of problems. Some of these variants can be found in the surveys by Poli et al. [20] or Banks et al. [2,3]. More recently, Ismail and Engelbrecht [14] proposed a self-adaptive PSO where the control parameters of the algorithm are optimized in a secondary swarm. Qu et al. [23] integrated a local search technique with a PSO to facilitate the search for multiple global or local optima. Wei et al. [30] developed an improved PSO where part of the population remains at a stable level while the rest uses harmony search. Akbari and Ziarati [1] proposed a rank based PSO where some of the best particles are used to update the position of a candidate particle. In addition, PSO has been modified to handle constraints (e.g., [12,13,18,28]).

Like most metaheuristics, PSO typically uses a large number of function evaluations so a standard implementation might not be suitable when function evaluations are computationally expensive. To deal with this issue, a natural approach is to use a surrogate model or metamodel to reduce the number of function evaluations required to obtain a relatively good solution in many practical problems. While surrogates have been used to assist evolutionary algorithms, they have been rarely used to assist PSO methods. For example, [22] used a radial basis function (RBF) metamodel to reduce the cost of PSO in two 20-D aerodynamic shape optimization problems. Parno et al. [19] used a DACE surrogate to improve the efficiency of PSO for simulation-based problems and applied it to a 6-D groundwater management problem and to several 2-D test problems. Tang et al. [27] used a hybrid surrogate model consisting of a quadratic polynomial and an RBF model to develop a surrogate-based PSO method and applied it to mostly low-dimensional test problems and engineering design problems. The surrogate-assisted PSO approach proposed in this paper is very different from these other approaches and it is applied to higher dimensional problems.

3. A standard particle swarm optimization method

Below is a standard PSO algorithm that mostly follows the notation in [29]. In every iteration of PSO, each particle is represented as a point in the search space with an associated velocity vector. This velocity vector is updated by using a linear combination of the velocity in the previous iteration, the direction of the best position so far of the particle, and the direction of the best position so far of any of the particles. The weights for the last two components of this linear combination are allowed to vary randomly from iteration to allow for the exploration of the search space.

In the algorithm below, $x^{(i)}(t)$ represents the position of particle i, where $i = 1, \ldots, s$, during time t and $x_j^{(i)}(t)$ represents the jth coordinate or component of $x^{(i)}(t)$, where $j = 1, \ldots, d$. That is, $x^{(i)}(t) = (x_1^{(i)}(t), x_2^{(i)}(t), \ldots, x_d^{(i)}(t))$. Moreover, $y^{(i)}(t)$ is the best position visited by particle i while $\hat{y}(t)$ is the best position visited by any of the particles up to time t. Here, only discrete time periods $t = 0, 1, 2, \ldots$ are considered.

Particle swarm optimization (PSO):

Inputs:

(1) Function to minimize: $f : [a, b] \to \mathbb{R}$, where $[a, b] \subseteq \mathbb{R}^d$
(2) Population size: s
(3) Initial swarm positions: $x^{(1)}(0), \ldots, x^{(s)}(0) \in [a, b]$
(4) Inertial weighting factor for each iteration: $i(t)$, where t is the iteration number
(5) Cognition parameter: μ
(6) Social parameter: ν
(7) Minimum and maximum velocities: v_{min} and v_{max}
(8) Maximum iterations: T_{max}

Output: The best point found by the algorithm.

Step 1. (**Evaluate initial swarm positions**) For each $i = 1, \ldots, s$, calculate $f(x^{(i)}(0))$.

Step 2. (**Determine initial particle velocities**)
For $i = 1, \ldots, s$,
(2a) Generate $u^{(i)}$ uniformly at random on $[a, b]$.
(2b) Set $v^{(i)}(0) = \frac{1}{2}(u^{(i)} - x^{(i)}(0))$.
End for.

Step 3. (**Initialize best position for each particle and overall best**) Set $y^{(i)}(0) = x^{(i)}(0)$, $i = 1, \ldots, s$, and let $\hat{y}(0)$ be the point in $\{y^{(1)}(0), \ldots, y^{(s)}(0)\}$ with the minimum value of f. (Choose the point with the smallest superscript in case of ties.) Set the iteration counter $t = 0$.

Step 4. (**Update particle velocities**)
For $i = 1, \ldots, s$
For $j = 1, \ldots, d$
$v_j^{(i)}(t+1) = i(t)v_j^{(i)}(t) + \mu\omega_{1,j}^{(i)}(t)(y_j^{(i)}(t) - x_j^{(i)}(t)) + \nu\omega_{2,j}^{(i)}(t)(\hat{y}_j(t) - x_j^{(i)}(t))$,
where $\omega_{1,j}^{(i)}(t), \omega_{2,j}^{(i)}(t) \sim U[0, 1]$
$v_j^{(i)}(t+1) = \min(\max(v_{min}, v_j^{(i)}(t+1)), v_{max})$
End for.
End for.

Step 5. (**Update particle positions**)
For $i = 1, \ldots, s$
$x^{(i)}(t+1) = x^{(i)}(t) + v^{(i)}(t+1)$
$x^{(i)}(t+1) = \text{proj}_{[a,b]}(x^{(i)}(t+1)) = \min(\max(a, x^{(i)}(t+1)), b)$, where the min and max are calculated componentwise.
End for.

Step 6. (**Evaluate swarm positions**) For each $i = 1, \ldots, s$, calculate $f(x^{(i)}(t+1))$.

Step 7. (**Update best position for each particle and overall best**) Set $\hat{y}(t+1) = \hat{y}(t)$.
For $i = 1, \ldots, s$
If $f(x^{(i)}(t+1)) < f(y^{(i)}(t+1))$, then
Set $y^{(i)}(t+1) = x^{(i)}(t+1)$.
If $f(y^{(i)}(t+1)) < f(\hat{y}(t+1))$, then $\hat{y}(t+1) = y^{(i)}(t+1)$.
Else
Set $y^{(i)}(t+1) = y^{(i)}(t)$.
End if.
End for.

Step 8. (**Check termination condition**) If $t < T_{max}$, then reset $t = t + 1$ and go back to Step 4. Else, STOP.

The above PSO algorithm begins with Step 1 where the objective function is evaluated at the initial swarm positions, which are typically chosen uniformly at random in the search space $[a, b]$. Step 2 determines the initial particle velocities using the *half-diff method* recommended in [11]. Step 3 initializes the best position for each particle to be the initial position and then it initializes the overall best position for any particle. Step 4 updates the particle velocities while ensuring that they are within the minimum and maximum velocities. Step 5 moves the particles into their new positions. To ensure that the particles remain within the search space, they are projected into the bounds if they leave the search space. In Step

6, the objective function is evaluated at the new positions of the particles. Then, in Step 7, the best position for each particle and the overall best position are all updated. Finally, in Step 8, the algorithm goes back to Step 4 if the termination condition has not been met. Otherwise, the algorithm stops and the overall best position of any particle is returned.

4. A new surrogate-assisted particle swarm optimization method

4.1. Overview of the proposed method

Surrogate models or metamodels have been used to assist particle swarm. In [22], an RBF metamodel is used in conjunction with PSO to reduce the cost of aerodynamic shape optimization problems. The basic idea in [22] is to use the expensive function to evaluate particle positions for a specified number of iterations at the beginning. Then, in subsequent iterations, an RBF model is used to pre-screen particles, identify a fraction of the particles that are most promising, and evaluate the expensive objective function at the positions of these particles only. Two main pre-screening strategies are used. In one strategy, the particles are ranked according to their predicted objective function values according to the metamodel and then a specified fraction of the best particles are selected. In the other more adaptive strategy, a particle is selected only if the metamodel predicts that the current position of this particle will improve its best position. Furthermore, the best position visited by a particle and the overall best position are updated using only the exact function values.

A different surrogate-assisted PSO was proposed by Parno et al. [19] where a DACE surrogate was used with PSO and applied to a 6-D groundwater management problem and to several 2-D test problems. In [19], an approximate global minimizer of the DACE surrogate is used in place of the overall best position visited by any of the particles if the actual objective function value at the approximate minimizer of the surrogate is better than the overall best function value obtained by the particles. A similar approach was used by Tang et al. [27] but using a hybrid surrogate model consisting of a quadratic polynomial and an RBF model instead of a DACE surrogate. Consequently, the method in [27] requires $(d+1)(d+2)/2$ data points before a surrogate model is constructed.

