Engineering Optimization
Vol. 00, No. 00, Month 200x, 1–29

Constrained Optimization by Radial Basis Function Interpolation for High-Dimensional Expensive Black-Box Problems with Infeasible Initial Points

Rommel G. Regis

ISSN: 0305-215X print/ISSN 1029-0273 online
© 200x Taylor & Francis
DOI: 10.1080/0305215X.YYYY.CATSid
http://www.informaworld.com

Engineering Optimization
Vol. 00, No. 00, Month 200x, 2–29

Constrained Optimization by Radial Basis Function Interpolation for High-Dimensional Expensive Black-Box Problems with Infeasible Initial Points

Rommel G. Regis*

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

(*December 20, 2012*)

This paper develops two new algorithms for constrained expensive black-box optimization that use radial basis function surrogates for the objective and constraint functions. These algorithms are called COBRA and Extended ConstrLMSRBF, and unlike previous surrogate-based approaches, they can be used for high-dimensional problems where all initial points are infeasible. They both follow a two-phase approach where the first phase finds a feasible point while the second phase improves this feasible point. COBRA and Extended ConstrLMSRBF are compared with alternative methods on 20 test problems and on the MOPTA08 automotive problem (Jones 2008), which has 124 decision variables and 68 black-box inequality constraints. The alternatives include a sequential penalty derivative-free algorithm, a direct search method with kriging surrogates, and two multistart methods. Numerical results show that COBRA algorithms are competitive with Extended ConstrLMSRBF and they generally outperform the alternatives on the MOPTA08 problem and most of the test problems.

Keywords: constrained optimization; surrogate model; radial basis function; expensive function; constraint satisfaction; high-dimensional optimization

*Corresponding author. Email: rregis@sju.edu

1. Introduction and Main Contributions

Some of the most challenging engineering optimization problems are those whose objective and constraint functions are black-box functions arising from computationally expensive computer simulations. If a feasible point is not initially available, finding such a point adds another level of difficulty to the problem. Even more challenging are expensive black-box optimization problems with a large number of decision variables and constraints. In many cases, the derivatives of these black-box functions are not available so traditional gradient-based methods might not be suitable. The focus of this paper is to solve optimization problems of the form:

$$\begin{array}{rl} & \min \ f(x) \\ \text{s.t.} & x \in \mathrm{R}^d, \ a \leq x \leq b \\ & g_i(x) \leq 0, \ i = 1, 2, \ldots, m \end{array} \tag{1}$$

where $a, b \in \mathrm{R}^d$ and $f, g_1, \ldots, g_m$ are deterministic, expensive black-box functions. Problems with noise in the objective or constraints will be addressed in future work. Define the vector-valued function $g(x) := (g_1(x), \ldots, g_m(x))$, let $[a, b] := \{x \in \mathrm{R}^d : a \leq x \leq b\}$, and let $\mathcal{D} := \{x \in [a, b] : g(x) \leq 0\}$ be the feasible region of (1). If $f, g_1, \ldots, g_m$ are all continuous on $[a, b]$, then $\mathcal{D}$ is a compact set and f has a global minimizer in $\mathcal{D}$.

Ideally, one wants to obtain a global minimum for (1) using only a relatively small number of objective and constraint function evaluations. However, it usually takes a large number of function evaluations to find a near optimal solution on low-dimensional bound constrained problems. For high-dimensional problems with many black-box constraints, finding the global minimum within a relatively small number of function evaluations is almost impossible. Hence, given a limited computational budget, a reasonable goal is to find a feasible solution that is a substantial improvement over some initial solution.

This paper presents two algorithms for problem (1) that use radial basis function (RBF) surrogates: COBRA (Constrained Optimization By RAdial basis function interpolation) and an extension of ConstrLMSRBF (Regis 2011). Unlike previous surrogate-based approaches, these algorithms can be used even if all initial points are infeasible. In general, constraint satisfaction (i.e., finding a feasible point) for an optimization problem with expensive black-box constraints is already challenging. In practice, an initial feasible point might not be available or the problem might not even have a feasible solution. There are constrained black-box optimization methods that allow infeasible starting points such as Mesh Adaptive Direct Search (MADS) (Audet and Dennis 2006) and SDPEN (Liuzzi *et al.* 2010), which is a sequential penalty derivative-free algorithm. However, COBRA and Extended ConstrLMSRBF are probably among the few surrogate-based approaches that can handle infeasible starting points.

Moreover, the proposed algorithms can be used for large-scale, expensive black-box problems. Previous surrogate-based methods have mostly been applied to low-dimensional expensive black-box problems, typically $d < 15$ (e.g., Queipo *et al.* 2005, 2009, Won and Ray 2005, Regis and Shoemaker 2005, 2007a,b, Huang *et al.* 2006, Holmström 2008, Egea *et al.* 2009, Villemonteix *et al.* 2009, Viana *et al.* 2010, Cassioli and Schoen 2011, Yahyaie and Filizadeh 2011, Basudhar *et al.* 2012, Parr *et al.* 2012). Relatively few surrogate-based methods have been developed and tested on high-dimensional and highly constrained problems where the objective and constraint functions are black-box and expensive. Hence, COBRA and Extended ConstrLMSRBF are significant contributions in surrogate-based optimization.

COBRA and Extended ConstrLMSRBF treat each constraint individually instead of lumping them into one penalty function. They both follow a two-phase structure where the first phase finds a feasible point while the second phase improves this feasible point. In these phases, RBF surrogates for the objective and constraint functions are used to select the iterates. In particular, for COBRA, each iterate is a minimizer of the RBF model of the objective subject to RBF surrogate constraints within some margin and also subject to some distance requirement from previous iterates. The margin is necessary for COBRA because it helps maintain the feasibility of the iterates.

This study compares several approaches for constrained black-box optimization on 20 test problems and on the MOPTA08 benchmark automotive problem (Jones 2008). The MOPTA08 problem has 124 decision variables and 68 inequality constraints, and is much larger than other problems in surrogate-based optimization. Two implementations of COBRA (called COBRA-Local and COBRA-Global) and Extended ConstrLMSRBF are compared to SDPEN (Liuzzi *et al.* 2010), a MADS algorithm that uses DACE surrogates, Matlab's Global Search with an interior-point solver, and multistart GLOBALm (Sendin *et al.* 2009) with a derivative-based solver and also with a derivative-free solver.

Computational results show that given a particular feasible initial point, COBRA-Local outperforms the alternatives on the MOPTA08 problem. In Regis (2011) and Jones (2008), ConstrLMSRBF-BCS was the best algorithm among 12 alternatives on the MOPTA08 problem given this feasible initial point. COBRA-Local outperforms ConstrLMSRBF-BCS on MOPTA08 so it is now among the best methods for this benchmark. Moreover, when the algorithms are initialized by a Latin hypercube design (LHD) whose points are all infeasible, COBRA-Local also outperforms the other methods on MOPTA08 in the sense of requiring less function evaluations to reach a target feasible objective function value. In addition, when feasible initial points are provided, the COBRA algorithms are competitive with Extended ConstrLMSRBF and are much better than the other methods on 12 of the 14 test problems in Regis (2011). On the other hand, when the algorithms are initialized by an infeasible LHD, the COBRA algorithms and Extended ConstrLMSRBF consistently find feasible points for the test problems. Furthermore, the COBRA algorithms are generally better than Extended ConstrLMSRBF and the other methods in reaching the target values on the test problems.

This paper is organized as follows. Section 2 provides a review of literature. Section 3 presents the two proposed RBF methods. Section 4 describes the computational experiments while Section 5 gives a discussion of the numerical results. Finally, Section 6 provides a summary of the paper.

2. Review of Literature

The traditional approach for solving continuous constrained optimization problems is to use a derivative-based local optimization algorithm coupled with a multistart approach. Examples of recent multistart methods for constrained optimization include OQNLP (Ugray *et al.* 2007), GLOBALm (Sendin *et al.* 2009), and the Tabu Tunneling and Tabu Cutting Method (Lasdon *et al.* 2010). This approach assumes that the objective and the constraint functions are smooth and their derivatives are easily obtained. However, in practice, the derivatives of these functions are sometimes not available so they would have to be obtained either by automatic differentiation or finite-differencing. Unfortunately, these procedures are not always applicable, reliable or efficient on expensive black-box functions. Hence, derivative-free methods (e.g., Kolda *et al.* 2003, Conn *et al.* 2009) are popular in practice. Furthermore, there are derivative-free heuristic methods for constrained problems such as evolutionary algorithms (e.g., Hernández Aguirre *et al.* 2004, Mezura-Montes and Coello Coello 2005, Santana-Quintero *et al.* 2010a, Mallipeddi and Suganthan 2010), swarm algorithms (e.g., Muñoz Zavala *et al.* 2005, Cagnina *et al.* 2011, Mezura-Montes and Cetina-Dominguez 2012), simulated annealing (e.g., Hedar and Fukushima 2006) and scatter search (e.g., Egea *et al.* 2007). A recent survey of constraint-handling techniques used in evolutionary algorithms and swarm intelligence algorithms is given by Mezura-Montes and Coello Coello (2011).

Surrogate models such as polynomials, kriging, RBFs, and neural networks have been widely used in expensive black-box optimization. Below are some references but they are not meant to be exhaustive. For example, Wang *et al.* (2001) developed the Adaptive Response Surface Method, which uses quadratic models for the expensive function. Kriging surrogates have been used in simulation experiments (Kleijnen 2008), engineering design (Simpson *et al.* 2001, Forrester *et al.* 2008), the EGO method (Jones *et al.* 1998) and other global optimization methods (e.g., Huang *et al.* 2006, Villemonteix *et al.* 2009). Moreover, RBF surrogates were used to develop global optimization methods by Gutmann (2001), Regis and Shoemaker (2007a,b), Holmström (2008), Jakobsson *et al.* (2010) and Cassioli and Schoen (2011). Kriging has been used with pattern search (Booker *et al.* 1999, Marsden *et al.* 2004) and with scatter search (Egea *et al.* 2009). Conn *et al.* (2009) and Powell (2006) used quadratic interpolation models in derivative-free trust-region methods for unconstrained optimization. In addition, surrogates have been widely used to assist evolutionary algorithms for expensive optimization problems (e.g., Won and Ray 2005, Karakasis and Giannakoglou 2006, Ray and Smith 2006, Annicchiarico 2007, Santana-Quintero *et al.* 2010b, Shahrokhi and Jahangirian 2010). For a discussion of the main issues in surrogate-based analysis and optimization, see Queipo *et al.* (2005).

Examples of surrogate-based methods for problems with expensive black-box constraints and with an expensive black-box objective function are CiMPS (Kazemi *et al.* 2011) and ConstrLMSRBF (Regis 2011). Moreover, Basudhar *et al.* (2012) developed a constrained version of EGO that uses a single support vector machine to approximate the feasible domain. Sasena *et al.* (2002) explored various infill sampling criteria for EGO and applied them to constrained optimization problems. Parr *et al.* (2012) developed a kriging-based method for constrained optimization that uses different infill criteria and selects multiple updates based on Pareto optimal solutions. Rashid *et al.* (2012) also developed an adaptive multiquadric RBF method for mixed-integer and constrained expensive black-box optimization. The ARBF algorithm by Holmström *et al.* (2008) can be used for mixed-integer nonlinearly constrained problems but these nonlinear constraints are either inexpensive or are incorporated into the objective function via penalty terms. In addition, Glaz *et al.* (2009) and Isaacs *et al.* (2009) showed the benefit of using multiple surrogates to approximate the expensive objective or constraint functions.

Local constrained optimization methods for expensive functions use local surrogates near the current iterate. For example, Brekelmans *et al.* (2005) developed a derivative-free trust-region method that uses local linear approximations and a filter method to select its iterates. COBYLA (Powell 1994) is also a derivative-free trust region method that uses linear interpolation models of the objective and constraint functions. NOMAD (Le Digabel 2011, Abramson 2007), which implements Mesh Adaptive Direct Search (MADS) (Audet and Dennis 2006, Abramson and Audet 2006), can be used with kriging surrogates (Abramson 2007) and quadratic models (Conn and Le Digabel 2011).

Another approach for problems with expensive black-box constraints are surrogate-assisted evolutionary algorithms. For example, Wanner *et al.* (2005) and Araujo *et al.*

(2009) use quadratic models to approximate the objective and constraint functions in genetic algorithms (GAs). The Adaptive Surrogate-Assisted Genetic Algorithm by Shi and Rasheed (2008) uses a penalty to handle constraints and a surrogate to approximate the fitness function of the GA. Mezura-Montes and Coello Coello (2011) notes that while surrogates for fitness functions have been extensively used in evolutionary algorithms for unconstrained problems, they are still rarely used to approximate constraint functions.

Relatively few surrogate-based methods can handle expensive black-box problems with large numbers of decision variables and constraints. One approach for these problems is to use dimension reduction methods such as sequential bifurcation (Bettonvil and Kleijnen 1997) before applying any optimization algorithms. Shan and Wang (2010) provided a survey of strategies for *HEB (High-dimensional, Expensive, Black-box)* problems and proposed promising approaches. Moreover, Shan and Wang (2011) developed the RBF-HDMR model and tested it on problems with up to 300 decision variables but with only bound constraints. Regis (2011) developed ConstrLMSRBF, which is suitable for constrained HEB problems but it assumes that a feasible initial point is available. Finally, SDPEN (Liuzzi *et al.* 2010) and PSD-MADS (Audet *et al.* 2008) do *not* use surrogates but they can be used for constrained HEB problems with infeasible initial points.

3. Radial Basis Function Methods for Constrained Black-Box Optimization

3.1. *The COBRA Algorithm*

Below is a description of a new surrogate-based derivative-free optimization method called *COBRA (Constrained Optimization By RAdial basis function interpolation)* that is suitable for problem (1) when the objective and constraint functions are black-box and expensive. COBRA treats each inequality constraint individually instead of lumping them into one penalty function and it uses RBF models to approximate the objective and constraint functions.

Like other surrogate-based methods, COBRA begins by evaluating the expensive functions at the points of a space-filling design in the region $[a, b]$ from (1). None of these design points are guaranteed to be feasible. Then COBRA implements a two-phase approach where Phase I finds a feasible point using RBF surrogates of the constraints and Phase II searches for a better feasible point using RBF surrogates of the objective and constraint functions. However, if an initial feasible point is given or is obtained by the space-filling design, then COBRA simply proceeds to Phase II.

At the beginning of each iteration in Phase I, COBRA fits multiple RBF models, one for each constraint function, using all previous iterates. The next iterate is a minimizer of the sum of the squares of the predicted constraint violations that satisfies RBF surrogates of the constraints within some small margin and that also satisfies a distance requirement from previous iterates. In Phase II, COBRA fits RBF models for the objective function and for each of the constraint functions and follows the same structure as ConstrLMSRBF (Regis 2011) but it uses a different approach for selecting an iterate. In particular, each iterate is a minimizer of the RBF surrogate of the objective that satisfies the same type of constraints as in Phase I. The margin is useful because it helps maintain feasibility of the iterates. On the other hand, ConstrLMSRBF simply chooses its iterate to be the best point from a set of random points with the minimum number of predicted constraint violations according to two criteria: estimated value of the objective function according to the RBF model and minimum distance from previous iterates.

COBRA performs exactly one evaluation of the objective function $f(x)$ and exactly one evaluation of the constraint function $g(x) = (g_1(x), \dots, g_m(x))$ in each iteration. In addition, it assumes that the values of $f(x)$ and $g(x)$ for any point $x \in \mathrm{R}^d$ are obtained in one call to a simulator so that each iteration requires one simulation.

In the description below, n_0 is the number of initial points, n is the number of previously evaluated points, $\mathcal{A}_n = \{x_1, \dots, x_n\}$ is the set of previously evaluated points, and $s_n^{(0)}(x), s_n^{(1)}(x), \dots, s_n^{(m)}(x)$ are the current RBF models for the objective and constraint functions $f(x), g_1(x), \dots, g_m(x)$, respectively. In addition, $NumViol(x)$ is the number of inequality constraint violations while $MaxViol(x)$ is the maximum (inequality) constraint violation of $x \in [a, b]$, i.e., $\mathrm{MaxViol}(x) := \max_{1 \le i \le m}(\max\{s_n^{(i)}(x), 0\})$.

There is a distance requirement ρ_n for the next iterate x_{n+1} on how close it can be to a previous iterate. Define $\rho_n := \gamma_n \ell([a, b])$, where $0 < \gamma_n < 1$ is called a *distance requirement factor* and $\ell([a, b])$ is the length of the smallest side of the box $[a, b] \subseteq \mathrm{R}^d$ in (1). Furthermore, there are two ordered sets of distance requirement factors $\Theta = \langle \theta^{(1)}, \dots, \theta^{(\kappa_1)} \rangle$ and $\Xi = \langle \xi^{(1)}, \dots, \xi^{(\kappa_2)} \rangle$, one for each phase, called the *distance requirement cycles*. The distance requirements γ_n are then obtained by cycling through these values.

COBRA: Constrained Optimization by Radial Basis Function Interpolation

Inputs:

(1) Deterministic black-box functions $f, g_1, \dots, g_m$ whose values at any input $x \in [a, b]$ are obtained in one computer simulation. (Assume that the simulator will not crash for any input $x \in [a, b]$.)
(2) Initial points $\mathcal{I} = \{x_1, x_2, \dots, x_{n_0}\} \subseteq [a, b]$.
(3) The maximum number of computer simulations allowed denoted by N_{max}.
(4) A particular RBF model. (The default is a cubic RBF with a linear polynomial tail.)
(5) The distance requirement cycles $\Theta = \langle \theta^{(1)}, \dots, \theta^{(\kappa_1)} \rangle$ and $\Xi = \langle \xi^{(1)}, \dots, \xi^{(\kappa_2)} \rangle$ for Phases I and II, respectively.
(6) The initial margin ϵ_{init}. (The default value is $\epsilon_{\mathrm{init}} = 0.001\ell([a, b])$.)
(7) The maximum margin ϵ_{max}. (The default value is $\epsilon_{\mathrm{max}} = 0.001\ell([a, b])$.)
(8) The threshold parameter $\mathcal{T}_{\mathrm{feas}}$ for the number of consecutive iterations that yield feasible solutions before the margin is reduced.
(9) The threshold parameter $\mathcal{T}_{\mathrm{infeas}}$ for the number of consecutive iterations that yield infeasible solutions before the margin is increased.

Output: The best point encountered by the algorithm.

Step 1. *(Evaluate Initial Points)* For each $i = 1, \dots, n_0$, compute $f(x_i)$ and $g(x_i) = (g_1(x_i), \dots, g_m(x_i))$. Let x_{best} be the best point found so far and let $f_{\mathrm{best}} := f(x_{\mathrm{best}})$. Set $n := n_0$ and $\mathcal{A}_n := \mathcal{I}$.

Step 2. *(Determine if a Feasible Point is Available)* If all points in $\mathcal{I}$ are infeasible, then proceed to Phase I (Step 3). Else, proceed to Phase II (Step 5).

Phase I (Find a Feasible Point):

Step 3. *(Initialize Margin)* Set $\epsilon_n^{(i)} = \epsilon_{\mathrm{init}}$ for $i = 1, \dots, m$.

Step 4. While a feasible point has not been found do

Step 4.1 *(Fit/Update RBF Models)* Using the data points $\mathcal{C}_n := \{(x_i, g(x_i)) : i = 1, \dots, n\}$, fit or update RBF models $s_n^{(1)}(x), \dots, s_n^{(m)}(x)$ for the m constraint functions.

```
    **Step 4.2** *(Determine Distance Requirement)* Set $\rho_n = \gamma_n \ell([a,b])$, where $\gamma_n$ is the element of the cycle $\Theta$ that corresponds to this iteration.
    **Step 4.3** *(Select Iterate)* Select $x_{n+1}$ to be a solution to the optimization subproblem:

    $$\begin{aligned} \min \sum_{i=1}^{m} \left[\max\left\{s_n^{(i)}(x), 0\right\}\right]^2 \\ \text{s.t.} \quad x \in \mathrm{R}^d,\ a \le x \le b \\ s_n^{(i)}(x) + \epsilon_n^{(i)} \le 0,\ i = 1, 2, \dots, m \\ \|x - x_j\| \ge \rho_n,\ j = 1, \dots, n \end{aligned} \qquad (2)$$

    **Step 4.4** *(Evaluate Functions)* Compute $g_1(x_{n+1}), \dots, g_m(x_{n+1})$ and $f(x_{n+1})$.
    **Step 4.5** *(Update Best Point)* Update the current best point, i.e. if $NumViol(x_{n+1}) < NumViol(x_{\text{best}})$ or if $NumViol(x_{n+1}) = NumViol(x_{\text{best}})$ and $MaxViol(x_{n+1}) < MaxViol(x_{\text{best}})$, then reset $x_{\text{best}} := x_{n+1}$.
    **Step 4.6** *(Update Information)* Set $\mathcal{A}_{n+1} := \mathcal{A}_n \cup \{x_{n+1}\}$, and reset $n := n + 1$.
End.
```

Phase II (Improve the Best Feasible Point):

```
**Step 5.** *(Initialize Margin and Counters)* Set $\epsilon_n^{(i)} = \epsilon_{\text{init}}$ for $i = 1, \dots, m$, $C_{\text{feas}} := 0$, and $C_{\text{infeas}} := 0$.
**Step 6.** While the termination condition is not satisfied (e.g., while $(n < N_{\max})$) do
    **Step 6.1** *(Fit/Update RBF Model)* Using the data points $\mathcal{B}_n = \{(x_i, f(x_i), g(x_i)) : i = 1, \dots, n\}$, fit or update RBF models $s_n^{(0)}(x), s_n^{(1)}(x), \dots, s_n^{(m)}(x)$ for the objective and $m$ constraint functions, respectively.
    **Step 6.2** *(Determine Distance Requirement)* Set $\rho_n = \gamma_n \ell([a,b])$, where $\gamma_n$ is the element of the cycle $\Xi$ that corresponds to this iteration.
    **Step 6.3** *(Select Iterate)* Select $x_{n+1}$ as a solution to the optimization subproblem:

    $$\begin{aligned} \min\ s_n^{(0)}(x) \\ \text{s.t.} \quad x \in \mathrm{R}^d,\ a \le x \le b \\ s_n^{(i)}(x) + \epsilon_n^{(i)} \le 0,\ i = 1, 2, \dots, m \\ \|x - x_j\| \ge \rho_n,\ j = 1, \dots, n \end{aligned} \qquad (3)$$

    **Step 6.4** *(Evaluate Functions)* Compute $g_1(x_{n+1}), \dots, g_m(x_{n+1})$ and $f(x_{n+1})$.
    **Step 6.5** *(Update Counters)* If $x_{n+1}$ is feasible, then reset $C_{\text{feas}} := C_{\text{feas}} + 1$ and $C_{\text{infeas}} := 0$. Otherwise, reset $C_{\text{infeas}} := C_{\text{infeas}} + 1$ and $C_{\text{feas}} := 0$.
    **Step 6.6** *(Adjust Margin)* If $C_{\text{feas}} \ge \mathcal{T}_{\text{feas}}$, reset $\epsilon_n^{(i)} := \epsilon_n^{(i)}/2$ for all $i$ and $C_{\text{feas}} := 0$. If $C_{\text{infeas}} \ge \mathcal{T}_{\text{infeas}}$, reset $\epsilon_n^{(i)} := \max(2\epsilon_n^{(i)}, \epsilon_{\max})$ for all $i$ and $C_{\text{infeas}} := 0$.
    **Step 6.7** *(Update Best Feasible Objective Function Value)* If $x_{n+1}$ is feasible and $f(x_{n+1}) < f_{\text{best}}$, then reset $x_{\text{best}} := x_{n+1}$ and $f_{\text{best}} := f(x_{n+1})$.
    **Step 6.8** *(Update Information)* Set $\mathcal{A}_{n+1} := \mathcal{A}_n \cup \{x_{n+1}\}$, and reset $n := n + 1$.
End.
**Step 7.** *(Return Best Feasible Solution Found)* Return $x_{\text{best}}$.
```

In Step 1, the objective and constraint functions $f, g_1, \ldots, g_m$ are evaluated at the n_0 initial points. Step 2 determines if one of the initial points is feasible. If none of them is feasible, the algorithm enters Phase I (Step 3) where it searches for a feasible point. Once a feasible point has been found or if one of the initial points is feasible, the algorithm proceeds to Phase II (Step 5) where it finds a better feasible point. In Steps 3 and 5, the margin for each inequality constraint $\epsilon_n^{(i)}$ is set to the initial margin ϵ_{init}. Moreover, in Step 5, the counters for the number of consecutive iterations that yielded feasible points (C_{feas}) and the number of consecutive iterations that yielded infeasible points (C_{infeas}) are both set to zero. In Steps 4 and 6, the algorithm goes through the loops that involve fitting or updating the RBF models for the objective and constraint functions, solving the constrained optimization subproblems to determine the next iterate, evaluating the objective and constraint functions at the selected point, and updating the counters, the margin for the inequality constraints and other information. Finally, in Step 7, the best feasible solution found is returned by the algorithm.

In Step 1, the initial points come from a space-filling design over the region $[a, b]$ defined by the bound constraints in (1). If a feasible starting point is provided, these initial points could be in the neighborhood of the feasible starting point since this might yield nearby feasible points with better objective function values. The best point found so far is recorded. Here, the best point among infeasible points is the one with the least number of constraint violations and the smallest maximum constraint violation among those with the least number of constraint violations.

In Phase I, RBF models for the constraint functions are built and used to select of the next iterate. In Phase II, an RBF surrogate for the objective, in addition to the RBF surrogates for the constraint functions, is also used to select the next iterate. In Steps 4.1 and 6.1, the default RBF model has the cubic form with a linear polynomial tail (see Section 3.3). Note that all of the x_i's in Phase I and some of the x_i's in Phase II could be infeasible but they are all used to build the RBF models. The distance requirement of the next iterate from previous iterates is set in Steps 4.2 and 6.2. Next, in Steps 4.3 and 6.3, the iterates are solutions to the optimization subproblems (2) and (3). That is, for Phase I (Step 4.3), the next iterate is a minimizer of the of the sum of the squares of the predicted constraint violations subject to satisfying RBF approximations of the inequality constraints within some margin and also satisfying a distance requirement from previous iterates. If Phase I turns out to be infeasible, then the iterate is a minimizer of the same objective in (2) subject only to the bound constraints. For Phase II (Step 6.3), the next iterate is a minimizer of the RBF approximation of the objective function subject to the same constraints in Phase I. Since RBFs are generally smooth (twice continuously differentiable), COBRA uses gradient-based algorithms such as Matlab's Fmincon to solve the above optimization subproblems.

The margin is meant to force the next iterate to be away from the RBF constraint boundaries to increase the chances that it will turn out to be feasible. In preliminary numerical experiments where a margin was not used, there were many iterations where the function evaluation point was at a boundary of one of the RBF constraints. Since the RBF constraints are only approximations of the black-box constraints, these points often turned out to be infeasible. Hence, a margin is introduced to increase the chances that all constraints will be satisfied by the next iterate. In addition, the distance requirement constraints are similar to those used by CORS-RBF (Regis and Shoemaker 2005) and they are meant to ensure that the iterate is not too close to previous iterates.

In Steps 4.4 and 6.4, the objective and constraint functions are evaluated at the selected point. The objective values are calculated in Step 4.4 even though they are not needed in Phase I because they are used to build RBF surrogates for the objective in Phase II.

In Steps 6.5 and 6.6, COBRA keeps track of the number of *consecutive* iterations that yielded feasible points, which is denoted by C_{feas}, and the number of *consecutive* iterations that yielded infeasible points, which is denoted by C_{infeas}. Whenever C_{feas} reaches the threshold parameter $\mathcal{T}_{\text{feas}}$, the current margin $\epsilon_n^{(i)}$ is reduced by half, C_{feas} is reset to zero, and the algorithm continues. The margin is reduced to gradually allow the algorithm to find the optimal solution if it happens to be located at a constraint boundary. Similarly, whenever C_{infeas} reaches the threshold parameter $\mathcal{T}_{\text{infeas}}$, the current margin $\epsilon_n^{(i)}$ is doubled, C_{infeas} is reset to zero, and the algorithm continues. The margin is doubled to increase the chances that the algorithm will find feasible iterates.

In Steps 4.5 and 6.7, the newly evaluated point x_{n+1} is compared with the current best point x_{best} to determine the new best point. In these comparisons, the point with fewer number of constraint violations wins. In particular, a feasible point is always better than an infeasible point. Between two points with the same number of constraint violations, the one with the smaller maximum constraint violation wins.

Additional implementation details, such as the setting of the algorithm parameters, can be found in Section 4.3.

3.2. *Extended ConstrLMSRBF for Infeasible Initial Points*

ConstrLMSRBF (Regis 2011) is a heuristic surrogate-based method that can be used for HEB problems when a feasible starting point is given. This section presents an extension of ConstrLMSRBF that can be used even if all initial points are infeasible. The *Extended ConstrLMSRBF* algorithm has the same two-phase structure as COBRA. It begins with a space-filling design over the region $[a, b]$ in (1). None of these design points are guaranteed to be feasible so Phase I finds a feasible point using the subroutine RBF_Local _Rand described below. Once a feasible point has been found, it enters Phase II and proceeds as in the original ConstrLMSRBF except that the step size σ_n in Regis (2011) is allowed to increase and the weight for the RBF criterion w_n^R is allowed to vary when the algorithm appears to be stuck in a local minimum.

In Phase I, the current best point is infeasible. In each iteration of RBF_Local_Rand, a set of random candidate points are generated by adding normal perturbations to some (or all) of the components of the current best point. Then the next iterate is chosen to be the one with the minimum number of predicted constraint violations among the candidate points. If there are several points with the minimum number of predicted constraint violations, select the one with the smallest maximum predicted constraint violation.

Below, $\mathcal{C}_n$ is the set of previous iterates and their constraint function values, x_{best} is the current best solution, and p_{select} is the probability of perturbing a coordinate of x_{best}. As in Regis (2011), the algorithm is referred to as *Extended ConstrLMSRBF-BCS* when $p_{select} < 1$. BCS stands for *Block Coordinate Search* and this indicates that only a subset of the coordinates are being perturbed. When $p_{select} = 1$, the algorithm is simply referred to as *Extended ConstrLMSRBF*.

function $x_{\text{best}} =$ **RBF_Local_Rand**$(\mathcal{C}_n, x_{\text{best}}, p_{select})$

While x_{best} is not feasible do

(a) *(Fit/Update RBF Models of Constraints)* Using the data points $\mathcal{C}_n = \{(x_i, g(x_i)) : i = 1, \ldots, n\}$, fit or update RBF models $s_n^{(1)}(x), \ldots, s_n^{(m)}(x)$ for the constraint functions.

(b) *(Generate Multiple Random Candidate Points)* Randomly generate t *candidate points* $\Omega_n = \{y_{n,1}, \ldots, y_{n,t}\} \subseteq [a,b]$ as follows: For $j = 1, \ldots, t$,

(b1) *(Select Coordinates to Perturb in Current Best Solution)* Generate d uniform random numbers $\omega_1, \ldots, \omega_d$ in $[0,1]$. Let $I_{\text{perturb}} := \{i : \omega_i < p_{select}\}$. If $I_{\text{perturb}} = \emptyset$, then select j uniformly at random from $\{1, \ldots, d\}$ and set $I_{\text{perturb}} = \{j\}$.

(b2) *(Generate Candidate Point)* Generate the jth candidate point $y_{n,j}$ by: $y_{n,j} = x_{best} + z_{n,j}$, where $z_{n,j}^{(i)} = 0$ for all $i \notin I_{\text{perturb}}$ and $z_{n,j}^{(i)} \sim N(0, \sigma_n^2)$ for all $i \in I_{\text{perturb}}$. Here, $x^{(i)}$ denotes the ith coordinate of the point $x \in \mathrm{R}^d$.

(b3) *(Ensure Candidate Point Satisfies Bound Constraints)* If $y_{n,j} \notin [a,b]$, then replace $y_{n,j}$ by a point in $[a,b]$ obtained by performing successive reflection of $y_{n,j}$ about the closest point on the boundary of $[a,b]$.

End.

(c) *(Collect Candidate Points with Minimum Number of Predicted Constraint Violations)* Use the RBF models for the constraint functions to determine which points in Ω_n are predicted to be feasible. Let Ω_n^{valid} be the collection of such points.

(d) *(Select Iterate)* Using the RBF models for the constraint functions and the data points $\mathcal{C}_n$, select the next iterate x_{n+1} from the candidate points in Ω_n^{valid} as the one with the smallest value of $MaxViol(x)$.

(e) *(Evaluate Functions)* Compute $g_1(x_{n+1}), \ldots, g_m(x_{n+1})$ and $f(x_{n+1})$.

(f) *(Update Best Point)* Update the current best point, i.e. if $NumViol(x_{n+1}) < NumViol(x_{\text{best}})$ or if $NumViol(x_{n+1}) = NumViol(x_{\text{best}})$ and $MaxViol(x_{n+1}) < MaxViol(x_{\text{best}})$, then reset $x_{\text{best}} := x_{n+1}$.

(g) *(Update Information)* Set $\mathcal{C}_{n+1} := \mathcal{C}_n \cup \{(x_{n+1}, g(x_{n+1}))\}$, and reset $n := n + 1$.

End.

3.3. *Radial Basis Function Interpolation*

The RBF model used in the implementation of COBRA and Extended ConstrLMSRBF is given below. This model is described in Powell (1992) and it is equivalent to *dual kriging* (see Cressie (1993)). Note that the procedure for fitting this model is different from that typically used in RBF networks in machine learning. The RBF model below is much easier to fit than a kriging surrogate thereby giving it an advantage on high-dimensional problems. However, it does not possess the error estimates that are normally associated with kriging. But then the numerical results below suggest that having a measure of error might not be as important as being able to interpolate the data points at least on the problems in this study.

Given n distinct points $x_1, \ldots, x_n \in \mathrm{R}^d$ and the function values $u(x_1), \ldots, u(x_n)$, where $u(x)$ could be the objective function or one of the constraint functions, COBRA and Extended ConstrLMSRBF use an interpolant of the form $s_n(x) = \sum_{i=1}^n \lambda_i \phi(\|x - x_i\|) + p(x)$, $x \in \mathrm{R}^d$. Here, $\|\cdot\|$ is the Euclidean norm, $\lambda_i \in \mathrm{R}$ for $i = 1, \ldots, n$, $p(x)$ is a linear polynomial in d variables, and ϕ has the *cubic* form: $\phi(r) = r^3$. Other possible choices for ϕ include the thin plate spline, multiquadric and Gaussian forms (see Powell (1992)). A cubic RBF model is used here because of prior success with this model in ConstrLMSRBF (Regis 2011).

Define the matrix $\Phi \in \mathrm{R}^{n\times n}$ by: $\Phi_{ij} := \phi(\|x_i - x_j\|), \quad i,j = 1, \ldots, n$. Also, define the matrix $P \in \mathrm{R}^{n\times(d+1)}$ so that its i^{th} row is $[1, x_i^T]$. Now, the cubic RBF model that interpolates the points $(x_1, u(x_1)), \ldots, (x_n, u(x_n))$ is obtained by solving the system

$$\begin{pmatrix} \Phi & P \\ P^T & 0_{(d+1)\times(d+1)} \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} U \\ 0_{d+1} \end{pmatrix}, \tag{4}$$

where $0_{(d+1)\times(d+1)} \in \mathrm{R}^{(d+1)\times(d+1)}$ is a matrix of zeros, $U = (u(x_1), \ldots, u(x_n))^T$, $0_{d+1} \in \mathrm{R}^{d+1}$ is a vector of zeros, $\lambda = (\lambda_1, \ldots, \lambda_n)^T \in \mathrm{R}^n$ and $c = (c_1, \ldots, c_{d+1})^T \in \mathrm{R}^{d+1}$ consists of the coefficients for the linear polynomial $p(x)$. The coefficient matrix in (4) is invertible if and only if $\mathrm{rank}(P) = d + 1$ (Powell 1992). This condition is equivalent to having a subset of $d + 1$ affinely independent points among the points $\{x_1, \ldots, x_n\}$.

The above RBF model is used to approximate the objective function $f(x)$ and each of the constraint functions $g_1(x), \ldots, g_m(x)$ in COBRA and Extended ConstrLMSRBF. For a given set of points where the objective and constraint function values are known, the same interpolation matrix is used. Hence, multiple RBF models are fit by solving (4) with multiple right hand sides and this can be done efficiently even when m is large (e.g., by the backslash operator in Matlab or by standard matrix factorization techniques). In fact, this is one advantage of the above RBF model over other surrogates where a separate procedure is needed to fit the objective and each of the constraint functions.

The framework of COBRA and Extended ConstrLMSRBF can also be implemented using other types of surrogates, including multiple-surrogate approaches (e.g., Glaz *et al.* (2009), Isaacs *et al.* (2009)). For COBRA, the only requirement is that the surrogates are differentiable and their derivatives are easy to calculate so that the optimization subproblems in Phases I and II can be solved by efficient gradient-based methods.

3.4. *Limitations of the Proposed Methods*

COBRA and Extended ConstrLMSRBF are designed to handle black-box inequality constraints and they can be used even if a feasible point is not provided. However, these methods are not intended to handle equality constraints. Although an equality constraint of the form $h(x) = 0$ can be replaced by an inequality constraint $|h(x)| \leq \epsilon$ for some small $\epsilon > 0$ (e.g., see Mezura-Montes and Coello Coello (2011)), preliminary experiments suggest that this might not be the best approach for COBRA and Extended ConstrLMSRBF. Moreover, the proposed methods assume that the simulator always yields objective and constraint function values within the region $[a, b]$ in (1) and they currently do not have a mechanism for recovering from failures of the simulator. In addition, these methods assume that the objective and constraint functions are deterministic and free of noise. These issues will all be addressed in future work.

4. Computational Experiments

4.1. *Alternative Optimization Methods*

COBRA and Extended ConstrLMSRBF are compared with alternative methods for constrained optimization. The choice of alternative methods is severely limited by the lack of publicly available software for surrogate-based constrained optimization. Two of the alternatives are the sequential penalty derivative-free algorithm SDPEN (Liuzzi *et al.* 2010) and the derivative-free NOMAD software (Le Digabel 2011). NOMAD is an implementation of the MADS algorithm (Audet and Dennis 2006), which is an extension of pattern search for constrained optimization that handles nonlinear constraints without using a penalty function. A Matlab version of NOMAD, called NOMADm (Abramson 2007), is used because it can be combined with a surrogate. NOMADm is run with a DACE surrogate and resulting algorithm is referred to as *NOMADm-DACE*.

The alternatives also include GLOBALm (Sendin *et al.* 2009), which is a multistart clustering algorithm for constrained global optimization. The local solvers used in GLOBALm are the derivative-based SOLNP solver (Ye 1989) and the derivative-free UNIRANDI solver. The resulting algorithms are referred to as GLOBALm-SOLNP and GLOBALm-UNIRANDI. Moreover, COBRA is compared with *Global Search* from the Matlab Global Optimization Toolbox (The Mathworks 2010). *Global Search* implements multistart OQNLP (Ugray *et al.* 2007) and the local solver used in Global Search is *Fmincon* from the Matlab Optimization Toolbox (The Mathworks 2009). Here, Fmincon is run using an interior point algorithm. Both SOLNP and Fmincon require derivatives of the objective and constraint functions and these are obtained by finite differencing.

In Regis (2011), ConstrLMSRBF was compared with several alternative methods including: a penalty-based modification of MLMSRBF (Regis and Shoemaker 2007a) called *MLMSRBF-Penalty*; *COBYLA* (Powell 1994); and Matlab's *Fmincon, Pattern Search*, and *Genetic Algorithm* solvers. These methods are not included here since they performed poorly in comparison with ConstrLMSRBF on many test problems used in this paper.

4.2. *A Benchmark Automotive Optimization Problem and Other Test Problems*

COBRA and Extended ConstrLMSRBF are compared with the alternatives on the MOPTA08 automotive problem (Jones 2008) available as a Fortran code at http://anjos.mgi.polymtl.ca/MOPTA2008Benchmark.html. The MOPTA08 problem has one black-box objective function to be minimized, 124 decision variables normalized to $[0, 1]$, and 68 black-box inequality constraints that are well normalized (Jones 2008). It is considered large-scale in the area of surrogate-based expensive black-box optimization because it has vastly more decision variables and constraints than most problems studied in this area. As mentioned earlier, previous papers in this area have mostly dealt with problems with less than 15 decision variables.

In a typical multidisciplinary mass minimization problem in the auto industry, the goal is to determine the values of the decision variables (e.g., shape variables) that minimize the mass of the vehicle subject to performance constraints (e.g., crashworthiness, durability). The MOPTA08 problem is a relatively inexpensive model of an actual automotive design problem. In particular, it is based on kriging response surfaces to a real automotive problem. Each simulation of this problem takes about 0.32 sec on an Intel(R) Core(TM) i7 CPU 860 2.8 Ghz desktop machine while each simulation of the real version could take 1–3 days (Jones 2008). However, as in Regis (2011), the different algorithms are compared by pretending that the objective and constraint functions are expensive.

The proposed methods are also compared with the alternatives on 20 well-known test problems used in Hedar and Fukushima (2006), Mezura-Montes and Cetina-Dominguez (2012) and other studies. Among these are four design problems: WB4 (Welded Beam Design), GTCD4 (Gas Transmission Compressor Design), PVD4 (Pressure Vessel Design), and SR7 (Speed Reducer Design). The number of variables and constraints in these problems and the best known feasible objective function values are given in Table A1 (Appendix). The best known values are taken from Hedar and Fukushima (2006) and Mezura-Montes and Cetina-Dominguez (2012) and from the results in this paper.

As before, the objective and constraint functions of these problems are not expensive to evaluate. In addition, the objective or constraint functions of some of the test problems are rescaled by either dividing by some positive constant or by applying a logarithmic transformation without changing the feasible region and the location of the optima before any algorithms are applied. The rescaling is meant to make the function values less extreme and avoid problems with fitting RBF surrogates.

4.3. *Experimental Setup*

Two sets of numerical experiments are used to compare COBRA and Extended ConstrLMSRBF with alternative methods. The computations are performed in Matlab 7.11.0 using an Intel(R) Core(TM) i7 CPU 860 2.8 Ghz desktop machine. In the first set of experiments, all algorithms begin with a feasible point and the comparisons are done in the same way as in Regis (2011). In particular, all algorithms evaluate the feasible initial point and the d points that are $0.05\ell([a,b])$ away from the initial point along each of the positive coordinate directions. Here, $\ell([a,b])$ is the length of the smallest side of $[a,b]$ in (1). In this case, COBRA and Extended ConstrLMSRBF begin with Phase II and the results for Extended ConstrLMSRBF are identical to ConstrLMSRBF from Regis (2011). In the second set of experiments, all algorithms begin with an affinely independent Latin hypercube design (LHD) of size $d+1$ over the region $[a,b]$ in (1). Here, the comparisons are only carried out on problems where the LHD points are infeasible so that COBRA and Extended ConstrLMSRBF will begin with Phase I.

For all problems except MOPTA08, eight algorithms are compared: two implementations of COBRA (called COBRA-Local and COBRA-Global), Extended ConstrLMSRBF, SDPEN, NOMADm-DACE, GLOBALm-SOLNP, GLOBALm-UNIRANDI, and Global Search. For MOPTA08, ConstrLMSRBF-BCS is also included in the first set of comparisons since this was the best algorithm on this problem in Regis (2011). Moreover, before applying any algorithm to any problem, the variables are rescaled so that the region $[a,b]$ in (1) becomes the unit hypercube $[0,1]^d$.

For each set of experiments, each algorithm is run 30 times on each test problem and 4 times on MOPTA08 where each run corresponds to a different feasible starting point (for the first set of experiments) or a different LHD (for the second set of experiments). However, the same feasible starting point or LHD is used for the different algorithms in a given run. For the first set of comparisons, each algorithm is given a computational budget of 200 objective and constraint function evaluations for the test problems and 4000 function evaluations for MOPTA08. COBRA and NOMADm-DACE are somewhat deterministic and Jones (2008) provided only one feasible starting point for MOPTA08. Hence, for the first set of comparisons, only one run of these algorithms are performed on MOPTA08. For the second set of comparisons, each algorithm is given a computational budget of 500 simulations for the test problems and 2000 simulations for MOPTA08. Here, one simulation yields the values of the objective and all constraint functions.

The two COBRA algorithms use the same distance requirement cycle of $\Theta = \langle 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\rangle$ for Phase I and they differ only in the distance requirement cycle for Phase II. For Phase II, COBRA-Local uses a locally-biased distance requirement cycle of $\Xi = \langle 0.01, 0.001, 0.0005\rangle$ while COBRA-Global uses a somewhat more diverse and global cycle of $\Xi = \langle 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\rangle$. Moreover, for both algorithms, the parameters ϵ_{init} (initial margin) and ϵ_{max} (the maximum margin) are set to $0.005\ell([a,b])$. The optimization subproblems are solved using Fmincon from the Matlab Optimization Toolbox (The Mathworks 2009). In addition, $\mathcal{T}_{\text{feas}}$ (the threshold parameter for deciding when to decrease the margin) and $\mathcal{T}_{\text{infeas}}$ (the tolerance parameter for deciding when to increase the margin) are both set to $\lceil 2\sqrt{d} \rceil$. These parameters are explained in Section 3.1. They may be adjusted based on prior information about a problem and these settings are probably not optimal for each problem. However, these parameters are fixed for all the problems to get a fair comparison among the different methods.

In the first set of experiments, Extended ConstrLMSRBF begins with Phase II so it is identical to the original ConstrLMSRBF. Hence, the results in Regis (2011) are used. For the second set of experiments, the parameters for ConstrLMSRBF are the same as in Regis (2011). That is, p_{select} is set to 0.1 for MOPTA08 and 1.0 for the test problems.

For NOMADm-DACE, the results in Regis (2011) are used for the first set of comparisons. These results were obtained by running NOMADm (Abramson 2007) with a kriging surrogate that uses a Gaussian correlation function. For the second set of comparisons, the parameters used for NOMADm-DACE are the same as in Regis (2011).

For Matlab's Global Search with Fmincon as the local solver, a minimum step size of $10^{-8}\ell([a,b])$ is used for calculating finite difference derivatives. Here, Fmincon runs an interior point algorithm. Moreover, the parameter *NumTrialPoints* in Global Search is set to $2(d+1)$ while the parameter *NumStageOnePoints* is set to $d+1$. These parameters are less than the default values so that Global Search does not spend too much time in the initialization stage since the function evaluations are assumed to be expensive. For all other parameters of Global Search and Fmincon, the default values are used.

For GLOBALm (Sendin *et al.* 2009), the local solvers are SOLNP and UNIRANDI. The number of sample points drawn uniformly in one cycle is set to $\min(10d, 100)$ and the maximum number of clusters is set to 10. The penalty weights for the constraints are set by using information from the initial feasible solutions for the different trials as was done in Regis (2011) for the MLMSRBF-Penalty algorithm. In particular, the penalty weights are set so that a violation of 10^{-6} in any of the inequality constraints results in a penalty function value equal to the maximum objective value of any of the initial feasible solutions. For all other parameters of GLOBALm, the default settings are used.

For the first set of comparisons, the performance of an algorithm on a particular problem is measured by keeping track of the best feasible objective function value after every objective or constraint function evaluation. COBRA performs one call to $f(x)$ and one call to $g(x) = (g_1(x), \ldots, g_m(x))$ in every iteration. However, other algorithms do *not* perform the same number of calls to $f(x)$ and $g(x)$. Moreover, one call to $g(x)$ could be more expensive than one call to $f(x)$ especially when m is large. This study simplifies the analysis and counts one call to either $f(x)$ or $g(x)$ as *one function evaluation*. Because there are multiple trials for each problem, the results are summarized in an *average progress curve*, which is a graph of the *mean of the best feasible objective function value* versus the number of function evaluations. To show variability in the results, error bars representing 95% t-confidence intervals for the mean are included in the plot.

For the second set of comparisons, the algorithms are compared according to the mean number of simulations (where one simulation is one call to $f(x)$ and one call to $g(x)$) required to reach feasibility and also to reach a particular target value on a given problem.

4.4. *Limitation of Comparative Studies of Optimization Algorithms*

One limitation of comparative studies of optimization algorithms is that it is sometimes difficult to guarantee that each method is run with the best parameter settings for the given problems. Although default and reasonable parameter settings are used for the other methods, it might still be possible to improve their performance by more carefully

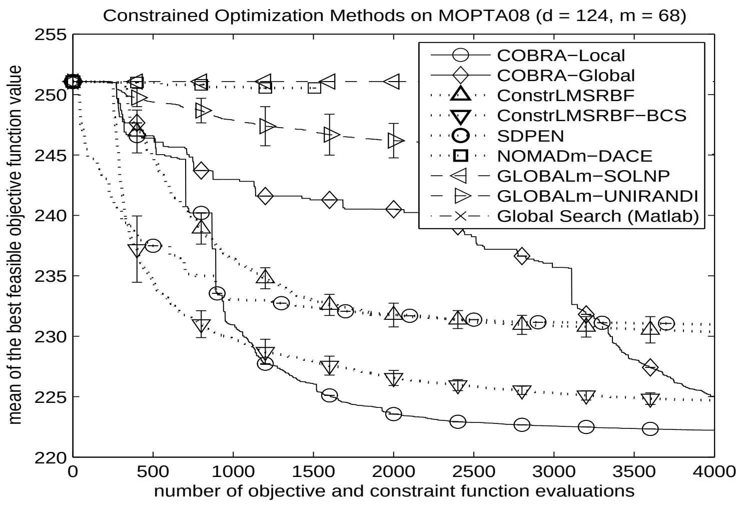


Figure 1. Objective function values of best feasible points found by optimization methods on the MOPTA08 problem. Error bars represent 95% t-confidence intervals about the mean.

tuning their parameters. In fact, even for COBRA and Extended ConstrLMSRBF, there is no guarantee that the default parameter settings are optimal for the problems in this study. The goal of the comparative studies in this paper is not really to prove that the proposed methods are superior to the other methods. Besides, it would be hard to generalize the results to other types of application problems and it is widely believed that there is no universally best algorithm in black-box optimization. Rather, the goal is simply to make the case that these new methods should be seriously considered for solving large-scale expensive black-box optimization problems. This is accomplished below by showing that the proposed methods perform reasonably well (better or comparable in many cases and worse in a few cases) in comparison with the alternatives on an important benchmark application problem and on a wide variety of test problems.

5. Results and Discussion

5.1. *Comparisons on the MOPTA08 Benchmark Given a Feasible Start Point*

Figure 1 shows the average progress curves of the different algorithms on the MOPTA08 problem when they are started at the initial solution provided by Jones (2008). This initial feasible point has an objective value of 251.0706. The average progress curve for Global Search with Fmincon is not included in Figure 1 because it hardly produced any feasible iterates. Table B2 (Appendix) provides the best feasible objective value obtained by the algorithms after 1000, 2000, 3000 and 4000 function evaluations.

From Figure 1 and Table B2, COBRA-Local is the best algorithm on MOPTA08 followed by ConstrLMSRBF-BCS. In particular, after about 1500 function evaluations, the best feasible objective value found by COBRA-Local is statistically significantly better than the mean of the best feasible objective values found by ConstrLMSRBF-BCS in four trials. Moreover, COBRA-Local is generally much better than SDPEN, ConstrLMSRBF, COBRA-Global, and the GLOBALm algorithms. The best feasible solution found by COBRA-Local had an objective function value of 222.2324 with 24 active constraints (using a constraint tolerance of 10^{-4}) and it is better than the best solution found by any of the four runs of ConstrLMSRBF-BCS on MOPTA08 in Regis (2011) and the best value reported by Jones (2008). ConstrLMSRBF-BCS was the best black-box optimization algorithm for MOPTA08 in Regis (2011) and it was also better than the other methods tried by Jones (2008) on MOPTA08, including Generalized Reduced Gradient (using iSIGHT-FD), SQP (Harwell routine VF13), an evolution strategy, a local search method with search directions from local surface approximations, and COBYLA (Powell 1994). Hence, COBRA-Local is now among the best methods on the MOPTA08 benchmark.

COBRA-Global performed well and is much better than NOMADm-DACE, Global Search and the GLOBALm algorithms on MOPTA08. Its progress is initially slow compared to SDPEN, ConstrLMSRBF and ConstrLMSRBF-BCS but it eventually caught up with these algorithms after 4000 function evaluations. Since COBRA-Local is much better than COBRA-Global on MOPTA08, this suggests that allowing new iterates to be closer to previous iterates is more effective for MOPTA08 and the given start point.

SDPEN performed reasonably well on MOPTA08. Although it does not utilize surrogates, it is very competitive with ConstrLMSRBF. In fact, among the non-RBF algorithms in this study and in Regis (2011) and Jones (2008), SDPEN is the best. SDPEN is a derivative-free extension of a sequential penalty method for nonlinear programming (Liuzzi *et al.* 2010). However, it did not perform as well as COBRA-Local or ConstrLMSRBF-BCS. It might be possible to develop a better derivative-free method for constrained black-box optimization by using surrogates within a trust-region framework.

In Regis (2011) and Figure 1, NOMADm-DACE did not perform well on MOPTA08. The code ran out of memory and terminated prematurely possibly because kriging is memory intensive. NOMADm-DACE is not really designed for high-dimensional problems. However, it performs relatively well on the smaller test problems in this study.

Figure B1 (Appendix) indicates the number of constraint violations of the iterates of the different algorithms. The corresponding plots for ConstrLMSRBF, ConstrLMSRBF-BCS and NOMADm-DACE can be found in Regis (2011). These plots are drawn to the same scale and the horizontal line marks an objective function value of 228, which represents 80% of the potential reduction that can be achieved from the starting value (Jones 2008). From Figure B1, the COBRA algorithms made steady progress in objective function value and crossed the horizontal line while obtaining a mixture of feasible and infeasible points. Most of the infeasible points in the trajectories of the COBRA algorithms involve only 1–5 constraint violations (squares). SDPEN also made good progress in objective function value with a mixture of feasible and infeasible points but it did not cross the line. In fact, among the algorithms in Figure B1, only the COBRA algorithms were able to obtain feasible points (circles) below the horizontal line.

From Figure 1, GLOBALm-SOLNP did not improve the best feasible objective value after 4000 function evaluations while GLOBALm-UNIRANDI only obtained a modest improvement. Figure B1 shows an improvement in objective value for GLOBALm-SOLNP but the corresponding solutions are mostly infeasible. For GLOBALm-UNIRANDI, there are many points in the initial trajectory with objective values that are worse than the initial feasible objective value and many of these points have > 5 constraint violations. However, after this initial phase for GLOBALm-UNIRANDI, there is some improvement in the objective value with a mixture of feasible points and infeasible points with only 1–5 constraint violations. Also, Global Search quickly obtained points with good objective values but these points have > 5 constraint violations. To be fair, GLOBALm and Global Search are not designed for expensive black-box functions. They are included to show the difficulty of MOPTA08 and provide some baseline performance on this problem.

Table 1. Number of simulations to reach a feasible point and to reach the target feasible objective value of 228 starting from a Latin Hypercube Design of size $d + 1 = 125$ on MOPTA08.

Trial Number	COBRA-Local		COBRA-Global		Extended ConstrLMSRBF-BCS		SDPEN	
	Feasible	Target	Feasible	Target	Feasible	Target	Feasible	Target
Trial 1	176	751	176	934	476	> 3000	959	> 3000
Trial 2	440	1026	440	1033	283	2577	> 3000	> 3000
Trial 3	429	1031	429	1298	307	> 3000	972	> 3000
Trial 4	425	1041	425	1033	227	> 3000	> 3000	> 3000

5.2. *Comparisons on the MOPTA08 Benchmark when all Initial Points are Infeasible*

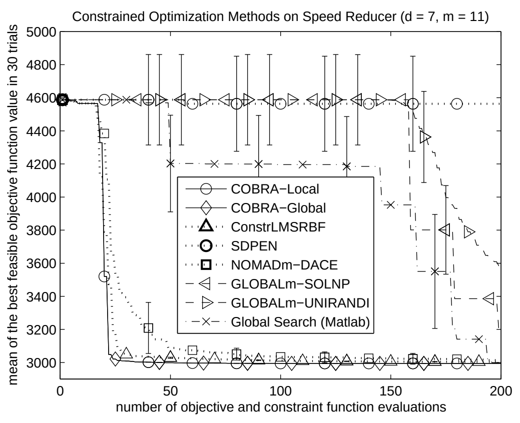
Table 1 provides the number of simulations required by the proposed algorithms and SDPEN to reach a feasible point and the target feasible objective value of 228 starting from an affinely independent LHD of size $d + 1 = 125$. COBRA-Local and COBRA-Global use the same distance requirement cycle for Phase I so their results in Table 1 to reach feasibility are identical. The other alternatives are excluded because of their relatively poor performance in Figure 1. Only four trials are performed because COBRA and Extended ConstrLMSRBF require a large amount of computational overhead.

Table 1 shows that the COBRA algorithms are quite robust in that they are able to reach a feasible point in less than 500 simulations and they are able to reach the target value in all four trials in less than 1300 simulations. Moreover, Extended ConstrLMSRBF is able to reach feasibility faster than the COBRA algorithms and SDPEN. However, the COBRA algorithms are much better than Extended ConstrLMSRBF and SDPEN in reaching the target value. Closer examination of the optimization trajectories show that Extended ConstrLMSRBF makes relatively quick improvements once it found a feasible point but it seems to stall when it is close to the target value. These results also suggest that it might be beneficial to combine Phase I of Extended ConstrLMSRBF with Phase II of COBRA. Phase I of Extended ConstrLMSRBF gets to a feasible point faster while Phase II of COBRA gets to the target value faster on MOPTA08.

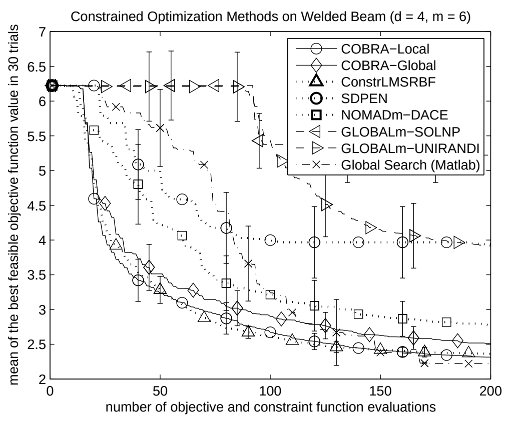
Table 1 also shows that COBRA-Local is better than COBRA-Global on three out of four trials on MOPTA08 when they are initialized by an infeasible LHD of size $d+1 = 125$. From Section 5.1, COBRA-Local is much better than COBRA-Global when they are started from the feasible point provided by Jones (2008). Hence, COBRA-Local seems to be the better algorithm on MOPTA08 whether or not a feasible initial point is available.

5.3. *Comparisons on the Smaller Test Problems Given Feasible Start Points*

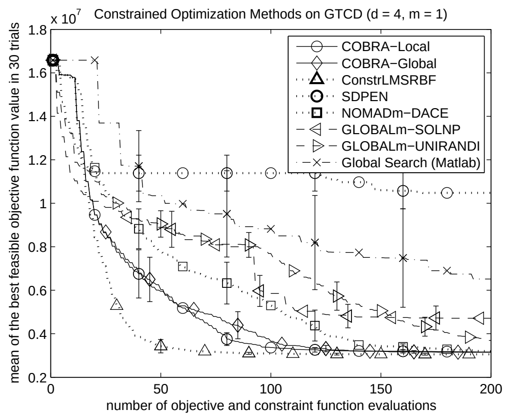
Figure 2 shows the average progress curves of the different algorithms given feasible start points on the 14 test problems used in Regis (2011) while Table B3 (Appendix) provides statistics (over 30 trials) on the best feasible objective values obtained by the algorithms after 100 function evaluations. Figure 2 and Table B3 show that the COBRA algorithms performed very well compared with the alternatives on these 14 test problems. In particular, both COBRA-Local and COBRA-Global are much better than SDPEN, NOMADm-DACE, the GLOBALm algorithms, and Global Search on 12 of these test problems (all except G2 and G10). Moreover, COBRA-Local is generally better than ConstrLMSRBF on six problems (SR7, G3MOD, Hesse, G4, G5MOD and G8) and it is comparable with ConstrLMSRBF on four other problems (WB4, PVD4, G7 and G9).



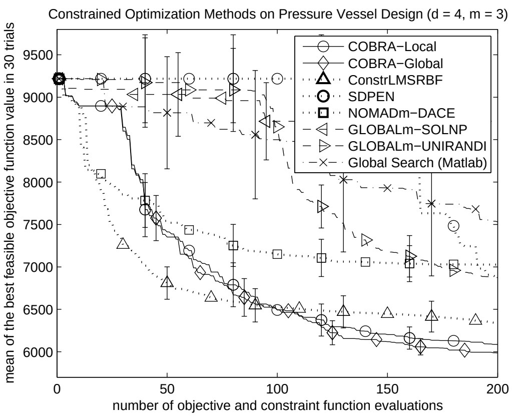
(a)



(b)



(c)

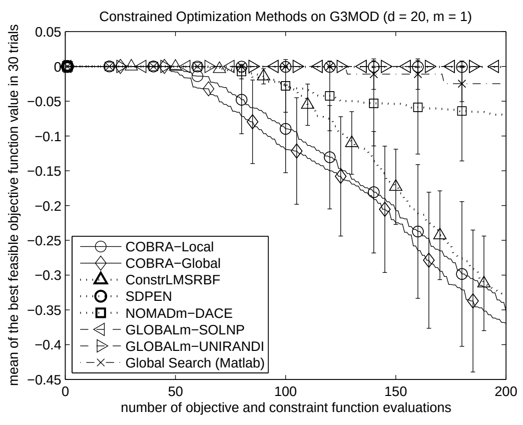


(d)

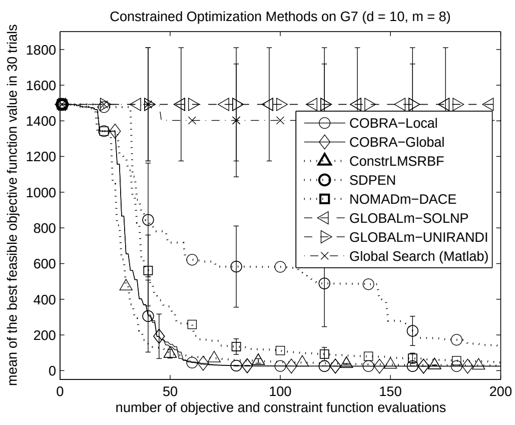
Figure 2. Mean of the best feasible objective function values in 30 trials obtained by optimization methods on test problems. Error bars are 95% t-confidence intervals about the mean.

Observe that neither COBRA-Local nor ConstrLMSRBF consistently dominated the other on the 14 test problems in Figure 2. COBRA selects iterates by solving a constrained optimization subproblem using a gradient-based method. On the other hand, ConstrLMSRBF uses a more heuristic approach of simply selecting the best point from a set of randomly generated candidate points with the minimum number of predicted constraint violations using an RBF criterion and a distance criterion. Note that these two different approaches resulted in competitive results. Moreover, since COBRA uses a more thorough search and uses the gradients of the RBF models, it is natural to expect that COBRA should yield much better results than ConstrLMSRBF. However, as can be seen from the results on GTCD, G2 and G10 in Figure 2, the more thorough search in COBRA did not always translate into faster improvement in the best feasible objective value. This is probably because for some of the test problems, the RBF models are somewhat crude early in the search so that a more thorough approach of using the gradients of these RBF models is not necessarily helpful in finding a better feasible solution.

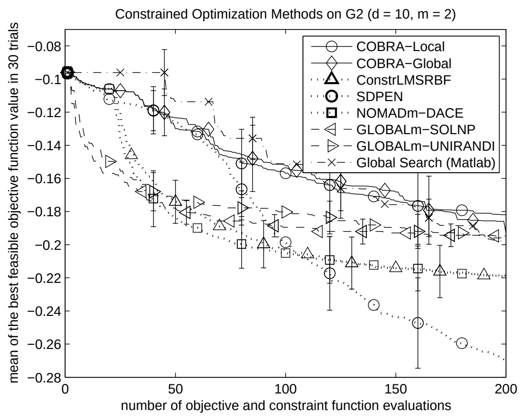
Next, the two COBRA algorithms have very similar performances on 7 of the 14 test problems in Figure 2. COBRA-Local is slightly better than COBRA-Global on three problems (WB4, GTCD and G2) while COBRA-Global is slightly better than



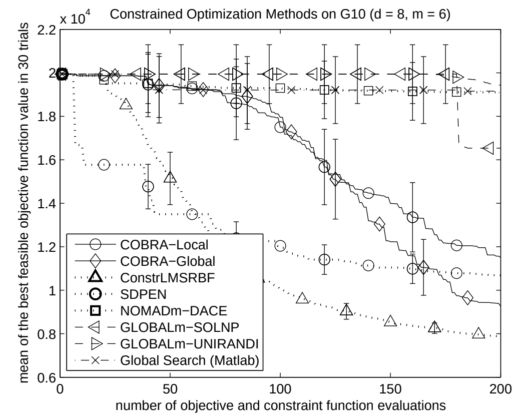
(e)



(f)



(g)



(h)

Figure 2. (Continued) Mean of the best feasible objective function values in 30 trials obtained by optimization methods on test problems. Error bars are 95% t-confidence intervals about the mean.

COBRA-Local on the remaining four problems (G3MOD, G10, Hesse and G8). COBRA-Global uses a more global and diverse distance requirement cycle for Phase II compared to COBRA-Local. Hence, COBRA is probably not very sensitive to the choice of the distance requirements on these test problems.

Figure 2 also shows that the COBRA methods are much better than two derivative-free methods with theoretical convergence analyses: SDPEN and NOMADm-DACE. In particular, they are better than SDPEN on 12 of the 14 problems in Figure 2 (all except G2 and G10) and are better than NOMADm-DACE on 13 problems (all except G2).

In addition, the COBRA algorithms are much better than the GLOBALm algorithms and the Matlab's Global Search with Fmincon, which was run with the interior-point solver option. GLOBALm and Global Search are multistart clustering algorithms for constrained global optimization. These results suggest that when the number of function evaluations is very limited, multistart methods are not expected to perform well because the effort spent in selecting good starting points for the local solvers are probably better spent doing actual local optimization from a good enough starting point. Moreover, when a multistart method is coupled with a derivative-based solver that uses finite difference derivatives, its performance is expected to be poor on expensive problems (Regis and

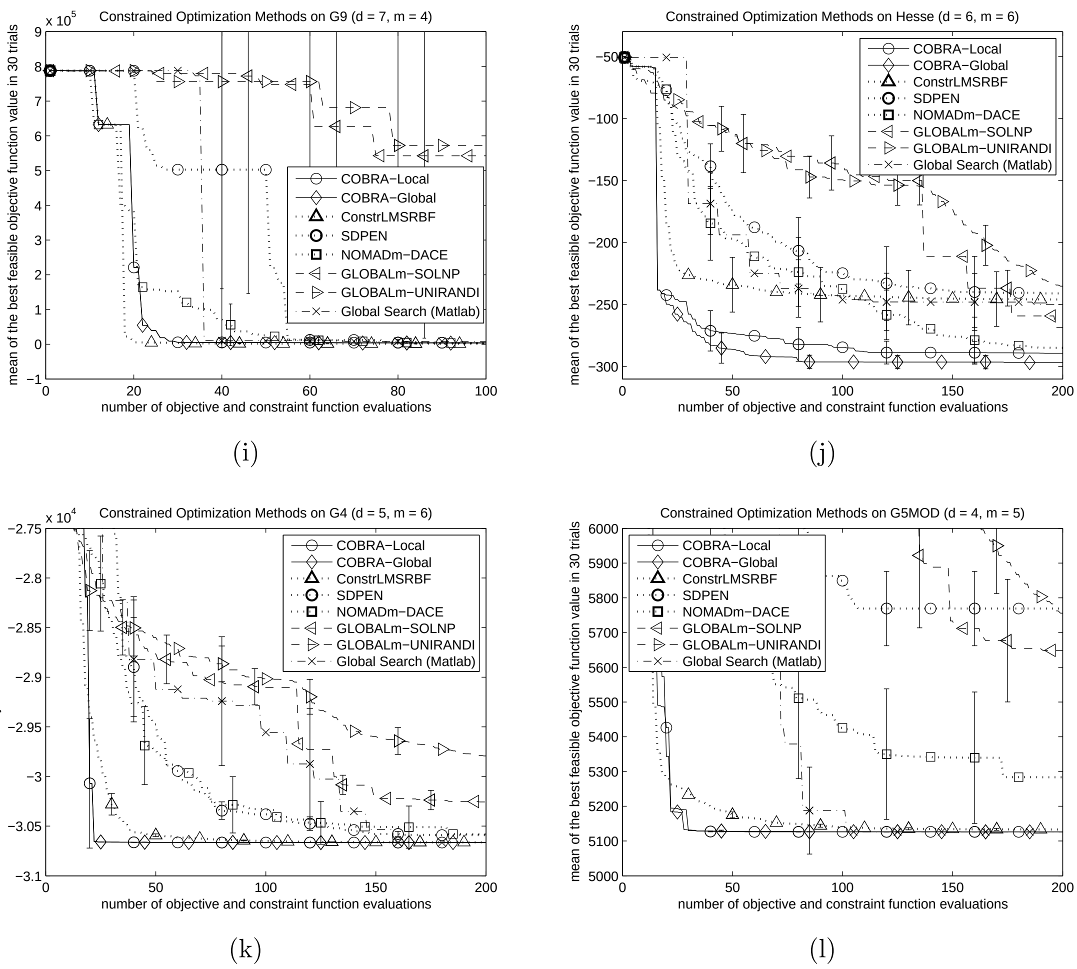


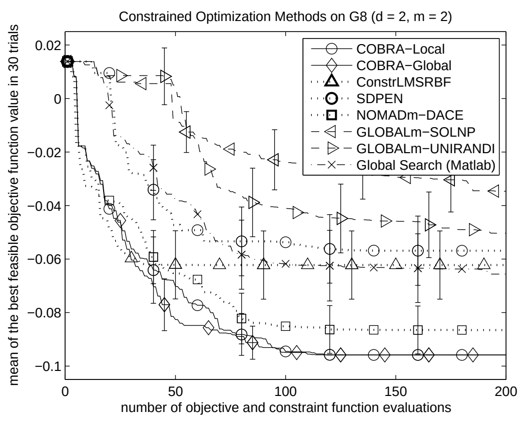
Figure 2. (Continued) Mean of the best feasible objective function values in 30 trials obtained by optimization methods on test problems. Error bars are 95% t-confidence intervals about the mean.

Shoemaker 2007a). Many points evaluated in finite differencing are very close to each other so they are not expected to improve the best feasible solution. Multistart methods will probably perform better if they are coupled with derivative-free trust region methods such as those by Conn *et al.* (2009) or Powell (2006) adapted for constrained problems.

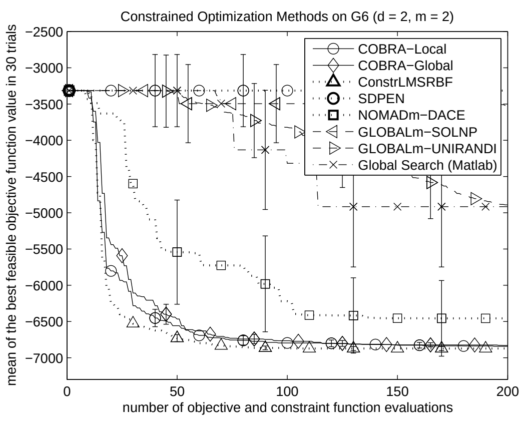
In Regis (2011), ConstrLMSRBF is much better than other alternatives on the 14 test problems in Figure 2. These alternatives include: MLMSRBF-Penalty, which is a penalty-based modification of MLMSRBF (Regis and Shoemaker 2007a); COBYLA (Powell 1994); and Matlab's Fmincon, Pattern Search, and Genetic Algorithm solvers. In Regis (2011), Fmincon implements SQP with finite-difference derivatives. The COBRA algorithms are also much better than these alternatives on most of the test problems.

5.4. *Comparisons on the Smaller Test Problems when all Initial Points are Infeasible*

Table 2 provides the mean number of simulations (over 30 trials) required by the proposed RBF algorithms, SDPEN and NOMADm-DACE to reach a feasible point and to reach a specified target feasible objective value (second column of the table) starting from



(m)



(n)

Figure 2. (Continued) Mean of the best feasible objective function values in 30 trials obtained by optimization methods on test problems. Error bars are 95% t-confidence intervals about the mean.

an affinely independent LHD of size $d + 1$. SDPEN and NOMADm-DACE are the only alternative methods in these comparisons because the other methods performed relatively poorly in Figure 2. Some of the test problems (GTCD, G2, G4 and Hesse) are excluded because the initial LHDs already include feasible points for these problems on most trials.

Table 2 shows that COBRA and Extended ConstrLMSRBF are consistently successful in finding feasible points for all problems in all 30 trials. Moreover, they reach feasibility in much less simulations than SDPEN on 15 out of 16 test problems (all except G24); they also reach feasibility in less simulations than NOMADm-DACE on 12 of the problems (all except G7, G8, G9 and G18) in Table 2. In addition, COBRA is better than Extended ConstrLMSRBF on 11 out of 16 test problems (all except WB4, G7, G8, G13MOD and G24) and it is as good as Extended ConstrLMSRBF on G24 in terms of finding a feasible point. This suggests that the more thorough and global strategy of COBRA (Phase I) is somewhat more effective in finding an initial feasible point than the more heuristic and local strategy of Extended ConstrLMSRBF (Phase I).

Table 2 also shows that all algorithms have difficulty reaching the target value on some problems. However, the COBRA algorithms performed well compared to the other methods on most of the problems. In particular, COBRA-Global is either the best or the second best on 13 out of 16 problems (all except G3MOD, G9 and G16) and COBRA-Local is also either the best or the second best on 10 problems (all except PVD4, G3MOD, G7, G9, G1 and G13MOD). Moreover, COBRA-Global and COBRA-Local are comparable on the test problems and they are both generally better than Extended ConstrLMSRBF when all the initial points are infeasible.

5.5. *Comparison of Running Times*

Table B4 (Appendix) provides the overhead running times (i.e., excluding total times spent on function evaluations) of the different algorithms on the MOPTA08 problem after 2000 and 4000 function evaluations. For example, the overhead running times of COBRA-Local on MOPTA08 are 11.05 hours and 41.74 hours for 2000 and 4000 function evaluations, respectively. The overhead running times for ConstrLMSRBF, ConstrLMSRBF-BCS and NOMADm-DACE on MOPTA08 are found in Regis (2011).

Table 2. Mean number of simulations (for 30 trials) to obtain a feasible point and to reach the target feasible objective value starting from an LHD of size $d+1$ for the different algorithms. The number inside the parenthesis is the standard error of the mean or the number of trials that did not reach the target value.

Problem	Best Known	Target	COBRA-Local		COBRA-Global		Extended ConstrLMSRBF	
	Value	Value	Feasible	Target	Feasible	Target	Feasible	Target
SR7	2994.42	2995	9.47 (0.11)	35.00 (2.67)	9.47 (0.11)	33.53 (1.57)	12.40 (0.36)	$>$ 500 (30)
WB4	1.7250	2.5	37.40 (5.92)	164.57 (12.23)	37.40 (5.92)	202.03 (12.94)	25.00 (4.10)	238.63 (19.99)
PVD4	5804.45	6000	7.87 (0.35)	$>$ 212.17 (2)	7.87 (0.35)	155.43 (38.19)	10.40 (0.65)	$>$ 263.53 (1)
G3MOD	−0.69	−0.33	23.53 (0.20)	$>$ 451.10 (24)	23.53 (0.20)	$>$ 439.97 (24)	31.20 (0.27)	141.70 (8.60)
G7	24.3062	25	47.47 (4.64)	199.47 (20.68)	47.47 (4.64)	99.80 (5.72)	39.83 (2.90)	264.53 (34.19)
G10	7049.3307	8000	22.80 (1.51)	276.43 (43.55)	22.80 (1.51)	$>$ 193.67 (1)	51.10 (6.50)	$>$ 394.30 (6)
G9	680.6301	1000	21.50 (1.85)	$>$ 275.47 (2)	21.50 (1.85)	176.40 (26.29)	23.10 (2.27)	$>$ 294.03 (1)
G5MOD	5126.50	5150	6.37 (0.09)	12.90 (0.46)	6.37 (0.09)	16.57 (1.84)	9.60 (0.29)	40.30 (1.39)
G8	−0.0958	−0.09	6.47 (0.15)	30.30 (2.79)	6.47 (0.15)	31.20 (2.53)	5.20 (0.18)	46.20 (6.20)
G6	−6961.8139	−6800	10.90 (0.30)	53.57 (13.95)	10.90 (0.30)	62.50 (10.51)	11.90 (0.24)	$>$ 101.23 (4)
G1	−15	−14.85	15 (0)	$>$ 387.77 (23)	15 (0)	125.17 (15.33)	19.07 (0.38)	$>$ 500 (30)
G13MOD	0.0035	0.005	9.37 (0.78)	221.67 (35.63)	9.37 (0.78)	169.00 (19.08)	8.60 (0.67)	146.43 (29.18)
G16	−1.9052	−1.8	14.70 (2.37)	38.77 (9.26)	14.70 (2.37)	46.33 (13.52)	19.57 (1.78)	38.40 (3.63)
G18	−0.8660	−0.8	108.57 (6.45)	$>$ 195.90 (6)	108.57 (6.45)	$>$ 212.83 (7)	122.03 (5.64)	$>$ 276.00 (9)
G19	32.6556	40	16.50 (0.50)	698.53 (75.25)	16.50 (0.50)	850.87 (70.63)	20.80 (0.77)	$>$ 1000 (30)
G24	−5.5080	−5	1.33 (0.09)	9.00 (0)	1.33 (0.09)	9.00 (0)	1.33 (0.09)	91.90 (5.98)

Problem	Best Known	Target	SDPEN		NOMADm-DACE	
	Value	Value	Feasible	Target	Feasible	Target
SR7	2994.42	2995	127.40 (7.43)	227.17 (1.98)	16.93 (1.49)	$>$ 82.00 (2)
WB4	1.7250	2.5	$>$ 199.20 (2)	$>$ 327.53 (30)	$>$ 129.33 (2)	$>$ 324.57 (13)
PVD4	5804.45	6000	$>$ 62.53 (2)	$>$ 142.13 (4)	14.87 (0.59)	$>$ 129.67 (30)
G3MOD	−0.69	−0.33	55.90 (0.89)	$>$ 702.27 (30)	58.17 (3.89)	$>$ 416.37 (28)
G7	24.3062	25	153.30 (9.23)	$>$ 644.47 (29)	40.37 (1.74)	58.90 (1.55)
G10	7049.3307	8000	$>$ 384.77 (28)	$>$ 400.67 (30)	$>$ 230.33 (30)	$>$ 230.33 (30)
G9	680.6301	1000	38.10 (3.56)	$>$ 88.63 (1)	22.90 (2.99)	135.50 (11.81)
G5MOD	5126.50	5150	$>$ 46.03 (2)	$>$ 220.10 (30)	$>$ 20.30 (3)	$>$ 30.03 (5)
G8	−0.0958	−0.09	19 (0)	32 (0)	5 (0)	46.87 (2.06)
G6	−6961.8139	−6800	$>$ 74 (30)	$>$ 74 (30)	$>$ 26.57 (4)	$>$ 34.93 (11)
G1	−15	−14.85	48.63 (0.92)	$>$ 323.50 (13)	30.23 (2.34)	$>$ 120.27 (1)
G13MOD	0.0035	0.005	20.63 (2.98)	$>$ 270.87 (30)	10.47 (1.17)	$>$ 97.03 (2)
G16	−1.9052	−1.8	$>$ 421.50 (30)	$>$ 421.50 (30)	64.63 (6.93)	$>$ 146.27 (2)
G18	−0.8660	−0.8	168.93 (6.79)	$>$ 495.60 (18)	86.23 (5.66)	$>$ 190.33 (14)
G19	32.6556	40	59.83 (3.69)	$>$ 1068.37 (30)	54.63 (5.75)	$>$ 1006.87 (30)
G24	−5.5080	−5	1.33 (0.09)	$>$ 77.70 (22)	NA	NA

Clearly, the running times of the COBRA algorithms on MOPTA08 are much longer than those of the alternative methods. However, for truly expensive functions, these running times are still much smaller than the total time spent on function evaluations. For example, if each objective or constraint function evaluation of the MOPTA08 problem takes 1 hour, then the total running time of COBRA-Local for 2000 function evaluations would be 2011.05 hours while the mean total running time of GLOBALm-UNIRANDI for 2000 function evaluations would be 2000.0046 hours. However, from Figure 1, the mean of the best feasible objective value obtained by COBRA-Local is much better than that obtained by GLOBALm-UNIRANDI after 2000 function evaluations.

The overhead running times on the test problems are not reported because they are much smaller than the overhead running time on MOPTA08. Besides, if a test problem is assumed to be expensive, then the total running time of an algorithm is dominated by the total time spent on function evaluations.

6. Summary and Conclusions

This paper introduced the COBRA and Extended ConstrLMSRBF algorithms for the optimization of expensive black-box objective functions subject to expensive black-box inequality constraints. These methods treat each constraint individually instead of lumping them into one penalty function and they use RBF surrogates for the objective and constraint functions in every iteration. Unlike many previous surrogate-based methods, COBRA and Extended ConstrLMSRBF can handle a large number of decision variables and inequality constraints and they can be used even if all initial points are infeasible. Both algorithms implement a two-phase approach where the first phase finds a feasible point while the second phase improves this feasible point.

The iterate in Phase I of COBRA is a minimizer of the sum of squares of the predicted constraint violations subject to satisfying the RBF approximations of the black-box inequality constraints within some margin and also subject to a distance requirement from previous iterates. The iterate in Phase II of COBRA is a minimizer of the RBF model of the black-box objective subject to the same constraints as in Phase I. The margin is necessary because it helps maintain the feasibility of the iterates. Because RBFs are generally smooth, gradient-based methods are used to obtain these iterates. On the other hand, the iterate in Phase I of Extended ConstrLMSRBF is chosen from a set of random candidate points as the one with the minimum number of predicted constraint violations. Finally, Phase II of Extended ConstrLMSRBF is identical to ConstrLMSRBF.

Extended ConstrLMSRBF and two implementations of COBRA that differ in the distance requirement parameters (COBRA-Local and COBRA-Global) are compared with alternative methods on 20 test problems and on the MOPTA08 benchmark automotive problem (Jones 2008). The MOPTA08 problem has 124 decision variables and 68 inequality constraints and is considered large scale in surrogate-based optimization. The alternatives include SDPEN, NOMADm-DACE, and the multistart GLOBALm and Global Search algorithms coupled with derivative-based and derivative-free solvers. Two sets of comparisons are performed for each problem. In the first set of comparisons, the algorithms are compared in terms of how quickly they can improve the objective function value from feasible starting points. In the second set of comparisons, the algorithms are compared in terms of the number of simulations required to reach feasibility and to reach a given target value for each problem.

The numerical results suggest that COBRA is now among the best methods for the

MOPTA08 problem. In particular, given the feasible start point provided by Jones (2008), COBRA-Local outperformed all alternatives including ConstrLMSRBF-BCS, which was the best algorithm on MOPTA08 in previous studies (Jones 2008, Regis 2011). Moreover, when the algorithms are initialized by a Latin hypercube design of size $d+1 = 125$ whose points are all infeasible, COBRA-Local is able to reach the target value more quickly compared to the other approaches. However, Extended ConstrLMSRBF finds a feasible point faster than the other methods but takes longer to reach the target value.

The results also show that when feasible start points are provided, the COBRA algorithms are much better than SDPEN, NOMADm-DACE and the multistart methods on 12 of the 14 test problems in Figure 2. Moreover, they are competitive with Extended ConstrLMSRBF on these 14 problems. In particular, COBRA-Local is better than Extended ConstrLMSRBF on six of these problems and it is comparable with Extended ConstrLMSRBF on another four problems. Although COBRA uses a gradient-based approach to select its iterates, it does not always outperform the more heuristic Extended ConstrLMSRBF. In addition, COBRA-Local and COBRA-Global have comparable performances on these test problems.

When the algorithms are initialized by an LHD of size $d+1$ whose points are all infeasible, the COBRA algorithms and Extended ConstrLMSRBF are consistently successful in finding feasible points for all 16 test problems in Table 2 and in all trials. Moreover, they are able to reach feasibility in much less simulations than either SDPEN or NOMADm-DACE on most of the test problems in Table 2. In addition, the COBRA algorithms performed generally better than Extended ConstrLMSRBF and the other methods in terms of reaching the target feasible objective function value faster on the test problems.

In summary, COBRA and Extended ConstrLMSRBF are very promising surrogate-based methods for high-dimensional and highly constrained, expensive, black-box optimization and constraint satisfaction problems.

Acknowledgements

I would like to thank Dr. Don Jones for the Fortran simulation code for the MOPTA08 Benchmark Problem from the automotive industry. I am grateful to thank Dr. Liuzzi, Prof. Lucidi, and Prof. Sciandrone for the Fortran code for the SDPEN algorithm, for Dr. Julio Banga's research group for the Matlab code for GLOBALm, and for Dr. Abramson and Prof. Audet's group for the NOMADm software. I am also grateful to Prof. Nick Sahinidis for providing me with a reference to the SDPEN algorithm. Finally, I would like to thank the referees for their helpful comments and suggestions.

References

Abramson, M.A., 2007. *NOMADm version 4.6 User's Guide.* Unpublished manuscript.

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

Annicchiarico, W., 2007. Metamodel-assisted distributed genetic algorithms applied to structural shape optimization problems. *Engineering Optimization*, 39(7), 757–772.

Araujo, M.C., Wanner, E.F., Aes, F.G.G., Takahashi, R.H.C., 2009. Constrained optimization based on quadratic approximations in genetic algorithms. *In*: E. Mezura-

Montes, ed. *Constraint-Handling in Evolutionary Computation: Studies in Computational Intelligence, Vol. 198*, Berlin: Springer, 193-217.

Audet, C., Dennis, J.E.Jr., 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(2), 188–217.

Audet, C., Dennis, J.E.Jr., Le Digabel, S., 2008. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization*, 19(3), 1150–1170.

Basudhar, A., Dribusch, C., Lacaze, S., Missoum, S., 2012. Constrained efficient global optimization with support vector machines. *Structural and Multidisciplinary Optimization*, DOI: 10.1007/s00158-011-0745-5.

Bettonvil, B., Kleijnen, J.P.C., 1997. Searching for important factors in simulation models with many factors: Sequential bifurcation. *European Journal of Operational Research*, 96(1), 180–194.

Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W., 1999. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1), 1–13.

Brekelmans, R., Driessen, L., Hamers, H., den Hertog, D., 2005. Constrained optimization involving expensive function evaluations: a sequential approach. *European Journal of Operational Research*, 160, 121–138.

Buhmann, M.D. 2003. *Radial Basis Functions.* Cambridge, UK: Cambridge Univ. Press.

Cagnina, L.C., Esquivel, S.C., Coello Coello, C.A., 2011. Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. *Engineering Optimization*, 43(8), 843-866.

Cassioli, A., Schoen, F., 2011. Global optimization of expensive black box problems with a known lower bound. *Journal of Global Optimization*, doi:10.1007/s10898-011-9834-7.

Conn, A.R., Le Digabel, S., 2011. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, DOI:10.1080/10556788.2011.623162.

Conn, A.R., Scheinberg, K., Vicente, L.N., 2009. *Introduction to Derivative-Free Optimization.* Philadelphia, PA: SIAM.

Cressie, N., 1993. *Statistics for Spatial Data.* New York: Wiley.

Egea, J.A., Rodriguez-Fernandez, M., Banga, J.R., Marti, R., 2007. Scatter Search for chemical and bioprocess optimization. *Journal of Global Optimization*, 37(3), 481–503.

Egea, J.A., Vazquez, E., Banga, J.R., Marti, R., 2009. Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization*, 43(2-3), 175–190.

Forrester, A.I.J., Sobester, A., Keane, A.J., 2008. *Engineering Design via Surrogate Modelling a Practical Guide.* John Wiley & Sons.

Glaz, B., Goel, T., Liu, L., Friedmann, P.P., Haftka, R.T. 2009. Multiple-surrogate approach to helicopter rotor blade vibration reduction. *AIAA Journal*, 47(1), 271–282.

Gutmann, H.-M., 2001. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3), 201–227.

Hedar, A., and M. Fukushima, M., 2006. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, 35: 521–549.

Hernández Aguirre, A., Botello, S., Lizarraga, G., Coello Coello, C.A., 2004. Handling Constraints using Multiobjective Optimization Concepts. *International Journal of Numerical Methods in Engineering*, 59(15), 1989–2017.

Holmström, K., 2008. An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. *Journal of Global Optimization*, 41(3), 447–464.

Holmström, K., Quttineh, N.H., Edvall, M.M., 2008. An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. *Optimization and Engineering*, 9(4), 311–339.

Huang, D., Allen, T.T., Notz, W.I., Zeng, N., 2006. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3), 441–466.

Isaacs, A., Ray, T., Smith, W., 2009. Multiobjective design optimization using multiple adaptive spatially distributed surrogates. *International Journal of Product Development*, 9(1–3), 188-217.

Jakobsson, S., Patriksson, M., Rudholm, J., Wojciechowski, A., 2010. A method for simulation based optimization using radial basis functions. *Optimization and Engineering*, 11(4), 501–532.

Jones, D.R., 2008. Large-scale multi-disciplinary mass optimization in the auto industry. Presented at the *Modeling and Optimization: Theory and Applications (MOPTA) 2008 Conference*. Ontario, Canada.

Jones, D.R., Schonlau, M., Welch, W.J., 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4), 455–492.

Kazemi, M., Wang, G.G., Rahnamayan, S., Gupta, K., 2011. Metamodel-based optimization for problems with expensive objective and constraint functions. *ASME Journal of Mechanical Design*, 133(1), 014505, doi:10.1115/1.4003035.

Karakasis, M.K., Giannakoglou, K.C., 2006. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8), 941–957.

Kleijnen, J.P.C., 2008. *Design and Analysis of Simulation Experiments.* Springer.

Kolda, T.G., Lewis, R.M., Torczon, V., 2003. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Review*, 45(3), 385–482.

Lasdon, L., Duarte, A., Glover, F., Laguna, M., Marti, R., 2010. Adaptive memory programming for constrained global optimization. *Computers & Operations Research*, 37(8), 1500–1509.

Le Digabel, S., 2011. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4), 44:1–44:15.

Liuzzi, G., Lucidi, S., Sciandrone, M., 2010. Sequential penalty derivative-free methods for nonlinear constrained optimization. *SIAM Journal on Optimization*, 20(5), 2614 – 2635.

Lophaven, S.N., Nielsen, H.B., Søndergaard, J., 2002. *DACE: A Matlab Kriging Toolbox, Version 2.0.* Technical Report IMM-TR-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

Mallipeddi, R., Suganthan, P.N., 2010. Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4), 561–579.

Marsden, A.L., Wang, M., Dennis, J.E.Jr., Moin, P., 2004. Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering*, 5(2), 235–262.

Mezura-Montes, E., Coello Coello, C.A., 2005. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1), 1–17.

Mezura-Montes, E., Coello Coello, C.A., 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Copmputation*, 1: 173–194.

Mezura-Montes, E., Cetina-Dominguez, O., 2012. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and*

Computation, 218(22): 10943–10973.

Muñoz Zavala, A.E., Hernández Aguirre, A., Villa Diharce, E.R., 2005. Constrained optimization via particle evolutionary swarm optimization algorithm (PESO). *Genetic and Evolutionary Computation Conference (GECCO) 2005*, Washington, DC, USA, 209-216.

Parr, J.M., Keane, A.J., Forrester, A.I.J., Holden, C.M.E., 2012. Infill sampling criteria for surrogate-based optimization with constraint handling. *Engineering Optimization*, DOI:10.1080/0305215X.2011.637556.

Powell, M.J.D., 1992. The theory of radial basis function approximation in 1990. *In*: W. Light, ed. *Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions*, Oxford, UK: Oxford Univ. Press, 105–210.

Powell, M.J.D., 1994. A direct search optimization methods that models the objective and constraint functions by linear interpolation. *In*: S. Gomez and J.P. Hennart, eds. *Advances in Optimization and Numerical Analysis*, Dordrecht: Kluwer, 51–67.

Powell, M.J.D., 2006. The NEWUOA software for unconstrained optimization without derivatives. *In*: G. Di Pillo, M. Roma, eds. *Large-Scale Nonlinear Optimization*, US: Springer, 255–297.

Queipo, N.V., Haftka, R.T., Shyya, W., Goel, T., Vaidyanathan, R., Tucker, P.K., 2005. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41, 1-28.

Queipo, N.V., Verde, A., Pintos, S., Haftka, R.T., 2009. Assessing the value of another cycle in Gaussian process surrogate-based optimization. *Structural and Multidisciplinary Optimization*, 39, 459-475.

Rashid, K., Ambanib, S., Cetinkaya, E., 2012. An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. *Engineering Optimization*, DOI:10.1080/0305215X.2012.665450.

Ray, T., Smith, W., 2006. A surrogate assisted parallel multiobjective evolutionary algorithm for robust engineering design. *Engineering Optimization*, 38(8), 997-1011.

Regis, R.G., 2011. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38(5), 837–853.

Regis, R.G., Shoemaker, C.A., 2005. Constrained global optimization using radial basis functions. *Journal of Global Optimization*, 31, 153–171.

Regis, R.G., Shoemaker, C.A., 2007a. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4), 497–509.

Regis, R.G., Shoemaker, C.A., 2007b. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37(1), 113–135.

Santana-Quintero, L.V., Hernandez-Diaz, A.G., Molina, J., Coello Coello, C.A., Caballero, R., 2010. DEMORS: a hybrid multi-objective optimization algorithm using differential evolution and rough sets for constrained problems. *Computers & Operations Research*, 37(3), 470–480.

Santana-Quintero, L.V., Montano, A.A., Coello Coello, C.A., 2010. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. *In*: Y. Tenne, C. Goh, eds. *Computational Intelligence in Expensive Optimization Problems*, Chapter 2, Berlin, Germany: Springer, 29–59.

Sasena, M.J., Papalambros, P., Goovaerts, P., 2002. Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. *Engineering Optimization*, 34(3), 263–278.

Sendin, J.O.H., Banga, J.R., Csendes, T., 2009. Extensions of a multistart clustering algorithm for constrained global optimization problems. *Industrial & Engineering Chemistry Research*, 48(6), 3014–3023.

Shahrokhi, A., Jahangirian, A., 2010. A surrogate assisted evolutionary optimization method with application to the transonic airfoil design. *Engineering Optimization*, 42(6), 497–515.

Shan, S., Wang, G., 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2), 219–241.

Shan, S., Wang, G.G., 2011. