Online Supplement for SOP: Parallel Surrogate Global Optimization with Pareto Center Selection for Computationally Expensive Single Objective Problems

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

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

2: School of Civil and Environmental Engineering, Cornell University, USA

3: Dept. of CEE, National University of Singapore, Singapore

4: Dept. of ISE, National University of Singapore, Singapore

The primary purpose of this supplementary material is to provide (A) pseudo-code of SOP Algorithm; (B) a proof of convergence of nSOP; (C) a definition of truncated Gaussian distribution; (D) descriptions of alternative algorithms; (E) test functions; and (F) additional experimental results.

A Pseudo-code of SOP Algorithm

We describe SOP algorithm presented in the main paper using a pseudo-code format.

Input:

- n_0 - the number of initial evaluation points in experimental design
- N_{cand} - the number of candidate points generated in Candidate Search
- N_{fail} - the maximum number of non-registered improvement before a particular center will be declared as tabu
- N_{tenure} - the tenure length
- r_{int} - the initial search radius
- τ - the tolerance for local improvement
- P - the number of centers per iteration (= the number of function evaluation points per iteration)
- MAXIT - the maximum number of optimization iterations
- f - a continuous real-valued objective function to minimize
- $\mathcal{D} = \{\text{lb} \leq x \leq \text{ub}\} \subset \mathbb{R}^d$ - the hyperrectangle domain of f

Output: The best point found

List of notation used in Algorithm 1:

- $x_i^{(n)}$ the ith evaluation point in nth optimization iteration ($1 \leq i \leq P$)
- $y_i^{(n)}$ the function value corresponding to $x_i^{(n)}$

Algorithm 1 Surrogate Optimization with Pareto center selection (SOP)

1. Set counter $n = 0$. Generate initial points $S^{(0)} = \left\langle x_1^{(0)}, ..., x_{n_0}^{(0)} \right\rangle$ and do function evaluations $y_i^{(0)} = f\left(x_i^{(0)}\right)$ for $i = 1, ..., n_0$. Calculate $d_i = \min\left\{\|s_i - s\| : s \in S^{(0)} \setminus \{s_i\}\right\}$, for $s_i \in S^{(0)}$. Let $T^{(0)} = \left\langle y_1^{(0)}, ..., y_{n_0}^{(0)} \right\rangle$, $D^{(0)} = \langle d_1, ..., d_{n_0} \rangle$ and $R^{(n_0)} = \langle \sigma_1, ..., \sigma_{n_0} \rangle$, where $\sigma_i = r_{\text{int}}$. In addition, let tabu_struct = [tabu_struct(1), tabu_struct(2)], where tabu_struct(1) = tabu_struct(2) = $\langle z_1, ... z_{n_0} \rangle$, where $z_i = 0$ for all $i = 1, ..., n_0$.
2. Define $F = (F_1, F_2) : S^{(0)} \to T^{(0)} \times -D^{(0)}$ by $F(s_i) = (t_i, -d_i)$ for $s_i \in S^{(0)}$, $t_i \in T^{(0)}$ and $d_i \in D^{(0)}$.
3. Iterative Loop (**while** $n <$ MAXIT)
   (a) Fit or update the response surface model $g^{(n)}(\cdot)$.
   (b) Find $[\mathcal{X}_{\text{int}}, \mathcal{P}_{\text{int}}] =$ **pareto_front**$\left(S^{(n)}, F_1, F_2\right)$.
   (c) Select P center points:
      i. $\mathcal{J} = [\mathcal{J}_1, ..., \mathcal{J}_P] =$ **select_P_indices**$(S^{(n)}, T^{(n)}, D^{(n)}, R^{(n)}$, tabu_struct, $P)$.
      ii. For $i = 1, ..., P$, $c_i = s_{(\mathcal{J}(i))} \in S^{(n)}$ is the selected center point and $r_i = \sigma_{(\mathcal{J}(i))} \in R^{(n)}$ the search radius of center i.
   (d) $p_{\textbf{select}} = \varphi(n)$. **for** $i = 1 : P$ **do**
      Generate a new point $x_i^{(n+1)} = ...$
         candidate_search$\left(c_i, r_i, \mathcal{D}, N_{\text{cand}}, g^{(n)}(\cdot), p_{\text{select}}\right)$.
      Evaluate $y_i^{(n+1)} = f(x_i^{(n+1)})$.
      end for
      Update $S^{(n+1)} = S^{(n)} \cup \left\langle x_1^{(n+1)}, ..., x_P^{(n+1)} \right\rangle$ and $T^{(n+1)} = T^{(n)} \cup \left\langle y_1^{(n+1)}, ..., y_P^{(n+1)} \right\rangle$.
      Calculate $d_i = \min\left\{\|s_i - s\| : s \in S^{(n+1)} \setminus \{s_i\}\right\}$ for all $s_i \in S^{(n+1)}$.
      Redefine $D^{(n+1)} = \left\langle d_1, ..., d_{n_0+(n+1)P} \right\rangle$.
      Update $F = (F_1, F_2) : S^{(n+1)} \to T^{(n+1)} \times -D^{(n+1)}$ by $F(s_i) = (t_i, -d_i)$ for $s_i \in S^{(n+1)}$, $t_i \in T^{(n+1)}$, and $d_i \in D^{(n+1)}$.
   (e) **for** $i = 1 : P$ **do**
      $\lambda_i^{(n+1)} = d_{n_0+(nP+i)} \in D^{(n+1)}$.
      $I_i =$ **hypervol_improv_indc**$\left(\mathcal{X}_{\text{int}}, \mathcal{P}_{\text{int}}, F_1, F_2, x_i^{(n+1)}, \left[y_i^{(n+1)}, \lambda_i^{(n+1)}\right], \tau\right)$.
      end for
   (f) Update $R^{(n+1)} = R^{(n)} \cup \langle \sigma_1, ..., \sigma_P \rangle$, where $\sigma_i = r_{\text{int}}$, and tabu_struct(1) = tabu_struct(1) $\cup \langle z_1, ... z_P \rangle$, tabu_struct(2) = tabu_struct(2) $\cup \langle z_1, ... z_P \rangle$, where $z_i = 0$. Set $n = n + 1$.
   (g) Update $\left[\text{tabu\_struct}, R^{(n)}\right] =$ **update_tabu_archive**$\left(\text{tabu\_struct}, I, \mathcal{J}, N_{\text{fail}}, N_{\text{tenure}}, r_{\text{int}}, R^{(n)}, P\right)$.
4. Terminate and return the best point found.