This paper develops a new surrogate-assisted PSO framework called *OPUS (Optimization by Particle swarm Using Surrogates)* that is different from the approaches mentioned above. The basic idea is to consider multiple trial velocities resulting in multiple trial positions for each particle in each iteration. The surrogate is updated in every iteration and, for each particle, it is used to identify the most promising among the trial positions for this particle. Each particle is then moved only to the most promising trial position and the expensive function is then evaluated at these new positions. The idea of selecting only the most promising trial position from multiple trial positions for each particle is similar to the one used in the surrogate-assisted evolutionary programming algorithm by Regis [24] where only the most promising offspring is selected from multiple trial offspring from each parent solution. In addition, the algorithm refines the current overall best position by finding a global minimizer of the updated surrogate within a relatively small box around that position. This minimizer of the surrogate is referred to as a *local refinement point* if it is not too close to a previously evaluated point. The expensive function is then evaluated at this point. Hence, the proposed OPUS method is essentially an accelerated PSO with the surrogate guiding where each particle should go and helping to refine the current overall best position.

The proposed OPUS framework is expected to perform better than the other surrogate-assisted approaches because it exploits the surrogate better and more thoroughly when selecting the new positions of the particles in the swarm (i.e., the function evaluation points) and this leads to faster improvement in the current best solution. In particular, the proposed OPUS approach considers many more trial positions that are pre-screened by the surrogate than the algorithm by Praveen and Duvigneau [22] and it performs the same number of function evaluations (equal to the population size s) in every iteration. In [22], the expensive function is only evaluated on a subset of the new particles, which uses $<s$ function evaluations. By using the surrogate to pre-screen a much larger set of trial positions, the chance of obtaining a better point is improved.

Moreover, unlike the approaches in [19] and [27], the proposed OPUS framework does not attempt to find a global minimizer of the surrogate over the entire search space. Instead, OPUS finds the global minimizer of the surrogate in a neighborhood around the current overall best position in order to improve that position. For high-dimensional problems, the surrogate is not expected to provide an accurate global approximation of the black-box function especially when the surface of that function is complex and relatively few data points are available. Because of this, the global minimizer of the surrogate over the entire search space might be very far from the true global minimizer of the function when the dimension is high. However, the surrogate can point to potential directions of improvement especially in the neighborhood of previously visited points. By limiting the global minimization search on the surrogate over a smaller region in the neighborhood of the current overall best position, as is done by model-based trust-region algorithms, the chances of getting an improvement is also increased.

In addition, the surrogate used in this study (a cubic RBF with a linear polynomial tail) only requires $d+1$ points to fit instead of the $(d+1)(d+2)/2$ points required by the hybrid surrogate in [27]. This enables the algorithm to make progress much earlier in the search especially on high dimensional problems. Finally, OPUS is applied to problems with much higher dimensions than the ones used in all these previous surrogate-based approaches.

4.2. Algorithmic framework

Below is a description of the OPUS framework for PSO using surrogates. The notation is the same as in Section 3 with a few additions. In this framework, $v^{(i,\ell)}(t)$ and $x^{(i,\ell)}(t)$ are the ℓth trial velocity and ℓth trial position, respectively, for particle i during time t.

Optimization by particle swarm using surrogates (OPUS):

Inputs:

(1) Function to minimize: $f : [a, b] \to \mathbb{R}$, where $[a, b] \subseteq \mathbb{R}^d$
(2) Population size: s
(3) Space-filling design: $\{z^{(1)}, \ldots, z^{(k)}\} \subseteq [a, b]$ with $k \geq s$ (e.g., a Latin hypercube design)
(4) Cognition parameter: μ
(5) Social parameter: ν
(6) Minimum and maximum velocities: v_{min} and v_{max}
(7) Number of trial positions for each particle: r
(8) Surrogate model (e.g., cubic RBF with linear polynomial tail)
(9) Side length of the box for local refinement: ξ
(10) Optimization solver for local refinement (e.g., multistart implementation of Matlab's Fmincon solver)
(11) Threshold distance for determining if one point is too close to another point: δ
(12) Maximum iterations: T_{max}

Output: The best point found by the algorithm.

Step 1. (**Evaluate points of space-filling design**) For $i = 1, \ldots, k$, calculate $f(z^{(i)})$.

Step 2. (**Determine initial swarm positions**) Choose initial swarm positions $x^{(1)}(0), \ldots, x^{(s)}(0)$ to be the s points from $\{z^{(1)}, \ldots, z^{(k)}\}$ with the best function values.

Step 3. (**Determine initial particle velocities**)
For $i = 1, \ldots, s$,
(3a) Generate $u^{(i)}$ uniformly at random on $[a, b]$.
(3b) Set $v^{(i)}(0) = \frac{1}{2}(u^{(i)} - x^{(i)}(0))$.
End for.

Step 4. (**Initialize best position for each particle and overall best**) Set $y^{(i)}(0) = x^{(i)}(0)$, $i = 1, \ldots, s$, and let $\hat{y}(0)$ be the point in $\{y^{(1)}(0), \ldots, y^{(s)}(0)\}$ with the minimum value of f. (Choose the point with the smallest superscript in case of ties.) Also, set $\mathcal{E}_0 = \emptyset$ and $t = 0$.

Step 5. (**Fit surrogate**) Use all previously evaluated points $\left(\bigcup_{j=0}^{t}\bigcup_{i=1}^{s}\{x^{(i)}(j)\}\right)\bigcup\mathcal{E}_t$ to build a surrogate model $s_t(x)$ for the objective function. (See Section 4.3)

Step 6. (**Determine new particle positions**)
For $i = 1, \ldots, s$
6(a) For ℓ=1, …, r
(i) (Generate trial velocities) For $j = 1, \ldots, d$

$$v_j^{(i,\ell)}(t+1) = i(t)v_j^{(i)}(t) + \mu\omega_{1,j}^{(i,\ell)}(t)(y_j^{(i)}(t) - x_j^{(i)}(t)) + \nu\omega_{2,j}^{(i,\ell)}(t)(\hat{y}_j(t) - x_j^{(i)}(t)),$$

where $\omega_{1,j}^{(i,\ell)}(t)$, $\omega_{2,j}^{(i,\ell)}(t) \sim U[0, 1]$

$$v_j^{(i,\ell)}(t+1) = \min(\max(v_{min}, v_j^{(i,\ell)}(t+1)), v_{max})$$

End for.
(ii) (*Generate trial positions*) $x^{(i,\ell)}(t+1) = x^{(i)}(t) + v^{(i,\ell)}(t+1)$
(iii) (*Project trial positions onto bounds*)
$x^{(i,\ell)}(t+1) = \text{proj}_{[a,b]}(x^{(i,\ell)}(t+1))$
End for.
6(b) (*Select promising position using surrogate*) Use the surrogate model $s_t(x)$ to select the most promising trial position for particle i among the points $\{x^{(i,1)}(t+1), x^{(i,2)}(t+1), \ldots, x^{(i,r)}(t+1)\}$. Let $x^{(i)}(t+1)$ be the most promising trial position and let $v^{(i)}(t+1)$ be the associated trial velocity.
End.

Step 7. (**Evaluate swarm positions**) For each $i = 1, \ldots, s$, calculate $f(x^{(i)}(t+1))$.

Step 8. (**Update best position for each particle and overall best**) Set $\hat{y}(t+1) = \hat{y}(t)$.
For $i = 1, \ldots, s$
If $f(x^{(i)}(t+1)) < f(y^{(i)}(t+1))$, then
Set $y^{(i)}(t+1) = x^{(i)}(t+1)$.
If $f(y^{(i)}(t+1)) < f(\hat{y}(t+1))$, then $\hat{y}(t+1) = y^{(i)}(t+1)$.
Else
Set $y^{(i)}(t+1) = y^{(i)}(t)$.
End if.
End for.

Step 9. (**Refit surrogate**) Use all previously evaluated points $\left(\bigcup_{j=0}^{t+1}\bigcup_{i=1}^{s}\{x^{(i)}(j)\}\right)\bigcup\mathcal{E}_t$ to build a new surrogate model $\tilde{s}_t(x)$ for the objective function. (See Section 4.3)

Step 10. (**Perform local refinement of overall best position**) Use the optimization solver for local refinement to find a global minimizer x_{t+1}^* of the surrogate $\tilde{s}_t(x)$ within the box $[\hat{y}(t+1) - \xi/2, \hat{y}(t+1) + \xi/2] \bigcap [a, b]$.

