Optimization and Engineering (2021) 22:2741–2777
https://doi.org/10.1007/s11081-020-09556-1

RESEARCH ARTICLE

GOPS: efficient RBF surrogate global optimization algorithm with high dimensions and many parallel processors including application to multimodal water quality PDE model calibration

Wei Xia[1] · Christine Shoemaker[1,2]

Received: 13 December 2019 / Revised: 24 August 2020 / Accepted: 24 August 2020 /
Published online: 17 September 2020

Abstract
This paper describes a new parallel global surrogate-based algorithm Global Optimization in Parallel with Surrogate (GOPS) for the minimization of continuous black-box objective functions that might have multiple local minima, are expensive to compute, and have no derivative information available. The task of picking *P* new evaluation points for *P* processors in each iteration is addressed by sampling around multiple center points at which the objective function has been previously evaluated. The GOPS algorithm improves on earlier algorithms by (a) new center points are selected based on bivariate non-dominated sorting of previously evaluated points with additional constraints to ensure the objective value is below a target percentile and (b) as iterations increase, the number of centers decreases, and the number of evaluation points per center increases. These strategies and the hyper-parameters controlling them significantly improve GOPS's parallel performance on high dimensional problems in comparison to other global optimization algorithms, especially with a larger number of processors. GOPS is tested with up to 128 processors in parallel on 14 synthetic black-box optimization benchmarking test problems (in 10, 21, and 40 dimensions) and one 21-dimensional parameter estimation problem for an expensive real-world nonlinear lake water quality model with partial differential equations that takes 22 min for each objective function evaluation. GOPS numerically significantly outperforms (especially on high dimensional problems and with larger numbers of processors) the earlier algorithms SOP and PSD-MADS-VNS (and these two algorithms have outperformed other algorithms in prior publications).

Keywords PDE-constrained optimization · Surrogate models · Parallel computing · Water quality models · Global optimization · Multi-modal and black-box objective

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11081-020-09556-1) contains supplementary material, which is available to authorized users.

Extended author information available on the last page of the article

1 Introduction

Optimization of numerical simulation models is important because they are widely used in numerous real-world applications in many fields, including science and engineering. One essential category of computer simulation models is those that are computing solutions to a system of partial differential equations (PDE) on, for instance, surface water and groundwater problems (Culver and Shoemaker 1992; Gorelick et al. 1993; Hinkelmann 2006; Pinder and Gray 1977; Yeh 2015), and aerodynamics problems (Bons et al. 2019; Sóbester and Forrester 2014). The computational time of these models tends to be significant (many minutes to hours per simulation).

For optimization of simulation models that are expensive, the optimization algorithm needs to be able to find a good solution with relatively few objective function evaluations. There are many efficient algorithms for linear, convex PDE optimization problems (e.g., Culver and Shoemaker 1992), which only have one local solution. However, when the simulation models contain multiple interacting nonlinear relationships, the objective function based on simulation results can have many local minima (Gorelick and Zheng 2015), so a global optimization method is necessary to find the global optimum. Optimizing multi-modal objectives is much harder because these non-global methods (e.g., linear, convex, or unimodal nonconvex algorithms) are not designed to find the best among multiple separated local minima. In addition, we assume no derivative information is available, and hence gradient-based methods or methods using an adjoint approach are not applicable.

Our goal is to present an algorithm that is effective for global optimization of expensive objective functions, including but not limited to objective functions subject to simulation models with partial differential equations. We propose a new parallel algorithm Global Optimization in Parallel with Surrogate (GOPS) that uses a surrogate model of the original expensive function to help guide the optimization search and reduce the number of evaluations on the expensive objective function. The surrogate model is cheap-to-compute, built with previously evaluated points, and is dynamically updated during the optimization process. The new algorithm enables evaluating multiple simulations simultaneously in one iteration. These multiple evaluation points are sampled around multiple centers selected from previously evaluated points. The parallel processing can help further to speed up the optimization processes and to reduce the wall-clock time that the user needs to spend on waiting for results.

GOPS uses some features of the earlier SOP algorithm (Krityakierne et al. 2016) but improves on earlier algorithms by (a) new centers are selected based on bivariate non-dominated sorting of previously evaluated points with additional constraints to ensure the objective value is below a target percentile and (b) as iterations increase, the number of centers decreases and the number of evaluation points per center increases. These features in GOPS are not present in earlier algorithms, which makes GOPS more robust and faster to converge.

We tested the GOPS algorithm on 14 analytical test functions (with 10, 21, 40 dimensions) and one real-word PDE-constrained parameter estimation problem (with 21 dimensions). The real-world test problem involves a highly nonlinear multi-modal model (solving partial differential equations) for fate and transport of many water quality constituents in a lake. This, hence, is an important example of the use of the GOPS algorithm on a PDE-based objective function. GOPS showed improved performance over SOP algorithms and other optimization methods.

The structure of this paper is as follows. In Sect. 2, the literature review is given. Section 3 describes the GOPS algorithm. In Sect. 4, we explain in detail the water quality model parameter estimation problem. In Sect. 5, we discuss the numerical results of algorithm performance on test functions and the real-world test problem. The Online Resource contains an extensive list of supplementary information, including definitions of symbols and parameters.

2 Literature review

Our focus is on global optimization of expensive, black-box, multi-modal objective functions for which no derivatives of the objective function available. Optimization algorithms that do not require derivative information are also referred to as derivative-free algorithms. A comprehensive literature review of different kinds of derivative-free algorithms, including both local and global optimization methods, can be found in Audet and Hare (2017) and Rios and Sahinidis (2013).

The global optimization algorithms can be classified into non-surrogate methods and surrogate methods based on whether the surrogate model is used to direct the algorithm search. A popular class of global non-surrogate methods for engineering problems are heuristic methods (e.g., Genetic Algorithm, Evolutionary Strategies, and Particle Swarm Optimization). These methods are straightforward to implement, and they can escape from local optima. However, such methods usually require many thousands of function evaluations (Jakobsson et al. 2010). Hence they are not suitable for problems that are computationally expensive to evaluate, such as an objective function that requires the solution of an expensive nonlinear PDE, and they are not considered in this paper.

There is another set of global non-surrogate methods that are combinations of a local optimization method and a global heuristic method that has global exploration features. Audet et al. (2008a) explored the combination of Mesh Adaptive Direct Search (MADS) with the metaheuristic Variable Neighborhood Search (VNS) algorithm. The MADS algorithm is an extension of the Generalized Pattern Search algorithm (Torczon 1997) and converges to a local minimum under appropriate assumptions. VNS is a metaheuristic method proposed by Mladenović and Hansen (1997). It uses a random perturbation method, which makes it able to move away from a local optimum solution and has been proven efficient on a broad range of problems. The study by Audet et al. (2008a) indicates that MADS with VNS allows the algorithm to move away from local solutions. MADS with VNS is available in NOMAD software (Le Digabel 2011), and it has three parallel versions: p-MADS, COOP-MADS, and PSD-MADS (Le Digabel et al. 2010). PSD-MADS performs better than other parallel MADS versions when the decision vector dimension is equal to or greater than 20 (Le Digabel 2011).

Global surrogate-based methods are suitable for expensive objective functions. These methods use an inexpensive surrogate model that approximates the black-box function to guide the search. Hence surrogate-based optimization models usually require a fewer number of evaluations on expensive black-box objective function than required by algorithms without surrogates.

There are two types of popular global surrogate-based optimization methods: (1) Gaussian Process (GP) based and (2) Radial Basis Functions (RBF) based. There are also other types of surrogates used in optimization, e.g., polynomial model and support vector regression. Detailed information of these surrogates could be found in Díaz-Manríquez et al. (2011), Forrester et al. (2008), and Müller and Shoemaker (2014). The most well-known GP-based method is EGO, which was introduced by Jones et al. (1998) and has gained popularity for some types of problems. However, a disadvantage of GP-based methods is that these methods can become computationally prohibitive in the non-evaluation phase of optimization and require an enormous amount of memory when the problem is high dimensional (Hensman et al. 2013; Regis 2013). Isaacs (2009) also showed that the time for the Gaussian process model to fit its surrogate (training time) is much longer than that for an RBF model of the same dimension.

RBF was first introduced in global optimization by Gutmann (2001), and there are various RBF-based serial methods proposed (Jakobsson et al. 2010; Regis and Shoemaker 2005, 2007b, 2009, 2013). RBF-based methods are proven to be effective for solving real-word computationally expensive problems, e.g., designing the specifics of trains (Björkman and Holmström 2000), groundwater problem (Christelis et al. 2018; Mugunthan et al. 2005), watershed problem (Regis and Shoemaker 2007b, 2013), methane emission problem (Müller et al. 2015), and aerodynamic regional airliner wing design (Sóbester et al. 2014). Jakobsson et al. (2010) applied an RBF-based global optimization method to the combustion engine design problem, which is a noisy function and computationally expensive with one simulation taking 42 h. There are also efforts made on using RBF-based methods to solve high dimensional problems. For example, DYCORS (Regis and Shoemaker 2013) has been successfully applied to 200-dimensional problems. RBF-based methods were applied to a 124-dimensional automotive problem with 68 black-box inequality constraints (Regis 2011, 2014). Díaz-Manríquez et al. (2011) compared RBF with GP (also known as kriging), polynomial model and support vector regression in term of accuracy, robustness, scalability and efficiency and suggested that for high dimensional problems (with $d > 15$) RBF is the best techniques to be combined with optimization algorithms.

There are also advances in the parallelization of surrogate-based algorithms to tackle expensive optimization problems with the assistance of parallel computing (Haftka et al. 2016). Sóbester et al. (2004) proposed a parallel version of the GP-based optimization method. Given P processors, in each iteration, the best P points with the maximum expected improvement value (based on the GP surrogate) are selected as evaluation points for the next iteration. Bischl et al. (2014) applied a multi-objective infill criterion on a parallel GP-based optimization algorithm to select multiple evaluation points that considered both diversity and expected improvement.

Regis and Shoemaker (2009), in the parallel Stochastic RBF (SRBF) algorithm, used a weighted metric to select P points sequentially in each iteration from candidate points generated around the best solution found so far. The selection of evaluation points in each iteration is not only dependent on the candidate point's estimated function value (based on surrogate model) but also its minimum distance from the evaluated and selected points in that iteration. The value of the weight between the surrogate estimation and distance criteria is varied to select as many evaluation points as there are processors.

Krityakierne et al. (2016) proposed the SOP algorithm for parallel computation and reported that there are a few studies on parallel surrogate global optimization that scaled up to many processors. The proposed SOP algorithm showed good speed up with up to 64 processors per iteration. In previous studies before their SOP study (Krityakierne et al. 2016), the maximum number of processors in parallel with global surrogate optimization was not larger than 10. Given P processors, SOP selects P evaluation points for the next iteration from candidate points generated around P center points. The P center points are selected from all previously evaluated points, based on bi-objective optimization on (1) the objective function value and (2) the distance from all other evaluated points. The utilization of multi-objective techniques is to balance the trade-off between exploration and exploitation during the search. Their study showed promising results that their optimization algorithm could be scaled up to use many processors effectively.

However, in the SOP study, the maximum dimension of the tested problem is 12. So, it is not clear whether the SOP algorithm is still efficient on higher dimensional optimization problems. Many real-world optimization problems involving PDE objective functions are high dimensional (Björkman and Holmström 2000; Shan and Wang 2010).

The new algorithm GOPS, introduced here, is significantly different from previous algorithms, including SOP. GOPS is designed to do well when the problem is high dimensional and/or the number of processors is large. The numerical result presented latter shows that GOPS has a significantly better numerical performance and can work with a larger number of parallel processors than other algorithms tested, even when the dimension of the decision vector is high.

3 GOPS

GOPS is a general purpose global optimization algorithm solving optimization problem in following form:

$$\begin{gathered}\min_{\boldsymbol{\mu}} f(\boldsymbol{\mu}) \\ \boldsymbol{\mu} \in \Theta \subset \mathbb{R}^d\end{gathered} \tag{1}$$

where $f(\boldsymbol{\mu})$ is the objective function to minimize and is assumed to be multi-modal, black-box (no derivative information available). $\boldsymbol{\mu}$ is the decision vector that in d dimensional. Θ is the d dimensional solution space usually defined by the upper bound and lower bound of the values of the parameters, so $\Theta = \{\mathbf{lb} \leq \boldsymbol{\mu} \leq \mathbf{ub}\} \subset \mathbb{R}^d$.

GOPS uses some features from the earlier parallel algorithm SOP (Krityakierne et al. 2016) and adds important new features to improve performance quite significantly, especially with many processors and decision vectors of high dimensions. GOPS has new strategies to dynamically change the diversity of candidate points generated by multiple sampling centers.

In following subsections, we will first describe the general framework of the GOPS algorithm and then specifically introduce new, improved strategies used in the iterative phases of GOPS that are dynamic by changing (1) $P_C^{(n)}$ the number of centers (around which candidate points are generated) in iteration n, (2) N_{c_j} the number of points around each center ($c_j, j \in \{1, \dots, P_C^{(n)}\}$), and (3) $P_{good}^{(n)}$, which is the percentile of the previously evaluated points (based on only function value) that are allowed to be selected as centers (which we refer to as "Good center candidate pool"). As discussed later, these three factors, which are dynamically changing as the iterations n increase, are helpful both in exploration and exploitation. Note that N_{c_j} is a function of n, but n is suppressed to reduce the complexity of the notation.