- $S^{(n)}$ the list (ordered set) containing all evaluated points up to nth iteration, i.e. $S^{(n)} = \langle s_1, ..., s_{n_0+nP} \rangle$ $= \left\langle \underbrace{x_1^{(0)}, ..., x_{n_0}^{(0)}}_{\text{initial pts}}, \quad \underbrace{x_1^{(1)}, ..., x_P^{(1)}}_{\text{iteration 1}}, \cdots, \quad \underbrace{x_1^{(n)}, ..., x_P^{(n)}}_{\text{iteration } n} \right\rangle$. Note $\left|S^{(n)}\right| = n_0 + nP$.
- $\lambda_i^{(n)}$ the minimum distance from $x_i^{(n)}$ to the set of already evaluated points $S^{(n)} \setminus \left\{x_i^{(n)}\right\}$
- $T^{(n)}$, $D^{(n)}$, $R^{(n)}$ a similar list to $S^{(n)}$ containing all function values, minimum distances, and search radii.
- $\langle \cdot \rangle$ is used to distinguish a list from a regular set.

We treat $S^{(n)}$, $T^{(n)}$, $D^{(n)}$, $R^{(n)}$ as a list. That is, the order of the members in these lists is important. Step 3 of Algorithm 1 is the iterative step that corresponds to Steps 2.1-2.4 of the General iterative framework of SOP presented in Figure 1 in the main paper.

Function $\mathcal{J} = [\mathcal{J}_1, ..., \mathcal{J}_P] =$ **select_P_indices**$(S^{(n)}, T^{(n)}, D^{(n)}, R^{(n)}, \text{tabu\_struct}, P)$

1. $\left[S^{\text{order}}, R^{\text{order}}, \text{tabu\_struct}^{\text{order}}\right] = ...$
   two_tier_ranking$\left(S^{(n)}, T^{(n)}, D^{(n)}, R^{(n)}, \text{tabu\_struct}\right)$
2. $[c_1, ..., c_P] =$ **P_centers_sel**$\left(S^{\text{order}}, R^{\text{order}}, \text{tabu\_struct}^{\text{order}}, P\right)$
3. For $j = 1, ..., P$, let $\mathcal{J}(j)$ be such that $c_j = s_{\mathcal{J}(i)} \in S^{(n)}$. That is, $\mathcal{J}$ is the P-vector containing the indices of the selected centers corresponding to the original list $S^{(n)}$.

Figure A.1: **select_P_indices** used in Step **3(c)i**

Function $\left[S^{\textbf{order}}, R^{\textbf{order}}, \textbf{tabu\_struct}^{\textbf{order}}\right] = ...$
two_tier_ranking$(S^{(n)}, T^{(n)}, D^{(n)}, R^{(n)}, \text{tabu\_struct})$

1. Sort all the evaluated points in the set $S^{(n)}$ into levels of non-dominated fronts according to the two following (pre-computed) objectives:
   (a) the objective function value (F_1) in $T^{(n)}$;
   (b) the negative of the minimum distance from all other evaluated points (F_2) in $-D^{(n)}$.
2. Let $x^{\text{Front}(i,j)}$ be the jth point of the ith front that is ordered such that for a fixed front i, $x^{\text{Front}(i,j)}$ is in an increasing order of objective function values, i.e. $f\left(x^{\text{Front}(i,1)}\right) \leq f\left(x^{\text{Front}(i,2)}\right) \leq \cdots$.
3. Denote by $S^{\text{order}} = \left\{x^{\text{Front}(1,1)}, x^{\text{Front}(1,2)}, \cdots x^{\text{Front}(2,1)}, x^{\text{Front}(2,2)}, ...\right\}$ the list of points obtained from this procedure starting from Front 1. Also, let R^{order} and $\text{tabu\_struct}^{\text{order}}$ be the sorted $R^{(n)}$ and sorted tabu_struct based on this order.

Figure A.2: **two_tier_ranking** used in **select_P_indices**

Function $[c_1, ..., c_P] =$ **P_centers_sel**$\Big(S^{\text{order}},\, R^{\text{order}},\, \text{tabu\_struct}^{\text{order}},\, P\Big)$

Write $S^{\text{order}} = \{s_{o_1}, s_{o_2}, ..., s_{o_N}\}$
$R^{\text{order}} = \{r_{o_1}, ... r_{o_N}\}$
$\text{tabu\_struct}^{\text{order}}(2) = \{t_{o_1}, ..., t_{o_N}\}$
$c_1 = s_{o_1}$, $\epsilon_1 = r_{o_1}$, $k = 1$
for $i = 2 : P$
$k = k + 1$
Do [Until $(\|c_j - s_{o_k}\| > \epsilon_j \, \forall j = 1, \ldots, i-1)$ and $(t_{o_k} = 0)$**]**
$k = k + 1$
[Exit Do]
$c_i = s_{o_k}$, $\epsilon_i = r_{o_k}$
end for