Step 11. (**Determine if minimizer of surrogate is far from previous points**)
If x_{t+1}^* is at least of distance δ from all previously evaluated points, then do
(11a) (*Evaluate minimizer of surrogate*) Calculate $f(x_{t+1}^*)$.
(11b) (*Update overall best position*) If $f(x_{t+1}^*) < f(\hat{y}(t+1))$, then $\hat{y}(t+1) = x_{t+1}^*$.
(11c) (*Update local refinement points*) Set $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{x_{t+1}^*\}$.
Else
(11d) (*Maintain local refinement points*) Set $\mathcal{E}_{t+1} = \mathcal{E}_t$.
End if.

Step 12. (**Check termination condition**) If $t < T_{max}$, then reset $t = t + 1$ and go back to Step 5. Else, STOP.

The OPUS framework begins by evaluating the points of the given space-filling design (Step 1). Then the initial swarm positions are selected to be the s points of the space-filling design with the best function values (Step 2). Step 3 determines the initial particle velocities using the half-diff method [11]. In Step 4, the best position for each particle is initially set to the starting position of the particle. Moreover, the overall best position is initially set to be the best among the starting positions in terms of objective function value. In addition, the set of local refinement points $\mathcal{E}_0$ is initialized to be the empty set.

Next, Step 5 fits a surrogate model $s_t(x)$ for the objective function using all available data points from all positions visited by any particle (given by $\bigcup_{j=0}^{t}\bigcup_{i=1}^{s}\{x^{(i)}(j)\}$) and all local refinement points (given by $\mathcal{E}_t$). Then Step 6 determines the new position of each particle by first considering multiple trial velocities within the velocity limits for that particle (Step 6(a)), generating the corresponding trial positions (Step 6(b)), projecting the trial positions into the bounds in case they leave the search space (Step 6(c)), and then using the surrogate to select the most promising among the trial positions and then choosing this to be the new position of the given particle (Step 6(d)). Once the new positions for the particles have been determined, the objective function is then evaluated at these positions (Step 7). Again, the best position for each particle and the overall best position by any particle are updated (Step 8).

The algorithm then builds a new surrogate model $\tilde{s}_t(x)$ that incorporates the newly evaluated points (Step 9) in preparation for local refinement of the overall best point (Step 10). In this local refinement step, an optimization solver is used to find a global minimizer x_{t+1}^* of the surrogate $\tilde{s}_t(x)$ within a box of side length ξ centered at the current overall best point $\hat{y}(t+1)$. In the numerical implementation, it is enough to find an approximate global minimizer of the surrogate within this box. If the global minimizer of the surrogate within the box (i.e., the local refinement point) is not too close to any previously evaluated point (at least distance δ from all previously evaluated points), then the objective function is evaluated at this local refinement point (Step 11(a)), the overall best position is updated (Step 11(b)), and the set of local refinement points is also updated (Step 11(c)). Finally, the algorithm goes back to Step 5 if the termination condition has not been satisfied. Otherwise, the algorithm stops and returns the overall best position found (Step 12).

4.3. A radial basis function model

The proposed method is implemented using the radial basis function (RBF) surrogate model described in [5,21] and the resulting algorithm is referred to as OPUS-RBF. This RBF model has been used in various surrogate-based optimization methods (e.g., [8,25]). The procedure for fitting this model differs from the one typically used for RBF networks in machine learning.

Given n distinct points $u^{(1)}, \ldots, u^{(n)} \in \mathbb{R}^d$ and the function values $f(u^{(1)}), \ldots, f(u^{(n)})$, the proposed OPUS-RBF uses an interpolant of the form

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

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 possible choices for ϕ include the thin plate spline, multiquadric and Gaussian forms. We use a cubic RBF model because of previous success with this model (e.g., [24,25]) Moreover, previous

work (e.g., [31]) suggest that cubic RBFs might be more suitable than Gaussian RBFs for surrogate-based optimization.

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

$$\begin{pmatrix} \Phi & P \\ P^T & 0_{(d+1)\times(d+1)} \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0_{d+1} \end{pmatrix}, \tag{2}$$

where $0_{(d+1)\times(d+1)} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a matrix of zeros, $F = (f(u^{(1)}), \ldots, f(u^{(n)}))^T$, $0_{d+1} \in \mathbb{R}^{d+1}$ is a vector of zeros, $\lambda = (\lambda_1, \ldots, \lambda_n)^T \in \mathbb{R}^n$ and $c = (c_1, \ldots, c_{d+1})^T \in \mathbb{R}^{d+1}$ consists of the coefficients for the linear polynomial $p(x)$. The coefficient matrix in (2) is invertible if and only if $\text{rank}(P) = d + 1$ [21]. This condition is equivalent to having a subset of $d + 1$ affinely independent points among the points $\{u^{(1)}, \ldots, u^{(n)}\}$. (A set of points $y^{(0)}, y^{(1)}, \ldots, y^{(k)}$ in $\mathbb{R}^d$ are *affinely independent* if the displacements $y^{(1)} - y^{(0)}, y^{(2)} - y^{(0)}, \ldots, y^{(k)} - y^{(0)}$ are linearly independent.)

5. Numerical experiments

5.1. Management of groundwater bioremediation

The proposed OPUS-RBF algorithm is compared with alternative methods on a 36-D groundwater bioremediation problem [32]. Groundwater bioremediation is the process of promoting the growth of soil bacteria that can transform groundwater contaminants into harmless substances. This study uses the optimization formulation in [32], which considers a setup that pumps oxygenated water into the groundwater by means of injection wells and uses monitoring wells to measure contaminant concentrations at specific locations. The optimization formulation uses a 2-D finite element simulation model that describes groundwater flow and changes in the concentrations of the contaminant, oxygen and biomass. The entire planning horizon is divided into management periods and the goal is to determine the pumping rates for each injection well at the beginning of each management period that minimizes the total pumping cost subject to the constraints that the contaminant concentrations at the monitoring wells are below a certain threshold during specified time periods. In [32], the constraints are incorporated into the total pumping cost objective function via a penalty term, resulting in a bound-constrained global optimization problem.

More precisely, this study uses the hypothetical contaminated aquifer described in [32] whose characteristics are symmetric about a horizontal axis. The aquifer is discretized using a two-dimensional finite element mesh with 18 nodes in the horizontal direction and 9 nodes in the vertical direction. There are 6 injection wells and 84 monitoring wells that are also symmetrically arranged so pumping decisions are only needed on the 3 injection wells on one side of the axis of symmetry. There are 12 management periods (where each management period is a month), giving $12 \times 3 = 36$ decision variables for the optimization problem. The decision variables are scaled so that the search space is $[0, 1]^{36}$. This groundwater bioremediation problem is referred to as GWB36.

This groundwater bioremediation model uses a relatively coarse grid so its simulation time is only about 0.1 s on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop. However, the above optimization problem is representative of more complex groundwater bioremediation problems, whose simulation times can range from a few minutes to many hours depending on the complexity of the model and the size of the region being modeled.

5.2. Calibration of a watershed simulation model

OPUS-RBF and alternative optimization methods are also 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 in upstate New York. This problem is described in [26]. The Cannonsville reservoir supplies drinking water to New York City and the model was created to assess the impact of changes in management practices on phosphorous loads from the watershed into the reservoir. Phosphorous promotes the growth of algae, which can clog the water supply. If the phosphorous levels become too high, New York City would either have to abandon the water supply or build a filtration plant that costs billions of dollars. Hence, it would be better to control the phosphorous at the watershed level than to build a plant.

Part of the above analysis requires modeling the flow of water in response to measured rainfall. This study considers the following global optimization problem where the goal is to calibrate the watershed model for flow against real measured flow data:

$$\min \quad SSE(x) = \sum_{t=1}^{T} (Q_t^{meas} - Q_t^{sim}(x))^2, \quad \text{s.t.} \quad 0 \le x \le 1.$$

Here, x is the vector of $d = 14$ model parameters that have been scaled so that their values are between 0 and 1, 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 [26]. Note that this problem is a nonlinear least squares problem with 1096 residual functions. This 14-D optimization problem is referred to as TownBrook14.

The simulation time for this example is only about 1.3 s on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop because it only covers a relatively small region inside the Cannonsville Reservoir. However, it is also representative of more complex watershed calibration problems whose simulation times can range from a few seconds to several minutes depending on the size of the modeled region and the length of time in the calibration period.