3.1 General description of GOPS

GOPS follows the iterative master-worker framework for the RBF surrogate algorithms (Regis and Shoemaker 2007a) and consists of three core steps, namely (1) *Initialization*, (2) *Iterative loop* and (3) *Termination*. The difference between GOPS and previous RBF-based algorithms is in the Iterative loop. The *Initialization* phase is to compute the objective function $f(\boldsymbol{\mu})$ at n_0 points, so there are multiple points $\{\boldsymbol{\mu}_i, f(\boldsymbol{\mu}_i)\}$ (for $i = 1, \dots, n_0$) that are used to initialize the surrogate model and start the iterative loop. These initial points in Step (1) could be obtained via any experimental design method (e.g., Latin hypercube sampling) where the number of points n_0 to be evaluated is given. In the *Termination* step, the only terminal condition for GOPS is computing budget, i.e., the maximum number of evaluations N_{max}, which is an input variable. In GOPS we set the number of evaluations in each iteration to be the number of processors available P. So, the terminal conditions can also be considered to be the maximum number of iterations, $MAXIT$ ($MAXIT = (N_{\max} - n_0)/P$). Note to make full use of the P processors, the values of N_{max} and n_0 are set to be multiples of the number of processors P.

The core of the algorithm and most complicated part is the *Iterative loop*, the main tasks of which are (a) to use the surrogate to select P points at which to simultaneously evaluate the objective function on the P available processors and then (b) to update the surrogate with the newly available values of the objective function. It is increasingly difficult to find P worthwhile points to evaluate on P processors as P gets large because one wants points that the surrogate indicates are likely to have low values (for minimization) and that are not too close together (so that there is some exploration). Our numerical experiments later use up to 128 processors, so we need to pick as many as 128 evaluation points in each iteration to assign to different processors.

There are five sub-steps within the iterative loop step of GOPS, including (1) Surrogate fit; (2) Center selection; (3) Candidate point generation and search; (4)

Objective function evaluation; (5) Adaptive learning. For sub-steps (1), (4) and (5) GOPS and SOP are the same. Sub-step (2) and (3) use some of the steps in SOP, including (a) *non-dominated sorting* and *tabu and radius constraints* for center selection and (b) *dynamic coordinate search* around the center for candidate search. GOPS is different from SOP in sub-step (2) and (3) by adding two more sampling strategies (a) the dynamic number of centers $P_C^{(n)}$ and evaluations of each center N_{c_j} and (b) $P_{good}^{(n)}$, which guarantees that points with poor objective function values do not become centers. In the following text, we will illustrate these steps that are common with SOP and then provide a detailed description of the two strategies in detail in Sects. 3.2 and 3.3.

3.1.1 Surrogate fit

In the surrogate fit step, the surrogate model of the original black-box function $f(\boldsymbol{\mu})$ is denoted as $\hat{f}(\boldsymbol{\mu})$. The surrogate $\hat{f}(\boldsymbol{\mu})$ is built with the points evaluated previously before the nth iteration, where n is the index of the iteration number $(1 \le n \le MAXIT)$. The reason to use the surrogate model is to help guide the optimization search to reduce the number of evaluations on the expensive objective function $f(\boldsymbol{\mu})$. The surrogate model is used to do a preliminary screening on the larger number of trial points such that only these points with a relatively small surrogate value (regard as "promising" points) and not too close to previously evaluated points will be selected to do the expensive function evaluation. A cubic Radial Basis Function (RBF) is selected as the surrogate model function.

Let $S^{(n)}$ be the set of evaluated sample points before n algorithm iterations and N_E the number of evaluation points in $S^{(n)}$, where $N_E = n_0 + P \times (n-1)$. The surrogate model is fit on $S^{(n)}$ with a cubic Radial Basis Function (RBF), which takes the interpolant of the form:

$$\hat{f}(\boldsymbol{\mu}) = \sum_{i=1}^{n} \lambda_i \phi(||\boldsymbol{\mu} - \boldsymbol{\mu}_i||) + p(\boldsymbol{\mu}), \boldsymbol{\mu} \in \mathbb{R}^d \tag{2}$$

where ||•|| is the Euclidean norm, $p(\boldsymbol{\mu})$ is a linear polynomial in d variables with $d+1$ coefficients $b_i \in \mathbb{R}$ for $i = 1, \dots d+1$, and ϕ has a cubic form: $\phi(r) = r^3$, the coefficients $\lambda_i \in \mathbb{R}$ for $i = 1, \dots, N_E$ [in Eq. (2)], are determined by solving the following linear system of equations:

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix} \tag{3}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N_E \times N_E}$ and $\boldsymbol{\Phi}_{x,y} = \phi(||\boldsymbol{\mu}_x - \boldsymbol{\mu}_y||)$, $x, y = 1, \dots, N_E$, $\mathbf{0} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a matrix of zeros, $\mathbf{F} = \left[f(\boldsymbol{\mu}_1) \ \dots \ f(\boldsymbol{\mu}_{N_E}) \right]^T$, $\mathbf{P} \in \mathbb{R}^{N_E \times (d+1)}$ and the ith row of the matrix $\mathbf{P}$ is $\left[\boldsymbol{\mu}_i^T \ 1 \right]$, $\boldsymbol{\lambda} = \left[\lambda_1 \ \dots \ \lambda_{N_E} \right]^T$, $\mathbf{B} = \left[b_1 \ \dots \ b_{d+1} \right]^T$. The matrix $\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix}$ in Eq. (3) is nonsingular and the linear system Eq. (3) has a unique solution if and only if rank($\mathbf{P}$) $= d + 1$ (Powell 1992). This condition is satisfied when there is a subset of $d + 1$ affinely independent points in $S^{(n)}$.

After the surrogate model $\hat{f}(\boldsymbol{\mu})$ is built, given any $\boldsymbol{\mu} \in \mathbb{R}^d$, there will be a surrogate function value $\hat{f}(\boldsymbol{\mu})$ as an estimation of the black-box function $f(\boldsymbol{\mu})$. The value $\hat{f}(\boldsymbol{\mu})$ is used to help guide the optimization search because it is much cheaper to evaluate than $f(\boldsymbol{\mu})$.

3.1.2 Center selection

During each iteration of the algorithm, a number of evaluated points are selected as center points, which will be used for generating candidate points, some of which will become evaluation points where the expensive $f(\boldsymbol{\mu})$ is evaluated. We will generate candidate points considered for expensive evaluation by many random perturbations around each center point. $P_C^{(n)}$ is the number of center points in the nth iteration, which can change as the number of iterations increases. In the earlier RBF algorithm SOP, the value of $P_C^{(n)}$ is equal to the number of processors P and does not change with iteration number n. The $P_C^{(n)}$ center points are selected from previously evaluated points (denoted as $S^{(n)}$). In GOPS, the number of centers $P_C^{(n)}$ is being reduced as the number iterations n increase. A detailed description of the computation of $P_C^{(n)}$ is discussed in Sect. 3.2.

Centers in each iteration are selected based on the *non-dominated sorting* techniques (Krityakierne et al. 2016). In each iteration, all the evaluated points in $S^{(n)}$ are ranked based on two objectives, (1) the objective function value $f(\boldsymbol{\mu}_i)$ and (2) the negative of minimum distance from $\boldsymbol{\mu}_i$ to all other evaluated points $S^{(n)} \setminus \{\boldsymbol{\mu}_i\}$ (denoted as $\eta^{(n)}(\boldsymbol{\mu}_i)$). In *non-dominated sorting*, the evaluation point $\boldsymbol{\mu}_a$ dominates $\boldsymbol{\mu}_b$ if both $f(\boldsymbol{\mu}_a) < f(\boldsymbol{\mu}_b)$ and $\eta^{(n)}(\boldsymbol{\mu}_a) < \eta^{(n)}(\boldsymbol{\mu}_b)$.

Note that as the iteration number n increases, there are new points added to $S^{(n)}$. Hence the value of the negative distance $\eta^{(n)}(\boldsymbol{\mu}_i)$ for the same evaluated point $\boldsymbol{\mu}_i$ is different at different iterations. We want to sample around points with a small value of $f(\boldsymbol{\mu}_i)$ for exploitation and with small $\eta^{(n)}(\boldsymbol{\mu}_i)$ for exploration (note $\eta^{(n)}(\boldsymbol{\mu}_i)$ is the negative of distance).

The *non-dominated sorting* ranks all previous evaluation points into different fronts, where the points in the jth front dominate all the points on the $j+1$th front. On any front m, all the points on the jth front are ranked in order of the value of $f(\boldsymbol{\mu})$ from smaller value to larger value. The detailed implementation of *non-dominated sorting* refers to Line 4 Step 1–3 in Algorithm 3 in Online Resource. The selection of center points begins from points in the first front to the last front and starts from points with the smallest value of $f(\boldsymbol{\mu})$ within each front. Note that the best solution found so far (denoted as $\boldsymbol{\mu}*$) is always selected as the first center c_1 (Line 6 in Algorithm 3 in Online Resource).

In GOPS, we add a constraint on the evaluated points for *non-dominated sorting*. Essentially, only evaluated points that are in the "Good center candidate pool" (which contains points in the best $P_{good}^{(n)}$ percent of all evaluated points, based on objective function values) are allowed to become centers and are included for *non-dominated sorting*. *Non-dominated sorting* has a time complexity of $O(MN^2)$ to generate non-dominated fronts for N evaluation points and M objective functions (In our case, $M=2$). Limiting evaluation points to the best $P_{good}^{(n)}$ percent of the objective function values cuts down the number of evaluation points N for *non-dominated sorting* and hence can significantly reduce the calculation time for non-dominating sorting compared with SOP. However, the biggest advantage of limiting the selection of candidate points to the best $P_{good}^{(n)}$ percent of objective functions is that it is likely to provide an improved set of candidate points. $P_{good}^{(n)}$ is updated by the algorithm dynamically in each iteration, which will be discussed in Sect. 3.3.

Besides the *non-dominated sorting* for center selection, two additional criteria, (1) *Tabu Rule* and (2) *Radius Rule*, are adopted from SOP to balance the exploration and exploitation further. Tabu rule is that points that were chosen as centers but did not induce an improvement in N_{fail} iteration will be forbidden from being selected as centers for a tenure of N_{tenure} iterations. The Radius constraint is that the points being selected as centers should be at least r_j distance from selected centers c_j in that iteration, where r_j is the search neighborhood sampling radius of center c_j and j is the index of center in iteration n $(j = 1, \dots, P_C^{(n)})$. The Tabu and Radius constraints in GOPS and SOP are the same. Only those points that do not violate the Tabu and Radius constraints can be selected as centers (as in Line 10 in Algorithm 3 in Online Resource). The implementation of center selection refers to Algorithm 3 in Online Resource.

3.1.3 Candidate search

To make full use of the P processors, we must in each iteration select P evaluation points $\left\{\boldsymbol{\mu}_i^{(n)}, i = 1, \dots, P\right\}$, at which the expensive black-box objective function $f(\boldsymbol{\mu}_i^{(n)})$ will be evaluated. In the original SOP, the P evaluation points are generated around P centers and the number of points selected for evaluation around each center is equal to 1. Let N_{c_j} be the number of samples around the center c_j. Hence in SOP $P_C^{(n)} = P$ $(\forall n \in \{1, \dots, MAXIT\})$, and $N_{c_j} = 1$ $(\forall j \in \{1, \dots, P_C^{(n)}\})$. In GOPS, the number of centers in the nth iteration $P_C^{(n)}$ is dynamically decreasing. Hence the number of samples around each center changes as the number of iterations increases. Note that we keep the total number of expensive evaluations $f(\boldsymbol{\mu})$ in each iteration as constant P (hence $\sum_{j=1}^{P_C^{(n)}} N_{c_j} = P$). We will demonstrate how the number of samples around each center changes as the number of iterations n increases in Sect. 3.2.

The samples around each center are generated by perturbing some selected coordinates of the current center point. We adopt the *dynamically coordinated search* from DYCORS (Regis and Shoemaker 2013) whereby the expected number of coordinates being chosen for perturbation is dynamically reduced during the search. This perturbation strategy is also used in SOP. For each center, a set of N_{cand} candidate points will be generated by perturbing only dimensions that have been randomly selected. Each coordinate of the center c_i has a probability of $p_{selected} = \varphi(n)$ being selected to be perturbed, where $\varphi(n)$ is reduced as the iteration number n increases by $\varphi(n) = \varphi_0 \times [1 - ln((n-1)P + 1)/ln(MAXIT \times P)]$, where $1 \le n \le MAXIT$. We set $\varphi_0 = min(20/d, 1)$, as in DYCORS and SOP. For those coordinates selected to vary (denoted as $I_{perturb}$), the variation of the trial points in each coordinate

$k \in I_{perturb}$ is sampled from truncated normal distribution $N_{truncated}(0, \sigma^2, a, b)$ with the standard deviation $\sigma = r_j$. r_j is the sampling radius of center c_j and the bound $[a, b] = [lb(k) - c_j(k), ub(k) - c_j(k)]$ (see Line 13 Algorithm 5 in Online Resource). For centers that are for the first time being selected as centers, the initial value of search radius r_j is equal to r_{int}. We adopt the value used in DYCORS and SOP, $r_{int} = 0.2 \times l(\Theta)$, where $l(\Theta)$ is the length of the shortest side of the hypercube Θ [as defined in Eq. (1)]. Detailed information about the truncated normal distribution can be seen in Krityakierne et al. (2016).

For center c_j we choose evaluation points by selecting N_{c_j} candidate points with the smallest surrogate value $\hat{f}(\boldsymbol{\mu})$ from the N_{cand} candidate points. Detailed implementation of the candidate search around centers is described in Algorithm 5 in Online Resource. These candidate points selected as evaluation points are sent to P processors to do all the objective function evaluations, with one evaluation per processor.

3.1.4 Adaptive learning

In the adaptive learning step, GOPS evaluates the candidate search around the center c_j, which is labeled success only if there is at least one evaluation point of the newly generated samples from center c_j (denoted as $S^{new}_{c_j}$) providing a significant improvement based on the hypervolume improvement metric (HI, used in Krityakierne et al. (2016)). The hypervolume of a set of evaluated points $S^{(n)}$ is the area that is dominated by $S^{(n)}$ on the objective space based on two objectives: (1) the objective function value $f(\boldsymbol{\mu})$ and (2) the negative of minimum distance from $\boldsymbol{\mu}$ to all other evaluated points $\eta^{(n)}(\boldsymbol{\mu})$. The hypervolume improvement is the difference between hypervolume of previously evaluated points with and without the newly evaluated point. If the value of HI exceeds a pre-defined threshold τ (usually set to be a small positive value), the search around center c_i is considered a success. Otherwise, the search around center c_i is a failure, in which case the search radius r_j (around center c_i) is reduced by half, and the failure count of the center point is increased by one (Line 3–5 in Algorithm 6 in Online Resource). Note that the value of r_j affects sampling of candidate points around centers. With a large value of r_j, the generated candidate points have a higher chance of being far from the center points. If the search around the center was not successful, it makes sense to search the region farther from that center. If the failure count exceeds a pre-defined threshold N_{fail}, the center point is added to Tabu list (Line 13–14 in Algorithm 6 in Online Resource) and will be removed from that Tabu list only after N_{tenure} iterations. The implementation of adaptive learning is explained in more detail in Algorithm 6 in Online Resource.

We can now give the general framework of the GOPS algorithm in Algorithm 1. The detailed implementation of each step in Algorithm 1 is demonstrated in Algorithm 2–5 in Online Resource. For example, "2.1" in Algorithm 1 refers to "Step 2.1" in Algorithm 2 in Online Resource. Symbols defined in definitions tables in Table B1–B2 in Online Resource. Note that the main difference between GOPS and SOP is the dynamic changes in the algorithm controlled by the varying numbers of centers $P^{(n)}_C$ and newly evaluated points around each center N_{c_j}, and $P^{(n)}_{good}$ that eliminates centers at points with very poor objective values.

Algorithm 1 General GOPS Framework

```
1  INITIALIZATION. Generate $n_0$ initial points from solution domain via
   experimental design method. Set the iteration number $n \leftarrow 1$.
2  ITERATIVE LOOP. While $n \leq MAXIT$ :
     2.1 SURROGATE FIT: Build or update the surrogate model $\hat{f}(\boldsymbol{\mu})$ based on
         previously evaluated points before nth iteration $S^{(n)}$.
     2.2 CENTER SELECTION: Calculate $P_C^{(n)}$ and Select $P_C^{(n)}$ center points
         based on non-dominated sorting, Tabu and Radius constraints (see
         Algorithm 3 in Online Resource).
     2.3 CANDIDATE SEARCH: Calculate $N_{c_j}$ for $j \in \{1,\ldots,P_C^{(n)}\}$ (see
         Algorithm 4 in Appendix B). Generate $P$ evaluation points for black-box
         evaluation (see Algorithm 5 in Online Resource).
     2.4 FUNCTION EVALUTION: Evaluate the $P$ evaluation points for black-
         box evaluation.
     2.5 ADAPTIVE LEARNING. Parameter update (see Algorithm 6 in Online
         Resource)
3  TERMINATION: While $n > MAXIT$ terminate and return $\boldsymbol{\mu}^*$, which is the
   best solution found so far.
```

Before explaining the calculation of $P_C^{(n)}$, N_{c_j} and $P_{good}^{(n)}$ in the following subsections, we first introduce a diversity factor $\beta_{diversity}^{(n)}$ which is used to control the value of $P_C^{(n)}$, N_{c_j} and $P_{good}^{(n)}$. The diversity factor $\beta_{diversity}^{(n)}$ is decreasing linearly as the number of iterations n increases. The value of $\beta_{diversity}^{(n)}$ at the nth iteration is calculated as:

$$\beta_{diversity}^{(n)} = 1 - (n-1)/(MAXIT - 1) \tag{4}$$

hence $\beta_{diversity}^{(n)} = 1$ when $n = 1$ to $\beta_{diversity}^{(n)} = 0$ when $n = MAXIT$.

3.2 Dynamic number of centers $P_C^{(n)}$ and evaluations per center N_{c_j}

In GOPS, to increase the exploitation ability of the algorithm during the optimization search, we dynamically reduce the number of centers in each iteration to focus on centers that seem to be especially promising as we obtain more information. We add a hard constraint on the number of centers in each iteration so that $P_C^{(n)} \leq P_C^{(n)max}$ for the nth iteration. The value of $P_C^{(n)max}$ is being dynamically reduced during the operation search process and controlled by the diversity factor $\beta_{diversity}^{(n)}$. Since we want at least one center to be selected in each iteration, the value of $P_C^{(n)max}$ should be at least one (Line 5 in Algorithm 3 in Online Resource). Hence the maximum number of centers in the nth iteration is

$$P_C^{(n)max} = \max\left(\left\lceil P \times \beta_{diversity}^{(n)} \right\rceil, 1\right) \tag{5}$$

Note the ceiling function $\lceil \mathbb{R} \rceil$ is used to make sure $P_C^{(n)max}$ is an integer in Eq. (5). The formula in Eq. (5) allows a larger number of centers to be selected in the initial search stage and allows only a smaller number of centers being selected in the final search stage. With a smaller number of centers, there is more focus put on exploitation in the later part of the search. Initially (i.e., when n is small), the number of centers is much larger, so the focus is more on exploration. As the number of iterations increases, GOPS eventually has more samples around one center to allow sufficient exploitation of each dimension of a good solution that is at the center point. This is very important for high dimensional problems. The original version of SOP only evaluates one sample around each of the centers, which limits exploitation, especially in high dimensional problems.

To further enhance the exploitation ability of the algorithm, we add one more constraint for the number of samples around the best solution found so far. Note that the best solution found so far will always be selected as the first center point c_1. We set the minimum number of samples around the center that is the best solution found so far to be $N_{c_1}^{min}$, which is dynamically increasing as the number of iteration increases:

$$N_{c_1}^{(n)min} = \max\left(\left\lceil P \times (1 - \beta_{diversity}^{(n)}) \right\rceil, 1\right) \tag{6}$$

In GOPS, we treat the best solution found so far differently from other centers, which is different from the SOP (Krityakierne et al. 2016) algorithm. We want to sample more around the best solution found so far, especially in the final search stage. We dynamically increase the value of $N_{c_1}^{min}$ as the optimization iteration increases. This helps exploitation in each dimension of a good solution in the final optimization search stage.

To actually implement GOPS, we first decide the number of evaluations points around the best center c_1 (Line 2–6 Algorithm 4 in Online Resource):

$$N_{c_1} = \begin{cases} \left\lceil P \big/ P_C^{(n)} \right\rceil & \text{if } \left\lceil P \big/ P_C^{(n)} \right\rceil > N_{c_1}^{\min} \\ N_{c_1}^{\min} & \text{else} \end{cases} \tag{7}$$

hence the number of evaluation points that could be assigned to the remaining $P_C^{(n)} - 1$ centers is $P - N_{c_1}$. We try to treat the reminding $P_C^{(n)} - 1$ centers $\left\{c_2, \dots, c_{P_C^{(n)}}\right\}$ equally. Hence the remaining $P - N_{c_1}$ evaluation points are distributed to the $P_C^{(n)} - 1$ centers by cycling though the reminding centers set $\left\{c_2, \dots, c_{P_C^{(n)}}\right\}$ until all the $P_C^{(n)} - 1$ evaluation points are assigned. The detailed implementation of the calculation of N_{c_j} for $j = 1, \dots, P_C^{(n)}$ refers to Line 2–15 Algorithm 4 in Online Resource.

3.3 $P_{good}^{(n)}$ for a "Good Center Candidate Pool"

In GOPS, we introduce $P_{good}^{(n)}$, which is a variable, to ensure a good center candidate pool. We rank all the evaluation points found so far based on their objective function value $f(\boldsymbol{\mu})$ (where lowest is best). Then in iteration n, the best $P_{good}^{(n)}$ percent of the previously evaluated points are put in the "good center candidate pool." The percentage of the "good center candidate pool" in the nth iteration $P_{good}^{(n)}$ declines as iteration n increases so (Line 1–3 Algorithm 3 in Online Resource):

$$P_{good}^{(n)} = P_{good}^{ini} \times \beta_{diversity}^{(n)} + P_{good}^{end} \times (1 - \beta_{diversity}^{(n)}) \tag{8}$$

where p_{good}^{ini} and p_{good}^{end} are parameters (values are given in Table B1 in Online Resource) to control the percentage of solutions that can be selected as center points in the initial and final iterations.

The introduction of the "good" center candidate pool is to prevent the selection of the "poor" solutions during the center selection. In the original SOP, the centers in the nth iteration are selected from all the evaluation points found so far before iteration n based on the non-dominated sorting on two objectives (1) objective function value $f(\boldsymbol{\mu})$ and (2) the negative minimum distance $\eta^{(n)}(\boldsymbol{\mu})$ to all other evaluated points. Recall that the distance function is used to encourage exploration into unexplored areas.

The center selection process in the SOP algorithm will iteratively select solutions from the first front and then from the remaining fronts (going in order front 2, front 3, etc.) until enough centers are selected. A drawback of the center selection method

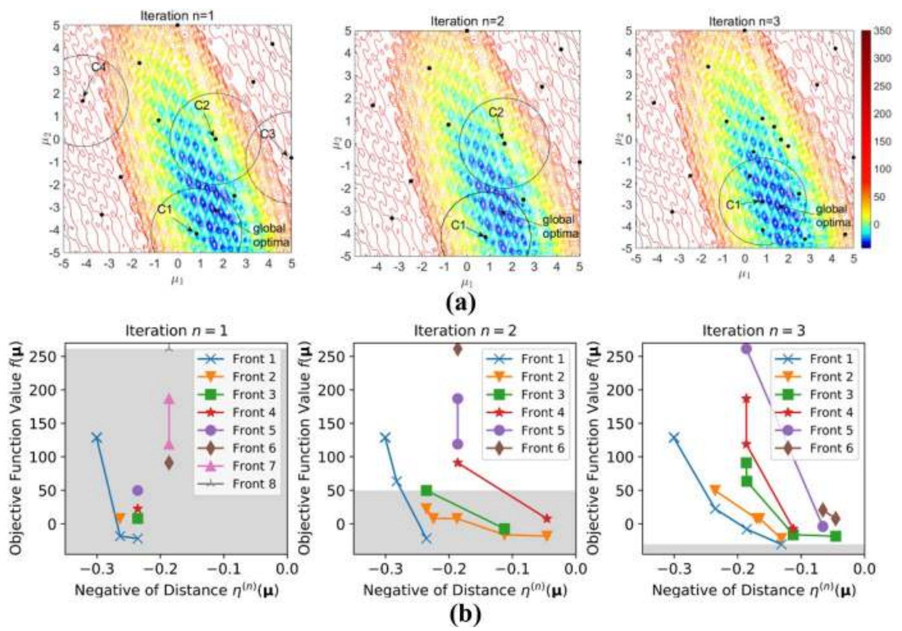

Fig. 1 Example of center selection in GOPS using the $P_C^{(n)max}$, $N_{c_1}^{min}$, and $P_{good}^{(n)}$ strategies on optimization of $f(\mathbf{\mu})$ that is the two-dimensional Rastrigin Function problem. Three successive optimization iterations are shown (i.e., $n = 1, 2, 3$, where n is iteration number). Previously evaluated points are plotted (pane a) in terms of decision variable values (black dots) and (pane b) in terms of objective function values $f(\mathbf{\mu})$ and negative of distance $\eta^{(n)}(\mathbf{\mu})$. (Colored lines for different fronts). In **a**, the surface of the Rastrigin function is shown in contour plot. The circles in pane **a** denote the search radius of the selected centers. In **b** the evaluated points on different levels of non-dominated fronts are shown. In **b** the shaded region denotes the "Good Center Candidate Pool". Evaluated points being selected as centers are noted with $C1$, $C2$, $C3$, $C4$ in both **a** and **b**

in SOP is that an evaluation point z with a very large objective function value $f(z)$ (a bad feature) has a reasonable chance of being selected in SOP as a center just because z is far from other previously evaluated points. Moreover, this will continue from the beginning of the optimization to the end of the optimization in each iteration. In the early iterations, search around these solutions might be useful for exploration, but as we get closer to the maximum number of iteration, we want to focus on the search around evaluation points with low objective function values. By contrast, SOP's approach will cause a waste of computing resources by searching around those centers that have evaluated points with high objective function value in the later phases of the search.

In Fig. 1, there is a simple example of the GOPS algorithm on center selection for three successive iterations on a two-dimensional optimization problem. The $f(\mathbf{\mu})$ test problem used in Fig. 1 is the F15 function from the 14 BBOB synthetical test problem (Hansen et al. 2009) that will be used to test GOPS's performance later but with 10, 21, and 40 dimensions. In Fig. 1, the range of the two decision variables μ_1 and μ_2 is $[-5, 5]$. For the example in Fig. 1, we used the result from a real optimization trial where we set the number of initial points $n_0 = 12$, the maximum number of iterations $MAXIT = 3$, and the number of samples in each iteration $P = 4$. The value of P^{ini}_{good} and P^{end}_{good} are set to be 100% and 1%, respectively. We show three successive iterations (i.e., $n = 1, 2, 3$). According to Eq. (8), the percentage of all evaluated points that are classified into "Good center candidate pool" at iterations 1, 2, and 3 are $P^{(1)}_{good} = 100\%$, $P^{(2)}_{good} = 50.5\%$, $P^{(3)}_{good} = 1\%$, respectively. The number of centers in each iteration $P^{(n)}_{C}$ is dynamically decreasing from four in iteration 1 to one in iteration 4. The "good center candidate pool" controlled by $P^{(n)}_{good}$ effectively prevents those points with a very large objective function value being selected as center points. Note that in the original SOP, these points with a large value of $f(\boldsymbol{\mu})$ in the first front in iteration 1 (i.e., the center points C3 and C4 in Fig. 1b) are most likely to be selected as center points again by SOP in iteration 2 and 3 just because they are far from other evaluated points. Exploring the region around these points, which are far from the global optima, is less likely to improve the best solution found so far. Hence selecting center points with poor objective diminishes somewhat the effectiveness of SOP, and this problem of selecting center points with poor objective values is eliminated in GOPS.

3.4 Convergence of GOPS

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 the number of evaluations per iteration* $P > 1$, *GOPS converges almost surely to the global minimum.*

The proof of Theorem 1 is given in Online Resource. The convergence analysis of GOPS is similar to that of SOP. Note that the changes between GOPS and SOP are (a) SOP has the number of centers always equal to the number of processors, and the number of samples per center is constant, whereas in GOPS the number of centers $P^{(n)}_{C}$ and number of samples around each center N_{c_j} can change in each iteration (in Sect. 3.2) and (b) GOPS adds constraints to ensure the objective value of the selected centers are below a target percentile to prevent poor evaluation points being selected as centers (in Sect. 3.3). From the original SOP paper, the convergence analysis of SOP is preserved when the following three conditions are met: (1) in each dimension of the vectors that are the P centers, there is a bounded-away-from-zero probability of being perturbed, (2) the range of sampling for a variable is a truncated normal distribution covering the entire compact hyperrectangle domain, (3) the variance of the normal distribution (perturbation distribution) is bounded above zero because it can only be reduced in half at most N_{fail} times. These conditions of SOP's convergence proof are independent of the number of centers and the number of samples around each center and is also independent of the location of centers. GOPS does not violate the three conditions above, and these features are used in the proof of convergence for GOPS.

4 A multi-modal optimization with objective function based on nonlinear PDE model

We consider the PDE model based parameter estimation problem [a particular case of the optimization problem in the form of Eq. (1)], which can be generalized in the following form:

$$\begin{aligned} &\min_{\boldsymbol{\mu}} f(\boldsymbol{\mu}) = g((\mathbf{u}(\boldsymbol{\mu}), \mathbf{u}) \\ &\text{s.t.} \mathbb{E}(\boldsymbol{\mu}) \text{ is solved} \\ &\boldsymbol{\mu} \in \Theta \subset \mathbb{R}^d \end{aligned} \tag{9}$$

where $\mathbb{E}(\boldsymbol{\mu})$ is a parameterized PDE model that involves a system of partial differential equations. $\mathbf{u}(\boldsymbol{\mu})$ is the solution from the PDE model $\mathbb{E}(\boldsymbol{\mu})$ given input parameter vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots \mu_d)$, where d is the number of parameters included in the calibration. Θ is the d dimensional solution space usually defined by the upper bound and lower bound of the values of the parameters, $\Theta = \{\mathbf{lb} \le \boldsymbol{\mu} \le \mathbf{ub}\} \subset \mathbb{R}^d$. The objective function of the optimization $f(\boldsymbol{\mu})$ equals an error function g that evaluates the difference between the simulated solution $\mathbf{u}(\boldsymbol{\mu})$ to the desired state $\hat{\mathbf{u}}$. These and other variables are defined in Table B3 in Online Resource.

Note that we consider an optimization problem where more than one state variable is simulated in the PDE model $\mathbb{E}(\boldsymbol{\mu})$. For example, the PDE model analyzed latter in Sect. 4.1 simulates different kinds of water quality substances in the water body simultaneously. Hence, the vector $\mathbf{u} = \{u_1, u_2, \dots u_{Ns}\}$ contains a set of simulation outputs for N_s state variables (i.e., different substances). $\mathbf{u} = \{u_1, u_2, \dots u_{Ns}\}$ is the desired state of these N_s variables. In our application, the desired state is a vector of observation data points used for model parameter calibration. Note that N_s is the number of state variables considered in the objective function, which can be smaller than the total number of state variables included in the PDE model. This situation could happen when there are state variables that do not appear in the objective function (because there is no observation data available) but that are necessary for the simulation of other essential state variables. For example perhaps no observation data is available on the organic matter in fast decomposing status (and hence no "desired state").

For the above PDE-constrained parameter optimization problem, it is the execution of the model simulation, i.e., the evaluation of $\mathbb{E}(\boldsymbol{\mu})$ in Eq. (9) that takes the majority of the computational time in optimization. In the following subsection, we provide the details of a real word PDE model for the water quality simulation of a tropical reservoir, which is referred to as WAQ in the following text. The GOPS algorithm is applicable to expensive, multi-modal functions without derivative information available in general, including objective functions that require the solution of a PDE model. In Sect. 4.1 and later, we discuss our real-world PDE model used as an application in this paper.

4.1 Partial differential equation models of water quality

One application of PDE equations is to simulate or predict the spatial and temporal behavior of water quality substances in lakes or reservoirs (Matta et al. 2016; Smits and van Beek 2013), groundwater aquifers (Gorelick et al. 1993; Mugunthan et al. 2005; Pinder and Celia 2006) and other water bodies. The water flow carries with it many substances, including some that are not beneficial, like algae or pollutants. These models are essential in water management so that water managers can have them as tools (1) to estimate the water quality in area or time (not measured) based on the model plus data measured at other points in time or space and (2) to estimate future events. For example, if there is going to be a significant change, e.g., nutrient emissions to the lake or reservoir or a change in water level (Matta et al. 2016), managers can evaluate what is the response of the water by feeding the current situation into the model and running it into the future.

In response to the demand for models to simulate the surface water systems, engineering firms have developed commercial and open-source software [e.g., Delft3D-WAQ (Hydraulics 2003), ELCOM (Hodges and Dallimore 2006), MIKE ECO lab (Butts et al. 2012), and WASP (Wool et al. 2006)]. This software is widely used around the world for water management.

These PDE models involve a large number of model parameters, which need to be calibrated to the measured data (known as solving an inverse problem) in order to simulate the studied system correctly. The PDE model used in this study dynamically simulates the water quality dynamics in an irregularly shaped tropical reservoir in Singapore that has over 250 ha of water surface and uneven depth. The water quality model describes the nonlinear dynamic process by which nutrients entering the lake are converted to different chemical forms and some nutrient species are taken up by algae. Hence the model has multiple nonlinear interactions, leading to an objective function (goodness of fit between model simulation output and measurement) that has multiple local minima. The multi-modal nature of the objective function will be discussed in Sect. 4.3. We will call the numerical PDE model based on Singapore data "WAQ."

The Delft3D-WAQ software suite (Hydraulics 2005) is employed in WAQ to simulate the transport of substances (e.g., nutrients) by solving the three-dimensional advection–diffusion equation as below:

$$\frac{\partial C_i(\boldsymbol{\mu})}{\partial t} = -\vec{v}\cdot\vec{\nabla}C_i(\boldsymbol{\mu}) + \vec{\nabla}\cdot(\vec{D}\cdot\vec{\nabla}C_i(\boldsymbol{\mu})) + S(C_i) + f_R(C_i(\boldsymbol{\mu}),t) \quad \text{in } \Omega\times[0,T] \tag{10}$$

where C_i is the concentration of the ith substance included in the PDE simulation model. There are a total of 64 substances included in the PDE model simulation. $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_d)^T$ is the model calibration parameters vector ($\boldsymbol{\mu} \in \mathbb{R}^d$), $\vec{\nabla}C_i = \left(\frac{\partial C_i}{\partial x}, \frac{\partial C_i}{\partial y}, \frac{\partial C_i}{\partial z}\right)$ is concentration gradient, where *x, y, z* represent coordinates in three spatial dimensions; $\Omega \subseteq \mathbb{R}^3$ represents the three-dimensional space domain. T is the simulation period length; $\vec{v}$ is the velocity vector.$\vec{D} = (D_x, D_y, D_z)$ represents the diffusion coefficient in different spatial directions. $S(C_i)$=sources or sinks of

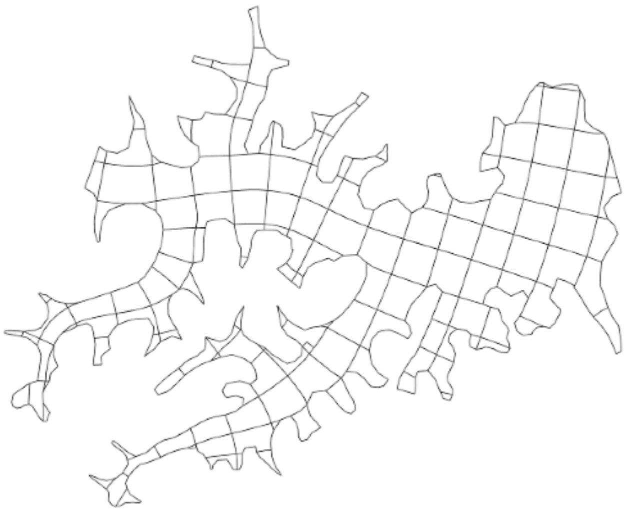

Fig. 2 Horizontal grid layout of the lake (has over 250 ha of water surface and uneven depth) for which the water quality model (WAQ) is computed

substance C_i, which could be time and spatial variant. $f_R(C_i(\boldsymbol{\mu}), t)$ is the reaction term involving the physical, chemical and biological processes. For physical processes, e.g., settling, evaporation or volatilization, $f_R(C_i(\boldsymbol{\mu}), t)$ describes the loss or increase rate of substance C_i at a specific location. For chemical or biological processes, $f_R(C_i(\boldsymbol{\mu}), t)$ generally characterizes the relation between substance C_i and all other substance $C_{j,j\neq i}$ at precisely the same location at that time. For example, ammonia and oxygen can form nitrite through chemical reactions. Dissolved nutrients are transformed into organic nutrients during the growth of algae. The decay of the substance organic nutrients to dissolved nutrients. In the aquatic environment, the relation between different substances is complicated, and in many forms, we are not going to list them all here (Hydraulics 2003).

The above Eq. (10) is only the simulation of one substance. Since there are, in total, 64 substances simulated in the WAQ model, for each substance, there is an advection–diffusion equation [i.e., Eq. (10)] to solve. The reaction term $f_R(C_i(\boldsymbol{\mu}), t)$ links the advection–diffusion equation [in Eq. (10)] of different substances C_i together, which makes the solving of the PDE simulation $\mathbb{E}(\boldsymbol{\mu})$ complicated.

The WAQ model is discretized in space by finite volume method leading to 1141 segments for our lake example. Figure 2 plots the horizontal grid layout of WAQ. The simulation period of the PDE model $\mathbb{E}(\boldsymbol{\mu})$ is 1 year. We set the time interval Δt to be 5 min, resulting in 105, 120 time steps. One run of the WAQ for a one-year simulation takes around 22 min to run on a Linux platform with CPU E5-2690 @2.60GHZ.

4.2 The objective function

In the parameter estimation problem for WAQ, the desired state of a substance s at specific locations $l \in \Omega$ and time $t \in [0, T]$ is the real-world observation data in the tropical reservoir, denoted as $C_{s,l,t}$. We want to find the value of the parameter vector $\boldsymbol{\mu}$ with which the simulated concentration of a substance s at the same location l at the same time t, $C_{s,l,t}(\boldsymbol{\mu})$, from the WAQ model, is as close to $C_{s,l,t}$ as possible when considering all substances, times, and locations. To assign a scalar measure of closeness between the simulated $C_{s,l,t}(\boldsymbol{\mu})$ and $C_{s,l,t}$, we use the goodness-of-fit metric adopted from previous studies in this field (Gibson et al. 2006):

$$gf_{l,s} = \frac{\frac{1}{m}\sum_{m=1}^{M}\left|\overline{C}_{l,s,m}(\boldsymbol{\mu}) - \overline{C}_{l,s,m}\right|}{\sigma_{l,s}} \tag{11}$$

where m is the index of the month, $m \in [1, M], M = 12$. $\bar{C}_{l,s,m}(\boldsymbol{\mu})$ is the monthly mean value of the simulated substance s at the location l in the month m. $\bar{\hat{C}}_{l,s,m}$ is the monthly mean value of the observed concentration in real-word of substance s at the location l in the month $m.\sigma_{l,s}$ is the standard deviation of $\bar{\hat{C}}_{l,s,m}$ with degree of freedom 11 (=12–1).

Hence our goal is to calibrate the model parameters to the observed data by optimizing the objective function of the optimization in Eq. (9), which is the sum of the goodness of fit $gf_{l,s}$ at different locations and for different substances, as below:

$$f(\boldsymbol{\mu}) = \sum_{l}\sum_{s} gf_{l,s} \tag{12}$$

For the studied case, there are bi-monthly observation data of 5 different substances, including Chlorophyll-A, Total Nitrogen, Total Phosphorus, Ammonia, and Nitrate at two locations in the tropical reservoir for 1 year. The biological connection between them is Chlorophyll-A that is an indicator of algal concentrations and algal growth is strongly affected by the availability of Nitrogen and Phosphorous, which are contained in the remaining four substances.

In total, 21 model parameters ($d = 21$) are selected for the model calibration. These parameters are from the reaction term $f_R(C_i(\boldsymbol{\mu}), t)$ in Eq. (10). They were introduced in the model to characterize the water quality physical (e.g., sedimentation), chemical (e.g., nitrification and denitrification), and biological processes (e.g., the phytoplankton growth) processes. The value of these 21 parameters affects the simulated concentration of all the five substances.

This problem is a good example of why the availability of parallel algorithms is so crucial because the number of model evaluations required by optimization algorithms to get a good solution is very high with such a high dimensional and computationally expensive problem. For example, assume 1200 evaluations are needed to get a good solution (for the value of 21 parameters), the time to compute these 1200 evaluations in serial would be about 18 days. Water managers typically oversee multiple water bodies, so having to wait 18 days for the analysis for each water body is inconvenient. With the 48 processors used in this example, it takes only 0.375 days to compute the 1200 evaluations. Besides, the cost for the computation time (in serial or parallel) using standard rates (US$0.019/core-hours) for the NSCC supercomputer was a total of US$8.37, so it is a very modest cost. Hence an efficient parallel algorithm like GOPS is needed.

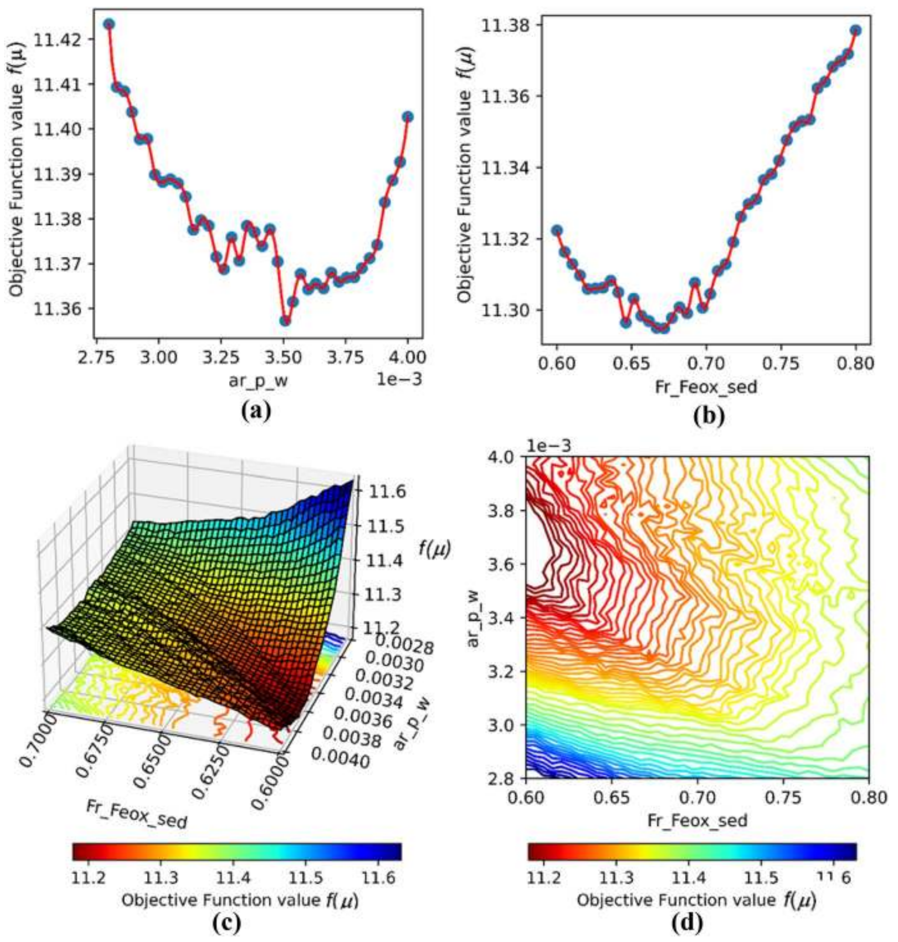

Fig. 3 The values of $f(\boldsymbol{\mu})$ for various values of two parameters *ar_p_w* and *Fr_Feox_sed* in $\boldsymbol{\mu}$. **a** Only the parameter *ar_p_w* is varied. **b** Only the parameter *Fr_Feox_sed* is varied. **c** Both the parameter *ar_p_w* and *Fr_Feox_sed* are varied. **d** The 2-D contour plot of the surface plot in **c**

4.3 Demonstration of the nonlinear, multi-modal features of water quality model calibration

As mentioned in Sect. 4.2, the WAQ model contains a system of partial differential equations describing many complicated physical, chemical, and biological processes. Hence, the relation between the objective function $f(\boldsymbol{\mu})$) and the value of these parameters $\boldsymbol{\mu}$ can be highly nonlinear and multi-modal.

We select two model parameters from the 21 parameters to demonstrate the nonlinearity and multi-modal feature of the optimization problem. The first parameter is *ar_p_w*, the stoichiometric constant for phosphorus in refractory detritus in water column. It controls the first order mineralization rate of the organic phosphorus detritus (Smits and van Beek 2013). The second parameter is *Fr_feox_sed*, the fraction of the iron (III) oxide over the reactive iron in sediment, which affects the adsorption of dissolved phosphorus in the sediment. Both parameters have a direct or indirect influence on the concentration of dissolved phosphorus in the water column. The concentration of the dissolved phosphorus will then affect the growth of the algae, which will affect many other water quality substances, such as dissolved nitrogen and chlorophyll-a. Hence, the response of the objective function $f(\boldsymbol{\mu})$ to the variation of the parameter value is complex and likely to be multi-modal.

A major focus of this research is to optimize multi-modal functions. To demonstrate the multimodality of the lake water quality objective function [Eq. (12)], we computed the objective function value of $f(\boldsymbol{\mu})$ when the value of the two parameters were changing independently, and all other parameters were kept at their original value. By computing the values of the PDE model (Delft3D-WAQ) and substituting the values into Eq. (12), we got the values of the objective function $f(\boldsymbol{\mu})$, which are plotted below in Fig. 3.

Figure 3a, b show the response of objective function $f(\boldsymbol{\mu})$ to the variation of one parameter at a time. In other words, the values of all the other 20 parameters are kept unchanged as the value of the one parameter is varied. It is quite apparent that the optimization problem in one dimension is nonlinear, nonconvex, and has multiple local minima. Changing one parameter can lead to improving the fit of some substances and worsening the fit of other substances with complex interactions, leading to the multi-modal objective function seen in Fig. 3a, b.

Figure 3c, d show the landscape of $f(\boldsymbol{\mu})$ when the values of the two parameters are changing simultaneously while the values of the other 19 parameters are kept at their original value. As shown in both Fig. 3c, d, the landscape of the objective function surface has a large number of local minima. Figure 3c, d indicate that the impact of one parameter on the model simulation output is affected by the value of another parameter. For example, when *ar_p_w* = 0.004, increase of *Fr_Fe_ox_sed*'s value leads to an increase in the objective function value $f(\boldsymbol{\mu})$. On the contrary, when *ar_p_w* = 0.0028, the objective function value $f(\boldsymbol{\mu})$ is generally decreasing with the increase of *Fr_Fe_ox_sed*'s value.

We only present in Fig. 3c, d the objective function $f(\boldsymbol{\mu})$ landscape over two parameters out of the 21 parameters, since it is challenging to include the investigation of all the combinations of parameters. For the 21-dimensional optimization problem, the relation between the objective function value $f(\boldsymbol{\mu})$ and the parameter value $\boldsymbol{\mu}$ is expected to be more complicated and to have even more local minima.

5 Numerical experiments

5.1 Alternative parallel optimization algorithms

We compared our algorithm to the original SOP algorithm and the parallel MADS algorithm Parallel Space Decomposition of MADS (PSD-MADS) (Audet et al. 2008b) with the use of variable neighborhood search (VNS) option (Le Digabel 2011). The NOMAD's user guide (Le Digabel 2011) indicates that PSD-MADS is much more efficient than other parallel versions of MADS algorithms on the larger problems with the number of decision variables above 20. The use of VNS with MADS enables the algorithm to escape from local minima and to search for the global minimum (Audet et al. 2008a, 2013; Le Digabel 2011). We refer to PSD-MADS with VNS as PSD-MADS-VNS. Krityakierne et al. (2016) compared the SOP algorithm with many alternative methods, including Parallel Stochastic RBF method (Regis and Shoemaker 2009) and an evolutionary algorithm that uses radial basis function approximation (ESGRBF) (Regis and Shoemaker 2004; Shoemaker et al. 2007). Their results show that SOP is more efficient than these algorithms, so in this paper, we did not consider these algorithms here.

We use Latin hypercube sampling for the generation of the initial evaluation points in both GOPS and SOP algorithms for all the following experiments. The number of candidate points around each center N_{cand} is set to be $\min(500d, 5000)$. The initial sampling radius r_{int} is $0.2 \times l(\Theta)$, where Θ is the solution domain, a hyperrectangle, and $l(\Theta)$ denotes the length of the shortest side of the hyperrectangle Θ. The threshold value of failure account N_{fail} and the tenure length N_{tenure} are set to be 3, and 5, respectively. The tolerance for local improvement τ is 10^{-5}. The values of the parameters above (applied to both GOPS and SOP) are kept the same as the value suggested in the original SOP paper. For the GOPS algorithm, there are two more user-defined parameters p^{ini}_{good} and p^{end}_{good}. Good results are obtained by setting $p^{ini}_{good} = 50\%$ and $p^{end}_{good} = 1\%$. Like the other hyperparameters in Table B1 (in Online Resource), we use these parameter values for all the numerical experiments, including synthetical test problem and the WAQ problem, so we do not tune parameters to specific problems.

The implementation of PSD-MADS-VNS is in NOMAD version 3.9, and we use MPI for parallel implementation (Snir et al. 1996). We followed MADS instructions on how to set up the problem, as described below. In PSD-MADS, the black-box problem is divided into lower dimension subproblems where only a subset of variables (ns out of d) are variant with the value of the rest of $d - ns$ variables fixed. ns is a parameter the value of which is chosen by users. The value of these $d - ns$ fixed variables are taken directly from the best solution found so far. Each subproblem is assigned to a worker that executes the MADS algorithm on the ns-dimensional subproblem. The worker terminates the MADS search on the assigned subproblem after

bbe_{max} black-box evolutions and sends the best solution it found back to the master, where bbe_{max} is another user-defined parameter. The master then collects the return solutions from all workers and updates the best solution found so far. The above processes are repeated until the terminal conditional of PSD-MADS-VNS algorithm is met (i.e., the maximum number of total black box evaluations).

For PSD-MADS-VNS tests, we set the maximal number of evaluations performed by each worker processor bbe_{max} to be ten, and the number of variables considered by the worker *ns* to be two as previous study (Le Digabel et al. 2010) suggests good results were obtained with this setting for problems with dimensions 20 and 50. NOMAD's VNS option has been used in order for the algorithm to be a global optimizer that can escape from local optima (by setting VNS to be 1). The PSD-MADS-VNS algorithm needs to start from a given initial trial point, which we set to be the best solution in the initial experimental design of GOPS and SOP for the respective trial.

5.2 Algorithm comparison on test function suit

The performance of the algorithms is investigated on 14 multi-modal benchmark functions F3, F4, F8, F9, and F15–F24 taken from the BBOB test suite (Hansen et al. 2009) before it is applied to the WAQ problem. There are in total 24 noiseless test functions in the BBOB test suite. The rest of the 10 noiseless functions are uni-modal, which is not the focus of this study. These 14 BBOB problems (F3, F4, F8, F9, and F15–F24) are all challenging multi-modal noiseless functions.

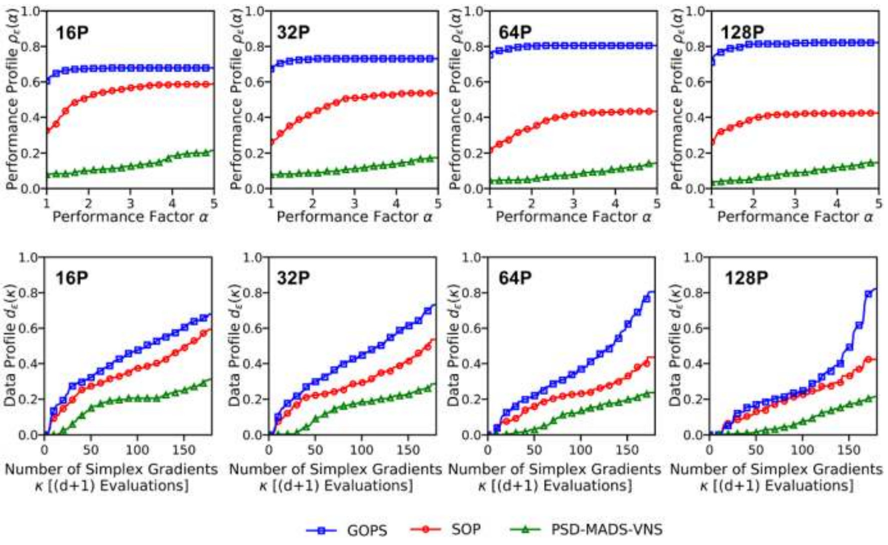

Fig. 4 Performance profile (upper pane) and data profile (lower pane) for accuracy level $tol = 10^{-3}$ for algorithms SOP, GOPS, PSD-MADS-VNS with 16, 32, 64, 128 processors on 14 10-dimensional BBOB test problems with 30 trials. High value is best

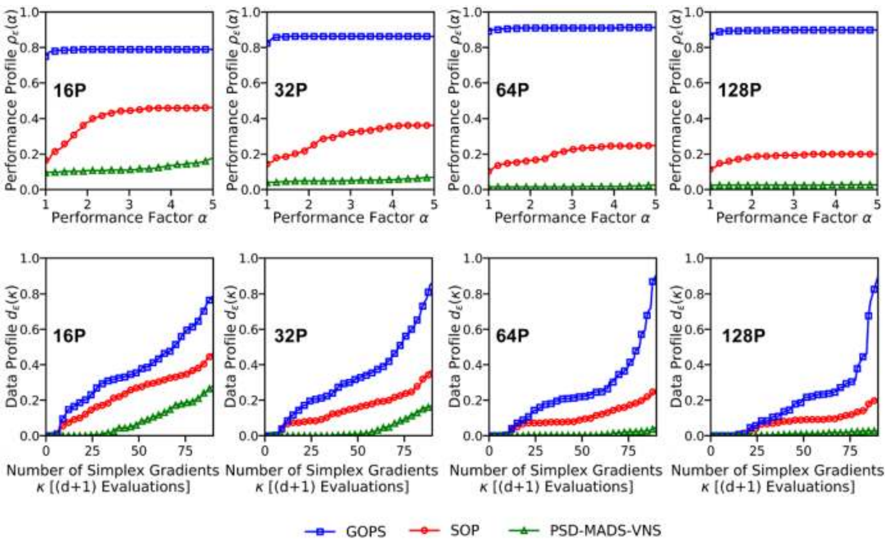

Fig. 5 Performance profile (upper pane) and data profile (lower pane) for accuracy level $tol = 10^{-3}$ for algorithms SOP, GOPS, PSD-MADS-VNS with 16, 32, 64, 128 processors on 14 21-dimensional BBOB test problems with 30 trials. High value is best

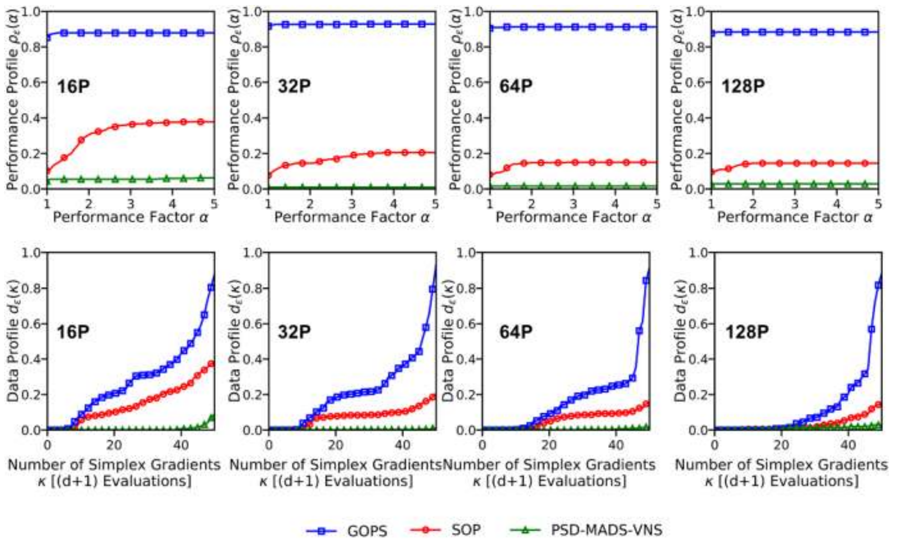

Fig. 6 Performance profile (upper pane) and data profile (lower pane) for accuracy level $tol = 10^{-3}$ for algorithms SOP, GOPS, PSD-MADS-VNS with 16, 32, 64, 128 processors on 14 40-dimensional BBOB test problems with 30 trials. High value is best

The BBOB test suite enables varying the dimensions of the problems. We test the algorithm's performance on the 14 test functions with three different dimensions (i.e., 10, 21, and 40 dimensions). We set the dimension of test functions to be 21 dimensional in order to be consistent with the dimension of the WAQ problem. The search domain for all the ten BBOB test problems is $[-5, 5]^d$, where $d = 10, 21, 40$.

All algorithms are tested with the number of processors $P = 16$, 32, 64, and 128. We use the notation *A-16P*, *A-32P*, *A-64P*, and *A-128P* in the text and figures below to distinguish the number of processors used by the algorithm *A*. For each problem, we repeated the optimization experiments with 30 trials for each algorithm. All algorithms use the same initial experiment design (randomly generated) in each trial in order to facilitate a fair comparison.

Figures 4, 5 and 6 shows data and performance profiles of all the three algorithms when different numbers of processors are used (i.e.,$P = 16, 32, 64, 128$) on 14 BBOB test problems in 10, 21, and 40 dimensional, respectively. We use methods from Moré and Wild (2009) and Müller (2016) to generate these data and performance profiles that consider the results of all trials on all problems to compare the overall performance of each algorithm.

The explanation of the calculations in these profiles is given in Online Resource. The performance profile demonstrates how well an algorithm performs over other algorithms on a set of problems. Data profile illustrates the percentage of problems that could be solved with the accuracy level of *tol* by an algorithm given a number of function evaluations. For both profile plots, high values indicate the best algorithms.

Both the data profiles and performance profiles (Figs. 4, 5 and 6) indicate that GOPS outperforms both SOP and PSD-MADS-VNS algorithm for all cases. SOP performs better than PSD-MADS-VNS for all cases (i.e., number of processors, $P = 16$, 32, 64, and 128) and on all the three different dimensions (i.e., 10, 21, and 40 dimensions).

The upper panes of Figs. 4, 5 and 6 show performance profiles with high accuracy levels $tol = 10^{-3}$ when 16, 32, 64, and 128 processors are used respectively on all 14 synthetic test problems in their 10, 21, and 40 dimensional versions.

Table 1 Percentage of solutions from SOP and PSD-MADS-VNS that is worse than that of GOPS (with 16, 32, 64, and 128 processors) in terms of mean objective function values over 30 trials after 1920 function evaluations (excluding 2(d + 1) evaluations in the initial experimental design)

	SOP	PSD-MADS-VNS	SOP	PSD-MADS-VNS	SOP	PSD-MADS-VNS	SOP	PSD-MADS-VNS
	16P (%)		32P (%)		64P (%)		128P (%)	
10D	1.2	25.7	5.6	31.4	1.4	30.3	23.8	69.4
21D	18.2	117.2	43.5	182.7	112.7	1632.5	126.8	2380.2
40D	64.1	1042.6	153.8	3542.2	290.3	7511.6	104.3	3197.5

The percentage is averaged over 14 BBOB test functions. Results on BBOB with three different dimensions (10, 21, and 40) are shown

From the performance profiles, we can see that the differences between GOPS and the other two algorithms (SOP and PSD-MADS-VNS) are more significant when larger numbers of processors are used and in higher dimensional problems. The percentage of problems for which GOPS is the fastest is increasing when a larger number of processors are used. The performance profiles also show that PSD-MASD-VNS is the worst among the three, and PSD-MASD-VNS is the fastest on only less than 10% of problems for all cases.

The data profiles (Figs. 4, 5 and 6 (lower panel)) show that the performance difference between GOPS and SOP increases as the number of evaluations (measured as simplex gradients) increases. This indicates that the adaptive diversity feature of GOPS (related to changes in $P_C^{(n)}$, N_{c_j}, and $P_{good}^{(n)}$) helps the exploration in the later stage of the algorithm. More detailed analysis of the performance and data profiles is in Online Resource.

Table 1 provides a summary of the performance of SOP and PSD-MADS-VNS over GOPS based on the mean objective function value (over 30 trials) that each algorithm achieved on all problems (in 10, 21, and 40 dimensional) after 1920 function evaluations (excluding $2*(d+1)$ evaluations in the initial experimental design) when the number of processors $P=16$, 32, 64, 128. The detailed results of mean objective function value with the standard error (over 30 trails) for each algorithm on each BBOB test problems is given in Table A1 to Table A3 of Online Resource. The percentage in Table 1 is calculated as follows: given the number of processors to be 16 (as an example), let X be the algorithm compared with GOPS-16P (e.g., $X=$SOP-16P or PSD-MADS-VNS-16P), and let Y_i^X ($i=1,\ldots,14$) be the solution of algorithm X on each of the 14 BBOB problem in d-dimension, Y_i ($i=1,\ldots,14$) be the solution of GOPS-16P on each of 14 BBOB problem in d-dimensional. The percentage for algorithm X on d-dimensional problems is $1/14\sum_{i=1}^{14}\left[(Y_i^X-Y_i)/|Y_i|\right]$. Since all the 14 BBOB test problems are minimization problems, the percentage in Table 1 denotes the percentage of time that algorithm X's solution is worse (if positive percentage) or better (if negative percentage) than GOPS's solution.

The numerical results in Table 1 show that GOPS in general obtained better solutions than SOP and PSD-MADS-VNS on all cases (when $P=16$, 32, 64, and 128), since the percentage in Table 1 are all positive. The positive percentages denote the percentage of SOP or PSD-MADS-VNS's solutions that are worse than GOPS's solution. The percentage is larger on higher dimensional problems (for X being either SOP or PSD-MADS-VNS), which means that GOPS's solution is much better than SOP or PSD-MADS-VNS on higher dimensional problems. The percentage for PSD-MADS-VNS is larger than that of SOP, which indicates that PSD-MADS-VNS is much worse than SOP's solution. These results are consistent with the conclusion from the data and performance profiles above. The percentages in Table 1 are the average value of all the 14 BBOB test problems. The detailed results on each problem in Table A1 to Table A3 in the Online Resource also show that GOPS obtained the best solution on most of the problems if not all and GOPS's solution is generally much better than SOP or PSD-MADS-VNS on higher dimensional problems.

Table 2 Speedup of GOPS, SOP, PSD-MADS-VNS on 14 10-dimensional BBOB test problems when $P = 16$, 32, 64, and 128 (over 30 trials)

Prob. ID	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS
	$P = 16$			$P = 32$			$P = 64$			$P = 128$		
F3	**8.5**	4.2	3.2	**30.1**	5.2	4.3	**53.6**	6.4	5.1	**80.3**	8.8	6.7
F4	**39.5**	4.4	8.8	**31.6**	5.9	13.0	**105.2**	7.2	15.5	**157.8**	12.5	20.4
F8	8.9	**10.9**	2.1	**32.5**	15.5	2.6	**53.1**	21.8	4.0	**63.0**	32.0	5.5
F9	**23.1**	7.0	2.1	**48.3**	10.5	3.2	**67.6**	13.7	3.7	**67.6**	21.1	5.1
F15	**16.0**	4.8	2.0	**27.6**	6.5	2.9	**38.0**	8.8	4.0	**40.5**	11.2	5.8
F16	**12.0**	6.9	2.4	**16.6**	8.2	6.4	**24.5**	10.5	8.9	**32.1**	14.7	12.8
F17	**60.9**	47.2	1.7	**99.4**	62.9	2.0	**125.9**	82.1	3.5	**145.2**	145.2	5.8
F18	**46.9**	28.3	4.1	**66.2**	60.0	6.0	**120.1**	76.8	6.6	**147.8**	147.8	12.4
F19	**38.6**	27.1	4.5	**75.6**	49.7	5.5	**99.5**	81.4	10.4	**171.9**	127.9	18.8
F20	**52.3**	10.6	2.2	**84.1**	16.1	3.0	**138.2**	24.1	4.0	**215.0**	27.1	6.0
F21	**102.3**	64.1	1.9	25.6	**123.6**	3.6	153.4	**192.3**	6.8	139.5	**247.3**	9.3
F22	**211.7**	146.5	1.6	**272.1**	272.1	2.8	**381.0**	317.5	4.0	**381.0**	381.0	5.4
F23	**42.4**	29.8	14.8	**70.6**	39.0	49.6	**127.1**	54.3	66.2	**211.8**	108.5	127.6
F24	**52.4**	39.4	2.9	**80.8**	48.5	6.7	**121.2**	78.8	8.0	**161.6**	126.0	11.7
Avg.	**51.1**	30.8	3.9	**68.7**	51.7	8.0	**114.9**	69.7	10.8	**143.9**	100.8	18.1

The average (avg.) speedup over the ten test problems is shown. The algorithm solvers with the best speedup are **bold**

Table 3 Speedup of GOPS, SOP, PSD-MADS-VNS on 14 21-dimensional BBOB test problems when $P = 16, 32, 64$, and 128 (over 30 trials)

Prob. ID	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS
	$P=16$			$P=32$			$P=64$			$P=128$		
F3	**6.1**	4.7	3.5	**10.8**	7.8	4.8	**17.2**	11.4	5.5	**26.9**	14.2	7.2
F4	**6.9**	6.2	6.7	**13.3**	8.2	9.8	**23.0**	12.1	11.2	**31.6**	16.9	14.7
F8	**8.2**	5.3	1.8	**13.2**	6.5	2.4	**16.8**	8.2	3.2	**18.9**	10.9	3.9
F9	8.5	**8.6**	1.9	**12.4**	8.2	2.5	**17.9**	9.6	3.4	**20.2**	10.2	4.6
F15	**7.8**	5.6	2.0	**12.6**	7.8	3.0	**19.3**	10.9	4.3	**22.9**	15.7	5.6
F16	**5.8**	4.3	4.4	**7.6**	5.5	5.1	**13.2**	7.0	7.0	**15.7**	9.6	11.4
F17	**35.8**	13.0	2.2	**59.6**	21.6	3.1	**80.5**	34.3	3.6	**100.6**	36.4	6.1
F18	**36.5**	7.1	1.6	**64.2**	12.3	2.7	**103.5**	16.9	3.0	**116.4**	16.9	6.3
F19	**37.2**	11.2	1.5	**58.1**	17.6	2.6	**80.8**	36.9	3.5	**154.9**	46.1	4.1
F20	**39.3**	7.6	2.6	**65.4**	10.1	3.5	**103.3**	13.3	4.3	**151.0**	13.3	5.9
F21	27.5	**53.0**	3.1	59.2	**65.4**	4.0	70.0	**150.9**	5.2	51.3	**163.5**	6.4
F22	**95.3**	85.8	2.8	**155.9**	122.5	3.4	**190.6**	171.5	4.3	**214.4**	171.5	5.7
F23	**27.5**	8.7	16.7	**46.6**	13.7	13.4	**75.2**	26.4	24.5	**122.2**	46.2	52.4
F24	**7.8**	5.3	0.9	**13.3**	8.2	2.3	**19.6**	16.0	2.9	**25.7**	18.9	4.1
Avg.	**25.0**	16.2	3.7	**42.3**	22.5	4.5	**59.4**	37.5	6.1	**76.6**	42.2	9.9

The average (avg.) speedup over the ten test problems is shown. The algorithm solvers with the best speedup are **bold**

Table 4 Speedup of GOPS, SOP, PSD-MADS-VNS on 14 40-dimensional BBOB test problems when $P = 16$, 32, 64, and 128 (over 30 trials)

Prob. ID	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS	GOPS	SOP	PSD-MADS-VNS
	$P = 16$			$P = 32$			$P = 64$			$P = 128$		
F3	**8.5**	7.2	2.8	**13.1**	10.9	3.4	**20.6**	12.8	4.2	**24.5**	13.6	5.6
F4	**9.5**	8.9	5.2	**16.7**	12.6	6.9	**27.0**	17.3	8.1	**35.4**	20.5	11.1
F8	**11.3**	7.1	2.6	**17.3**	8.8	3.3	**20.5**	7.8	4.2	**20.5**	8.8	5.5
F9	**13.7**	10.5	2.7	**21.9**	13.9	3.7	**30.1**	14.9	4.6	**30.1**	13.1	6.3
F15	**9.5**	7.0	2.2	**15.7**	11.0	3.3	**21.5**	15.4	4.3	**25.5**	19.3	6.1
F16	**9.8**	6.2	3.9	**14.7**	9.0	6.4	**19.3**	11.1	13.0	**31.3**	15.2	19.6
F17	**7.8**	6.0	1.5	**15.5**	7.7	3.2	**20.9**	11.1	4.0	**30.1**	12.4	6.9
F18	**13.6**	6.5	3.0	**20.7**	10.4	4.0	**34.7**	13.3	5.8	**47.8**	15.0	7.6
F19	**35.0**	31.7	2.7	**77.0**	55.8	3.7	70.0	**73.2**	5.9	**96.2**	73.2	6.3
F20	**11.8**	11.6	3.3	**19.8**	17.1	4.3	**27.6**	21.6	5.3	**31.0**	20.3	7.0
F21	**12.6**	6.4	3.4	**18.9**	8.7	4.7	**23.1**	12.5	5.6	**26.0**	14.1	7.3
F22	**10.0**	7.1	3.1	**16.4**	10.3	4.4	**18.6**	14.9	5.2	17.4	**17.9**	6.9
F23	**7.9**	3.2	2.9	**13.9**	5.6	6.1	**24.1**	19.2	15.4	**39.2**	26.5	23.5
F24	**10.5**	8.0	2.4	18.1	**19.0**	3.1	21.5	**30.5**	4.7	28.3	**28.6**	5.8
Avg.	**12.3**	9.1	3.0	**21.4**	14.3	4.3	**27.1**	19.7	6.4	**34.5**	21.3	9.0

The average (avg.) speedup over the ten test problems is shown. The algorithm solvers with the best speedup are **bold**

5.3 Relative speedup on test function suite

Speedup is an important measure of a parallel algorithm since it indicates the efficiency of the algorithm in using multiple processors synchronously. In this section, we investigate the parallel speedups of all the three algorithms on the 14 BBOB test function suit with three different dimensions (10, 21, and 40). We did not do a speedup investigation for the PDE problem because the time required for each evaluation makes this very time and resource intensive and because we can use the inexpensive test function to evaluate the algorithm's speedup. We follow the relative speedup calculation in Krityakierne et al. (2016) that the relative speedup is calculated as:

$$\alpha - \text{Speedup}(P) = n^{(\alpha)}(1) \Big/ n^{(\alpha)}(P) \tag{13}$$

where $n^{(\alpha)}(1)$ is the number of iterations that the fast serial algorithm required to reach a specified level of accuracy α. $n^{(\alpha)}(P)$ is the number of iterations that the parallel algorithm with P processors needs to reach the same level of accuracy α. Note for the serial algorithm, the number of iterations equals the number of evaluations. Equation (13) requires the knowledge of the fastest serial algorithm, which is hard to know. In the SOP paper (Krityakierne et al. 2016), the serial StochRBF (Regis and Shoemaker 2007b) is used as the serial algorithm to compute $n^{(\alpha)}(1)$. However, results showed that DYCORS is more efficient than StochRBF (Regis and Shoemaker 2013). We thus used DYCORS as the serial algorithm to compute $n^{(\alpha)}(1)$ from Eq. (13).

The relative $\alpha - \text{Speedup}(P)$ for each test function is calculated, given the accuracy level of α. Let y_1^*, y_{16}^*, y_{32}^*, y_{64}^*, y_{128}^* be the average (over 30 trials) of the best objective function values obtained from serial DYCORS, and the parallel algorithms GOPS-16P, GOPS-32P, GOPS-64P, GOPS-128P, respectively. We set $\alpha = \max\{y_1^*, y_{16}^*, y_{32}^*, y_{64}^*, y_{128}^*\}$ for algorithm GOPS. We calculate the accuracy level α for SOP and PSD-MADS-VNS in a similar approach. Tables 2, 3 and 4 show the $\alpha - \text{Speedup}(P)$ values Eq. (13) for all three algorithms when a different number of processors are used with the 14 BBOB test problems with 10, 21, and 40 dimensions, respectively.

A parallel algorithm has the favorable trait "scalability" if it has good speedups for a significant number for processors. As shown in Table 2 (in 10-dimensional case), for problems F4, F8-F9, F17-F20 and F22-F24, GOPS has outstanding scalability and is even superlinear in some cases. Superlinear speedup (i.e., $\alpha - \text{Speedup}(P)$ is larger than P) indicates that the use of P processors has a speedup that is higher than P, and so its efficiency is greater than 100%. Note that superlinear scalability could happen in cases when the serial algorithms perform much worse than parallel algorithms. Figure A1 in Online Resource gives two examples where serial algorithm DYCORS performance is relatively worse than GOPS and SOP on problem F21 (in 10-dimensional) and F22 (in 21-dimensional), which could lead to a high value of speedup. The superlinear speedup holds for algorithm GOPS on problems F17-F20, and F22–F24 for all cases when $P = 16$, 32, 64, 128, on problems F9 when $P = 16$, 32, 64, on problem F4 and F21 when $P = 16$, 64, 128, on problem

F8 when P = 32. There are also many problems for which SOP has a superlinear speedup. For example on problem F17–F18, F21-F22, and F24 for all cases when P = 16, 32, 64, 128. While in 21-dimensional cases (Table 3), GOPS has a superlinear speedup than SOP on more problems. The superlinear speedup also holds for GOPS on problem F17–F20 and F22–F23 for all cases when $P = 16$, 32, 64, 128, and on problem F21 for $P = 16$, 32, 64. By contrast, for SOP algorithm, superlinear speedup only holds on problem F21 and F22 when $P = 16$, 32, 64, 128. For PSD-MADS-VNS superlinear speedup only holds for problem F23 when P = 32, 64, 128 (in 10-dimensional case) and when $P = 16$ (in 21-dimensional case). The results suggest that GOPS has a significantly improved speedup over SOP on almost all problems, especially for problems F3, F4, F15–F16, F20, F23, F24 (in 10 dimensional), for problems F17, F18, F19, F20, and F23 (in 21 dimensional) and for problems F8, F18, F19, F21, F23 (in 40 dimensional).

GOPS has much better scalability over PSD-MADS-VNS on all problems (in 10, 21, and 40 dimensional) for all cases when $P = 16$, 32, 64, 128. SOP has better scalability over PSD-MADS-VNS on most of the problems except for F4 (on 10 and 21 dimensional cases), F16 (on 21 and 40 dimensional), and F23. For function F16 and F23, Krityakierne et al. (2016) reported SOP had rather poor scalability.

Hence, GOPS has, by far, the best average speedup over all the problems and dimensions compared to SOP and PSD-MADS-VNS. Remarkably there is not a single combination of problems and dimensions for which PSD-MADS-VNS has the best speedup.

The good performance of GOPS and SOP over PSD-MADS-VNS might be because firstly, GOPS and SOP use a surrogate to guide the search, while PSD-MADS-VNS does not use a surrogate. The use of surrogates helps guide the search to reach the regions where values of the objective function are lower, and often the

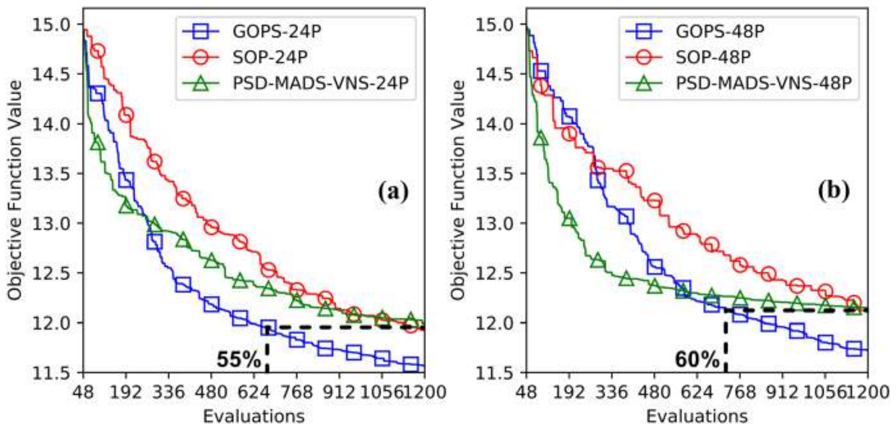

Fig. 7 Calibration progress plot of average best solution on the WQ PDE problem found so far in terms of objective function value (over ten trials) vs the number of evaluations for algorithms SOP, GOPS, PSD-MADS-VNS on WAQ when 24 processors are used (**a**) and 48 processors are used (**b**). Lower objective function value is better. Note in **a** GOPS required only 55% as many evaluations to get the final answer of PSD-MADS-VNS after 1200 evaluations. In **b** this percentage is 60%

global minimum is located more quickly; Secondly, GOPS and SOP use multiple perturbation centers, while PSD-MADS-VNS generates simulation points around only one center (which is the best solution found so far). Hence GOPS and SOP have a more diverse set of evaluation points per iteration. The diversity of evaluation points can help to escape from local minima and also helps to locate promising regions faster.

5.4 Algorithm comparison on WAQ

In this section, we compare GOPS, SOP, and PSD-MADS-VNS on the expensive PDE-constrained calibration optimization problem on lake water quality, WAQ (refer to Sect. 4.1)

Figure 7 shows that with 1200 evaluations the best solution on the WAQ PDE model calibration problem is always obtained by GOPS. However, even more important is that with 24 processors, GOPS was able to obtain in only 660 evaluations the solution that was the best solutions found by the other two algorithms (SOP and PSD-MADS-VNS) in 1200 evaluations, which means GOPS only needed about 55% (660/1200) as many evaluations as the other two algorithms to get the same result. This is equivalent to saying GOPS was 1.8 (=1/0.55) times as fast as the other two algorithms on the WAQ problem with 24 processors. With 48 processors, the GOPS algorithm used only about 60% as many evaluations as the other two algorithms, which is equivalent to a speed up of 1.7 (=1/0.6). For the optimization experiments, each algorithm was repeated for 10 trials on the WAQ problem and the analysis above is based on the average over the 10 trials as plotted in Fig. 7.

The performance improvement of GOPS over SOP provides the evidence that the introduction of the new elements $P_C^{(n)}$, N_{c_j}, and $P_{good}^{(n)}$ did improve the search on the WAQ problem. Especially in case when 48 processor are used, GOPS shows the same convergence speed as SOP in the first 300 evaluations, and later GOPS converges much faster than SOP, as shown in Fig. 7b. Note that one difference between GOPS and SOP is that GOPS dynamically reduces the number of sampling centers $P_C^{(n)}$ and increases the sampling around the best solution found so far N_{c_1} as the number of iterations increases. This difference allows GOPS to have exploration ability that is similar to SOP in the beginning iterations but stronger exploitation ability in the latter search stages. Also, since the sampling around the best solution found so far in GOPS (and also in SOP) is based on the truncated normal distribution in the solution domain, GOPS is a global optimizer that can get out of local minima. The ability of GOPS to be a global optimizer is described in the Theorem 1 in Sect. 3.4, and the proof of that theorem is given in the Online Resource.