Figure A.3: **P_centers_sel** used in **select_P_indices**

Function $x =$ **candidate_search**$(c,\, r,\, \mathcal{D},\, N_{\text{cand}},\, g(\cdot),\, p_{\text{select}})$
for $j = 1 : N_{\text{cand}}$
for $i = 1 : d$, $u_i = \text{rand}(1)$, **end for**
$I_{\text{perturb}} = \{i :\, u_i < p_{\text{select}}\}$
if $I_{\text{perturb}} = \emptyset$, $I_{\text{perturb}} = \text{randi}(d)$, **end if**
generate $v_j = \mathcal{L}\,(c,\, r,\, \mathcal{D},\, I_{\text{perturb}})$
end for
$x = \arg\min_{1 \le j \le N_{\text{cand}}} g\,(v_j)$

Figure A.4: **candidate_search** used in Step **3d**

Function HI = **hypervol_improv_indc**$(\mathcal{X}_{\text{int}},\, \mathcal{P}_{\text{int}},\, F_1,\, F_2,\, x_{\text{new}},\, [y_{\text{new}},\, \lambda_{\text{new}}]\,,\, \tau)$
if for some $x \in \mathcal{X}_{\text{int}}$, $x \preceq x_{\text{new}}$
HI= 0
else
$v_{\text{int\_best}} = (\min F_1\,[\mathcal{X}_{\text{int}}]\,,\, \min F_2\,[\mathcal{X}_{\text{int}}])$
$v_{\text{ref}} = (\max F_1\,[\mathcal{X}_{\text{int}} \cup \{x_{\text{new}}\}]\,,\, \max F_2\,[\mathcal{X}_{\text{int}} \cup \{x_{\text{new}}\}])$
$[\mathcal{X}_{\text{new}},\, \mathcal{P}_{\text{new}}] =$**pareto_front**$(\mathcal{X}_{\text{int}} \cup \{x_{\text{new}}\}\,,\, F_1,\, F_2)$
$\mu = \big(\mathbf{HV}(\mathcal{P}_{\text{new}},\, v_{\text{ref}}) - \mathbf{HV}(\mathcal{P}_{\text{int}},\, v_{\text{ref}})\big) / \prod_{i=1}^{2} \big(v_{\text{ref}}(i) - v_{\text{int\_best}}(i)\big)$
$\text{HI} = 1_{\mu > \tau}$
end if

Figure A.5: **hypervol_improv_indc** used in Step **3e**

Function $\left[\text{tabu\_struct}, R^{(n)}\right] = ...$
update_tabu_archive$\left(\text{tabu\_struct}, I, \mathcal{J}, N_{\text{fail}}, N_{\text{tenure}}, r_{\text{int}}, R^{(n)}, P\right)$
$t_1 = \text{tabu\_struct}(1)$
$t_2 = \text{tabu\_struct}(2)$
for $j = 1 : P$
if $I_j = 0$
$t_1\left(\mathcal{J}\left(j\right)\right) = t_1\left(\mathcal{J}\left(j\right)\right) + 1$
$R^{(n)}\left(\mathcal{J}\left(j\right)\right) = R^{(n)}\left(\mathcal{J}\left(j\right)\right)/2$
end if
end for
for $i = 1 : n_0 + (n-1)P$
if $t_2(i) > 0$
$t_2(i) = t_2(i) - 1$
else if $t_1(i) > N_{\text{fail}}$
$t_2(i) = m$
$t_1(i) = 0$
$R^{(n)}\left(i\right) = r_{\text{int}}$
end if
end for

Figure A.6: **update_tabu_archive** used in Step **3g**

B Convergence of nSOP

In this section, we will give some conditions which ensure the global convergence in a probabilistic sense (with probability 1) of nSOP. First, note the main differences between SOP, StochRBF [9] and ParStochRBF [10].

Table B.1: Difference between SOP, StochRBF and ParStochRBF

	StochRBF	ParStochRBF	SOP
# of simulated points per iteration	1	P	P
perturbation center(s)	x_{best}	x_{best}	$c_1, ..., c_P$
# of candidate point batches per iteration	1	1	P

Since ParStochRBF is essentially the same as StochRBF except for the number of points selected for evaluation in each iteration, one can easily apply Theorem 1 in [9] and prove the convergence for ParStochRBF. In contrast, each center of SOP does its own candidate search, resulting in P batches of candidate points per iteration. Moreover, only some of the coordinates of the centers in SOP are selected for perturbation, whereas all coordinates of the center x_{best} in [9] are perturbed for generating candidate points.

A major difference between the proof of Theorem B.4 and the proof in [9] is the multiple centers (indexed by i) and that in [9] all dimensions are perturbed at once, whereas in SOP only some of the dimensions are perturbed which necessitates the definition and use of the class of random variables $\mathcal{F}_{n-1}$ and $\mathcal{Q}_{n-1}$ defined in Definition B.2 below.

Definition B.1. For $1 \le j \le N_{\text{cand}}$, $n \ge 1$, and $1 \le i \le P$,

$V_{i,j}^{(n-1)}$ is the random vector representing the jth random candidate point $v_{i,j}^{(n-1)}$ generated around center c_i in Candidate Search of SOP.