5.3. Test problems

OPUS-RBF is further compared with alternative methods on ten test functions with at least 30 dimensions. Six of these test functions are the 30-D versions of the Ackley, Rastrigin, Griewank, Keane, Levy and Michalewicz functions that are well-known in the engineering optimization community. The remaining four test functions are the 30-D versions of the Extended Rosenbrock, Trigonometric and Broyden Tridiagonal functions and the 32-D version of the Extended Powell Singular function. These functions are taken from the Moré et al. [16] test set that is well-known in the mathematical programming community. The dimension of the Extended Powell Singular is different from the others because it must be multiple of 4. The first six test problems each have a large number of local minima while the last four each have only one local minimum. Note that most of the surrogate-based optimization methods in the literature have only been applied to problems with dimensions $d < 15$. Hence, the test problems in this study are considered high-dimensional in this area. Table 1 summarizes the characteristics of the test problems in this study.

The above test problems are not really computationally expensive to evaluate and the different algorithms are compared by pretending that these functions are expensive. This is accomplished below by keeping track of the best function values obtained by the different algorithms as the number of function evaluations increase. The relative performance of algorithms on the test

Table 1
Test problems for the computational experiments.

Test function	Dimensions	Domain	Global min value
Ackley	30	$[-15, 20]^d$	$-20-e$
Rastrigin	30	$[-4, 5]^d$	$-d$
Griewank	30	$[-500, 700]^d$	0
Keane	30	$[1, 10]^d$	<−0.39
Levy	30	$[-5, 5]^{30}$	<−11
Michalewicz	30	$[0, \pi]^{30}$	<−23
Extended Rosenbrock	30	$[-2, 2]^d$	0
Extended Powell Singular	32	$[-1, 3]^d$	0
Trigonometric	30	$[-1, 3]^d$	0
Broyden Tridiagonal	30	$[-1, 1]^d$	0

problems are expected to be similar to their relative performance on truly expensive functions that have the same general shape as these test problems.

5.4. Alternative methods and experimental setup

All computational runs are carried out in Matlab 7.11 on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop machine. The OPUS-RBF method is compared with various alternative methods on the groundwater bioremediation problem (GWB36), the watershed calibration problem (TownBrook14) and the above test problems. Each method is run for 30 trials on all problems. The alternatives include a standard particle swarm method as described in Section 3, the well-known evolution strategy with covariance matrix adaptation (CMA-ES) ([9,10]), an RBF-assisted evolution strategy (ESGRBF) ([25,26]), and two other RBF-assisted PSO algorithms that are described next.

One of the alternative RBF-assisted PSO methods is an implementation of the surrogate-assisted PSO (sPSO) algorithm by Parno et al. [19] that uses the same RBF surrogate in OPUS-RBF. This algorithm is referred to as sPSO-RBF. The original sPSO algorithm in [19] uses the DACE kriging surrogate model but there is no publicly available code for it. The implementation of sPSO in this study uses RBF because the problems here have much higher dimensions than those in [19] and it is well-known that kriging becomes computationally and memory intensive in high dimensions. Moreover, to properly compare the strategies used in the OPUS and sPSO algorithms, the same type of surrogate model is used.

Another alternative RBF-assisted PSO method is an implementation of a variant of the metamodel-assisted PSO with Inexact Pre-Evaluation (IPE) by Praveen and Duvigneau [22] that also uses the same RBF surrogate in OPUS-RBF. This algorithm is referred to as mPSO-IPE-RBF. The original implementations of this algorithm in [22] use a Gaussian RBF model, but again, there are no publicly available codes. In this study, the algorithm in [22] is implemented using the same cubic RBF model with a linear tail used by OPUS-RBF to properly compare the two strategies. Moreover, the mPSO-IPE-RBF implementation in this study uses a global RBF metamodel in the IPE phase instead of the local RBF metamodels in [22] because it is simpler and previous work (e.g., [26]) showed little difference in algorithm performance when using a global RBF model or a local RBF model constructed using only points in the vicinity of a point to be estimated (as long as there is enough of these points). Moreover, Praveen and Duvigneau [22] found little difference in the performance of the algorithm when varying the number of points used to construct the local RBF metamodel. This is because global RBF models tend to produce good local fits so that there is little difference in RBF approximation at a point whether one uses the entire global RBF model or only a local RBF model.

Surrogate-assisted methods (such as OPUS-RBF, sPSO-RBF, mPSO-IPE-RBF, ESGRBF) require a space-filling experimental design to fit the initial surrogate model. In this study, an affinely independent Latin Hypercube Design (LHD) with $d+1$ points is used to initialize the surrogate-assisted methods. The initial population of particles is chosen as a subset of the LHD with the best objective function values. If there are not enough LHD points to form the initial population, it is augmented by uniform random points over the search space. The LHD is not needed by standard PSO. However, preliminary experiments on the test problems suggest that the performance of PSO when initialized by uniform random points over the search space is similar to its performance when initialized by an LHD. To ensure fair comparison among the different methods, all methods use the same LHD in a given trial. Hence, the standard PSO implementation in this study is referred to as PSO-LHD.

In this study, the population size for OPUS-RBF, sPSO-RBF and PSO-LHD is set to $s=20$, which is a commonly used value (e.g., [13,29]). In some PSO implementations, the inertial weighting factor $i(t)$ varies with the iterations from a high value (close to 1) to a low value. Here, it is fixed at $i(t)=0.72984$ and the cognition and social parameters are set to $\mu=\nu=1.496172$ as recommended in [4]. The minimum and maximum values for the components of the velocity vectors are set to $-\min_{1\le i\le d}(b_i-a_i)/4$ and $\min_{1\le i\le d}(b_i-a_i)/4$, respectively. For OPUS-RBF, the number of trial positions for each particle is $r=10d$, the side length of the box for local refinement is $\xi=0.1\min_{1\le i\le d}(b_i-a_i)$, and the threshold distance for determining when a point is too close to another is $\delta=0.0005\sqrt{d}\min_{1\le i\le d}(b_i-a_i)$.

For mPSO-IPE-RBF, the population size is set to $s=120$ of which the best 10% of the particle positions in terms of metamodel value are evaluated by the expensive function as was done in [22]. The global minimization of the RBF surrogates in OPUS-RBF and sPSO-RBF are both carried out using a multistart implementation of the Fmincon solver from the Matlab Optimization toolbox where the gradients of the RBF model are supplied to the solver. In addition, the maximum number of iterations is set so that the computational budget for each problem is 500 objective function evaluations.

The parameters used for ESGRBF ([25,26]) are $\mu=8$ parents and $\lambda=50$ trial offspring in each generation where the best $\nu=20$ of the trial offspring become the actual offspring where the expensive objective function is evaluated. As mentioned above, the initial parent population is taken to be the best μ points from an affinely independent LHD of size $d+1$.

6. Results and discussion

6.1. Performance and data profiles

The proposed OPUS-RBF method is evaluated and compared with alternatives using performance and data profiles ([7,17]). Let $\mathcal{P}$ be the set of problems where a given problem p corresponds to a particular test problem and a particular starting point. In this case, the starting point is the first point in the LHD used to initialize all methods. Since there are 12 optimization problems (GWB36, TownBrook14 and the 10 test problems) and 30 starting points (corresponding to the 30 trials), there are $12\times 30=360$ problems for the performance profiles. Moreover, let $\mathcal{S}$ be the set of solvers. Here, there are 6 solvers (OPUS-RBF, sPSO-RBF, mPSO-IPE-RBF, PSO-LHD, CMA-ES and ESGRBF). For any pair (p, s) of a problem p and a solver s, the *performance ratio* is

$$r_{p,s}=\frac{t_{p,s}}{\min\{t_{p,s}: s\in\mathcal{S}\}},$$

where $t_{p,s}$ is the number of function evaluations required to satisfy the convergence test that is defined below. Clearly, $r_{p,s}\ge 1$ for any $p\in\mathcal{P}$ and $s\in\mathcal{S}$. Moreover, for a given problem p, the best solver s for this problem attains $r_{p,s}=1$. Furthermore, by convention, $r_{p,s}=\infty$

Table 2
Statistics on the best objective function values after 300 function evaluations of the test problems (taken over 30 trials).

