J Glob Optim (2016) 66:417–437
DOI 10.1007/s10898-016-0407-7

# SOP: parallel surrogate global optimization with Pareto center selection for computationally expensive single objective problems

**Tipaluck Krityakierne**[1] · **Taimoor Akhtar**[2,3] · **Christine A. Shoemaker**[2,3,4]

Received: 6 October 2014 / Accepted: 19 January 2016 / Published online: 2 February 2016


**Abstract** This paper presents a parallel surrogate-based global optimization method for computationally expensive objective functions that is more effective for larger numbers of processors. To reach this goal, we integrated concepts from multi-objective optimization and tabu search into, single objective, surrogate optimization. Our proposed derivative-free algorithm, called SOP, uses non-dominated sorting of points for which the expensive function has been previously evaluated. The two objectives are the expensive function value of the point and the minimum distance of the point to previously evaluated points. Based on the results of non-dominated sorting, $P$ points from the sorted fronts are selected as centers from which many candidate points are generated by random perturbations. Based on surrogate approximation, the best candidate point is subsequently selected for expensive evaluation for each of the $P$ centers, with simultaneous computation on $P$ processors. Centers that previously did not generate good solutions are tabu with a given tenure. We show almost sure convergence of this algorithm under some conditions. The performance of SOP is compared with two RBF based methods. The test results show that SOP is an efficient method that can reduce time required to find a good near optimal solution. In a number of cases the efficiency

**Electronic supplementary material** The online version of this article (doi:10.1007/s10898-016-0407-7) contains supplementary material, which is available to authorized users.

✉ Tipaluck Krityakierne
tk338@cornell.edu

Taimoor Akhtar
taimoor.akhtar@gmail.com

Christine A. Shoemaker
cas12@cornell.edu

1 Institute of Mathematical Statistics and Actuarial Science, University of Bern, Bern, Switzerland

2 School of Civil and Environmental Engineering, Cornell University, Ithaca, NY, USA

3 Department of CEE, National University of Singapore, Singapore, Singapore

4 Department of ISE, National University of Singapore, Singapore, Singapore

of SOP is so good that SOP with 8 processors found an accurate answer in less wall-clock time than the other algorithms did with 32 processors.

**Keywords** Radial basis functions · Tabu · Computationally expensive · Blackbox · Simulation optimization · Response surface · Metamodel

## 1 Introduction

Real-world applications in various fields, such as physics, engineering, or economics, often have a simulation model which is multimodal, computationally expensive, and blackbox. "Blackbox" implies that many mathematical characteristics are not known, including derivatives or number of local minima. Many existing optimization methods for black-box functions such as genetic algorithm, simulated annealing, or particle swarm are not suitable for this type of problem due to the large number of objective function evaluations that these methods generally require.

One approach for dealing with this type of problem is to use a surrogate model (alternatively called metamodel or response surface) to approximate the objective function. Response surface based optimization methods start by building a (computationally inexpensive) surrogate surface, which is then used to iteratively select new points for the expensive function evaluation. The surrogate surface is updated in each iteration.

We consider a real-valued global optimization problem of the form:

$$\min_{x \in \mathcal{D}} f(x) \tag{1}$$

where $\mathcal{D} = \{\text{lb} \leq x \leq \text{ub}\} \subset \mathbb{R}^d$. Here, $\text{lb}, \text{ub} \in \mathbb{R}^d$ are the lower and upper variable bounds, respectively. $f(x)$ is a computationally expensive black-box function that is continuous but not differentiable or its gradient is computationally intractable.

The purpose of this research is to develop a new way to solve surrogate optimization problems for computationally expensive functions in parallel. Previous efforts (mostly serial) have involved generating candidate points by normal perturbations around one center point (usually the best solution found so far), by uniform sampling in the domain, or by using an optimization method on the surrogate to find the point that satisfies some criterion. In this work, we use a different approach involving non-dominated sorting on previously evaluated points to select multiple centers, which are points on the sorted fronts. Hence, we are using concepts from multi-objective optimization for single objective optimization of computationally expensive functions.

### 1.1 Literature review

Many authors have demonstrated the effectiveness of using response surfaces for optimization of computationally expensive problems within a limited number of function evaluations. Jones et al. [10] used kriging as a basis function to develop a global optimization method, Efficient Global Optimization (EGO), where the next point to evaluate is obtained by maximizing an expected improvement, balancing the response surface value with the uncertainty of the surface. Huang et al. [9] extended Jones' EGO algorithm and developed a global optimization method for noisy cost functions. Booker et al. [3] also used kriging surface to speed up the pattern search algorithms. Gutmann [6] built the response surface with radial basis functions where the next point to evaluate is obtained by minimizing the bumpiness of

the interpolation surface. Sóbester et al. [23] used Gaussian radial basis functions and proposed weighted expected improvement to control the balance of exploitation (of the response surface) and exploration (of the decision variable space) to select the next evaluation point. Regis and Shoemaker [18] also used radial basis functions in the Metric Stochastic Response Surface algorithm (also known as the Stochastic RBF algorithm), which is a global optimization algorithm wherein the next point to evaluate is chosen by randomly generating a large number of candidate points and selecting the point that minimizes a weighted score of response surface predictions and a distance metric. Optimization of computationally expensive problems is still an active field of research as can be seen by several workshops devoted to this subject. More recent papers on this subject include e.g. [11,12,15].

Due to the pervasiveness of parallel computing, there is a need to develop surrogate algorithms that select and evaluate multiple points in each iteration to reduce the wall-clock time (which is proportional to the number of optimization iterations). For example, Sóbester et al. [22] developed a parallel version of EGO. Several local maximum points of the expected improvement function are selected for the expensive evaluations in parallel. In 2009, Regis and Shoemaker proposed a parallel version of the Stochastic RBF algorithm [19]. In each iteration, a fixed number of points are selected for doing the expensive function evaluations. The selection is done sequentially and based on the weighted score of (1) the surrogate value, and (2) the minimum distance from previously evaluated points and previously selected points within that parallel iteration, until the desired number of points are selected. The expensive function evaluations at the selected points are done simultaneously. The experimental results showed that the algorithm is very efficient compared to their previous methods. As a counterpart of EGO, Viana et al. [25] proposed MSEGO that is based on the idea of maximizing the expected improvement (EI) functions, but instead of using just one surrogate as in EGO, multiple surrogates are used. Since different EI functions are obtained for different surrogates, multiple points can be selected per iteration. Although MSEGO was shown to reduce the number of iterations, they found that the numerical convergence rate did not scale up with the number of points selected in each iteration for evaluation.

There is an inherent trade-off between exploration and exploitation in surrogate-based optimization methods. Recently, Bischl et al. [2] attempted at analyzing this trade-off and proposed MOI-MBO which is a parallel kriging based optimization algorithm that uses a multi-objective infill criterion that rewards the diversity and the expected improvement for selecting the next expensive evaluation points. Many versions of MOI-MBO were proposed based on various multi-objective infill criteria: mean of the surrogate model, model uncertainty, expected improvement, distance to the nearest neighbor, and the distance to the nearest better neighbor. Evolutionary optimization was used to handle the embedded multi-objective optimization problem. The overall test results suggested that the version that used a combination of mean, model uncertainty, and nearest neighbor worked best.

### 1.2 Differences between SOP and previous algorithms

The major difference between our algorithm and other existing parallel optimization algorithms is the use of a Pareto non-dominated sorting technique to select $P$ distinct evaluated points whose objective function values are small and that are far away from other evaluated points, which will then serve as centers. The selected centers are subsequently used with the addition of random perturbations for generating a set of candidate points from which the next function evaluation points are chosen and evaluated. In addition the selection of the $P$ points is subject to tabu constraints. This contrasts with the approach Regis and Shoemaker [19] used to generate the $P$ points for parallel expensive function evaluations. In [19], all the $P$ points

are obtained from the same center that is the best point found so far. Selecting the $P$ points from different centers (as is done in this work) allows the algorithm to search more globally simultaneously in each iteration, which is especially helpful as the number of processors increases. The concept of non-dominated sorting has been widely used in multi-objective optimization [1,4,26]. SOP considers the trade-off between exploration and exploitation in single objective optimization as a bi-objective optimization problem and incorporates non-dominated sorting into the algorithm framework. The method proposed in [2] is also based on multi-objective optimization. However, their embedded multi-objective optimization problem was solved on the surrogate while in our approach, a bi-objective optimization problem is solved on previously evaluated expensive function evaluation points.