$X_i^{(n)}$ is the random vector representing $x_i^{(n)}$, the ith function evaluation point of iteration n (chosen from candidate points $\left\{V_{i,\,j}^{(n-1)} : j = 1, ..., N_{\text{cand}}\right\}$)

Fix $n \geq 0$ (optimization iteration counter) and $1 \leq i \leq P$. Let $c_i^{(n)} \in S^{(n)}$ be the ith selected center for perturbation in iteration n. Recall that candidate points $\left\{v_{i,\,j}^{(n)} : j = 1, ..., N_{\text{cand}}\right\}$ are generated by perturbing randomly selected coordinates of $c_i^{(n)}$. Each coordinate of $c_i^{(n)}$ has a probability $\varphi(n) \in (0,\, 1]$ to be perturbed. If no variable of $c_i^{(n)}$ is selected for perturbation, one variable is chosen at random.

We next define the σ-algebra generated by relevant information available up to iteration n.

Definition B.2. Let $n \geq 1$ and $1 \leq i \leq P$. For $1 \leq j \leq N_{\text{cand}}$, let $Q_{i,\,j}^{(n-1)}$ be the random vector that determines which coordinates of $c_i^{(n-1)} \in S^{(n-1)}$ are chosen for a perturbation to obtain $V_{i,\,j}^{(n-1)}$.

For each $n \geq 1$, define $\mathcal{F}_{n-1},\ \mathcal{Q}_{n-1} \subset \mathbb{R}^d$ by

$$\mathcal{F}_{n-1} := \left\{ \begin{array}{ccc} \underbrace{X_1^{(0)}, .., X_{n_0}^{(0)}}_{\text{initial pts}} & \begin{array}{c} V_{1,\,1}^{(0)}, ..., V_{1,\,N_{\text{cand}}}^{(0)}, \\ \vdots \\ V_{P,\,1}^{(0)}, ..., V_{P,\,N_{\text{cand}}}^{(0)}, \end{array} & \begin{array}{c} V_{1,\,1}^{(n-1)}, ..., V_{1,\,N_{\text{cand}}}^{(n-1)}, \\ \vdots \\ V_{P,\,1}^{(n-1)}, ..., V_{P,\,N_{\text{cand}}}^{(n-1)} \end{array} \\ & \underbrace{\qquad\qquad}_{\text{cand pts iter } 0} \quad \cdots \quad \underbrace{\qquad\qquad}_{\text{cand pts iter } n-1} & \end{array} \right\} \begin{array}{c} \leftarrow \text{center1} \\ \vdots \\ \leftarrow \text{center}P \end{array} \tag{B.1}$$

$$\mathcal{Q}_{n-1} := \left\{ \begin{array}{ccc} \begin{array}{c} Q_{1,\,1}^{(0)}, ..., Q_{1,\,N_{\text{cand}}}^{(0)}, \\ \vdots \\ Q_{P,\,1}^{(0)}, ..., Q_{P,\,N_{\text{cand}}}^{(0)}, \end{array} & & \begin{array}{c} Q_{1,\,1}^{(n-1)}, ..., Q_{1,\,N_{\text{cand}}}^{(n-1)}, \\ \vdots \\ Q_{P,\,1}^{(n-1)}, ..., Q_{P,\,N_{\text{cand}}}^{(n-1)} \end{array} \\ \underbrace{\qquad\qquad}_{\text{iteration } 0} & \cdots & \underbrace{\qquad\qquad}_{\text{iteration } n-1} \end{array} \right\} \begin{array}{c} \leftarrow \text{center1} \\ \vdots \\ \leftarrow \text{center}P \end{array} . \tag{B.2}$$

Then, define

$$\mathcal{E}_{n-1} = \mathcal{F}_{n-1} \cup \mathcal{Q}_{n-1} \tag{B.3}$$

for $n \geq 1$, and $\mathcal{E}_{-1} = \left\{X_1^{(0)}, .., X_{n_0}^{(0)}\right\}$.

Thus, $\mathcal{F}_{n-1}$ contains the initial data used to fit the initial surrogate and all the candidate points $V_{i,\,j}^{(\cdot)}$ generated up to iteration $n-1$. $\mathcal{Q}_{n-1}$ contains all $Q_{i,\,j}^{(\cdot)}$, the random vector that determines which coordinates of selected centers are chosen for a perturbation, up to iteration $n-1$.

Remark B.3. For $n \geq 1$, since the new ith evaluation point $X_i^{(n)}$, $1 \leq i \leq P$, is selected deterministically from the values of the random vectors $\left\{\left(V_{i,\,1}^{(n-1)}\right), ..., \left(V_{i,\,N_{\text{cand}}}^{(n-1)}\right)\right\}$, the entire path of algorithm up to optimization iteration n is completely determined by $\sigma(\mathcal{E}_{n-1})$, where $\sigma(\mathcal{E}_{n-1})$ is the $\sigma - algebra$ generated by the random vectors in $\mathcal{E}_{n-1}$.

Theorem B.4 below builds upon Theorem 1 in [9]. Unlike [9] whose sequence $\{X_n^*\}_{n\geq 1}$ is defined for every function evaluation point (since the number of function evaluations = number of optimization iteration in [9]), $\left\{X^{**(n)}\right\}_{n\geq 0}$ in our case is defined for n that is the number of optimization iterations.