Test problem	Algorithm	Best	Median	Worst	Mean	Standard error
Groundwater bioremediation	OPUS-RBF	**361.98**	**432.87**	**532.63**	**434.41**	**7.82**
	sPSO-RBF	504.56	587.31	823.97	599.13	13.46
	mPSO-IPE-RBF	391.37	470.96	582.11	476.39	9.53
	PSO-LHD	461.05	636.53	772.83	620.85	13.88
	CMA-ES	410.52	597.28	789.25	596.95	16.39
	ESGRBF	374.69	453.51	554.32	447.63	8.00
Town Brook	OPUS-RBF	**631.33**	**703.62**	826.62	**712.82**	10.11
	sPSO-RBF	667.59	727.17	**786.44**	721.62	**6.65**
	mPSO-IPE-RBF	661.37	711.64	933.05	736.83	12.00
	PSO-LHD	645.88	744.94	874.71	741.95	9.13
	CMA-ES	644.54	753.60	1122.86	769.78	16.81
	ESGRBF	662.32	717.00	851.42	724.23	7.45
Ackley	OPUS-RBF	**-20.65**	**-19.89**	**-19.43**	**-19.90**	**0.05**
	sPSO-RBF	−19.56	−18.99	−18.31	−18.96	0.07
	mPSO-IPE-RBF	−15.38	−12.88	−11.44	−12.93	0.15
	PSO-LHD	−12.86	−11.44	−10.02	−11.47	0.12
	CMA-ES	−12.06	−10.85	−8.97	−10.75	0.13
	ESGRBF	−13.01	−11.16	−9.27	−11.17	0.16
Rastrigin	OPUS-RBF	**-15.93**	**-6.82**	2.09	**-6.97**	0.78
	sPSO-RBF	−11.91	−6.15	**1.84**	−5.70	**0.68**
	mPSO-IPE-RBF	0.08	10.35	16.55	10.06	0.73
	PSO-LHD	10.85	19.75	27.06	18.73	0.75
	CMA-ES	9.86	22.52	38.87	22.58	1.46
	ESGRBF	6.21	20.82	40.58	21.74	1.18
Griewank	OPUS-RBF	**0.71**	**0.97**	**1.05**	**0.96**	0.0136
	sPSO-RBF	1.04	1.07	1.13	1.07	**0.0041**
	mPSO-IPE-RBF	28.73	59.56	76.43	57.21	2.44
	PSO-LHD	50.66	86.37	124.62	86.49	3.49
	CMA-ES	67.16	103.31	165.94	105.50	4.00
	ESGRBF	60.13	101.94	150.18	103.89	4.19
Keane	OPUS-RBF	**-0.32**	**-0.24**	**-0.18**	**-0.24**	5.90×10^{-3}
	sPSO-RBF	−0.24	−0.17	−0.13	−0.17	3.68×10^{-3}
	mPSO-IPE-RBF	−0.20	−0.17	−0.14	−0.17	2.67×10^{-3}
	PSO-LHD	−0.20	−0.17	−0.15	−0.17	$\mathbf{2.38} \times 10^{-3}$
	CMA-ES	−0.25	−0.19	−0.16	−0.19	3.40×10^{-3}
	ESGRBF	−0.23	−0.19	−0.15	−0.19	3.74×10^{-3}
Levy	OPUS-RBF	**-8.25**	**-3.91**	**0.12**	**-3.50**	0.38
	sPSO-RBF	−7.55	−2.89	0.41	−3.24	**0.36**
	mPSO-IPE-RBF	6.78	21.20	37.31	21.40	1.35
	PSO-LHD	21.19	38.43	56.65	37.58	1.76
	CMA-ES	28.39	54.83	119.60	57.93	3.67
	ESGRBF	21.69	36.64	71.56	37.41	2.09
Michalewicz	OPUS-RBF	**-11.87**	**-9.53**	−7.81	**-9.51**	0.16
	sPSO-RBF	−11.08	−9.30	−8.04	−9.38	0.13
	mPSO-IPE-RBF	−11.00	−8.90	−7.17	−8.92	0.18
	PSO-LHD	−10.85	−9.37	**-8.42**	−9.44	**0.10**
	CMA-ES	−10.08	−7.98	−5.97	−8.02	0.17
	ESGRBF	−9.72	−8.16	−6.65	−8.28	0.14
Extended Rosenbrock	OPUS-RBF	**25.28**	**37.28**	**61.15**	**39.43**	**1.71**
	sPSO-RBF	61.94	103.86	192.09	109.00	5.71
	mPSO-IPE-RBF	65.84	123.76	369.36	132.11	10.48
	PSO-LHD	93.81	164.82	308.71	173.99	8.85
	CMA-ES	100.74	227.94	450.37	236.90	15.25
	ESGRBF	145.98	287.30	469.46	281.97	14.12
Extended Powell Singular	OPUS-RBF	**21.30**	**71.70**	**152.36**	**75.21**	**6.04**
	sPSO-RBF	116.48	198.40	460.20	211.14	13.09
	mPSO-IPE-RBF	110.40	259.73	427.10	259.19	14.99
	PSO-LHD	145.27	335.74	635.99	338.18	23.46
	CMA-ES	160.72	405.26	766.20	390.30	29.46
	ESGRBF	239.93	439.87	824.03	438.70	27.69
Trigonometric	OPUS-RBF	**2.74**	**6.91**	**14.65**	**7.66**	**0.61**
	sPSO-RBF	14.80	40.92	63.52	38.52	2.45
	mPSO-IPE-RBF	181.92	445.86	932.84	499.58	34.84
	PSO-LHD	453.70	1249.45	3474.96	1419.15	132.55
	CMA-ES	169.63	845.65	3094.28	1045.73	125.51
	ESGRBF	168.73	371.30	1031.57	414.15	35.65

Table 2 (*Continued*)

Test problem	Algorithm	Best	Median	Worst	Mean	Standard error
Broyden Tridiagonal	OPUS-RBF	**2.42**	**8.63**	**13.76**	**8.10**	**0.52**
	sPSO-RBF	4.73	8.74	16.32	9.59	0.65
	mPSO-IPE-RBF	10.80	18.61	27.64	19.24	0.77
	PSO-LHD	13.39	21.51	29.26	21.27	0.76
	CMA-ES	12.63	18.32	27.43	18.29	0.63
	ESGRBF	6.91	15.05	29.93	16.17	1.00

whenever solver s fails to yield a solution that satisfies the convergence test.

Now, for any solver $s \in \mathcal{S}$ and for any $\alpha \geq 1$, the *performance profile of s with respect to* α is the fraction of problems where the performance ratio is at most α [7], i.e.,

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \{p \in \mathcal{P} : r_{p,s} \leq \alpha\} \right|.$$

For any solver $s \in \mathcal{S}$, the *performance profile curve of s* is the graph of the performance profiles of s for a range of values of α.

In the context of derivative-free, expensive black-box optimization, algorithms are compared given a fixed and relatively limited number of function evaluations. Hence, the convergence test proposed by Moré and Wild [17] uses a tolerance $\tau > 0$ and the minimum function value f_L obtained by *any* of the solvers on a particular problem within a given number μ_f of function evaluations and it checks if a point x obtained by a solver satisfies

$$f(x^{(0)}) - f(x) \geq (1 - \tau)(f(x^{(0)}) - f_L),$$

where $x^{(0)}$ is a starting point corresponding to the problem under consideration. In the above expression, x is required to achieve a reduction that is $1 - \tau$ times the best possible reduction $f(x^{(0)}) - f_L$.

Next, given a solver $s \in \mathcal{S}$ and $\alpha > 0$, the *data profile of a solver s with respect to* α [17] is given by

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\} \right|,$$

where $t_{p,s}$ is the number of function evaluations required by solver s to satisfy the convergence test on problem p and n_p is the number of variables in problem p. For any solver $s \in \mathcal{S}$, the *data profile curve of s* is the graph of the data profiles of s for a range of values of α. For a given solver s and any $\alpha > 0$, $d_s(\alpha)$ is the fraction of problems "solved" (i.e., problems where the solver generated a point satisfying the convergence test) by s within $\alpha(n_p + 1)$ function evaluations (equivalent to α simplex gradient estimates [17]).

Fig. 1 shows the performance profiles and data profiles of the various solvers on the optimization problems in this study. In both profiles, it is very clear that OPUS-RBF is a dramatic improvement over PSO on the problems considered in this study. Moreover, both profiles also show that OPUS-RBF is better than the alternative surrogate-assisted PSO algorithms (sPSO-RBF and mPSO-IPE-RBF) and the other alternative methods CMA-ES and ESGRBF on the problems considered. In particular, from the performance profile curve for OPUS-RBF, the fraction of problems for which OPUS-RBF is the best (given by the value of $\rho_{\text{OPUS-RBF}}(1)$) is more than 50% while the corresponding fraction for sPSO-RBF is less than 40% and the corresponding fractions for the other methods are all less than 10%. Also, from the data profiles, OPUS-RBF satisfied the convergence test for more than 90% of the problems compared to less than 60% for the other methods after the equivalent of about 15 simplex gradient estimates.

6.2. Explanation of additional numerical comparisons

Fig. 2 shows the plot of the mean of the best objective function value obtained by each algorithm on each problem as the number of function evaluations increases. These plots are referred to as *average progress curves*. The error bars are 95% t confidence intervals for the mean. That is, the length of each side of the error bar is equal to 2.045 times the standard deviation of the best function values divided by the square root of the number of trials. The factor 2.045 is the critical value corresponding to a 95% confidence level for a t distribution with 29 degrees of freedom. Since all algorithms start with the same set of initial points (an affinely independent LHD) in each trial, all curves begin at the same height. As the number of function evaluations increase, the curves that go down faster compared to the others are the better algorithms.

When function evaluations are computationally expensive, average progress curves are also suitable for comparing the performance of algorithms for several reasons. First, finding the global minimum of a high-dimensional objective function given a very limited computational budget is an unrealistic goal. Hence, comparing the number of function evaluations required to reach the global minimum within some tolerance, as is typically done for evolutionary and swarm algorithms, is not appropriate. Second, in the computationally expensive setting, the total running time of each algorithm is completely dominated by the time spent on function evaluations and the emphasis of the comparisons is on how well an algorithm performs given a very limited number of function evaluations. Imagine, for example, that each function evaluation of the Ackley function takes about an hour. Then the average progress curves give a good idea of where each algorithm stands after say 100 h or at other computational budgets.

In addition to the average progress curves, Table 2 provides statistics (best, median, worst, mean and standard error of the mean) for the best objective function value found by the algorithms after a fixed number of function evaluations (300 function evaluations in this study). For each problem and for each statistic in Table 2, the best value among the different methods is written in boldface. The purpose of this table is to allow comparison with algorithms developed by other researchers. Moreover, as suggested in [6], Table 3 provides the results of Friedman's nonparametric

Table 3
Non-parametric multiple comparisons between OPUS-RBF and other methods after 300 function evaluations for the test problems. The symbol + indicates that the mean rank of OPUS-RBF is significantly better than that of the other algorithm, − indicates that it is significantly worse, and 0 means the difference is not significant at the 5% level.

Problem	sPSO-RBF	mPSO-IPE-RBF	PSO-LHD	CMA-ES	ESGRBF
Groundwater bioremediation	+	0	+	+	0
Town Brook	0	0	0	+	0
Ackley	0	+	+	+	+
Rastrigin	0	+	+	+	+
Griewank	0	+	+	+	+
Keane	+	+	+	+	+
Levy	0	+	+	+	+
Michalewicz	0	+	0	+	+
Extended Rosenbrock	+	+	+	+	+
Extended Powell Singular	+	+	+	+	+
Trigonometric	0	+	+	+	+
Broyden Tridiagonal	0	+	+	+	+

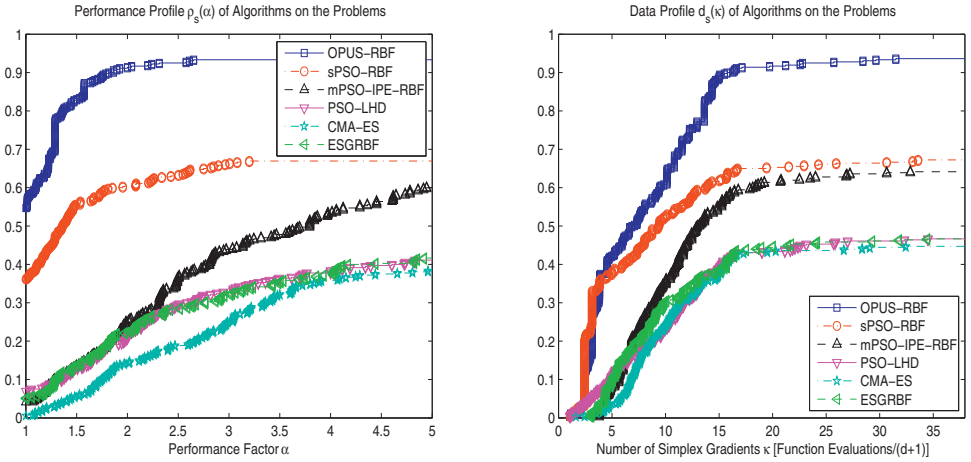

Fig. 1. Performance and data profiles for optimization methods on the test problems.

statistical test for determining if the median of the best objective value (after 300 function evaluation) of at least one algorithm is significantly different from that of the other algorithms for each test problem. In Friedman's test, the treatments are the different algorithms and the blocks are the different trials. Friedman's test is then followed by a multiple comparison procedure to determine whether the mean rank of OPUS-RBF is significantly better than the mean rank of an alternative method. In Table 3, the symbol + indicates that the mean rank of OPUS-RBF is significantly better than that of the other algorithm, – indicates that it is significantly worse, and 0 means the difference is not significant at the 5% level. Friedman's test and the multiple comparison procedure are performed using Matlab's Statistics Toolbox.

Note that the objective functions used here are not really computationally expensive to evaluate. However, the relative performance of the algorithms on these test problems are expected to be similar to their relative performance on truly expensive real-world objective functions with characteristics similar to the test functions. Besides, performing these comparisons on cheap test problems enables one to perform multiple trials and compare average case performance.

6.3. Discussion of the average progress curves and statistical comparisons

The average progress curves in Fig. 2 clearly show that OPUS-RBF is an improvement over a standard PSO implementation initialized by an LHD, namely PSO-LHD, on 11 out of 12 problems (all except Michalewicz). In particular, OPUS-RBF is better than PSO-LHD on the 36-D groundwater bioremediation problem (GWB36) and on the 14-D watershed calibration problem (TownBrook14). On the Michalewicz problem, the performance of OPUS-RBF is close to that of PSO-LHD and it is slightly better than PSO-LHD after more than 400 function evaluations. These results suggest that using RBF surrogates as proposed in this paper generally improves the performance of a standard PSO algorithm, and in the worst case, it does not seem to hurt performance.

Next, the average progress curves in Fig. 2 also show that OPUS-RBF is consistently much better than mPSO-IPE-RBF on all test problems, including the two environmental applications GWB36 and TownBrook14. Moreover, OPUS-RBF is better than sPSO-RBF on 9 of the 12 problems (all except Griewank, Levy and Trigonometric) and it has roughly the same performance as sPSO-RBF on the remaining three problems. Recall that mPSO-IPE-RBF is an RBF-assisted PSO algorithm that uses the same strategy as one of the variants of the metamodel-assisted PSO algorithm by Praveen and Duvigneau [22]. Moreover, sPSO-RBF is also an RBF-assisted PSO algorithm that uses the same strategy as that used by the surrogate-assisted PSO algorithm by Parno et al. [19]. This suggests that the strategy used by OPUS-RBF is an improvement over the ones used in two previous surrogate-assisted PSO algorithms.

In addition, Fig. 2 shows that although PSO-LHD is not always better than CMA-ES, OPUS-RBF is consistently better than CMA-ES on all problems in this study, including the environmental applications. Moreover, OPUS-RBF is also much better than ESGRBF on all 12 problems. CMA-ES is a well-known method that has been shown to work well on many problems. Also, ESGRBF is an evolution strategy with RBF surrogates that is designed for expensive black-box optimization and it has been shown to work well on TownBrook14 [26] and on a 12-D version of GWB36 [25].

The statistics on the best objective function value obtained by the different algorithms after 300 function evaluations (shown in Table 2) also show that OPUS-RBF is much better than the alternatives. In particular, the best solutions for each of the 12 problems (including the environmental applications) after 300 function evaluations were all obtained by OPUS-RBF. Moreover, the median and mean of the best objective function values after 300 function evaluations for OPUS-RBF is consistently better than those for the other algorithms for all 12 problems. In addition, the results of Friedman's test in Table 3 show that the performance of OPUS-RBF is statistically significantly better than the performance of mPSO-IPE-RBF, PSO-LHD, CMA-ES and ESGRBF on most of the test problems after 300 function evaluations. Also, OPUS-RBF has significantly better performance than sPSO-RBF after 300 function evaluations on four of the problems, including the groundwater application, and it is at least as good as the other algorithms on the other problems.

