2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

SO-MODS: Optimization for High Dimensional Computationally Expensive Multi-Modal Functions with Surrogate Search

Juliane Müller
School of Civil & Env. Engineering
Cornell University, Ithaca, NY 14853
Email: juliane.mueller2901@gmail.com

Tipaluck Krityakierne
Center of Applied Mathematics
Cornell University, Ithaca, NY 14853
Email: tk338@cornell.edu

Christine A. Shoemaker
School of Civil & Env. Engineering
Cornell University, Ithaca, NY 14853
Email: cas12@cornell.edu

***Abstract*—SO-MODS is a new algorithm that combines surrogate global optimization methods with local search. SO-MODS is an extension of prior algorithms that sought to find near optimal solutions for computationally very expensive functions for which the number of allowable evaluations is strictly limited. The global search method in SO-MODS perturbs the best point found so far in order to find a new sample point. The number of decision variables being perturbed is dynamically adjusted in each iteration in order to be more effective for higher dimensional problems. The procedure for dynamically changing the dimensions perturbed is drawn from earlier work on the DYCORS algorithm. We use a cubic radial basis function as surrogate model and investigate two approaches to improve the solution accuracy. The numerical results show that SO-MODS is able to reduce the objective function value dramatically with just a few hundred evaluations even for 30-dimensional problems. The local search is then able to reduce the objective function value further.**

I. Introduction

Evolutionary algorithms mimic the natural behavior of survival of the fittest. These algorithms are based on the use of a "population" that serves to carry information about objective function values of multiple decision vectors that have been evaluated in a previous iteration into the search for new solutions in the current iteration. This is a major strength of evolutionary algorithms that is absent from derivative-based search (e.g. SQP) that chooses the next evaluation point based primarily on the values of the current evaluation point and all its derivatives.

In this paper we introduce the algorithm SO-MODS (Surrogate-assisted Optimization with Memetic Optimization Dimensionally-controlled Search), which has been developed for global optimization of high-dimensional, computationally expensive black-box functions when a high accuracy of the solution is desired. SO-MODS differs from most evolutionary algorithms in that the population in iteration n includes all values of the decision vector that have had their objective function evaluated in any previous iteration. Hence, the size of the population is growing with each iteration.

For computationally expensive black-box objective functions, for which due to the black-box nature derivatives are not available, the goal is to obtain good solutions with relatively few objective function evaluations in order to reduce the total computation time. SO-MODS achieves this goal by forming a multivariate approximation (a response surface) of the expensive function $f(x)$. This response surface $s(x)$ is used as a surrogate for the expensive function $f(x)$ during the search, and therefore the number of required expensive function evaluations for finding accurate solutions can be significantly reduced. The novelty in SO-MODS is a memetic approach to improve the accuracy of the solution in which we use a local optimization search around a subset of the previously evaluated (decision vector) points.

SO-MODS has three main phases:

1) high dimensional global optimization search with an extension of DYCORS [1] that employs additional logic for local optimization searches on the response surface.
2) ORBIT [2], which is a derivative-free local optimizer, and GORBIT which is multi-start ORBIT [3].
3) gradient-based local optimization with fmincon.

Phases 1 and 2 use a radial basis function surrogate for $f(x)$ to avoid the need to do expensive finite difference calculations. Phase 3 is implemented to improve the precision of the final solution and uses numerical differentiation to approximate the derivatives of the true objective function. Phase 3 (fmincon) requires in each iteration at least d (problem dimension) function evaluations to numerically compute the derivatives. At least one additional function evaluation is then needed to identify the point for starting the next iteration. Hence, at least $d+1$ expensive evaluations are required in each fmincon iteration. Hence, the goal is to get close to the global minimum during Phases 1 and 2 of SO-MODS such that only very few iterations with fmincon will be required.

The purpose of this study is to develop and demonstrate a global optimization algorithm for relatively high dimensional problems that improves the accuracy of the solution while maintaining high efficiency for computationally expensive functions for which there is a strict limitation on the number of objective function evaluations that can be allowed.

A. Problem Formulation

We consider the global optimization problem in the form:

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

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

where the objective function $f : \mathcal{D} \subset \mathbb{R}^d \to \mathbb{R}$ is a deterministic black-box function, and $\mathcal{D} = \prod_{i=1}^{d}[a_i, b_i]$ a box-constrained set with $-\infty < a_i \leq x_i \leq b_i < \infty$ for $1 \leq i \leq d$. We are particularly interested in problems that have multiple local minima, a relatively high dimension d, and for which the gradient of $f(x)$ is assumed to be unavailable. Furthermore, we assume that there exists a global minimum $x^* \in \mathcal{D}$ for which $f(x) \geq f(x^*)$ for all $x \in \mathcal{D}$ and that f is continuous at the point x^*.

B. Related Work

Since simulations in engineering applications can be computationally extremely expensive and unacceptably time consuming, several surrogate based methods have been proposed to solve such global optimization problems. The first use of a non-quadratic surrogate response surface for traditional (e.g. fixed population sized) evolutionary algorithms was by Regis and Shoemaker [4]. The first surrogate global optimization method that keeps all available points was Jones et al.'s EGO method [5] which is based on a kriging surface and involves the maximization of an expected improvement function. Radial basis functions were used in Regis and Shoemaker's LMSRBF method [6], [7] which uses a weighted score computed based on the objective function value prediction by the response surface and the distance to previously evaluated points to iteratively select sample points. This sampling method has also been employed as a framework in DYCORS [1]. The interested reader is referred to [8]–[11] for a review and more information as well as applications of existing response surface based methods for global optimization. A recent survey of surrogate-assisted evolutionary optimization techniques can also be found in [12]. Other response surfaces such as polynomial regression models [13], MARS (regression splines) [14], as well as mixture surrogate models [15], [16] have been successfully applied to solve global optimization problems arising in engineering.