Theorem B.4. *Let f be a function defined on $\mathcal{D} \subseteq \mathbb{R}^d$ and 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 $\eta > 0$. Suppose further that the algorithm generates the random vectors $\left\{X_i^{(0)}\right\}_{1\leq i\leq n_0}$, $\left\{X_i^{(n)} : n \geq 1,\, 1 \leq i \leq P\right\}$ and $\left\{V_{i,\,1}^{(n)}, ..., V_{i,\,N_{cand}}^{(n)} : n \geq 0,\, 1 \leq i \leq P\right\}$ satisfying the following two conditions:*

[E1] *For each $n \geq 0$, the random vectors $\left\{V_{i,\,j}^{(n)} : 1 \leq i \leq P,\, 1 \leq j \leq N_{cand}\right\}$ are jointly independent conditional on $\mathcal{E}_{n-1}$.*

[E2] *For any $j = 1, ..., N_{cand}$, $x \in \mathcal{D}$ and $\delta > 0$, there exists $\nu_j(x, \delta) > 0$ such that*

$$Pr[V_{i,j}^{(n)} \in B(x, \delta) \cap \mathcal{D} | \sigma(\mathcal{E}_{n-1})] \geq \nu_j(x, \delta)$$

for all $n \geq 0$, $1 \leq i \leq P$, where $B(x, \delta)$ is the open ball of radius δ centered at x.

If the sequence of random vectors $\{X^{**(n)}\}_{n\geq 0}$ is defined by $X^{**(0)} = argmin_{x \in \{X_1^{(0)}, ..., X_{n_0}^{(0)}\}} f(x)$ and

$$X^{**(n)} = \begin{cases} argmin_{x \in \{X_1^{(n)}, ..., X_P^{(n)}\}} f(x) & \text{if } \min\left\{f(X_1^{(n)}), ..., f(X_P^{(n)})\right\} < f(X^{**(n-1)}) \\ X^{**(n-1)} & \text{otherwise} \end{cases}$$

for $n \geq 1$, then $X^{**(n)} \to x^*$ almost surely.

Proof. In order to avoid tedious arguments, we will use parts of Theorem 1 in [9] to prove our theorem. First, replace $\mathcal{E}_n$ defined in [9] with the one defined in Definition B.2 above. We define random vectors $\left\{\widetilde{X}_k\right\}_{k\geq 1}$ and $\{Y_{k,1}, ..., Y_{k,t}\}_{k \geq n_0}$ that satisfy the two conditions **[C1]**–**[C2]** of Theorem 1 in [9] as follow:

$$\widetilde{X}_k = \begin{cases} X_k^{(0)} & \text{for } 1 \leq k \leq n_0 \\ \text{argmin}_{x \in \{X_1^{(k-n_0)}, ..., X_P^{(k-n_0)}\}} f(x) & \text{for } k \geq n_0 + 1 \end{cases}. \tag{B.4}$$

Let $t = P \times N_{\text{cand}}$. Then, $\{Y_{k,1}, ..., Y_{k,t}\}_{k \geq n_0}$ is defined as $\left\{V_{i,j}^{(k-n_0)} : i = 1, ..., P, j = 1, ..., N_{\text{cand}}\right\}_{k \geq n_0}$. That is, all the candidate points generated from each of the P centers in the same optimization iteration are combined into one set. Since the two conditions **[E1]**–**[E2]** in this theorem are assumed, it follows immediately that the two conditions **[C1]**–**[C2]** in [9] also hold. Finally, we define $X_1^* = \widetilde{X}_1$ and $X_k^* = \widetilde{X}_k$ if $f(\widetilde{X}_k) < f(X_{k-1}^*)$ and $X_k^* = X_{k-1}^*$ otherwise. Then, by Theorem 1 of [9], we can conclude that $X_k^* \xrightarrow{a.s.} x^*$. Since $\left(X^{**(k)}\right)$, (X_k^*) have the same tail, they converge to the same limit. Therefore, $X^{**(n)} \xrightarrow{a.s.} x^*$. □

The condition **[E1]** of the theorem trivially holds for nSOP. Condition **[E2]** requires that the algorithm is able to sample at any point of the variable domain. Lemma B.5 and Lemma B.6 below are needed to prove that the condition **[E2]** holds for nSOP .

Lemma B.5. *For a fixed $j \in \{1, ... N_{cand}\}$, let H be the event that all coordinates of the center $c_i^{(n)}$ are selected for perturbation with search radius $r_i^{(n)} > 0$ (and $v_{i,j}^{(n)}$ is generated). Let $h_{i,j}^{(n)}$ be the conditional density of $V_{i,j}^{(n)}$ in nSOP given $\sigma(\mathcal{E}_{n-1})$ and H. Then, there is a constant $C > 0$ such that $h_{i,j}^{(n)}(u) \geq C$ for all $u \in \mathcal{D} = \{x : lb \leq x \leq ub\}$, $1 \leq i \leq P$ and $n \geq 0$.*

Proof. Assume that all coordinates of $c_i^{(n)}$ are selected for perturbation and that all the information $\sigma(\mathcal{E}_{n-1})$ is known. Since candidate points of nSOP follow multivariate truncated normal distributions, given that all coordinates of $c_i^{(n)}$ are selected for perturbation, the conditional density $h_{i,j}^{(n)}$ can be written as: $h_{i,j}^{(n)}(u) = A \exp\left\{\frac{-\left\|u - c_i^{(n)}\right\|^2}{2\sigma^2}\right\}$ for $u \in \mathcal{D}$ and 0 otherwise, where $\sigma = r_i^{(n)}$ and $A > 0$ is a normalizing constant for truncated multivariate normal density. Recall that each time when the local improvement of a particular center is not registered, the search radius of that center is shrunk by half. Moreover, the center whose local improvement is not registered for N_{fail} search efforts will be marked as tabu and the search radius of that center is reset back to r_{int}. Therefore, $r_i^{(n)} \geq r_{\text{int}}/2^{N_{\text{fail}}} := B > 0$. Taking $C := A \exp\left\{\frac{-\|\text{ub}-\text{lb}\|^2}{2B^2}\right\}$. It follows that $h_{i,j}^{(n)}(u) \geq C > 0$ for all $u \in \mathcal{D}$, $1 \leq i \leq P$ and $n \geq 0$. □