6 Conclusion

Global Optimization in Parallel with Surrogate (GOPS) is a new algorithm for parallel optimization of computationally expensive, multi-modal continuous functions with no objective function derivatives available. Its effectiveness is demonstrated here in highly parallelized computations both on test functions and on a very complex PDE model (based on real data) that generates a multi-modal objective function. The PDE model involves a set of nonlinear partial differential equations describing the spatial distribution of concentration of many constituents and is a challenging calibration problem.

The difficulty in designing parallel global optimization algorithms is how to pick P new points to evaluate on the objective function simultaneously, where P is the number of processors. This can be very inefficient unless we can decide how to explore primarily promising areas while not allowing sampled points to be too close to one another. This problem becomes more difficult as P gets larger.

GOPS addresses this problem by (a) first some promising centers are selected from previously evaluated points and (b) new candidate points are created by adding random perturbations to each of these center points. The new features in GOPS are controlled by the new variables $P_C^{(n)}$, N_{c_j}, and $P_{good}^{(n)}$. GOPS promotes exploration in the early iteration by having many centers and hence fewer evaluation points picked from each center. However, gradually as the iterations increase, the number of centers decreases, and hence the number of points selected from each center becomes larger. In addition, previously evaluated points with sufficiently poor objective functions are not allowed to become center points. These new elements help enable the algorithm to search better even when it has to do as P expensive function evaluations in each iteration even when P is large.

The new GOPS algorithm is very efficient up to $P=128$ processors, which enables GOPS to use a large number of computing resources for solving PDE-constrained optimization problems in a relatively short wall-clock time.

The performance of GOPS was tested on 14 synthetic BBOB test problems (in 10, 21, and 40 dimensions) and one real-world PDE-constrained parameter estimation problem (WAQ) that is 21 dimensional. GOPS was compared with the SOP algorithm and the widely used MADS algorithm with its parallel global optimization option, PSD-MADS-VNS. GOPS performance was clearly the best on all test problems (especially on high dimensional problems and/or with larger number of processors) based on performance and data profile plots that evaluate all the test results. Numerical experimental results indicate GOPS dramatically outperformed the other algorithms with regard to (a) accuracy of solution for a fixed number of evaluations, (b) speedup and efficiency for up to 128 processors, (c) GOPS' ability to find the global minimum of multi-modal test functions without derivatives for objective functions with up to 40 dimensions, and (d) its ability to efficiently solve a parameter calibration problem of a 21 dimensional nonlinear PDE model describing spatial and temporal dynamics of multiple water quality constituents in a large lake utilizing real data.

GOPS performed the best on the calibration of real-world water quality PDE problem WAQ, which is multi-modal. There have been very few studies of global optimization of multi-modal, PDE models because these models are expensive, and popular global methods like particle swarm optimization or genetic algorithms take too many evaluations to be practical for expensive functions. Hence GOPS is an essential tool for this kind of environmental problem as well as for many other problems described by nonlinear PDE's that result in the occurrence of multiple local minima in the objective function. This lake water quality application illustrates the practical use of GOPS algorithm to solve real world problems.

The numerical result indicates that the use of the $P_C^{(n)}$, N_{c_j}, and $P_{good}^{(n)}$ strategies (in Sects. 3.2 and 3.3) in the GOPS algorithm improves the algorithm's exploitation ability, especially in later iterations, which in turn enables the algorithm to find more accurate solutions than SOP and PSD-MADS-VNS.

As is noted in Sect. 5.1, the values of all algorithm parameters in GOPS are given, and all tests were performed with this same set of parameters. So the expectation is that the GOPS algorithm will be used with these algorithm parameters, and parameter tuning is not required.

GOPS is a general-purpose global optimization method that is not limited to PDE-constrained global optimization only. For problems with an objective function that is computationally expensive, multi-modal, with no available derivatives, GOPS is a very promising option compared with other parallel global optimization methods, e.g., SOP, PSD-MADS-VNS or non-surrogate metaheuristics.

Acknowledgements This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme and Prof. Shoemaker's grant from the National University of Singapore (NUS). Water quality numerical model and Data models were provided by PUB, Singapore's National Water Agency. The computational work for this article was entirely performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). The GOPS software was built in the pySOT toolbox (see (Eriksson et al. 2019) in references). (The GOPS software will be publicly (open source) available soon after this paper is published. The GOPS code is also available from the first author upon request (xiawei@u.nus.edu)).

References

Audet C, Hare W (2017) Derivative-free and blackbox optimization. Springer, Berlin

Audet C, Béchard V, Le Digabel S (2008a) Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J Glob Optim 41:299–318. https://doi.org/10.1007/s10898-007-9234-1

Audet C, Dennis JE Jr, Digabel SL (2008b) Parallel space decomposition of the mesh adaptive direct search algorithm. SIAM J Optim 19:1150–1170. https://doi.org/10.1137/070707518

Audet C, Diest K, Le Digabel S, Sweatlock LA, Marthaler DE (2013) Metamaterial design by mesh adaptive direct search. Numerical methods for metamaterial design, vol 127. Springer, Berlin, pp 71–96

Bischl B, Wessing S, Bauer N, Friedrichs K, Weihs C (2014) MOI-MBO: multiobjective infill for parallel model-based optimization. In: International conference on learning and intelligent optimization, Gainesville, FL, USA, 2014. Springer, Cham, pp 173–186. https://doi.org/10.1007/978-3-319-09584-4

Björkman M, Holmström K (2000) Global optimization of costly nonconvex functions using radial basis functions. Optim Eng 1:373–397. https://doi.org/10.1023/A:1011584207202

Bons NP, He X, Mader CA, Martins JR (2019) Multimodality in aerodynamic wing design optimization. AIAA J 57:1004–1018. https://doi.org/10.2514/1.J057294

Butts M, Loinaz M, Bauer-Gottwein P, Unnasch R, Gross D (2012) MIKE SHE-ECOLAB: an integrated catchment-scale eco-hydrological modelling tool. In: 19th international conference on computational methods in water resources, University of Illinois at Urbana, Champaign, 2012

Christelis V, Regis RG, Mantoglou A (2018) Surrogate-based pumping optimization of coastal aquifers under limited computational budgets. J Hydroinform 20:164–176. https://doi.org/10.2166/hydro.2017.063

Culver TB, Shoemaker CA (1992) Dynamic optimal control for groundwater remediation with flexible management periods. Water Resour Res 28:629–641

Díaz-Manríquez A, Toscano-Pulido G, Gómez-Flores W (2011) On the selection of surrogate models in evolutionary optimization algorithms. In: 2011 IEEE congress of evolutionary computation (CEC), New Orleans, LA, USA, 2011. IEEE, pp 2155–2162. https://doi.org/10.1109/CEC.2011.5949881

Eriksson D, Bindel D, Shoemaker CA (2019) pySOT and POAP: an event-driven asynchronous framework for surrogate optimization. arXiv preprint arXiv:00420

Forrester A, Sobester A, Keane A (2008) Engineering design via surrogate modelling: a practical guide. Wiley, Chichester

Gibson R, Atkinson R, Gordon J (2006) Review of three-dimensional ecological modelling related to the North Sea shelf system. Part II: model validation and data needs. In: Hawkins SJ, Allcock AL, Bates AE, Firth LB, Smith IP, Swearer SE, Todd PA (eds) Oceanography marine biology: an annual review, vol 44. CRC Press, Boca Raton

Gorelick SM, Zheng C (2015) Global change and the groundwater management challenge. Water Resour Res 51:3031–3051. https://doi.org/10.1002/2014WR016825

Gorelick SM, Freeze RA, Donohue D, Keely JF (1993) Groundwater contamination: optimal capture and containment. Lewis Publishers Inc., Chelsea

Gutmann H-M (2001) A radial basis function method for global optimization. J Glob Optim 19:201–227

Haftka RT, Villanueva D, Chaudhuri A (2016) Parallel surrogate-assisted global optimization with expensive functions: a survey. Struct Multidiscip Optim 54:3–13

Hansen N, Finck S, Ros R, Auger A (2009) Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. RR-6829, INRIA. https://hal.inria.fr/inria-00362633v2. Accessed 10 Sept 2020

Hensman J, Fusi N, Lawrence ND (2013) Gaussian processes for big data. arXiv preprint arXiv:13096835

Hinkelmann R (2006) Efficient numerical methods and information-processing techniques for modeling hydro-and environmental systems, vol 21. Springer, Berlin

Hodges B, Dallimore C (2006) Estuary, lake and coastal ocean model: ELCOM v2. 2 science manual. Centre for Water Research, University of Western Australia

Hydraulics D (2003) Delft3D-WAQ: technical reference manual. WL| Delft Hydraulics, Delft

Hydraulics D (2005) Delft3D-WAQ users manual. WL Delft Hydraulics, Delft

Isaacs A (2009) Development of optimization methods to solve computationally expensive problems. University of New South Wales, Australian Defence Force Academy, School of Engineering and Information Technology

Jakobsson S, Patriksson M, Rudholm J, Wojciechowski A (2010) A method for simulation based optimization using radial basis functions. Optim Eng 11:501–532. https://doi.org/10.1007/s11081-009-9087-1

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13:455–492

Krityakierne T, Akhtar T, Shoemaker CA (2016) SOP: parallel surrogate global optimization with Pareto center selection for computationally expensive single objective problems. J Glob Optim 66:417–437

Le Digabel S (2011) Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. ACM Trans Math Softw 37:44

Le Digabel S, Abramson MA, Audet C, Dennis Jr J (2010) Parallel versions of the MADS algorithm for black-box optimization. In: Optimization days, Montreal, 2010

Matta E, Selge F, Gunkel G, Rossiter K, Jourieh A, Hinkelmann R (2016) Simulations of nutrient emissions from a net cage aquaculture system in a Brazilian bay. Water Sci Technol 73:2430–2435

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100

Moré JJ, Wild SM (2009) Benchmarking derivative-free optimization algorithms. SIAM J Optim 20:172–191

Mugunthan P, Shoemaker CA, Regis RG (2005) Comparison of function approximation, heuristic, and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. Water Resour Res 41:W11427

Müller J (2016) MISO: mixed-integer surrogate optimization framework. Optim Eng Optim 17:177–203

Müller J, Shoemaker CA (2014) Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. J Glob Optim 60:123–144

Müller J, Paudel R, Shoemaker C, Woodbury J, Wang Y, Mahowald N (2015) CH 4 parameter estimation in CLM4. 5bgc using surrogate global optimization. Geosci Model Dev 8:3285–3310

Pinder GF, Celia MA (2006) Subsurface hydrology. Wiley, Hoboken

Pinder GF, Gray WG (1977) Finite element simulation in surface and subsurface hydrology. Academic, New York

Powell M (1992) The theory of radial basis function approximation in 1990. In: Light WA (ed) Advances in numerical analysis II: wavelets, subdivision, and radial functions, vol 105. Oxford University Press, Oxford

Regis RG (2011) Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions Computers. Oper Res 38:837–853

Regis RG (2013) An initialization strategy for high-dimensional surrogate-based expensive black-box optimization. In: Takáč M, Terlaky T (eds) Modeling and optimization: theory and applications. Springer, Berlin, pp 51–85

Regis RG (2014) Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Eng Optim 46:218–243

Regis RG, Shoemaker CA (2004) Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE Trans Evol Comput 8:490–505

Regis RG, Shoemaker CA (2005) Constrained global optimization of expensive black box functions using radial basis functions. J Glob Optim 31:153–171

Regis RG, Shoemaker CA (2007a) Parallel radial basis function methods for the global optimization of expensive functions. Eur J Oper Res 182:514–535

Regis RG, Shoemaker CA (2007b) A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J Comput 19:497–509

Regis RG, Shoemaker CA (2009) Parallel stochastic global optimization using radial basis functions. INFORMS J Comput 21:411–426

Regis RG, Shoemaker CA (2013) Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. Eng Optim 45:529–555

Rios LM, Sahinidis NV (2013) Derivative-free optimization: a review of algorithms and comparison of software implementations. J Glob Optim 56:1247–1293

Shan S, Wang GG (2010) Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct Multidiscip Optim 41:219–241

Shoemaker CA, Regis RG, Fleming RC (2007) Watershed calibration using multistart local optimization and evolutionary optimization with radial basis function approximation. Hydrol Sci J 52:450–465

Smits JG, van Beek JK (2013) ECO: a generic eutrophication model including comprehensive sediment-water interaction. PLoS ONE 8:e68104

Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J (1996) MPI: the complete reference—the MPI core. MIT Press, Cambridge

Sóbester A, Forrester AI (2014) Aircraft aerodynamic design: geometry and optimization. Wiley, West Sussex

Sóbester A, Leary SJ, Keane AJ (2004) A parallel updating scheme for approximating and optimizing high fidelity computer simulations. Struct Multidiscip Optim 27:371–383

Sóbester A, Forrester AI, Toal DJ, Tresidder E, Tucker S (2014) Engineering design applications of surrogate-assisted optimization techniques. Optim Eng 15:243–265

Torczon V (1997) On the convergence of pattern search algorithms. SIAM J Optim 7:1–25

Wool TA, Ambrose RB, Martin JL, Comer EA, Tech T (2006) Water quality analysis simulation program (WASP), vol 6. User's Manual, Version 6

Yeh WW (2015) Optimization methods for groundwater modeling and management. Hydrogeol J 23:1051–1065

Affiliations

Wei Xia[1] · **Christine Shoemaker**[1,2]

✉ Christine Shoemaker
shoemaker@nus.edu.sg

[1] Department of Civil and Environmental Engineering, National University of Singapore, Singapore 117576, Singapore

[2] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576, Singapore