II. SO-MODS Framework

A. Stochastic Response Surface

Regis and Shoemaker [7] introduced a class of Stochastic Response Surface (SRS) algorithms which is a framework for expensive global optimization. Examples of methods based on SRS are LMSRBF [6], [7], DYCORS [1], SO-I [17], and SO-MI [18]. The SRS framework is given in Algorithm 1.

Algorithm 1 SRS

1: Build an initial response surface.
2: **while** Termination condition not met **do**
3: Randomly generate candidate points.
4: Select a candidate point for the expensive function evaluation.
5: Update the best point found so far, x_{best}.
6: Update the response surface.
7: **end while**
8: **return** Approximate global minimum.

In this paper, we adopt the SRS framework in the global search phase (Phase 1) of SO-MODS.

B. SO-MODS

SO-MODS involves a memetic search, which is a combination of global and local search methods. Phase 1 utilizes an extension of the DYCORS algorithm [1] because it has been shown to be able to efficiently find good solutions for high-dimensional, computationally expensive multi-modal black-box problems when the number of function evaluations is strictly limited (e.g., 1000 function evaluations for 30 dimensional problems). The extension of DYCORS developed here uses additional local optimization searches on the inexpensive response surface. Both DYCORS and Phase 1 of SO-MODS differ from earlier algorithms by the authors (e.g., [6], [7]) in that the number of perturbed dimensions is random and the expected number of perturbations decreases with iteration. This dynamical dimensioned search feature has been shown to be efficient for higher dimensional problems [19]. The individual steps of SO-MODS are shown in Algorithm 2.

Inputs:

I-1 A real-valued function f defined on a hyper-rectangle $\mathcal{D} = \prod_{i=1}^{d}[a_i, b_i] \subset \mathbb{R}^d$.
I-2 A response surface model.
I-3 A set of initial evaluation points $\mathcal{S}_0 = \{x_1, ..., x_{n_0}\}$ determined by a randomly generated Latin hypercube design.
I-4 The number of candidate points randomly generated in each iteration, denoted by N_{cand}.
I-5 The maximum number of allowed function evaluations, denoted by $N_{\max}$.
I-6 A function $\varphi(n)$ defined for all positive integers $n_0 \leq n \leq N_{\max} - 1$ whose values are in $(0, 1]$.
I-7 The initial step size σ_{init} and the minimum step size σ_{min}.
I-8 The tolerance for the number of consecutive failed iterations τ_{fail} and the threshold for the number of consecutive successful iterations τ_{success}.
I-9 The maximum number of times the step size can be reduced ($T_{\max}$) before starting the local search.

Algorithm 2 SO-MODS

1: **Initialization.** Set $\sigma_0 \leftarrow \sigma_{\text{int}}$, $C_{\text{fail}} \leftarrow 0$, $C_{\text{success}} \leftarrow 0$, $T_{\text{shrink}} \leftarrow 0$.
2: **Initial point evaluation and initial surrogate.** Set $n = n_0$. Denote the best point found so far by x_{best}, i.e., $x_{\text{best}} = \text{argmin}_{x \in S_0} f(x)$, and $f_{\text{best}} = f(x_{\text{best}})$. Build the initial response surface $s_0(x)$ based on the initial n_0 evaluated points.
3: **while** $n < N_{\max}$ **do**
4: **Global search phase**
5: **Local search phase**
6: **end while**
7: **return** Best solution found: x_{best} and f_{best}.

We introduce two versions of SO-MODS that use the same global and local search strategy in Phase 1 and Phase 3, respectively, but that use different search approaches in Phase 2.

1) Global Search Phase: The global search phase is summarized in Algorithm 3 and works as DYCORS [1]. Both SO-

MODS versions use the same global stopping criterion (GSC), which is either the maximum number of allowed function evaluations (in which case Algorithm 2 stops after Step 4), or $T_{\text{shrink}} = T_{\max}$ (see Algorithm 3, Step 9), whichever criterion is satisfied first.

Algorithm 3 Global Search [1]

1: **while** Global stopping criterion not met (GSC = false) **do**
2: Compute the probability of perturbing a coordinate: $p_{\text{select}} = \varphi(n)$.
3: Generate N_{cand} candidate points around x_{best} : $\Omega_n = \{y_{n,1}, ..., y_{n,N_{\text{cand}}}\}$ by perturbing each variable with probability p_{select}. The perturbation magnitude is an $N(0, \sigma_n)$ random variable.
4: Select the next point for evaluation (x_{n+1}) from Ω_n based on the information from the surrogate $s_n(x)$ and the distance to previously evaluated points through the function Select_Evaluation_Point(Ω_n, $s_n(x)$, $\mathcal{S}_n$).
5: Do the expensive function evaluation $f(x_{n+1})$.
6: Update the set of evaluated points: $\mathcal{S}_{n+1} = \mathcal{S}_n \cup \{x_{n+1}\}$.
7: Update x_{best} and $f(x_{\text{best}})$. Update the response surface $s_n \leftarrow s_{n+1}(x)$.
8: Update counters. If $f(x_{n+1}) < f_{\text{best}}$, reset $C_{\text{success}} = C_{\text{success}}+1$ and $C_{\text{fail}} = 0$; otherwise reset $C_{\text{fail}} = C_{\text{fail}}+1$ and $C_{\text{success}} = 0$.
9: Adjust step size σ_{n+1}: $[\sigma_{n+1}, C_{\text{success}}, C_{\text{fail}}] =$ Adjust_Step_Size(σ_n, C_{success}, τ_{success}, C_{fail}, τ_{fail}). If $\sigma_{n+1} < \sigma_n, T_{\text{shrink}} = T_{\text{shrink}} + 1$.
10: Reset $n = n + 1$
11: **end while**

The global search phase described in Algorithm 3 shows that after building an initial response surface (Algorithm 2, Step 2), we create in Step 3 of Algorithm 3 candidate (decision vector) points by adding a random multivariate perturbation to the current best point. Generating this random perturbation involves computing first (Algorithm 3, Step 2) $p_{\text{select}} = \varphi(n)$, which is the probability that any one dimension is perturbed. In case no variable of x_{best} is selected for perturbation, we select one variable at random. N_{cand} candidate points are generated by perturbing the selected dimensions by adding an $N(0, \sigma_n)$ random variable. Then in Step 4 of Algorithm 3, one of the candidate points is selected for evaluation. This is the point with a minimum score which is computed as the weighted sum of the surrogate surface value $s(x_{\text{cand}})$ and a metric that is based on the distance between x_{cand} and the set of previously evaluated points $\mathcal{S}_n$. Hence, the algorithm is selecting a point that has both a low response surface value and is not too close to previously evaluated points. The weights for both criteria cycle through the pattern Υ which allows a repeated transition from local to global search. After updating x_{best} in Step 7, the surrogate surface $s(x)$ is updated. The variance σ_n (for $N(0, \sigma_n)$) is adjusted in Step 9 of Algorithm 3 to speed up the convergence. The term T_{shrink} in Step 9 is a measure of the iterations done without improvement and when T_{shrink} reaches the value $T_{\max}$, the global search stops and Phase 2 (search for improvements on the response surface) starts.

Note that for generating the candidate points, adjusting $\varphi(n)$, Select_Evaluation_Point, and Adjust_Step_Size (Algorithm 3, Steps 3, 2, 4, and 9, respectively), we follow the method suggested by Regis and Shoemaker [1]. For space considerations, we will not repeat the description here and refer the reader to this paper for more details.

For Phase 2 of SO-MODS, we examined two approaches, namely

1) SO-MODS version A: use GORBIT and explore the various local minima of the response surface
2) SO-MODS version B: use ORBIT

which will be described in the following. Also, we note here that SO-MODS version A does not use any scaling of the objective function values of the evaluated points for fitting the response surface, whereas in version B function values larger than the median of all function values computed so far are replaced by the median value.

2) SO-MODS version A: Algorithm 4 describes SO-MODS version A (denoted by SO-MODS-A in the following). SO-MODS-A starts as soon as the stopping criterion of Algorithm 3 has been satisfied (if the budget of function evaluations has not been exhausted).

Algorithm 4 SO-MODS-A

1: Use GORBIT with first ORBIT run starting from x_{best}.
2: **if** Function evaluations left **then**
3: **repeat**
4: Update the response surface using all the function evaluation data obtained so far.
5: Apply MLSL to find the minimum points of the response surface (denote the set of points by $\mathcal{M}$).
6: Delete points in $\mathcal{M}$ that are too close to already evaluated points.
7: **if** $\mathcal{M} \neq \emptyset$ **then**
8: Do the expensive evaluations at the points in $\mathcal{M}$.
9: Update x_{best} and f_{best}.
10: **end if**
11: **until** $\mathcal{M} = \emptyset$
12: Local search on the true objective function starting from x_{best}.
13: **end if**

After SO-MODS' Phase 1 has finished (Algorithm 2, Step 4), Phase 2 starts. We use GORBIT (described in [20]), a multi-start version of the ORBIT algorithm [21] where we start the first ORBIT run from the best point found so far (x_{best}). The purpose of using GORBIT is to improve the accuracy of the solution in a computationally cheap way without the need of numerical differentiation. Hence, we can improve x_{best} in a computationally efficient way. We allow GORBIT to do at most $\min\{N_{\max}/4, N_{\max} - n\}$ expensive function evaluations. After GORBIT has finished, we update the response surface with all the points that have been evaluated so far. We then identify the local and global minima of the response surface. Since the response surface is in general multi-modal, we use MLSL (multi level single-linkage [22]) together with MATLAB's local minimizer fmincon to identify the various local minima of the response surface. Note that for finding the minima of the response surface the expensive objective function is not evaluated and this step is therefore computationally cheap. We denote the set of local minimum points by $\mathcal{M}$. We compute the distance of the points in $\mathcal{M}$ to

the set of already evaluated points and discard points in $\mathcal{M}$ that are too close to already evaluated points. If there are points left in $\mathcal{M}$, we do the computationally expensive evaluations at these points, update the response surface with the new data, and update x_{best} and $f_{\text{best}} = f(x_{\text{best}})$. We iterate (find the local minima of the response surface, compute distances to already evaluated points, do the expensive evaluations, update the response surface) until $\mathcal{M} = \emptyset$.

Lastly, in Phase 3, we start a local search with MATLAB's local optimization algorithm fmincon on the true objective function starting from x_{best} in an attempt to further improve the solution. We must note here that fmincon requires in each iteration at least d function evaluations to numerically compute the derivatives (we already know the function value at the current point) and then it needs one or more additional function evaluations to find the point for starting the next iteration. Hence, for a 30-dimensional problem, at least $d + 1 = 31$ expensive evaluations are required in each fmincon iteration, and therefore the budget of remaining function evaluations is quickly used up. As a result, we must use a global variable to count expensive function evaluations rather than counting the algorithm's iterations. The performance of this last local optimization heavily depends on x_{best}. Local optimization algorithms converge to local minima and, if x_{best} is not in the vicinity of the global minimum, fmincon will stop at a local minimum. The improvement achieved by fmincon depends on where in the vicinity of a (local) minimum x_{best} is located. If x_{best} is on a very steep slope of the objective function landscape, then fmincon can find improvements within relatively few evaluations. However, if x_{best} is in a very shallow valley, fmincon's progress will be slower (in terms of function evaluations needed for finding improvements). If there are function evaluations left after fmincon stops, the algorithm goes back to the global search phase (see Algorithm 2, Step 5).

3) SO-MODS version B: Our second approach for Phases 2 and 3 of SO-MODS is described in Algorithm 5 and will be denoted in the following by SO-MODS-B. Phase 1 of SO-MODS-B is the same as Phase 1 of SO-MODS-A.

Algorithm 5 SO-MODS-B

1: Use a local optimizer starting from x_{best} to find the minimum of the response surface. Denote the minimum point found by x_s.
2: Use the ORBIT algorithm starting from x_s to further improve the solution. The best point found by ORBIT is denoted by x_o. The maximum number of allowed evaluations for ORBIT is 300.
3: Use MATLAB's fmincon starting from x_o on the true objective function to further improve the solution.

Algorithm 5 uses in Step 1 MATLAB's local optimizer fmincon to find the minimum of the response surface starting from x_{best}, which is the best point found during the global search phase. In Step 1 of Algorithm 5, the computationally expensive objective function is not evaluated. Also, the minimum point of the response surface found with fmincon is not guaranteed to be the global minimum of the response surface. The goal of this step is to find the best starting point for the ORBIT algorithm [21] in Step 2 of Algorithm 5. The number of expensive evaluations ORBIT is allowed to do is fixed to 300. Numerical experiments showed that this budget delivers the best results and ORBIT terminates in many cases before the budget is used up. After ORBIT has stopped, we apply fmincon on the true objective function to further improve the accuracy of the solution. We want to stress at this point again that fmincon does in each iteration at least $d+1$ evaluations, and hence the remaining budget of function evaluations decreases fast. However, should there be function evaluations left after fmincon stops, the algorithm goes back to the global search phase (Algorithm 2, Step 4).

C. Radial Basis Function Interpolation

The surrogate model used in our algorithm is a radial basis function (RBF) interpolant. We chose the RBF model based on the results reported in [16]. Denote in the following $x_\iota \in \mathbb{R}^d$, $\iota = 1, \ldots, n$, the already evaluated points. The RBF model can then be represented as

$$s(x) = \sum_{\iota=1}^{n} \lambda_\iota \phi(\|x - x_\iota\|) + p(x), \tag{2}$$

where $s(x)$ is the surrogate model prediction at the point x, $\phi : \mathbb{R}^d \mapsto \mathbb{R}$ denotes the radial basis function (we use the cubic RBF $\phi(r) = r^3$), and $p(x) = \beta^T x + \alpha$ is the associated polynomial tail. Here, $\beta = [\beta_1, \ldots, \beta_d]^T \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$. The parameters λ_ι, $\iota = 1, \ldots, n$, β_j, $j = 1, \ldots, d$, and α are determined by solving a linear system of equations:

$$\begin{bmatrix} \Phi & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ c \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{3}$$

where $\Phi_{\iota\nu} = \phi(\|x_\iota - x_\nu\|)$, $\iota, \nu = 1, \ldots, n$, 0 is a matrix with all entries 0 of appropriate dimension, and

$$P = \begin{bmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} \quad c = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \\ \alpha \end{bmatrix}, \quad F = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}. \tag{4}$$

The matrix in (3) is invertible if and only if $\text{rank}(P) = d + 1$ [23].

III. Numerical Experiments

The CEC'14 suite of expensive optimization test problems consists of 8 benchmark problems, each with dimensions 10, 20, and 30. Hence, there are totally 24 problems which we use to test SO-MODS. We follow closely the instruction given in [24]. Hence, although we do know the analytic description of the test problems, we treat them as black boxes. The test problems are computationally cheap to evaluate which facilitates efficient testing of SO-MODS. The test problems have characteristics such as unimodal, multi-modal, step function, and very long and narrow valleys where the global minimum is located. The maximum number of function evaluations is $50d$. We do 20 trials for each problem with each algorithm version.

A. Algorithm Parameters

Parameters in the global search phase of SO-MODS were set following [1], except for the parameters for generating the initial experimental design. The parameter values used in this paper are provided in Table I, where $l(\mathcal{D})$ represents the length of the shortest side of the hyper-rectangle $\mathcal{D}$. For SO-MODS-A, we used totally $n_0 = 2(d+1)$ points for the initial experimental design, where $d+1$ points were created by a symmetric Latin hypercube design and the remaining $d+1$ points were created with a random Latin hypercube design using MATLAB's built-in function lhsdesign. For SO-MODS-B we created $n_0 = 10d$ points for the initial experimental design with a symmetric Latin hypercube design strategy.

TABLE I. SO-MODS PARAMETER VALUES FOR GLOBAL SEARCH

Parameter	Value
$N_{\text{cand}} = \lvert\Omega_n\rvert$ (number of candidates points)	$\min(500d, 5000)$
Υ (weight pattern)	$< 0.3, 0.5, 0.8, 0.95 >$
σ_{int} (initial step size)	$0.2l(\mathcal{D})$
$\sigma_{\min}$ (minimum step size)	$0.2\frac{1}{2^6}l(\mathcal{D})$
τ_{success}	3
τ_{fail}	$\max(d, 5)$

B. Experimental Results

The numerical results for SO-MODS versions A and B for the test problems are summarized in Tables II, III, and IV. The better result of both algorithm versions is marked in bold. The global minimum of all functions is zero. Hence, the reported numbers show the statistics for the absolute errors between the final solution found by the SO-MODS versions and the true minimum. Results with a function value $\leq 10^{-8}$ are considered zero, and thus we do not report decimals for these problems. Problems with function values larger than 10^{-8} are reported with four decimal places. The discussion of the results for the test problems is in the following grouped into unimodal and multi-modal problems.

TABLE II. SO-MODS RESULTS, VERSIONS A AND B FOR 10-DIMENSIONAL PROBLEMS.[1]

Alg.	Problem	Best	Worst	Median	Mean	Std.
A	F1	0	0.0000	**0**	0.0000	0
	F4	0	0.0018	0.0002	0.0003	0.0004
	F7	0	0.0057	0.0007	**0.0012**	0.0015
	F10	0	0	**0**	**0**	0
	F13	0.0001	0.0018	0.0005	**0.0006**	0.0005
	F16	0.0001	0.1600	0.0101	0.0259	0.0362
	F19	0.2284	101.4595	8.3836	13.3185	21.2667
	F22	6.9648	41.7884	**19.4017**	**22.2374**	10.1582
B	F1	0	0	**0**	**0**	0
	F4	0	0.0000	**0.0000**	**0.0000**	0.0000
	F7	0.0000	0.0128	**0.0005**	0.0021	0.0037
	F10	0	0	**0**	**0**	0
	F13	0.0000	1.1551	**0.0001**	0.1156	0.3555
	F16	0.0000	0.0885	**0.0099**	**0.0154**	0.0212
	F19	5.6356	72.2342	**7.1151**	**10.4275**	14.5768
	F22	12.9345	58.7023	25.3714	27.0131	12.6030

[1] Bold numbers denote the better result of both algorithms for the respective test problem.

1) Unimodal Problems: Test problems F1-F12 are unimodal, i.e., there is only one minimum, which is the global minimum. Test problems F10-F12 are three instances of the step function, which is discontinuous. In general, if we know that our objective function is unimodal, we should use a local optimization algorithm because convergence to the minimum will be faster. If the objective function is, however, a black box as in this investigation, we do not know *a priori* how many local and global optima there are, and thus a global optimization algorithm must be used to avoid getting trapped in a local optimum. In that case, many function evaluations may be done without improving the solution because the global search selects samples that are relatively far away from the best point found so far in unexplored regions of the variable domain (exploration) rather than focusing on the best point and locally searching in its vicinity (exploitation).

TABLE III. SO-MODS RESULTS, VERSIONS A AND B FOR 20-DIMENSIONAL PROBLEMS.[2]

Alg.	Problem	Best	Worst	Median	Mean	Std.
A	F2	0	0.0000	0.0000	0.0000	0
	F5	0.0001	0.0038	0.0013	**0.0015**	0.0011
	F8	0	0.0161	0.0022	0.0030	0.0038
	F11	0	0	**0**	**0**	0
	F14	0.0004	0.0036	0.0015	0.0015	0.0008
	F17	0.0001	0.0833	0.0013	0.0106	0.0190
	F20	15.8335	60.6626	**17.5148**	19.7182	9.6773
	F23	27.8589	70.6418	**42.8138**	**43.3055**	10.8154
B	2	0	0	**0**	**0**	0
	F5	0.0001	0.1611	**0.0008**	0.0102	0.0357
	F8	0.0000	0.0227	**0.0005**	**0.0023**	0.0053
	F11	0	0	**0**	**0**	0
	F14	0.0000	0.0007	**0.0001**	**0.0001**	0.0001
	F17	0.0000	0.0472	**0.0001**	**0.0038**	0.0107
	F20	16.0645	19.3468	17.8302	**17.9037**	0.9123
	F23	30.8437	102.4802	56.7125	59.7471	20.7312

[2] Bold numbers denote the better result of both algorithms for the respective test problem.

TABLE IV. SO-MODS RESULTS, VERSIONS A AND B FOR 30-DIMENSIONAL PROBLEMS.[3]

Alg.	Problem	Best	Worst	Median	Mean	Std.
A	F3	0	0.0000	0.0000	**0.0000**	0
	F6	0.0005	0.0160	**0.0025**	**0.0039**	0.0038
	F9	0.0304	2.0614	**0.1537**	**0.3564**	0.5050
	F12	0	0	**0**	**0**	0
	F15	0.0013	0.0080	0.0020	0.0024	0.0015
	F18	0.0002	0.1133	0.0093	0.0144	0.0252
	F21	26.2578	140.4813	71.9988	61.9795	32.2668
	F24	54.7270	108.4505	**84.5724**	**83.9804**	15.7449
B	F3	0	0.0018	**0**	0.0001	0.0004
	F6	0.0015	3.0618	0.0368	0.3278	0.8651
	F9	0.9225	130.7708	9.6791	20.2410	30.7410
	F12	0	0	**0**	**0**	0
	F15	0.0000	0.0026	**0.0000**	**0.0002**	0.0006
	F18	0.0001	0.0756	**0.0024**	**0.0110**	0.0193
	F21	25.9159	122.7319	**28.4763**	**36.5977**	25.6698
	F24	56.7139	173.6795	94.5187	109.5081	38.8189

[3] Bold numbers denote the better result of both algorithms for the respective test problem.

For problems F1-F8, SO-MODS found near-optimal solutions, where the results for the shifted sphere function are best (F1-F3). For problems F4-F6 and F7-F9, respectively, we can see that with increasing dimension (from 10 to 30), it becomes increasingly more difficult to find accurate solutions within the given budget of allowable function evaluations. This can be noted especially for the shifted rotated ellipsoid function (F7-F9) for which the errors for the 30-dimensional problem are significantly larger than for the 10-dimensional problem.

For problems F10-F12, both SO-MODS versions detected the global minimum already before Phase 2 started. We should

note here that using a local solver as, for example, MATLAB's fmincon is unlikely to find the optimum of the step function. The local solver fmincon numerically computes the derivatives from the current best point. However, because of the step-shape, all the points in the vicinity of the current best point have the same objective function value (the gradient is zero), and therefore fmincon will terminate without improving the solution. On the other hand, it is not claimed that fmincon would perform well on this problem class (fmincon is a gradient based optimizer that requires the objective function and first derivative to be continuous), which emphasizes the need of derivative-free global optimization algorithms that are able to deal with these kinds of problems.

2) Multi-modal Problems: Test problems F13-F24 have several local minima, which makes it significantly more difficult to find the global optimum within a very restricted number of function evaluations since local optima (if detected) must be explored until no further improvements can be found and thereafter the search must continue globally in order to detect new valleys in the objective function landscape where the global minimum may be located. The results show that also for most of these test problems the mean absolute error increases with the dimension.

The mean errors are lowest for the Griewank function (test functions F16-F18). This is the only multi-modal function for which SO-MODS finds lower errors for the 20- and 30-dimensional instances than for the 10-dimensional problem. One reason for this behavior may be related to the fact that finding the global minimum of the Griewank function becomes easier as the dimension increases [25].

For the remaining multi-modal test functions, the algorithms were on average not able to get as close to the global optimum as for the unimodal problems. For the shifted and rotated Rastrigin function (F22-F24), the errors are largest, which may be related to the large number of local minima. If the local search phase of SO-MODS (Phase 3) does not start anywhere close to the global optimum, the search will stop at a local optimum. SO-MODS-A finds better solutions for this problem than SO-MODS-B which may be related to the exploration of the various local minima of the response surface in SO-MODS-A which enables a more global search for improvements.

The shifted and rotated Rosenbrock function (F19-F21) has a very long and narrow valley with a local and a global optimum. Hence, it can be expected that the local search will converge slowly if it is not trapped in the local minimum. The reported standard deviations of both SO-MODS versions are for this function large compared to the other test problems, which is also reflected in the large difference between the best and the worst function values found over all trials.

3) Analysis of Phases 1 and 2 of SO-MODS-A: We investigated how many objective function evaluations were done in Phases 1 and 2 of SO-MODS-A. Table V summarizes the number of function evaluations averaged over the 20 trials done in each phase. The dashes denote problems for which an improvement was not found in all 20 trials during the respective optimization phase. GS-D denotes the global search (average number of evaluations done during Algorithm 3), GS-G denotes the search with GORBIT, and LS-M denotes the search with MLSL on the response surface (both done in Phase 2). The final local search with fmincon on the true objective function (Phase 3) used up the remaining budget of function evaluations except for problems F10-F12 where the global minimum was detected already during GS-D.

The table shows that for the 10-dimensional problems, GS-D uses for all problems less than half of the allowed number of function evaluations ($N_{\max} = 500$). For the 20-dimensional problems, GS-D uses except for problems F11, F17, and F23 more than half of the budget ($N_{\max} = 1000$) indicating that GS-D keeps making progress. For the 30-dimensional problems, GS-D uses except for problems F12 and F18 always more than half of the budget. The more function evaluations used up by GS-D, the fewer function evaluations are left for the local search.

We set GS-G's budget of function evaluations to about one fourth of the total budget. LS-M's budget of allowed function evaluations is not fixed and depends on the progress made. Hence, we can see that for the unimodal problems only very few evaluations are done during this stage. The minumum of the response surface is for these unimodal problems already relatively well-explored, and hence improvements by sampling at the minimum point of the response surface are rarely possible. For the highly multi-modal Rastrigin function (F22-F24), LS-M finds several improvements by sampling at the local minimum points of the response surface. Hence, for multi-modal problems, exploring these local minima is of advantage which is also reflected in the results in Tables II, III, and IV.

TABLE V. AVERAGE NUMBER OF FUNCTION EVALUATIONS DONE IN EACH PHASE OF SO-MODS-A.

d	Problem	GS-D[4]	GS-G[5]	LS-M[6]
10	F1	196	125	1
	F4	208	126	1
	F7	223	126	2
	F10	122	-	-
	F13	213	125	12
	F16	118	129	2
	F19	205	124	5
	F22	177	125	10
20	F2	583	247	2
	F5	616	249	7
	F8	591	250	4
	F11	309	-	-
	F14	514	249	14
	F17	279	251	2
	F20	550	247	4
	F23	475	249	23
30	F3	907	373	-
	F6	1026	365	-
	F9	1134	329	-
	F12	485	-	-
	F15	897	375	24
	F18	493	374	3
	F21	864	373	5
	F24	843	374	55

[4] Global search with DYCORS (Phase 1).
[5] Global search with GORBIT (Phase 2).
[6] Global search with MLSL for minima of response surface (Phase 2).

4) Progress Plots for SO-MODS: Figures 1, 2, and 3 show the progress plots of SO-MODS versions A and B for the 10-, 20-, and 30-dimensional problems, respectively. The plots show the average best objective function value found after 10%, 20%,. . .,100% of the total number of allowed function

evaluations. Note that the function values are shown on a log-scale, and thus graphs that do not reach until the maximum number of allowed evaluations indicate that the minimum value zero has been found within fewer than the allowed number of evaluations. The graphs show that, except for test problems F19-F21, SO-MODS-A performs better than SO-MODS-B and is outperformed by SO-MODS-B only near the end of the budget of allowable function evaluations. For problems F10-F12 (step function), SO-MODS-A finds the minimum within significantly fewer function evaluations. This behavior may in part be related to the number of points in the initial experimental design which is for SO-MODS-B $10d$ and for SO-MODS-A $2(d+1)$. Hence, SO-MODS-A starts the systematic search for improved solutions earlier and is thus able to find improved objective function values within fewer evaluations than SO-MODS-B. Hence, if the budget of allowed function evaluations is significantly lower than the $50d$ evaluations allowed in this study, SO-MODS-A should be preferred. Otherwise, SO-MODS-B should be used.

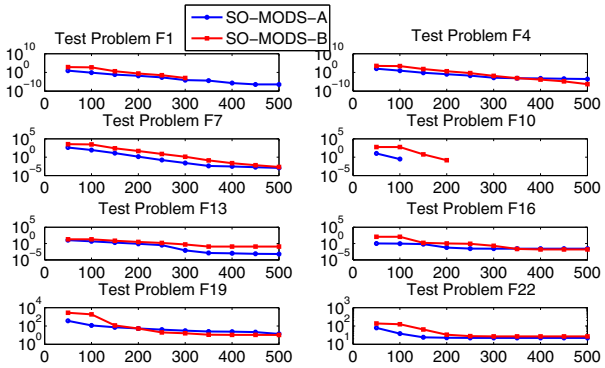

Fig. 1. The graphs show the number of function evaluations vs. the average objective function value for the 10-dimensional problems.

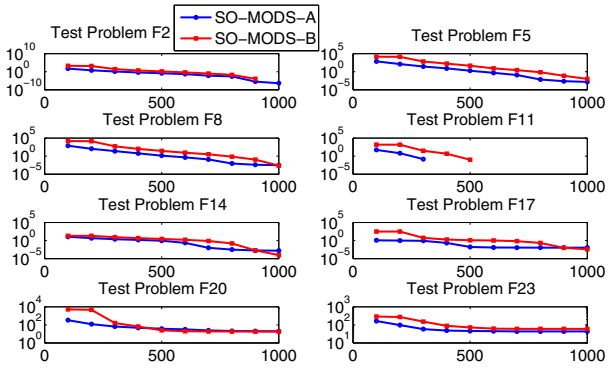

Fig. 2. The graphs show the number of function evaluations vs. the average objective function value for the 20-dimensional problems.

5) Computational Complexity: The computational complexity of both SO-MODS versions is shown in Table VI. As may be expected, the computational complexity increases with the problem dimension. For both algorithm versions, the complexity for test problems F10-F12 is lowest because SO-MODS stopped before the total budget of function evaluations was used up. We can also see that the computational complexity of SO-MODS-B is two to three times larger than that of SO-MODS-A which may be related to the different sizes of initial experimental designs used for both versions and the number of evaluations allowed for ORBIT and GORBIT, respectively.

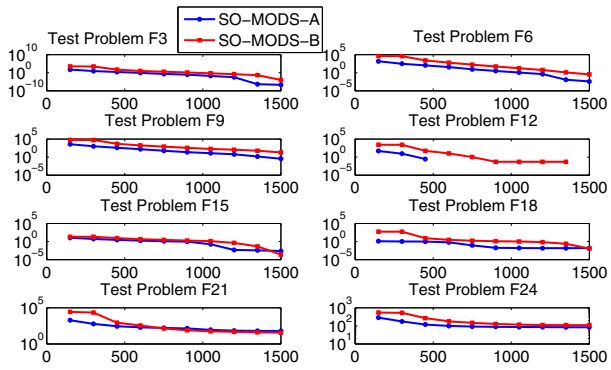

Fig. 3. The graphs show the number of function evaluations vs. the average objective function value for the 30-dimensional problems.

TABLE VI. COMPUTATIONAL COMPLEXITY OF SO-MODS, VERSIONS A AND B

d	Problem	SO-MODS-A	SO-MODS-B
10	F1	76	218
	F4	96	226
	F7	90	147
	F10	13	108
	F13	108	251
	F16	73	143
	F19	84	345
	F22	82	236
30	F2	843	1979
	F5	630	1949
	F8	586	928
	F11	137	710
	F14	648	1489
	F17	310	1079
	F20	563	1750
	F23	565	1367
30	F3	2708	6578
	F6	2394	6903
	F9	2758	5739
	F12	453	2243
	F15	2297	5203
	F18	810	4728
	F21	1858	7760
	F24	2632	4883

IV. CONCLUSIONS

In this paper we introduced the algorithm SO-MODS, which is a surrogate model based optimization algorithm for computationally expensive black-box global optimization problems. SO-MODS is an extension of the DYCORS algorithm introduced by Regis and Shoemaker [1]. SO-MODS extends DYCORS by two additional search phases, namely ORBIT/GORBIT to further improve the best solution found during the global DYCORS search, and a gradient based local optimization on the true objective function to improve the accuracy of the solution. We introduced two versions of SO-MODS where version A uses GORBIT and version B uses ORBIT. Version A explores all the local and global minima of the response surface before it starts the local search on the true objective function. Version B starts the local search from the best point found during ORBIT.

We evaluated the performance of SO-MODS on the CEC'14 test problem suite consisting of 24 problem instances that contained unimodal, multi-modal, continuous, and discontinuous test problems. The problems had 10, 20, and 30 dimensions, and the maximum number of allowed function evaluations was restricted to 50×problem dimension. Numerical experiments showed that both SO-MODS versions are able to find near-optimal solutions. For the multi-modal problems, where the existence of several local minima makes the search for the global minimum significantly more difficult, SO-MODS was able to efficiently decrease the objective function value within very few function evaluations. In general, we found that SO-MODS version B reaches more accurate solutions than version A for most test problems if 50×problem dimension function evaluations are allowed. For lower budgets of allowed evaluations, version A performed for most test problems better.

We found that our results improved by using ORBIT/GORBIT in Phase 2 of SO-MODS as compared to switching directly from the global search in Phase 1 to the gradient-based local search (e.g., with fmincon) in Phase 3. The big advantage of ORBIT/GORBIT over fmincon is that ORBIT/GORBIT does not need at least $d+1$ function evaluations in each iteration to numerically compute the derivatives. However, the response surface used in ORBIT/GORBIT is not exact, and thus, in the final Phase 3 of SO-MODS, it is best to do a gradient-based search with fmincon on the true objective function since the goal in this contest is to find high accuracy solutions (absolute errors less than 10^{-8}) for the CEC'14 benchmark problems. Most real-world application optimization problems of complex systems are not precise and have model errors expected to be at least 5%. For these real models, SO-MODS' Phase 3 may be shortened or even eliminated since finding a solution within 1% or 5% of the optimum is all that is needed.

ACKNOWLEDGMENT

This research was sponsored in part by grant 1116298 from the Computer & Information Science & Engr. Division of the U.S. National Science foundation to Prof. Shoemaker and grant DE-SC0006791 from the SciDAC (Scientific Discovery through Advanced Computing) program at the U.S. Department of Energy to Profs. Mahowald and Shoemaker. The codes used were written at Cornell by the authors.

REFERENCES

[1] R. Regis and C. Shoemaker, "Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization," *Engineering Optimization*, vol. 45, pp. 529–555, 2013.

[2] S. Wild and C. Shoemaker, "Global convergence of radial basis function trust region derivative-free algorithms," *SIAM Journal on Optimization*, vol. 21, pp. 761–781, 2011.

[3] ——, "Global convergence of radial basis function trust-region algorithms for derivative-free optimization," *SIAM Review*, vol. 55, pp. 349–371, 2013.

[4] R. Regis and C. Shoemaker, "Local function approximation in evolutionary algorithms for the optimization of costly functions," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 490–505, 2004.

[5] D. Jones, M. Schonlau, and W. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[6] R. Regis and C. Shoemaker, "Parallel stochastic global optimization using radial basis functions," *INFORMS Journal on Computing*, vol. 21, pp. 411–426, 2009.

[7] ——, "A stochastic radial basis function method for the global optimization of expensive functions," *INFORMS Journal on Computing*, vol. 19, pp. 497–509, 2007.

[8] S. Koziel and L. Leifsson, "Surroagte-based modeling and optimization: Applications in engineering," S. Koziel and L. Leifsson, Eds. Springer, 2013.

[9] A. Forrester, A. Sóbester, and A. Keane, *Engineering Design via Surrogate Modelling - A Practical Guide*. Wiley, 2008.

[10] D. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, pp. 345–383, 2001.

[11] M. Le, Y. Ong, S. Menzel, Y. Jin, and B. Sendhoff, "Evolution by adapting surrogates," *Evolutionary Computation*, vol. 21, pp. 313–340, 2013.

[12] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, pp. 61–70, 2011.

[13] R. Myers and D. Montgomery, *Response Surface Methodology, Process and Product Optimization using Designed Experiments*, J. W. . S. Inc., Ed. Wiley-Interscience Publication, 1995.

[14] J. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, pp. 1–141, 1991.

[15] J. Müller and R. Piché, "Mixture surrogate models based on Dempster-Shafer theory for global optimization problems," *Journal of Global Optimization*, vol. 51, pp. 79–104, 2011.

[16] J. Müller and C. Shoemaker, "Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems," *Journal of Global Optimization, DOI:10.1007/s10898-014-0184-0*, 2014.

[17] J. Müller, C. Shoemaker, and R. Piché, "SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications," *Journal of Global Optimization, DOI: 10.1007/s10898-013-0101-y*, 2013.

[18] ——, "SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems," *Computers and Operations Research*, vol. 40, pp. 1383–1400, 2013.

[19] B. Tolson and C. Shoemaker, "Dynamically dimensioned search algorithm for computationally efficient watershed model calibration," *Water Resources Research*, vol. 43, pp. W01 413, 16 pages, 2007.

[20] S. Wild, "Derivative-free optimization algorithms for computationally expensive functions," Ph.D. dissertation, Cornell University, Operations Research and Information Engineering, 2009.

[21] S. Wild, R. Regis, and C. Shoemaker, "ORBIT: Optimization by radial basis function interpolation in trust-regions," *SIAM Journal on Scientific Computing*, vol. 30, pp. 3197–3219, 2007.

[22] A. Rinnooy Kan and G. Timmer, "Stochastic global optimization methods, part II: multi level methods," *Mathematical Programming*, vol. 39, pp. 57–78, 1987.

[23] M. Powell, *The Theory of Radial Basis Function Approximation in 1990*, Light, Ed. Advances in Numerical Analysis, vol. 2: wavelets, subdivision algorithms and radial basis functions. Oxford University Press, Oxford, pp. 105-210, 1992.

[24] B. Liu, Q. Chen, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, "Problem definitions and evaluation criteria for computationally expensive optimization," CEC 2014, Tech. Rep., 2013.

[25] M. Locatelli, "A note on the Griewank test function," *Journal of Global Optimization*, vol. 25, pp. 169–174, 2003.