Lemma B.6. *If $\inf_{n\geq 0} \varphi(n) > 0$, the condition [E2] holds for nSOP.*

Proof. Recall that $\varphi(n)$ is the probability for selecting a variable of the current center $c_i^{(n)}$ for perturbation. Assume that $\inf_{n\geq 0}\varphi(n) > 0$. Let $j \in \{1, \dots N_{\text{cand}}\}$, $x \in \mathcal{D}$ and $\delta > 0$ be given. With the notation used in Lemma B.5 and the definition of H,

$$\Pr[V_{i,j}^{(n)} \in B(x,\,\delta)\cap\mathcal{D}|\sigma(\mathcal{E}_{n-1})] \tag{B.5}$$

$$\geq \Pr[\Big(V_{i,j}^{(n)} \in B(x,\,\delta)\cap\mathcal{D}\Big)\cap H|\sigma(\mathcal{E}_{n-1})] \tag{B.6}$$

$$= \Pr[V_{i,j}^{(n)} \in B(x,\,\delta)\cap\mathcal{D}|\sigma(\mathcal{E}_{n-1}),\,H]\times\Pr(H) \tag{B.7}$$

$$= \left(\int\limits_{B(x,\,\delta)\cap\mathcal{D}} h_{i,j}^{(n)}(u)du\right)\times\varphi(n)^d \tag{B.8}$$

$$\geq C\mu\left(B(x,\,\delta)\cap\mathcal{D}\right)\times\left(\inf_{n\geq 0}\varphi(n)\right)^d := \nu_j(x,\,\delta) \tag{B.9}$$

for any $n \geq 0$, where $h_{i,j}^{(n)}$ is defined in Lemma B.5 and $C > 0$ is a non-negative constant existing in Lemma B.5. Due to the compactness of hypercube $\mathcal{D}$ and our assumption on φ that $\inf_{n\geq 0}\varphi(n) > 0$, it follows that $\nu_j(x,\,\delta) > 0$. Note also that $\nu_j(x,\,\delta)$ is independent of n and i. Thus, the condition **[E2]** is now verified. □

Since the conditions of Theorem B.4 hold for nSOP, we can conclude that nSOP converges to the global minimum with probability 1.

Example B.7. One example of φ that satisfies the sufficient condition given in Lemma B.6 is:

$$\varphi(n) = \begin{cases} \varphi_0\times[1-\ln(n-n_0+1)/\ln(M-n_0)] & \text{if } n_0 \leq n \leq M-2 \\ \varphi(M-2) & n > M-2 \end{cases},$$

where $\varphi_0 > 0$ and $M >> 0$. This is a modification of the $\varphi(n)$ used in [11], for which the maximum number of function evaluations is bounded.

C Truncated Gaussian Distribution

The truncated Gaussian distribution (or the truncated normal distribution) is the probability distribution of a normally distributed random variable whose value is bounded in the domain $[a,\,b]$, where $-\infty < a < b < \infty$.

Suppose $X \sim N(\mu,\,\sigma^2)$ has a normal distribution. Then, X conditional on $a \leq X \leq b$ has a truncated normal distribution with the probability density function given by [4]

$$f(x;\,\mu,\,\sigma,\,a,\,b) = \begin{cases} \frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma})-\Phi(\frac{a-\mu}{\sigma})} & ;\, a \leq x \leq b \\ 0 & ;\, \text{otherwise} \end{cases} \tag{C.1}$$

where $\phi(z) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}z^2)$ is the probability density function of the standard normal distribution $N(0,\,1)$ and Φ is its corresponding cumulative distribution function. We denote this distribution by $N_{\text{truncated}}(\mu,\,\sigma^2;\,a,\,b)$.

D Alternative Optimization Algorithms

We compare our algorithm to Parallel Stochastic RBF [10] and an evolutionary algorithm that uses radial basis function approximation (ESGRBF) [8, 12]. Both of these algorithms were shown to be very efficient for computationally expensive functions. In terms of applications, Parallel Stochastic RBF was used to solve an optimization problem arising in groundwater bioremediation introduced in [7] and ESGRBF was used to calibrate computationally expensive watershed models.

1. Parallel StochRBF: Parallel Stochastic RBF is a parallel version of Stochastic RBF by Regis and Shoemaker [9]. In each iteration, the algorithm generates candidate points around x_{best} and the best P points are selected for expensive function evaluations. The selection is done sequentially, 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. The weights for these two criteria are adjusted in a cycling manner according to a predefined weight pattern.

2. ESGRBF: An evolutionary algorithm that uses radial basis function approximations ESRBF within a standard evolutionary strategy (ES) was proposed in [8]. ESRBF uses multiple local RBF models in each generation of an evolutionary algorithm, one RBF model for each offspring. Based on self-adaptive algorithm parameters, in each generation, $\lambda \geq \nu$ offspring are generated and their objective function values are estimated using the RBF surrogate model. The true objective function is evaluated at the ν best individuals. The μ parents for the next generation are selected from these ν best offspring. In ESRBF, to estimate the objective function value of any offspring x, first the k nearest neighbors of x among all previously evaluated parameter vectors are located. Then these nearest neighbors are used to build a local RBF approximation that is then used to estimate $f(x)$. In contrast, ESGRBF [12] uses a single RBF model to estimate the fitness values of the offspring in each generation. Shoemaker et al. [12] found that the performances of ESRBF and ESGRBF are almost identical because the global RBF approximation in ESGRBF typically agrees with the individual local RBF approximations. However, as mentioned in [12] that ESGRBF is much easier to implement and it involves less overhead than ESRBF, therefore we also use ESGRBF in this study.

E Test Functions

E.1 Real-Parameter Black-Box Optimization Benchmarking (BBOB) Testbed

Ten benchmark functions F15–F24 were selected from the BBOB test suite [5]. All of them are 10 dimensional problems. These functions are multimodal. The functions are listed in Table E.1. For definition and properties of these functions, refer to [5].

Table E.1: Benchmark functions for SOP

No.	Test Function
F15	Rastrigin Function
F16	Weierstrass Function
F17	Schaffers Function
F18	Schaffers Function, moderately ill-conditioned
F19	Composite Griewank-Rosenbrock Function F8F2
F20	Schwefel Function
F21	Gallagher's Gaussian 101-me Peaks Function
F22	Gallagher's Gaussian 21-hi Peaks Function
F23	Katsuura Function
F24	Lunacek bi-Rastrigin Function

E.2 Groundwater Bioremediation Problem

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 [13]. The constrained global optimization problem aims to find the cheapest cleanup strategy by determining the pumping rates of oxygenated water into the ground to promote the degradation of the contaminants by microbes. The locations of the injection wells are known. Monitoring wells are also set up in order to measure the concentration of the contaminant and to ensure that it will be below some threshold level at specified time periods. Because

the pumping of oxygenated water is expensive, the goal is to determine the pumping rates for each injection well at the beginning of each management period in order to minimize the total pumping cost required to reduce the contaminant concentration at the monitoring wells to the maximum allowed by EPA regulations. The authors introduced a penalty term to incorporate a contamination constraint into the total pumping cost (objective function). This implementation resulted in a box-constrained global optimization problem. The decision variables are the pumping rates at each of the 3 injection wells in each of the management periods. The four time period problem (resulting in 12 decision variables) is considered. We refer to this groundwater problem as GWB12D.

F Additional Experimental Results

The mean and standard deviation of the best function value obtained after the algorithms terminate (60 hours) when using 8 and 32 processors are reported in Tables F.1 and F.2, respectively. In addition to ParStochRBF and ESGRBF, we also report the results of two serial surrogate-based algorithms, namely Stochastic RBF [9] and NOMADm-DACE (NOMADm with kriging surrogate) [1, 2, 3, 6] after 480 function evaluations in Table F.1.

Table F.1: Mean and standard deviation of the best function value after 480 function evaluations for F15–F24 from BBOB testbed using 8 processors for parallel algorithms

Alg	nSOP-8P		uSOP-8P		StochRBF-8P		ESGRBF-8P		StochRBF*		NOMADm-DACE*	
Test Func	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
F15	**-15.276**	12.608	-14.993	13.575	15.616	10.299	17.834	7.193	-2.421	22.896	27.586	23.731
F16	-254.080	3.144	-255.652	1.768	**-256.633**	1.591	-247.473	4.957	-256.387	1.366	-253.097	3.530
F17	-37.648	0.307	-37.460	0.508	-37.616	0.608	**-37.667**	0.612	-35.371	1.936	-34.576	2.821
F18	**-35.795**	1.091	-33.681	4.181	-33.904	1.706	-33.130	2.228	-28.136	5.279	-21.989	11.048
F19	44.967	0.767	**44.375**	0.999	44.974	0.962	44.759	0.896	44.549	1.121	44.787	1.773
F20	185.634	0.248	185.447	0.279	185.828	0.407	185.637	0.516	**184.658**	0.416	185.102	0.543
F21	**312.711**	1.397	313.432	2.806	315.551	3.058	320.990	11.478	313.676	2.958	317.403	8.411
F22	45.380	0.528	46.634	3.901	48.310	7.498	58.931	19.076	**45.318**	1.917	60.297	18.733
F23	212.922	0.671	212.868	0.582	213.057	0.449	212.888	0.699	**211.733**	0.438	212.072	1.062
F24	**110.676**	6.868	113.276	13.650	116.597	9.293	121.462	10.161	111.856	14.105	150.201	37.613

*StochRBF and NOMADm-DACE are serial algorithms that do 1 function evaluation per iteration.

Table F.2: Mean and standard deviation of the best function value after 1920 function evaluations for F15–F24 from BBOB testbed using 32 processors

Alg	nSOP-32P		uSOP-32P		StochRBF-32P		ESGRBF-32P	
Test Func	mean	std	mean	std	mean	std	mean	std
F15	-23.385	5.541	**-30.612**	3.304	12.689	6.762	-0.159	6.700
F16	-257.141	0.707	-257.538	1.052	**-257.637**	1.686	-244.834	1.868
F17	-38.168	0.325	-38.075	0.379	-38.144	0.275	**-38.609**	0.038
F18	-37.224	0.352	-36.604	0.877	-35.666	0.979	**-37.852**	0.604
F19	43.855	0.498	43.885	0.462	44.151	0.683	**43.668**	0.896
F20	185.237	0.241	185.117	0.336	185.480	0.292	**185.081**	0.253
F21	312.001	0.414	**311.774**	0.901	317.249	2.963	315.660	5.573
F22	45.572	0.001	**45.503**	0.216	51.512	10.272	47.490	6.068
F23	**212.361**	0.551	212.584	0.442	212.683	0.524	212.813	0.608
F24	**96.234**	11.835	96.954	10.063	105.678	7.581	96.665	8.889

We conduct a Mann-Whitney-Wilcoxon non-parametric test to compare the results of the final solution when using 8 processors. The one-tailed test is performed at the 5% level of significance. Tables F.3(a) and (b) show the number of test functions (out of ten) in which nSOP or uSOP is significantly better or worse than the compared algorithm (in terms of the final solution).

From Table F.3, the values of zero in the first column indicate that nSOP-8P and uSOP-8P are not significantly different. However, in total nSOP-8P outperforms the compared algorithms on more test functions than uSOP-8P does.

While nSOP-8P, uSOP-8P, StochRBF-8P, and ESGRBF-8P do eight function evaluations per iteration, the serial algorithms Stochastic RBF and NOMADm-DACE do one point per iteration and are therefore able to exploit more information during the optimization, and the surrogate is updated after each function evaluation. Hence, given the same number of function evaluations (480), one would expect these serial algorithms to outperform the algorithms that simulate $P = 8$ function evaluations per iteration.

We see that both nSOP-8P and uSOP-8P could attain a better final solution than serial StochRBF for 2 test problems and are significantly better than serial NOMADm-DACE for 6 out of 10 test problems.

Table F.3: The total number of test functions (out of ten) that SOP-8P can attain a better or worse final solution than the compared algorithm at 5% significance level. The data is collected after 480 function evaluations.

(a) nSOP-8P

nSOP-8P vs.	uSOP-8P	StochRBF-8P	ESGRBF-8P	StochRBF*	NOMADm-DACE*
better	0	3	6	2	6
worse	0	1	0	4	2

(b) uSOP-8P

uSOP-8P vs.	nSOP-8P	StochRBF-8P	ESGRBF-8P	StochRBF*	NOMADm-DACE*
better	0	3	3	2	6
worse	0	0	0	3	1

*StochRBF and NOMADm-DACE are serial algorithms that simulate 1 function evaluation per iteration.

Similarly, we conduct a Mann-Whitney-Wilcoxon test to compare the final results for 32 processors at the 5% level of significance. Tables F.4(a) and (b) show the number of test functions in which nSOP-32P or uSOP-32P is significantly better or worse than the compared algorithm.

From Table F.4, we see that both nSOP-32P and uSOP-32 outperform each other on one test function. Both algorithms are better than StochRBF-32P on 6 test functions. nSOP-32P is better than ESGRBF-32P on 4 test functions while uSOP-32P is better than ESGRBF-32P on 3 test functions.

We note that Tables F.3 and F.4 are based only on the final values at the maximum number of evaluations. However, looking at the progress curves (Figure 5 of the main paper), the reader can see that in many cases (e.g. especially for F21 and F22) SOP could achieve accurate solutions much more quickly (e.g. less than 20 hours of wall-clock time) than the other methods. This is not reflected in Tables F.3 and F.4.

Finally, the α-levels for nSOP and uSOP used in Section 3.6 of the main paper are shown in Table F.5.

References

[1] M.A. Abramson. *NOMADm Version 4.5 User's Guide.*, 2007.

Table F.4: The total number of test functions (out of ten) that SOP-32P can attain a better or worse final solution than the compared algorithm at 5% significance level. The data is collected after 1920 function evaluations.

(a) nSOP-32P

nSOP-32P vs.	uSOP-32P	StochRBF-32P	ESGRBF-32P
better	1	6	4
worse	1	0	3

(b) uSOP-32P

uSOP-32P vs.	nSOP-32P	StochRBF-32P	ESGRBF-32P
better	1	6	3
worse	1	0	2

Table F.5: α-levels

Test Function	nSOP			uSOP		
	α_1	α_2	α_3	α_1	α_2	α_3
F15	-2.876	-2.997	-3.027	-2.876	-2.997	-3.027
F16	-241.649	-251.824	-254.367	-243.050	-253.284	-255.842
F17	-33.626	-35.041	-35.395	-33.626	-35.041	-35.395
F18	-26.742	-27.868	-28.149	-26.742	-27.868	-28.149
F19	47.176	45.378	44.929	46.686	44.908	44.463
F20	194.821	187.399	185.543	194.693	187.276	185.422
F21	329.358	316.811	313.675	329.358	316.811	313.675
F22	47.851	46.028	45.572	48.958	47.093	46.627
F23	223.550	215.034	212.905	223.511	214.996	212.868
F24	115.991	111.572	110.467	118.940	114.409	113.276
GWB12D	367.716	353.708	350.206	362.463	348.655	345.203

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

[3] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[4] William Greene. H.(2003): Econometric analysis. *New Jersey, ua: Prentice Hall*, 2003.

[5] Nikolaus Hansen, Steffen Finck, Raymond Ros, Anne Auger, et al. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. 2009.

[6] Søren Nymand Lophaven, Hans Bruun Nielsen, and Jacob Søndergaard. Dace-a matlab kriging toolbox, version 2.0. Technical report, 2002.

[7] Pradeep Mugunthan, Christine A Shoemaker, and Rommel G Regis. Comparison of function approximation, heuristic, and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. *Water Resources Research*, 41(11), 2005.

[8] Rommel G Regis and Christine A Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *Evolutionary Computation, IEEE Transactions on*, 8(5):490–505, 2004.

[9] Rommel G Regis and Christine A Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

[10] Rommel G Regis and Christine A Shoemaker. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing*, 21(3):411–426, 2009.

[11] Rommel G Regis and Christine A Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[12] Christine A Shoemaker, Rommel G Regis, and Ryan C Fleming. Watershed calibration using multistart local optimization and evolutionary optimization with radial basis function approximation. *Hydrological sciences journal*, 52(3):450–465, 2007.

[13] Jae-Heung Yoon and Christine A Shoemaker. Comparison of optimization methods for ground-water bioremediation. *Journal of Water Resources Planning and Management*, 125(1):54–63, 1999.