We found no journal papers on surrogate global optimization that select a large number of evaluation points in each iteration in order to efficiently use a large number of processors. For example, the maximum number of points used in [2,19], and [25] were 5, 8, and 10 points, respectively. On the other hand, SOP can do many expensive objective function evaluations simultaneously. (SOP was tested on as many as 64 points per parallel iteration.) SOP thus has the potential to greatly reduce wall-clock time.

The structure of this paper is as follows: The new algorithm, SOP, is described in Sect. 2, and its theoretical properties are described in Theorem 1. In Sect. 3, we illustrate the performance of SOP and compare algorithms on a number of test functions as well as a groundwater bioremediation problem. Lastly, we give our concluding remarks in Sect. 4.

## 2 Algorithm description

Surrogate Optimization with Pareto Selection (SOP) is a parallel surrogate based algorithm where simultaneous surrogate-assisted candidate searches are performed around selected evaluated points (referred to as centers in subsequent discussions). The search mechanism of the algorithm incorporates bi-objective optimization, where the conflicting objectives focus the search to achieve a balance between exploration and exploitation, tabu search, and surrogate assisted candidate search. The motivation for using this combination of methods is so we can generate $P$ points (where $P$ can be large) for evaluation in parallel that will provide useful information for the search. Hence, these points need to be efficiently distributed and among other factors should not be too close to each other.

A synchronous Master-slave parallel framework is employed for simultaneous surrogate-assisted candidate search. The algorithm is adaptive and learns from the search results of prior iterations.

### 2.1 General algorithm framework

The general algorithm framework[1] is iterative and is described by a flowchart in Fig. 1. The algorithm consists of three core steps, namely (1) Initialization, (2) Iterative loop and (3) Termination.

The algorithm initialization phase (Step 1) starts by evaluating the expensive $f(x)$ on $n_0$ initial points from the decision space. The initial point may be selected via any experimental design method, e.g. Latin hypercube sampling. The expensive points are subsequently evaluated before initiation of the iterative loop.

[1] Pseudo-code for SOP algorithm is given in Algorithm 1 of the Online Supplement.

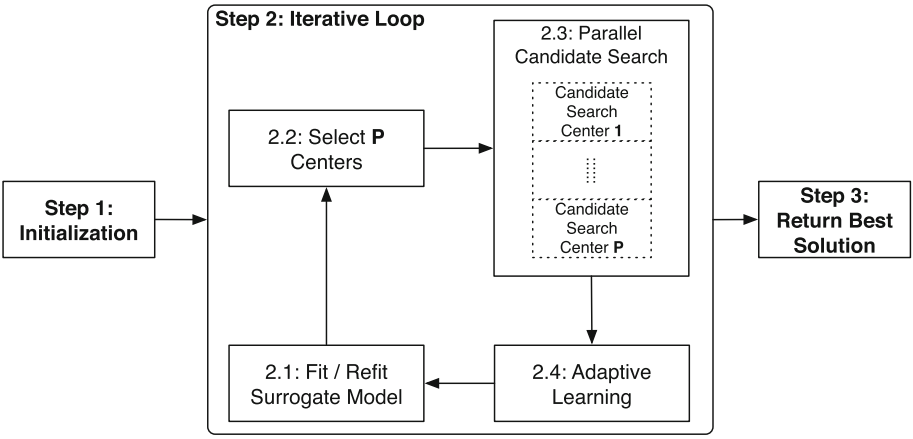


**Fig. 1** General iterative framework of SOP

Step 2 of the algorithm corresponds to the iterative loop, which has four sub-components, namely (2.1) Fit/refit surrogate model, (2.2) Select the $P$ centers, (2.3) Parallel candidate search, and (2.4) Adaptive learning.

In Step 2.1, all previously evaluated points and their corresponding function values are used to fit a surrogate model. Steps 2.2, 2.3 and 2.4 constitute the core components that define the search mechanics of SOP by maintaining a balance between exploration and exploitation, and incorporating the power of parallelism for improving efficiency of the surrogate-assisted candidate search. Each of these steps will be discussed in more detail later.

At the end of the iterative loop (Step 3), the algorithm terminates after a pre-specified number of iterations (or number of function evaluations) and returns the best solution found so far.

### Step 2.2 non-dominated sorting and $P$ center selection

Given that $P$ processors are available, Step 2.2 of SOP selects $P$ points from the set of already evaluated points as **center points** for parallel surrogate-assisted candidate search. The centers are selected by using non-dominated sorting (an idea from multi-objective optimization) on points where the expensive $f(x)$ has been evaluated. This approach uses two objective functions to find a diverse set of points that are likely to provide good function values and be diverse enough to improve the surrogate surface when many points are being evaluated simultaneously by parallel processors. While previous methods, such as Parallel Stochastic RBF, define the best solution found in all previous iterations as a center for perturbations, we instead create as many centers for perturbations as there are processors to facilitate improved parallel performance.

Let $S^{(n)}$ be the set of already evaluated points after $n$ algorithm iterations. SOP considers the challenge of balancing the trade-off between exploration and exploitation as a bi-objective optimization problem.

The objective function corresponding to exploitation is simply the objective function of the expensive optimization problem and is referred to as $F_1(x)$. The second objective $F_2(x)$ focusing on exploration is the minimum distance of an evaluated point, $x \in S^{(n)}$, from all other evaluated points, $S^{(n)} \setminus \{x\}$. A large value of minimum distance intuitively indicates that a potential center point is in a sparsely explored region of the decision space where the accuracy of the current solution can possibly be improved with candidate search in that

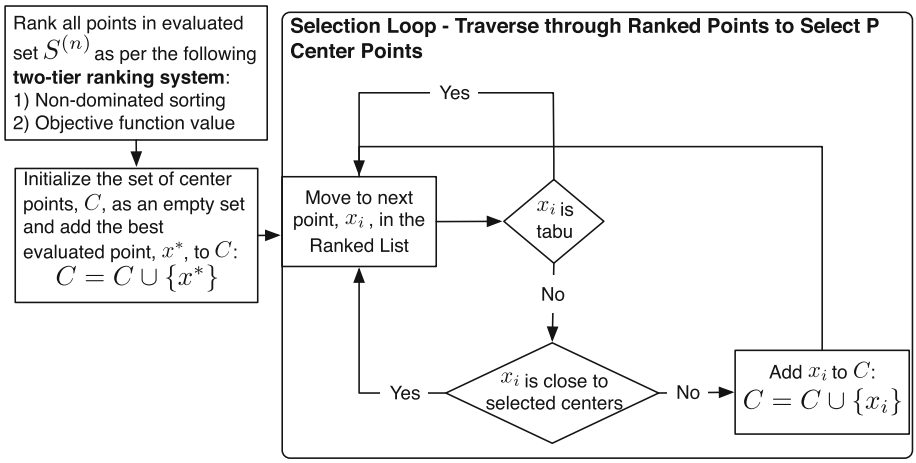


**Fig. 2** The process for selection of $P$ center points (Step 2.2)

region. Thus, to explore previously unexplored regions, a large value of $F_2(x)$ is desirable. Maximizing $F_2(x)$ is the same as minimizing its negative, so the second objective $F_2(x)$ is defined as the negative of the minimum distance to the set of already evaluated points.

Mathematically, the following bi-objective optimization over a finite set $S^{(n)}$ is considered for center selection:

$$\min_{x \in S^{(n)}} [F_1(x),\ F_2(x)], \tag{2}$$

where $F_1(x) = f(x)$ from Eq. 1 and $F_2(x) = -\min_{s \in S^{(n)} \setminus \{x\}} \|s - x\|$ are two conflicting objective functions that we want to minimize simultaneously.

Given that SOP considers the process of selection of centers as the bi-objective problem defined above, $P$ points are selected as centers from the set of evaluated points. The process for selection of $P$ centers is depicted in Fig. 2.

The selection process initiates by ranking all evaluated points $(x_i, f(x_i))$ according to the **two-tier ranking system**.[2] This corresponds to the upper left box in Fig. 2. The first tier of ranking is based on the concept of non-dominated sorting method [5] on the bi-objective problem defined in Eq. 2, which divides the evaluated set into mutually exclusive subsets which are referred to as fronts, where each front has a unique rank. The first rank (Pareto front) is given to the non-dominated solutions in the evaluated set. These solutions are then removed, and the non-dominated solutions identified in the remaining set are given the second rank. This process continues until all solutions are sorted into fronts (see Fig. 3b).

Since multiple solutions may be on the same front, we introduce the second tier of ranking to differentiate between these solutions. The solutions on the same front are ranked according to their objective function values $f(x)$ (best to worst). Hence, the best evaluated point found so far has the best rank (since it lies on the first front and has the best objective function value). The secondary ranking tends to focus more on exploitation than exploration. We will denote by **Ranked List** the set (list) of evaluated points sorted according to this two-tier ranking system.

Let $C$ denote the set of (to be) selected centers, where initially $C = \emptyset$. First, the best evaluated solution found so far, $x^*$, is always selected as the first center $c_1$. This corresponds

[2] Pseudo-code describing the two-tier ranking system can be found in Figure A.2 (two_tier_ranking) of the Online Supplement.

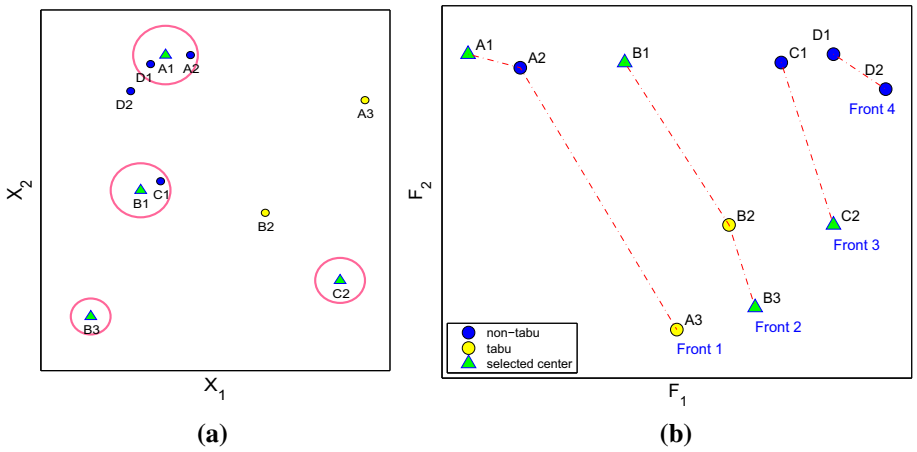


**Fig. 3** Example of point classification into tabu, non-tabu or selected center. Previously evaluated points $x \in S^{(n)}$ are shown in (**a**) in terms of decision variable values and (**b**) in terms of objective function $[F_1(x), F_2(x)]$ values. **a** Decision space. (**b**) Objective space

to the second box of the flowchart in Fig. 2. Next, the algorithm traverses through points in the Ranked List, sequentially adds a selected point to $C$, and stops when $C = \{c_1, ..., c_P\}$. While traversing through the ranked points, two additional criteria are checked before points are selected as centers:[3]

**The first selection criteria** is that a point in the Ranked List is only selected as a center if it is not in the **Tabu List**. The Tabu List is a subset of evaluated points and contains points which, after being chosen as center points in $N_{\text{fail}}$ algorithm iterations, did not induce an improvement in the exploration-exploitation trade-off. Details of the process of how points are added and removed from the Tabu List are later discussed in Step 2.4 of SOP. The premise behind maintaining a Tabu List is to induce adaptive learning within the SOP search mechanism that could ensure that points which do not induce improvement in the exploration-exploitation trade-off after repeated candidate searches are identified and subsequently removed from consideration as search centers in subsequent algorithm iterations. This selection condition is referred to as the **Tabu Rule**.

**The second selection criteria** is that a point is only selected as a center if its distance from every point that is already selected as a center is greater than the candidate search neighborhood sampling radius of the selected center. The candidate search neighborhood sampling radius for all evaluated points is adaptive, with an initial value equal to $r_{\text{int}}$. The effect of changing this parameter was examined in [20]. Their results suggest that on more complicated surfaces a relatively large initial search radius is more effective since it allows the algorithm to explore a large region initially before focusing the search on a smaller region. Step 2.4 of SOP explains how the value of the candidate search neighborhood of a center point adapts. This selection criteria maintains the notion of exploration within the center selection framework by not selecting a point as a center if it lies within the candidate search sampling radius of an already selected center of the current algorithm iteration. This selection condition is referred to as the **Radius Rule**.

[3] Pseudo-code describing the $P$ center selection procedure can be found in Figure A.3 (P_centers_sel) of the Online Supplement.

At the end of this process, we will have obtained $C = \{c_1, \ldots, c_P\}$, the set of $P$ selected centers.

Figure 3 illustrates an example of the center selection process. First the points in $S^{(n)}$ are sorted according to the two-tier ranking system (Fig. 3b). In this example, four center points ($P = 4$) are selected sequentially from the Ranked List. First, the point A1 on the first front is selected because A1 has the minimum value of $F_1(x)$. Since A2 is within the candidate search neighborhood radius of A1, and A3 is in the Tabu List, neither of these two points are selected as centers. We move on to the second front and the next point that will be selected is therefore B1 (since it has the lowest objective function $F_1(x)$ value on this front). Continuing with the selection process, the four points that satisfy both the Radius Rule and the Tabu Rule are A1, B1, B3, and C2, and are thus selected as centers, and the process stops.

In the special case where all points in the Ranked List have already been examined but the number of selected centers is less than $P$ (the number of centers needed), we re-examine points in the Ranked List with only the Radius Rule imposed. If the number of centers selected is still less than $P$, the next $P - i$ centers, $c_{i+h}$, $h = 1, .., P - i$, will be selected iteratively by cycling through the set of already selected centers $\{c_1, ..., c_i\}$, i.e. $c_{qi+l} = c_l$ for $q \geq 1$ and $1 \leq l \leq i$, until a total of $P$ centers are selected. This is because good solutions often lie in a very small neighborhood. Therefore, it may be beneficial to focus the local candidate search around those promising centers.

### Step 2.3 candidate search

Once $P$ center points have been selected, in Step 2.3 of SOP, simultaneous surrogate-assisted candidate searches are performed around each center for selection of new points for expensive evaluation. For every center point, one new point is selected for expensive evaluation after candidate search, and subsequently evaluated.

The candidate search around each of the $P$ center points is performed in parallel, within a synchronous Master-slave parallel framework. The master process selects the $P$ center points (Step 2.2) and sends one center point to each slave process. There are $P$ slave processes. Let us assume that the center $c_i$ is sent to slave process $i$ for a fixed $i \in \{1, \ldots, P\}$. Slave process $i$ performs a surrogate-assisted candidate search around the center point $c_i$ to select a new evaluation point, $x_i^{\text{new}}$, for expensive evaluation before returning the evaluated point to the master process. The rest of this section provides a description of the mechanism of the surrogate-assisted candidate search used to select $x_i^{\text{new}}$ from a center $c_i$.

The surrogate-assisted candidate search mechanism employed in SOP is referred to as the **candidate search method**, and is based on the idea of a candidate point method in which we randomly generate a large number of candidate points around the center $c_i$, and then evaluate them on the surrogate. Müller and Shoemaker [14] compared the candidate point method to the alternative, which is to search for a minimum with an optimization method on the surrogate surface. Based on comparison of a number of test functions and surrogate types, they found no statistical difference between overall algorithm performance when the surrogate surface is searched either with candidate points or with an optimization search such as Genetic Algorithm.

In the candidate search method, the center point $c_i$ is perturbed to generate a set of $N_{\text{cand}}$ candidate points, $\{v_1, ..., v_{N_{\text{cand}}}\}$.[4] For every candidate point $v_j$ where $j = 1, \ldots, N_{\text{cand}}$, we randomly select the dimensions of $c_i$ that will be perturbed, where each coordinate of the center $c_i$ has a probability of $\varphi(n)$ to be perturbed. If no dimension $k$ of $c_i$ is selected for

[4] Pseudo-code describing the candidate search method can be found in Figure A.4 (candidate_search) of the Online Supplement.

perturbation, then one dimension is chosen at random. Any candidate point is then generated by perturbing the selected variables of the center.

The algorithm updates $\varphi(n)$ by

$$\varphi(n) = \varphi_0 \times [1 - \ln(nP + 1)/\ln(\text{MAXIT} \times P)], \tag{3}$$

for all $0 \leq n \leq \text{MAXIT} - 1$, where $\varphi_0 = \min(20/d, 1)$, and MAXIT is the maximum number of optimization iterations.

Equation 3 is adapted from [20] with $n$ denoting here the number of optimization iterations rather than the number of function evaluations. The rationale for the use of $\varphi(n)$ is to reduce the number of dimensions that are perturbed at any iteration. $\varphi(n)$ gets smaller with each iteration, gradually reducing the expected number of dimensions perturbed. An intuitive explanation for why this approach is effective is that after many iterations, the solution is possibly quite a good solution so one probably does not want to change the values in all of a large number of dimensions at once. For many black-box problems the sensitivity of the objective function to different decision variables may vary, and hence perturbing only a subset of decision space may improve search efficiency. The efficiency of this approach has been demonstrated on a related algorithm in [20], but the evidence supporting that efficiency increases by reducing the number of perturbed dimensions is empirical, and is based on the test problems previously studied.

Two candidate generation methodologies, using truncated normal and uniform random perturbation, are employed within SOP and are referred to as nSOP and uSOP, respectively.

Let $I_{\text{perturb}}$ be the subset of dimensions of $c_i$ that are selected to be perturbed, and define $\text{ub}(k)$ and $\text{lb}(k)$ as the upper and lower bounds for values of variables of dimension $k$ in the domain. The following explains how a candidate point $v_j$ of the two versions of SOP is generated via random perturbation of the center point $c_i$:

**V1** nSOP: $v_j = c_i + Z$, where $Z(k) \sim N_{\text{truncated}}\left(0,\, r_i^2;\, a(k),\, b(k)\right)$, $a(k) = \text{lb}(k) - c_i(k)$ and $b(k) = \text{ub}(k) - c_i(k)$ for $k \in I_{\text{perturb}}$, and $Z(k)$ is zero for $k \notin I_{\text{perturb}}$. See Online Supplement for a description of $N_{\text{truncated}}$, the truncated normal distribution.

**V2** uSOP: $v_j(k) = \mathcal{U}[a(k),\, b(k)]$ where $a(k) = \max(\text{lb}(k),\, c_i(k) - r_i)$, $b(k) = \min(c_i(k) + r_i,\, \text{ub}(k))$ for $k \in I_{\text{perturb}}$, and $v_j(k) = c_i(k)$ for $k \notin I_{\text{perturb}}$. Here, $\mathcal{U}[u_1,\, u_2]$ is a uniform random variable on $[u_1,\, u_2]$.

The perturbation neighborhood of $c_i$ in both of the above random candidate generation methods is dependent upon the candidate search neighborhood radius $r_i$. This radius $r_i$ is adaptive. Step 2.4 of SOP provides an overview of how the value adapts for each center point.

After $N_{\text{cand}}$ candidate points are generated, the candidate point whose estimated objective function value (from the surrogate model) is smallest is selected as the next function evaluation point $x_i^{\text{new}}$. Because the surrogate evaluation is inexpensive, this calculation is very fast, even for large $N_{\text{cand}}$. Finally, the slave process evaluates (via expensive simulation) and returns both $x_i^{\text{new}}$ and its function value $f\left(x_i^{\text{new}}\right)$ to the master process.

### Step 2.4 adaptive learning and tabu archive

Once new points have been selected and evaluated through simultaneous candidate searches in Step 2.3, Step 2.4 of SOP archives the performance of the search at the end of the current algorithm iteration. This is done for adaptive learning and for subsequently changing the search mechanism of the algorithm.

There are two core components of the adaptive learning process of SOP. In the first component, the algorithm evaluates the candidate search around each center point $c_i$ as a

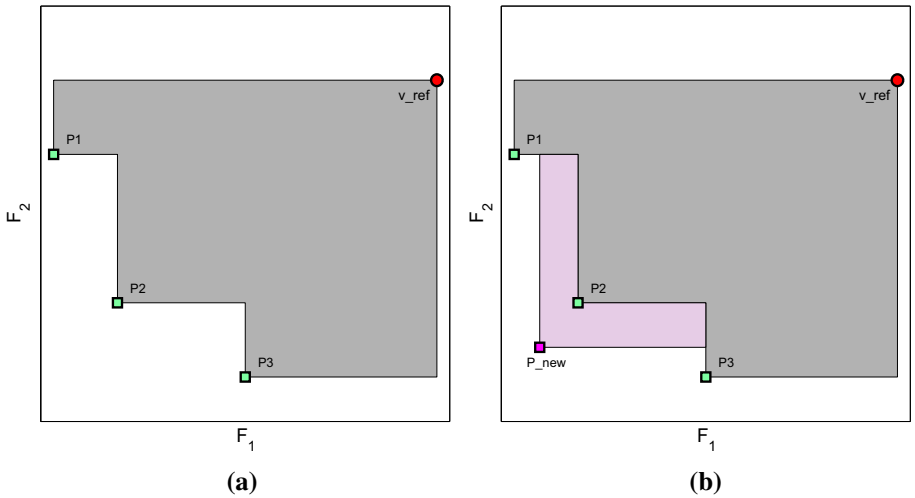


**Fig. 4** Example of hypervolume improvement. The initial Pareto front is {P1, P2, P3}. The hypervolume corresponding to this Pareto front is the shaded area in (**a**). After adding the new point, the Pareto front becomes {P1, P_new, P3}. The light area in (**b**) is the improvement made after the new point is added. **a** Hypervolume. **b** Hypervolume improvement

success or a failure so as to (1) update the candidate search neighborhood radius $r_i$ of the center point $c_i$, and (2) update the failure count corresponding to the center point.

Candidate search around a center point $c_i$ is considered a success if $x_i^{\text{new}}$ induces a significant improvement in the exploration-exploitation trade-off defined in Eq. 2. The hypervolume improvement metric is employed to quantitatively measure this improvement.

Let $S_{\text{Pareto}}^{(n)}$ be the set of points in the decision space on Front 1 (Pareto set) based on the bi-objective problem in Eq. 2 derived from the set $S^{(n)}$.

The decision about which points on the Ranked List will be made tabu will be made on basis of the quantitative assessment of the performance of the best candidate point generated around that point in the past. This assessment is based on the hypervolume. The hypervolume of $S_{\text{Pareto}}^{(n)}$ is the area of the objective space that is dominated by $S_{\text{Pareto}}^{(n)}$. Hypervolume improvement of $x_i^{\text{new}}$ is simply the difference in the hypervolume of previously evaluated points (non-dominated) with and without the newly evaluated point:[5]

$$\text{HI}_i = \text{HV}\left(S_{\text{Pareto}}^{(n)} \cup x_i^{\text{new}}\right) - \text{HV}\left(S_{\text{Pareto}}^{(n)}\right), \tag{4}$$

where $\text{HV}(S)$ refers to the hypervolume of a set $S$, and $\text{HI}_i$ refers to the hypervolume improvement of $x_i^{\text{new}}$ corresponding to the center $c_i$. Figure 4 shows an example of the hypervolume improvement in the objective space.

If $x_i^{\text{new}}$ does not generate significant hypervolume improvement, i.e. $\text{HI}_i$ is smaller than a pre-defined threshold $\tau$, then the candidate search around $c_i$ is tagged as a failure. In this case the algorithm search adapts via reduction of candidate search neighborhood of the center point $c_i$ by a factor of two. This is done to concentrate the search around closer to this center point, since a wider search radius did not induce an improvement. Furthermore, the failure count of the center point is increased by one. If the failure count goes beyond a

[5] Pseudo-code for hypervolume improvement can be found in Figure A.5 (hypervol_improv_indc) of the Online Supplement.

pre-defined threshold, $N_{fail}$, the center point is added to the Tabu List, as it did not produce an improvement in the exploration-exploitation trade-off after multiple candidate searches.[6]

However, we do not wish to keep a center point in the Tabu List forever. This is because as the algorithm progresses and new points are evaluated, the surrogate model is updated and it may be beneficial to reconsider a tabu point for center selection. Hence, SOP also incorporates a mechanism for removing a point from the Tabu List. In the second component of the adaptive mechanism, the Tabu List is updated, where a center point is removed from the Tabu List if it has been in the Tabu List for the last $N_{\text{tenure}}$ algorithm iterations. $N_{\text{tenure}}$ is referred to as the **tenure length** and is an algorithm parameter.

### 2.2 Convergence of nSOP

**Theorem 1** *Suppose that* $x^* = \min_{x\in\mathcal{D}} f(x) > -\infty$ *is the unique global minimizer of* $f$ *in* $\mathcal{D}$ *such that* $\min_{x\in\mathcal{D},\, \|x-x^*\|\geq\eta} f(x) > f(x^*)$ *for all* $\eta > 0$. *If* $\inf_{n\geq 0} \varphi(n) > 0$, *nSOP converges almost surely to the global minimum.*

The proof of convergence of the nSOP algorithm can be intuitively explained. First, in each iteration, each dimension of the $P$ centers has a bounded-away-from-zero probability of being perturbed. Second, the range of sampling for a variable is a truncated normal distribution covering the entire compact hyperrectangle domain $\mathcal{D}$. Finally, the variance of the normal distribution (perturbation distribution) is bounded above zero because it can only be reduced in half at most $N_{\text{fail}}$ times. Based on these three conditions, the probability of trial points being inside a ball with a small radius centered at any point $x$ in the compact domain is bounded away from zero. Hence, as the number of iterations goes to infinity, the sequence of random trial points visited by the algorithm is dense in $\mathcal{D}$. The proof in the Online Supplement addresses the convergence of stochastic nSOP by building on an earlier proof with only one center [18], which is in turn based on Theorem 2.1 in [24] for stochastic algorithms.

This argument that the trial points form a dense set does not apply to the uSOP version. This is because when a dimension is perturbed, the range of perturbation does not cover the whole domain $\mathcal{D}$, so every ball around a point in the domain does not necessarily have a positive probability of being sampled.

## 3 Numerical experiments

### 3.1 Alternative parallel optimization algorithms

We compare our algorithm to Parallel Stochastic RBF [19] and an evolutionary algorithm that uses radial basis function approximation (ESGRBF) [17,21]. Parallel Stochastic RBF was used to solve an optimization problem arising in groundwater bioremediation introduced in [13], and ESGRBF was used to calibrate computationally expensive watershed models. Regis and Shoemaker reported good results for their Parallel Stochastic RBF method compared to many alternative methods including asynchronous parallel pattern search [8] and a parallel evolutionary algorithm. ESGRBF was shown to significantly outperform conventional Evolution Strategy (ES) algorithms and ES algorithms initialized by SLHDs [17]. We thus compare our algorithms with these two methods.

[6] Pseudo-code for tabu archiving can be found in Figure A.6 (update_tabu_archive) of the Online Supplement.

**Table 1** Parameter values for SOP

| Parameter | Value | Corresponding step |
|---|---|---|
| $N_{\text{cand}}$ | min (500d, 5000) | Step 2.3 |
| $r_{\text{int}}$ (nSOP) | $0.2 \times l\,(\mathcal{D})$ | Steps 2.2–2.4 |
| $r_{\text{int}}$ (uSOP) | $0.1 \times l\,(\mathcal{D})$ | Steps 2.2–2.4 |
| $N_{\text{fail}}$ | 3 | Step 2.4 |
| $N_{\text{tenure}}$ | 5 | Step 2.4 |
| $\tau$ | $10^{-5}$ | Step 2.4 |

### 3.2 Experimental setup

All algorithms are run with $P = 8$ and $P = 32$ function evaluations per iteration. We use the notation $\mathcal{A}$-8P and $\mathcal{A}$-32P to distinguish the number of function evaluations per iteration of algorithm $\mathcal{A}$. For example, Parallel StochRBF algorithm simulating 8 function evaluations per iteration is denoted by StochRBF-8P.

We use a cubic RBF interpolation model [16] for the surrogate and Latin hypercube sampling [27] to generate the initial evaluation points for all three examined algorithms. The size of the initial experimental designs was set to $n_0 = \min\{n \in \mathbb{N} : n \geq 2(d+1) \text{ and } P|n\}$, which is the smallest integer larger than $2(d+1)$ and divisible by $P$. The number $2(d+1)$ has previously been used and shown to be an efficient size for initial experimental data set (see e.g. [18,19]).

The definition of $n_0$ is based on the assumption that (1) each function evaluation takes approximately the same time and (2) $P$ parallel processors are available so that the $P$ expensive evaluations can be distributed to these $P$ processors to use the available computing power efficiently.

We do ten trials for each algorithm and each test problem. All algorithms use the same initial experimental design in order to facilitate a fair comparison. Table 1 summarizes the values of the algorithm parameters for SOP. $l\,(\mathcal{D})$ denotes the length of the shortest side of the hyperrectangle $\mathcal{D}$. For Parallel StochRBF, all algorithm parameter values are set as recommended in [19]. For ESGRBF, the parameters are set to $\mu = 4$, $\lambda = 20$, and $\nu = 8$ for $P = 8$ and $\mu = 14$, $\lambda = 80$, and $\nu = 32$ for $P = 32$.

### 3.3 Test functions

We use ten benchmark functions taken from the BBOB test suite [7], namely the functions F15–F24. All of them are 10 dimensional problems and multimodal. The functions are listed in the Online Supplement. For definition and properties of these functions, refer to [7].

### 3.4 Progress curve in wall-clock time

Although the test functions are computationally inexpensive, we assume that each function evaluation takes one hour computation time and that other computational overhead arising, for example, from updating the response surface is negligible. When the objective functions are computationally expensive and function evaluations are done in parallel, the stopping criterion for the optimization is typically a given limit on the allowable wall-clock time.

Under the assumption that $P$ function evaluations are simulated simultaneously in each unit of wall-clock time, we plot a progress curve as a function of wall-clock time. The progress curve enables us to compare the performance of the different algorithms over a range of the allowable computation time.

We set the maximum wall-clock time to be 60 h. The total number of function evaluations ($= 60P$) for $P = 8$ and $P = 32$ will then be 480 and 1920, respectively.

### 3.5 Experimental results and discussion

Figure 5 shows the progress curves of selected test functions. The mean of the best objective function value is plotted on the vertical axis and the wall-clock time is plotted on the horizontal axis. The figure shows both the case for doing simultaneously 8 expensive evaluations and 32 parallel evaluations for a comparison. For each test function, in addition to the main plot, the last 15 iterations are plotted separately in the small sub-figure so that the tail of the plot before the algorithm terminates can be seen clearly.

Overall, SOP together with either normal or uniform strategies leads to a very good performance, clearly outperforming the other two alternative methods. We also find that for $P = 8$, StochRBF-8P performs better than ESGRBF-8P. However, when $P = 32$, ESGRBF-32P performs better than StochRBF-32P.

When doing more function evaluations per iteration, it should be expected that the algorithm improves the objective function value in less wall-clock time since more information is obtained for refining the response surface. However, in some cases, SOP-8P outperformed the alternative methods that do 32 evaluations per iteration. For example, for function F15, nSOP-8P and uSOP-8P with 8 processors got better results in a shorter wall-clock time than StochRBF-32P and ESGRBF-32P. For function F16, both ESGRBF-8P and ESGRBF-32P are worse than nSOP-8P, uSOP-8P, and StochRBF-8P. In addition, ESGRBF-8P even outperformed ESGRBF-32P after around 15 h indicating that parallel ESGRBF is not efficient for higher number of processors on some functions.

As for functions F21 and F22, nSOP and uSOP converge fastest and to a better final solution. Moreover, nSOP-8P and uSOP-8P again outperformed both StochRBF-32P and ESGRBF-32P by finding a lower objective function value in less wall-clock time. StochRBF-32P did very poorly on these two test functions and StochRBF-8P even surpassed StochRBF-32P in both functions.

In contrast to Stochastic RBF and ESGRBF, SOP with 32 processors always found the solution with lower value in less wall-clock time than for SOP with 8 processors, indicating that SOP was much more efficient at utilizing higher number of processors.

Although Parallel StochRBF was shown to work well in [19], we find that our method outperformed it here. While SOP sophisticatedly selects various centers for generating candidate points using the Pareto trade-off strategy, Parallel StochRBF uses the current best point as the only center and generates only one batch of candidate points from which the next $P$ function evaluation points are selected. We assume this is why Parallel StochRBF is not as effective when $P$ is large.

Additional results can be found in the Online Supplement. In particular, there is a comparison of the number of problems on which nSOP or uSOP is significantly better than alternative algorithms in terms of the final solution. In this regard, both nSOP and uSOP are significantly better than Parallel StochRBF on three and six (while worse on one and no) problems when using 8 and 32 processors, respectively. We note that these results in the Online Supplement are based only on the final values at the maximum number of iterations. However, looking at

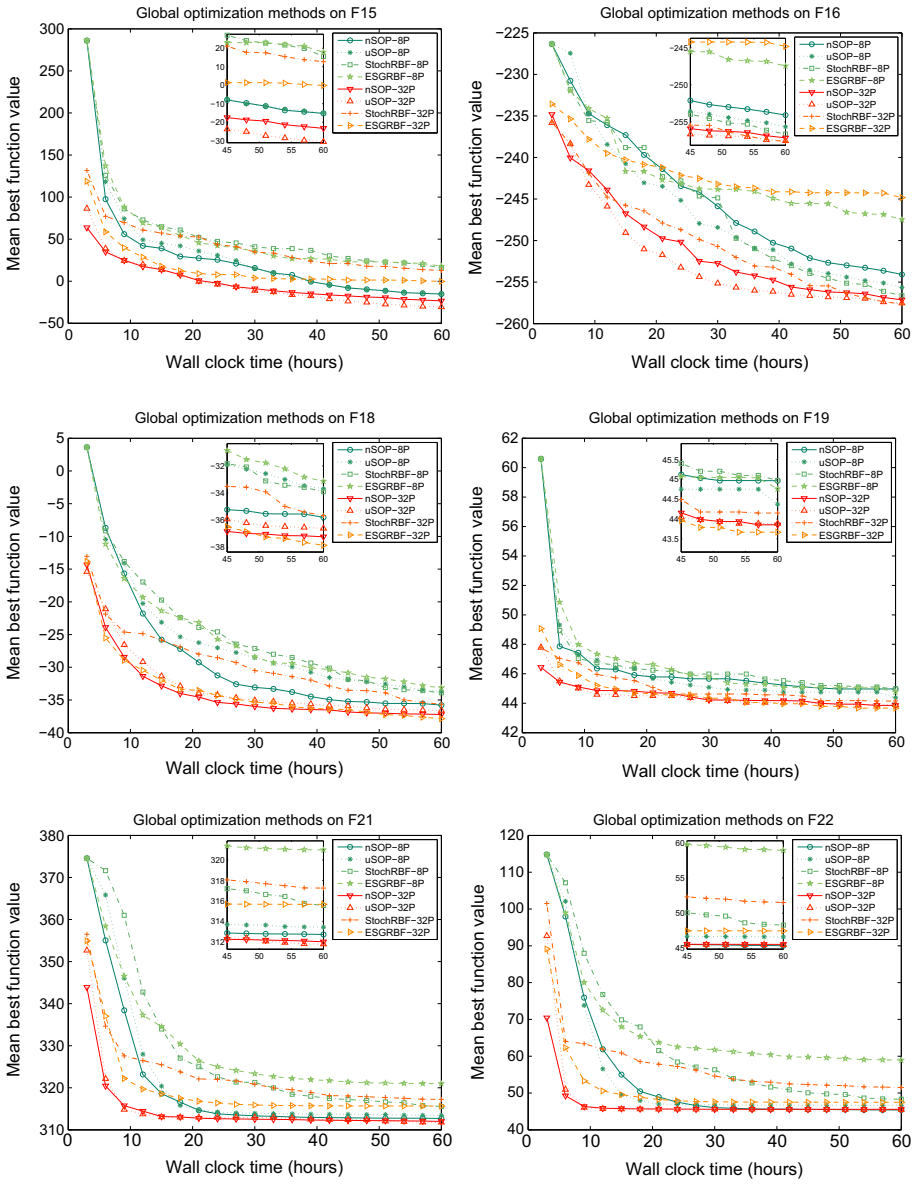


**Fig. 5** Best objective function value found averaged over ten trials versus wall-clock time. Eight and thirty-two points are simulated per iteration. Assume that each function evaluation takes 1 h

the progress curves (Fig. 5), the reader can see that in many cases SOP could achieve accurate solutions much more quickly (e.g. less than 20 h of wall-clock time) than the other methods, which is not reflected in these tables.

In addition to the BBOB testbed, we also tested our algorithms on a 12-dimensional problem arising in the detoxification of contaminated groundwater using aerobic bioremediation [28]. The description of this problem is given in the Online Supplement. The parameters

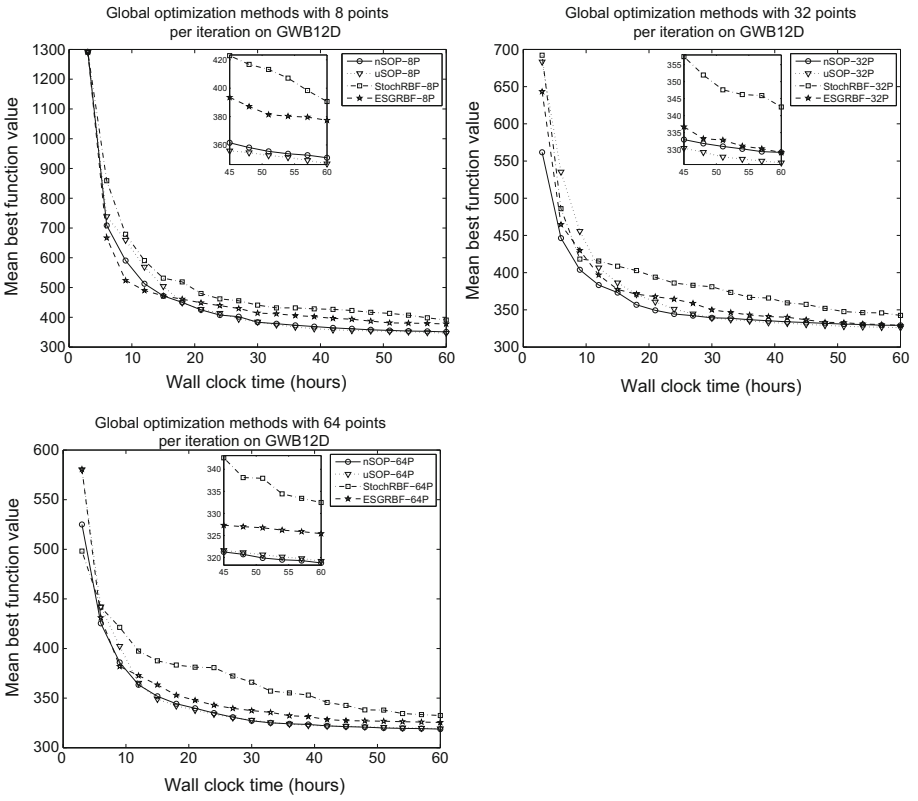


**Fig. 6** Best objective function value on groundwater problem GWB12D averaged over ten trials versus wall-clock time. Eight (*top left*), thirty-two (*top right*) and sixty-four (*lower left*) expensive evaluations are done simultaneously per iteration

$\mu = 26$, $\lambda = 160$, and $\nu = 64$ are set for ESGRBF-64P. All other algorithm parameters are the same as those used for BBOB testbed, and the results are given in Fig. 6.

From Fig. 6, both nSOP and uSOP can yield a faster reduction in function values than the alternative methods at each wall-clock time unit. The mean and the standard deviation of the best function value after 60 iterations are also reported in Table 2. We see that SOP can reach the lowest average final value with the smallest standard deviation in all cases (uSOP is best on $P = 8, 32$ while nSOP is best on $P = 64$).

We carry out a Mann-Whitney-Wilcoxon test to compare the final results of GWB12D obtained in Table 2(a). For any two algorithms $\mathcal{A}_1$, $\mathcal{A}_2$, we write $\mathcal{A}_1 \approx \mathcal{A}_2$ if $\mathcal{A}_1$ and $\mathcal{A}_2$ are not significantly different and write $\mathcal{A}_1 \prec \mathcal{A}_2$ if $\mathcal{A}_1$ is significantly better than $\mathcal{A}_2$ at the 5 % level of significance. The statistical results are summarized in Table 2(b). As can be seen, the results indicate that SOP can reduce the function values faster and achieve a better final solution than the alternative methods.

### 3.6 Relative speedup

To compute parallel speedups for computationally expensive functions, we need to be able to compare the number of function evaluations an algorithm required to reach a solution of a

**Table 2** Results for GWB12D after 60 h

| Alg | nSOP | | uSOP | | StochRBF | | ESGRBF | |
|---|---|---|---|---|---|---|---|---|
| *P* | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| *(a) Mean and standard deviation of the best function value after 60 h for P = 8, 32 and 64 processors on GWB12D. The best value for each P is shown in bold font* | | | | | | | | |
| **8** | 350.964 | 17.940 | **346.727** | **15.606** | 390.614 | 31.145 | 377.373 | 27.752 |
| **32** | 329.166 | 8.034 | **326.112** | **5.016** | 342.550 | 13.997 | 329.128 | 9.875 |
| **64** | **318.774** | **2.465** | 319.182 | 3.658 | 332.473 | 4.390 | 325.420 | 8.837 |
| *P* | Mann–Whitney–Wilcoxon test results | | | | | | | |
| *(b) Mann–Whitney–Wilcoxon test results for GWB12D at 5 % significance level* | | | | | | | | |
| **8** | nSOP ≈ uSOP ≺ ESGRBF ≈ StochRBF | | | | | | | |
| **32** | nSOP ≈ uSOP ≈ ESGRBF ≺ StochRBF | | | | | | | |
| **64** | nSOP ≈ uSOP ≺ ESGRBF ≺ StochRBF | | | | | | | |

certain accuracy. The results can vary according to the accuracy level, which we call $\alpha-$level. We define

$$\alpha\text{-Speedup(P)} := \mathcal{I}^{(\alpha)}(1)/\mathcal{I}^{(\alpha)}(P) = n^{(\alpha)}(1)/\left\lceil n^{(\alpha)}(P)/P \right\rceil, \tag{5}$$

where

$$n^{(\alpha)}(P) = \text{argmin}_i\{f(x_i) \leq \alpha \text{ given } P \text{ processors}\} \tag{6}$$

and

$$\mathcal{I}^{(\alpha)}(P) = \left\lceil n^{(\alpha)}(P)/P \right\rceil. \tag{7}$$

So $n^{(\alpha)}(P)$ is the number of function evaluations and $\mathcal{I}^{(\alpha)}(P)$ is the number of iterations that an algorithm required to reach a specified $\alpha-$level of accuracy. Note that SOP is not designed to run in serial. If SOP selects only one center per iteration, the point with the lowest objective function value will be selected as the center in each iteration regardless of the results obtained from the Pareto non-dominated sorting. We thus use Stochastic RBF, which was proven to be very efficient [18] as a serial algorithm to compute $\mathcal{I}^{(\alpha)}(1)$ and $n^{(\alpha)}(1)$.

The *relative* $\alpha$-Speedup(P) for each test function is calculated for different $\alpha$-levels, $\alpha_1 > \alpha_2 > \alpha_3$ of accuracy. To get the three $\alpha$-levels of nSOP for test functions in BBOB testbed, nSOP(8) and Stochastic RBF are run for 496 function evaluations and nSOP(32) for 1984 function evaluations. Let $y_1^*$, $y_8^*$, and $y_{32}^*$ be the average best objective function values obtained from Stochastic RBF, nSOP(8) and nSOP(32), respectively. We set $\alpha_3 = \max\left\{y_1^*, y_8^*, y_{32}^*\right\}$, $\alpha_2 = \alpha_3 + |\alpha_3| \times 0.01$ and $\alpha_1 = \alpha_3 + |\alpha_3| \times 0.05$. For the GWB12D test function, we ran in addition nSOP(64) for 3968 function evaluations. Let $y_{64}^*$ be the average best objective function values obtained from nSOP(64), and define $\alpha_3 = \max\left\{y_1^*, y_8^*, y_{32}^*, y_{64}^*\right\}$, $\alpha_2 = \alpha_3 + |\alpha_3| \times 0.01$ and $\alpha_1 = \alpha_3 + |\alpha_3| \times 0.05$.

We calculate also three $\alpha$-levels for uSOP using the same approach as in nSOP. The $\alpha$-levels for nSOP and uSOP are given in the Online Supplement. Table 3 shows the $\alpha$-Speedup(P) values.

The algorithm results will change as $P$ changes since $P$ centers are used to generate candidate points and the response surface is updated only once every $P$ evaluations. On some problems, the change is quite helpful and actually reduces the number of evaluations

**Table 3** $\alpha$-Speedup of nSOP and uSOP

| Test function | $P$ | nSOP | | | uSOP | | |
|---|---|---|---|---|---|---|---|
| | | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ |
| F15 | 8 | 11.805 | 11.976 | 12.195 | 11.524 | 11.690 | 11.905 |
| | 32 | 19.360 | 19.640 | 20.000 | 20.167 | 20.458 | 20.833 |
| F16 | 8 | 2.318 | 2.545 | 3.159 | 3.105 | 3.182 | 4.889 |
| | 32 | 5.100 | 4.308 | 5.378 | 6.556 | 5.600 | 9.059 |
| F17 | 8 | 10.333 | 23.706 | 27.500 | 8.857 | 21.211 | 24.750 |
| | 32 | 24.800 | 57.571 | 61.875 | 20.667 | 44.778 | 49.500 |
| F18 | 8 | 17.529 | 23.000 | 24.800 | 12.417 | 15.607 | 17.103 |
| | 32 | 37.250 | 48.556 | 55.111 | 29.800 | 39.727 | 45.091 |
| F19 | 8 | 8.800 | 6.500 | 5.689 | 7.600 | 9.639 | 8.267 |
| | 32 | 29.333 | 37.143 | 34.700 | 22.800 | 31.545 | 19.840 |
| F20 | 8 | 5.636 | 5.333 | 2.032 | 6.200 | 6.400 | 2.143 |
| | 32 | 20.667 | 16.000 | 3.048 | 15.500 | 12.800 | 3.649 |
| F21 | 8 | 3.818 | 14.056 | 19.192 | 3.500 | 14.056 | 10.848 |
| | 32 | 8.400 | 31.625 | 38.385 | 7.000 | 31.625 | 38.385 |
| F22 | 8 | 12.391 | 10.258 | 10.578 | 17.375 | 13.714 | 4.698 |
| | 32 | 40.714 | 31.800 | 8.207 | 39.714 | 36.000 | 32.889 |
| F23 | 8 | 2.000 | 8.375 | 1.746 | 2.000 | 13.800 | 2.651 |
| | 32 | 2.000 | 33.500 | 5.000 | 2.000 | 23.000 | 4.071 |
| F24 | 8 | 7.275 | 9.226 | 8.197 | 6.577 | 7.220 | 7.267 |
| | 32 | 13.741 | 14.382 | 14.706 | 24.429 | 20.286 | 18.957 |
| GWB12D | 8 | 3.850 | 3.255 | 3.000 | 4.289 | 3.345 | 3.397 |
| | 32 | 9.059 | 8.950 | 9.000 | 8.150 | 7.462 | 7.926 |
| | 64 | 12.833 | 11.933 | 11.812 | 12.538 | 12.125 | 12.588 |

required to reach an $\alpha$-level. In this case, the speedup is "superlinear", i.e. $\alpha$-Speedup(P) is greater than $P$. The superlinear speedup holds at many $\alpha$-levels for problems F17, F18, F21, and F22. F17 is Schaffers function, which is highly multimodal– both the frequency and amplitude of the modulation vary. F18 is a moderately ill-conditioned counterpart to F17, with conditioning of about 1000. Both F21 and F22 are Gallagher's Gaussian function, where F21 consists of 101 local optima, and the conditioning around the global optimum is about 30. F22 consists of 21 local optima and the conditioning around the global optimum is about 1000 [7].

The three functions that have rather poor scalability (i.e. speedup low compared to $P$) are F16, F20, and F23. These functions are described in [7] as being highly rugged and moderately to highly repetitive.

For F15, the superlinear speedup holds for 8 processors but not for 32 processors. As for F19, and F24, although they do not achieve a superlinear speedup, we can see that the speedup seems to improve in the number of processors. These three functions, despite being highly multimodal, are not as rugged as those mentioned in the previous paragraph. The global amplitude of F15 is large compared to local amplitudes. As for F24, the function was constructed to be deceptive for some evolutionary algorithms with large population size. This might be the reason why the scalability of SOP on this test function deteriorates with 32 processors.

Finally, we do not get good scalability for the groundwater bioremediation problem. The speedup is around 3, 9, and 12 for 8, 32, and 64 processors. Recall however none of the other algorithms did as well as SOP on this black-box problem (Fig. 6).

### 3.7 Sensitivity of SOP to $N_{\text{fail}}$

A larger $N_{\text{fail}}$ allows the algorithm to search longer around a particular center with no improvement until it is declared as tabu (search more locally). On the other hand, a small $N_{\text{fail}}$ allows the algorithm to leave the region around centers with no improvement earlier (search more globally).

To examine the effect of changing $N_{\text{fail}}$, SOP is implemented also with $N_{\text{fail}} = 0$ (which is the extreme case where any unimproved center is immediately tabu) and $N_{\text{fail}} = 10$ (longer length before being tabu), while our default implementation is $N_{\text{fail}} = 3$. Figure 7 shows the comparison between these implementations on some problems with nSOP-8P and uSOP-8P.

The results suggest that the best $N_{\text{fail}}$ varies from problem to problem. For example, for uSOP-8P (Fig. 7b), on the test function F18, the algorithm implemented with $N_{\text{fail}} = 10$ results in faster progress while the algorithm with the default $N_{\text{fail}} = 3$ is better on the test function F15. In addition, for the same test problem, the best $N_{\text{fail}}$ of nSOP and uSOP can also be different (e.g. F18).

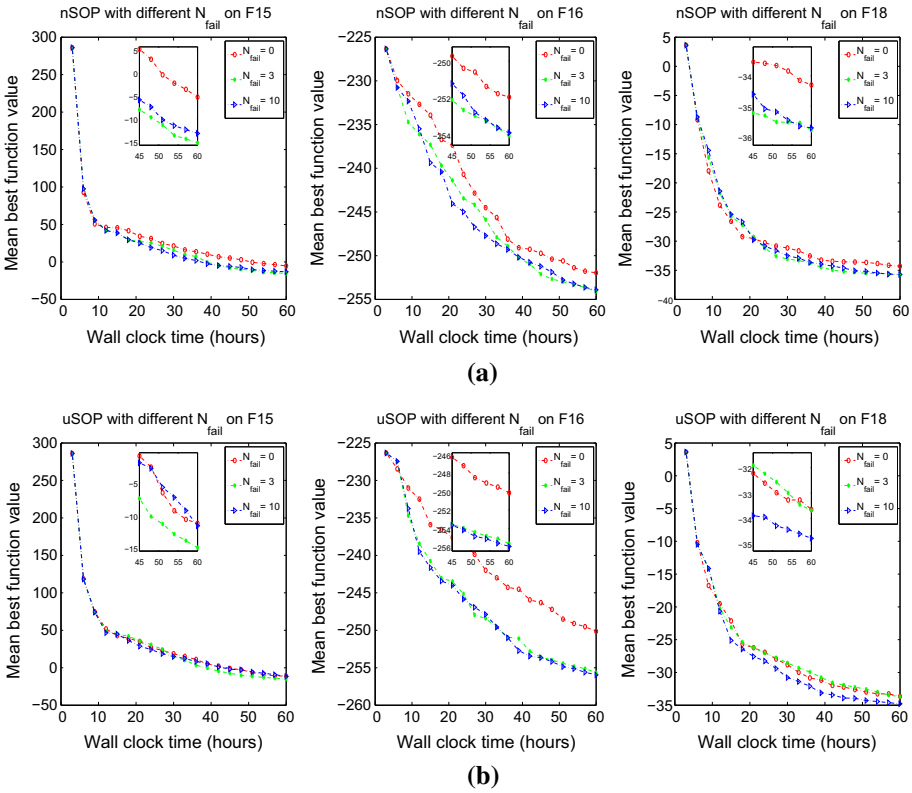


**Fig. 7** Best objective function value found by SOP-8P with different $N_{\text{fail}}$ averaged over ten trials versus wall-clock time. **a** nSOP-8P. **b** uSOP-8P

We briefly discuss the results using a large number of processors without providing the plots. When $P = 32$, similar results are obtained when using $N_{\text{fail}} = 3$ and $N_{\text{fail}} = 10$. This is because when $P$ is large, multiple tasks take place at the same time around each of the $P$ centers, and so the memory archive for tabu list does not matter that much. Also, we found that in one case the result obtained with $N_{\text{fail}} = 0$ is best. Although this is not typical, the reason might be that immediate tabu allows the algorithm to explore the surface more globally.

## 4 Conclusions

Parallel computation has the potential to greatly reduce the wall-clock time to solve a global optimization problem for a computationally expensive objective function, but that potential can only be realized if the parallel algorithm is able to effectively select the work to be computed in parallel. The efficiency of the parallel surrogate algorithms depends on the $P$ decision vectors selected for function evaluations in each iteration to provide the most useful information over the course of many iterations. We want these values to both help improve the surrogate surface accuracy and to help identify the neighborhood of the global minimum.

In this paper we introduced the algorithm, SOP, which is based on the selection of center points through non-dominated sorting. To improve the diversity, promising points on the sorted fronts whose function values are small and are far away from other evaluated points are selected as centers. The selected centers are then used for generating a set of candidate points from which the next function evaluation points are chosen. Multiple centers generate a more diverse set of candidate points. We also incorporate a tabu structure to further diversify the search by not allowing some points on the better fronts to be selected under certain conditions. Two versions of SOP, nSOP and uSOP, relying on the different candidate generation methodologies are proposed. It can be shown that the nSOP algorithm converges to the global optimum almost surely.

In the numerical experiments, we compared SOP with two parallel RBF-based algorithms, Parallel StochRBF and ESGRBF. Our numerical results indicate that overall SOP was more effective than the alternative methods on the synthetic test functions. In some cases, SOP with just 8 processors could obtain a better result in less wall-clock time than the alternative algorithms with 32 processors. The results on a groundwater bioremediation problem using up to 64 processors also showed that our algorithm was significantly better than the alternative methods. Hence, these results are expected to have a positive impact in the field of expensive black-box optimization, which applies to many areas, in particular in (but not limited to) engineering.

**Acknowledgments** This research was conducted primarily when Dr. Krityakierne was a PhD student in Applied Mathematics at Cornell University. During that time, her time and that of Prof. Shoemaker were supported in part by NSF grants 1116298 (CISE) and 1049033 and DOE-SciDAC DE-SC0006791 and by the Fulbright and the King Anandamahidol Foundation of Thailand fellowships to Dr. Krityakierne. All the authors continued to work on the manuscript in their new positions after leaving Cornell. Financial support for completing the manuscript was provided by National University of Singapore.

## References


1. Akhtar, T., Shoemaker, C.A.: Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection. J. Glob. Optim. **64**, 17–32 (2015)
2. Bischl, B., Wessing, S., Bauer, N., Friedrichs, K., Weihs, C.: Moi-mbo: multiobjective infill for parallel model-based optimization. In: Learning and Intelligent Optimization, pp. 173–186. Springer, New York (2014)
3. Booker, A.J., Dennis Jr, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Struct. Optim. **17**(1), 1–13 (1999)
4. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Lect Notes Comput Sci **1917**, 849–858 (2000)
5. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms, vol. 16. Wiley, London (2001)
6. Gutmann, H.-M.: A radial basis function method for global optimization. J. Glob. Optim. **19**, 201–227 (2001)
7. Hansen, N., Finck, S., Ros, R., Auger, A., et al: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions (2009)
8. Hough, P.D., Kolda, T.G., Torczon, V.J.: Asynchronous parallel pattern search for nonlinear optimization. SIAM J. Sci. Comput. **23**(1), 134–156 (2001)
9. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. J. Glob. Optimiz. **34**(3), 441–466 (2006)
10. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
11. Li, Y., Liu, L., Long, T., Chen, X.: Multiple-optima search method based on a metamodel and mathematical morphology. Eng. Optim. **48**(3), 437–453 (2016)
12. Liu, H., Shengli, X., Ma, Y., Wang, X.: Global optimization of expensive black box functions using potential lipschitz constants and response surfaces. J. Glob. Optimiz. **63**(2), 229–251 (2015)
13. Mugunthan, P., Shoemaker, C.A., Regis, R.G.: Comparison of function approximation, heuristic, and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. Water Resour. Res. **41**(11) (2005). doi:10.1029/2005WR004134
14. Müller, J., Shoemaker, C.A.: Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. J. Glob. Optimiz. **60**(2), 123–144 (2014)
15. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased disimpl algorithm for expensive global optimization. J. Glob. Optimiz. **59**(2–3), 545–567 (2014)
16. Powell, M.J.D.: The theory of radial basis function approximation in 1990. In: Advances in Numerical Analysis, vol. 2: Wavelets, Subdivision Algorithms and Radial Basis Functions. Oxford University Press, Oxford, pp. 105–210 (1992)
17. Regis, R.G., Shoemaker, C.A.: Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE Trans. Evol. Comput. **8**(5), 490–505 (2004)
18. Regis, R.G., Shoemaker, C.A.: A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J. Comput. **19**(4), 497–509 (2007)
19. Regis, R.G., Shoemaker, C.A.: Parallel stochastic global optimization using radial basis functions. INFORMS J. Comput. **21**(3), 411–426 (2009)
20. Regis, R.G., Shoemaker, C.A.: Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. Eng. Optim. **45**(5), 529–555 (2013)
21. Shoemaker, C.A., Regis, R.G., Fleming, R.C.: Watershed calibration using multistart local optimization and evolutionary optimization with radial basis function approximation. Hydrol. Sci. J. **52**(3), 450–465 (2007)
22. Sobester, A., Leary, S.J., Keane, A.J.: A parallel updating scheme for approximating and optimizing high fidelity computer simulations. Struct. Multidiscip. Optim. **27**(5), 371–383 (2004)
23. Sóbester, A., Leary, S.J., Keane, A.J.: On the design of optimization strategies based on global response surface approximation models. J. Glob. Optimiz. **33**(1), 31–59 (2005)
24. Spall, J.C.: Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, vol. 65. Wiley, London (2005)
25. Viana, F.A.C., Haftka, R.T., Watson, L.T.: Efficient global optimization algorithm assisted by multiple surrogate techniques. J. Glob. Optimiz. **56**(2), 669–689 (2013)
26. Vrugt, J. A., Robinson, B. A.: Improved evolutionary optimization from genetically adaptive multimethod search, Proc Nat Acad Sci, **104**(3), 708–711 (2007)

27. Ye, K.Q., Li, W., Sudjianto, A.: Algorithmic construction of optimal symmetric latin hypercube designs. J. Stat. Plan. Inference **90**(1), 145–159 (2000)
28. Yoon, J.-H., Shoemaker, C.A.: Comparison of optimization methods for ground-water bioremediation. J. Water Resour. Plan. Manag. **125**(1), 54–63 (1999)