6.4. Summary of the numerical results and limitations of the comparisons

The results of the performance and data profiles in Section 6.1 and the average progress curves and statistical comparisons in Section 6.3 consistently indicate that OPUS-RBF is much better than the alternative methods, including the two other RBF-assisted PSO algorithms, on the problems in this study. However, this paper does *not* really prove that the proposed OPUS-RBF is superior to the alternative methods. For one thing, the results are limited to two environmental applications and ten other test problems. These results do not necessarily generalize to other applications or test problems. Besides, it is widely believed that there is *no* universally best algorithm. Moreover, although reasonable parameter settings

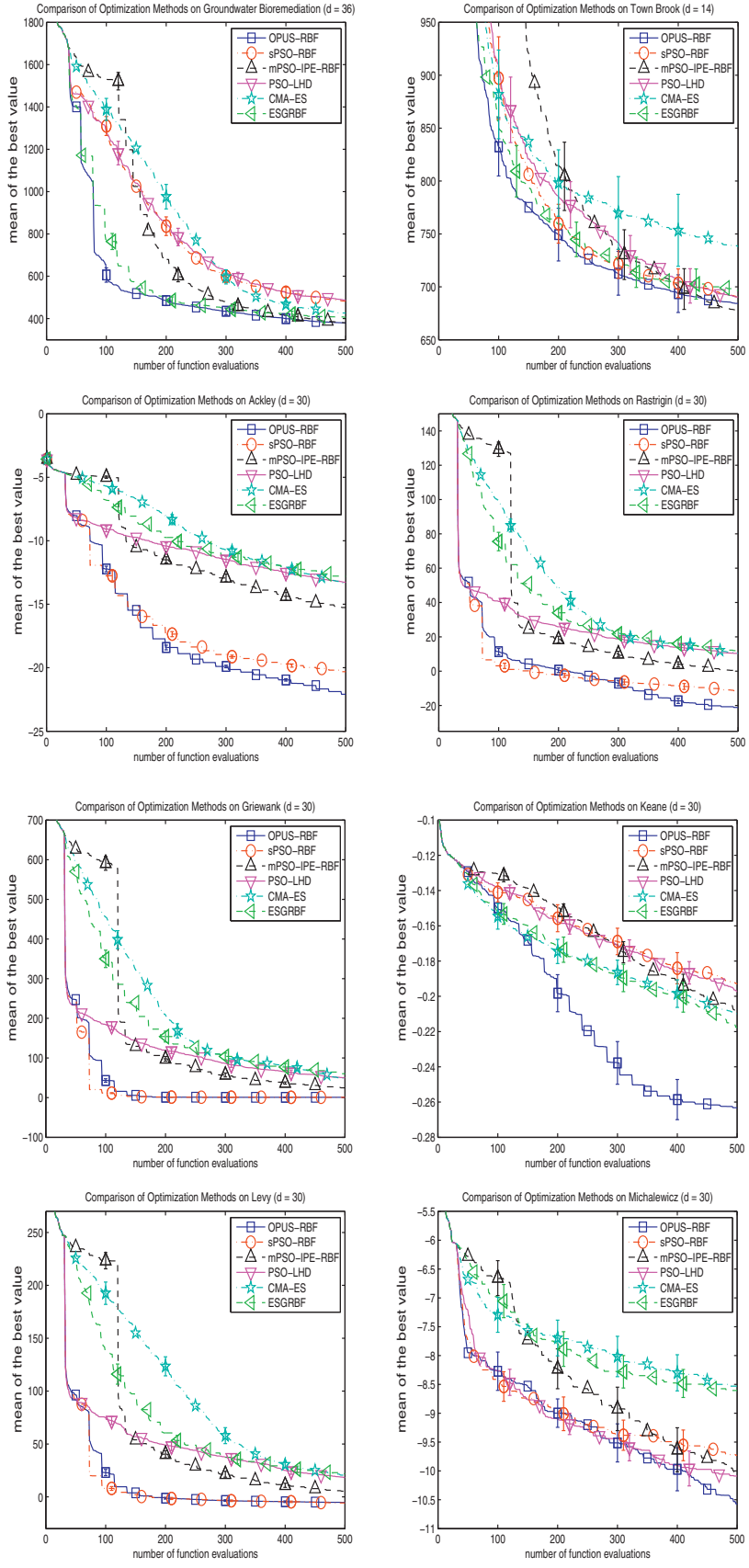

Fig. 2. Mean of the best feasible objective function value (over 30 trials) vs number of function evaluations for the alternative optimization methods on the optimization problems. Error bars represent 95% t-confidence intervals for the mean.

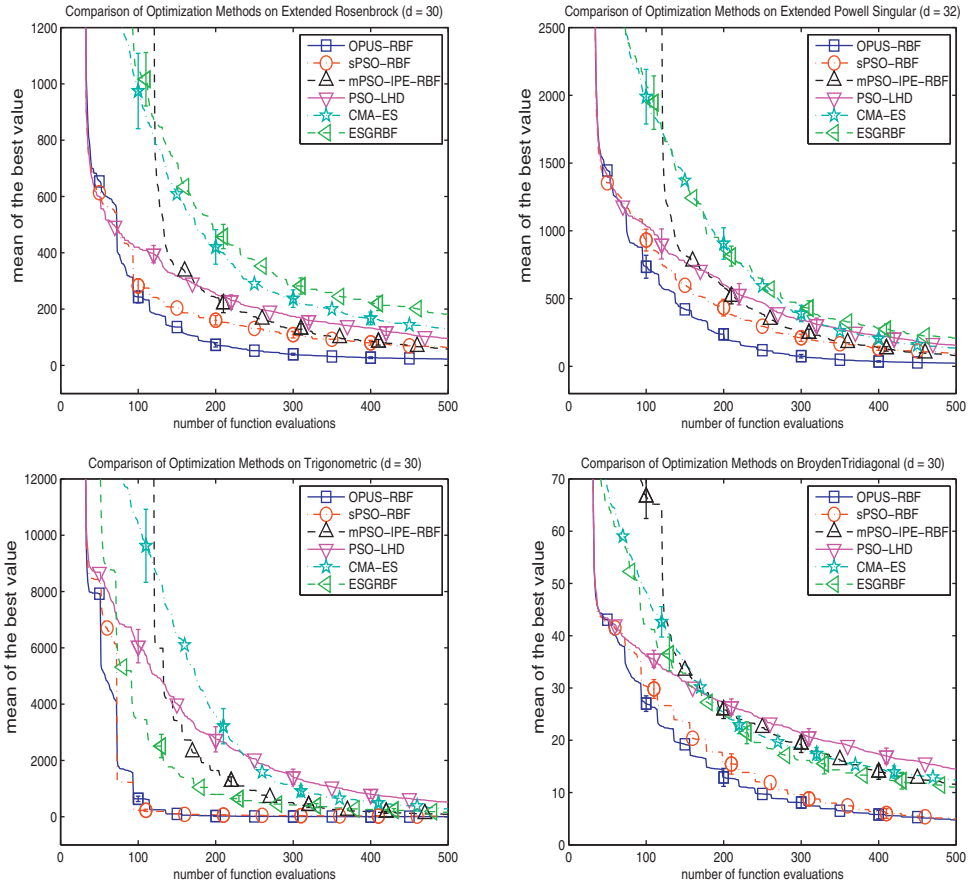

Fig. 2. (*Continue*).

are used for OPUS-RBF and the alternative methods, these settings are probably not optimal for the problems in this study. One of the goals of this paper is to demonstrate that the proposed OPUS-RBF performs better than or at least compares favorably with reasonable alternative methods on *some* important applications and on an adequate collection of widely used test problems. The numerical results showed that this goal has been achieved. Hence, OPUS-RBF is an excellent alternative for expensive black-box optimization that is worthy of further consideration.

6.5. Average running times

To get an idea of the overhead computation times of OPUS-RBF, Table 4 reports the mean running times on the 36-D groundwater bioremediation problem (the highest dimensional problem in this study) for 250 and 500 function evaluations, excluding the time spent on function evaluations. As expected, the surrogate-assisted algorithms (OPUS-RBF, sPSO-RBF, mPSO-IPE-RBF and ESGRBF) have longer average running times compared to PSO-LHD and CMA-ES because of the extra time required to fit the RBF surrogates, screen out trial solutions, or perform optimization on RBF surfaces. Moreover, among the surrogate-assisted algorithms, OPUS-RBF and sPSO-RBF require longer computation times because these algorithms solve a minimization problem on the RBF surface in every iteration. However, the extra computation times are still relatively small, and for computationally expensive problems, these running times are negligible compared to the total time spent on all function evaluations. In addition, the overhead time for OPUS-RBF is much smaller than those of other surrogate-based methods in the literature. This is because the RBF model used here takes much less time to fit compared to other surrogate models such as kriging, which requires numerically solving a maximum likelihood estimation problem.

Table 4
Mean running times of the different algorithms on the GWB36 problem (excluding time spent on function evaluations) on an Intel(R) Core(TM) i7 CPU 860 @ 2.8 GHz desktop machine.

Algorithm	After 250 function evaluations	After 500 function evaluations
OPUS-RBF	109.77 s (1.83 min)	507.96 s (8.47 min)
sPSO-RBF	120.48 s (2.01 min)	454.09 s (7.57 min)
mPSO-IPE-RBF	0.27 s	1.18 s
PSO-LHD	0.04 s	0.08 s
CMA-ES	0.23 s	0.58 s
ESGRBF	0.72 s	1.85 s

7. Summary and conclusions

This paper introduced a new framework for a surrogate-assisted PSO, called OPUS (Optimization by Particle swarm Using Surrogates), that is suitable for computationally expensive black-box optimization. The strategy used by OPUS differs from the strategies used by the few other surrogate-assisted PSO algorithms in the literature and it is more suitable for high dimensional problems. The basic strategy in OPUS is to consider multiple trial velocities and multiple trial positions for each particle in each iteration. The surrogate is then used to select the most promising trial position for each particle (in terms of predicted objective function value) and the particle then moves into that position. Then the algorithm attempts to improve the current overall best position by finding the global minimum of the surrogate within a box centered at that

position. A cubic RBF model with a linear polynomial tail is used in the implementation and the resulting algorithm is called OPUS-RBF. Numerical results on a 36-D groundwater bioremediation problem, a 14-D watershed calibration problem and 10 other test problems with about 30 dimensions suggest that OPUS-RBF is an improvement over a more standard PSO implementation and it is better than two other RBF-assisted PSO algorithms (sPSO-RBF and mPSO-IPE-RBF) that implement the strategies used by the surrogate-assisted PSO algorithms by Parno et al. [19] and by Praveen and Duvigneau [22]. In addition, OPUS-RBF is better than an evolution strategy with covariance matrix adaptation (CMA-ES) and an evolution strategy with RBF surrogates (ESGRBF) on the same problems. Overall, OPUS-RBF is a promising alternative for expensive black-box optimization.

Acknowledgements

I would like to thank Dr. Bryan Tolson and Ryan Fleming for providing the simulation code for the watershed calibration problem. I am also grateful to Dr. Nikolaus Hansen and his colleagues for the CMA-ES Matlab code. Finally, I would to thank Jorge More and Stefan Wild for their Matlab code that creates performance and data profiles.

References

[1] R. Akbari, K. Ziarati, A rank based particle swarm optimization algorithm with dynamic adaptation, J. Comput. Appl. Math. 235 (8) (2011) 2694–2714.
[2] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: Background and development, Nat. Comput. 6 (4) (2007) 467–484.
[3] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Nat. Comput. 7 (1) (2008) 109–124.
[4] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2007, pp. 120–127.
[5] M.D. Buhmann, Radial Basis Functions, Cambridge Univ. Press, Cambridge, UK, 2003.
[6] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolutionary Comput. 1 (1) (2011) 3–18.
[7] E.D. Dolan, J.J. More, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213.
[8] H.-M. Gutmann, A radial basis function method for global optimization, J. Global Optim. 19 (3) (2001) 201–227.
[9] N. Hansen, The CMA evolution strategy: a comparing review, in: J.A. Lozano, P. Larranga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation, Springer, Berlin, Heidelberg, 2006, pp. 75–102.
[10] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolut. Comput. 9 (2) (2001) 159–195.
[11] S. Helwig, R. Wanka, Theoretical analysis of initial particle swarm behavior, in: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN08), Springer, Dortmund, 2008, pp. 889–898.
[12] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2007) 1407–1422.
[13] X. Hu, R.C. Eberhart, Solving constrained nonlinear optimization problems with particle swarm optimization, in: N. Callaos (Ed.), Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL, 2002, pp. 203–206.
[14] A. Ismail, A.P. Engelbrecht, Self-adaptive particle swarm optimization, in: Simulated Evolution and Learning, Lecture Notes in Computer Science, vol. 7673, Springer, Berlin, Heidelberg, 2012, pp. 228–237.
[15] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[16] J. Moré, B. Garbow, K. Hillstrom, Testing unconstrained optimization software, ACM Trans. Math. Software 7 (1981) 17–41.
[17] J. Moré, S. Wild, Benchmarking derivative-free optimization algorithms, SIAM J. Optim. 20 (1) (2009) 172–191.
[18] A.-E. Munoz-Zavala, A.H. Aguirre, E.R.V. Diharce, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), in: H.-G. Beyer (Ed.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005), vol. 1, ACM Press, New York, 2005, pp. 209–216.
[19] M.D. Parno, T. Hemker, K.R. Fowler, Applicability of surrogates to improve efficiency of particle swarm optimization for simulation-based problems, Eng. Optim. 44 (5) (2012) 521–535.
[20] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: an overview, Swarm Intell. 1 (1) (2007) 33–57.
[21] M.J.D. Powell, The theory of radial basis function approximation in 1990, in: W. Light (Ed.), Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions, Oxford University Press, Oxford, UK, 1992, pp. 105–210.
[22] C. Praveen, R. Duvigneau, Low cost PSO using metamodels and inexact preevaluation: application to aerodynamic shape design, Comput. Meth. Appl. Mech. Eng. 198 (2009) 1087–1096.
[23] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Inform. Sci. 197 (2012) 131–143.
[24] R.G. Regis, Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions, IEEE Trans. Evolut. Comput. (2013), http://dx.doi.org/10.1109/TEVC.2013.2262111.
[25] R.G. Regis, C.A. Shoemaker, Local function approximation in evolutionary algorithms for the optimization of costly functions, IEEE Trans. Evolut. Comput. 8 (5) (2004) 490–505.
[26] C.A. Shoemaker, R.G. Regis, R.C. Fleming, Watershed calibration using multistart local optimization and evolutionary optimization with radial basis function approximation, Hydrol. Sci. J. [J. Sci. Hydrol.] 52 (3) (2007) 450–465.
[27] Y. Tang, J. Chen, J. Wei, A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions, Eng. Optim. 45 (5) (2013) 557–576.
[28] G. Toscano-Pulido, C.A. Coello Coello, A constraint-handling mechanism for particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation 2004 (CEC 2004), vol. 2, IEEE Service Center, Piscataway, New Jersey, 2004, pp. 1396–1403.
[29] A.I.F. Vaz, L.N. Vicente, A particle swarm pattern search method for bound constrained global optimization, J. Global Optim. 39 (2007) 197–219.
[30] B. Wei, Y. Li, D. Shen, F. Yu, X. Xu, Local stable mechanism for particle swarm optimization algorithm, in: Information Computing and Applications, Communications in Computer and Information Science, vol. 308, Springer, Berlin, Heidelberg, 2012, pp. 466–473.
[31] S.M. Wild, C.A. Shoemaker, Global convergence of radial basis function trust region derivative-free algorithms, SIAM J. Optim. 21 (3) (2011) 761–781.
[32] J.-H. Yoon, C.A. Shoemaker, Comparison of optimization methods for groundwater bioremediation, J. Water Resour. Plan. Manag. 125 (1999) 54–63.

Rommel G. Regis is an Associate Professor at the Department of Mathematics of Saint Joseph's University in Philadelphia, Pennsylvania, USA. He received a Master's degree in Mathematics at the University of Florida in 1998 and a Ph.D. in Operations Research at Cornell University in 2004 with minors in Computer Science and Applied Probability & Statistics. His current research interests include engineering optimization, global optimization, and heuristic methods and he has published several journal papers on surrogate-based expensive black-box optimization.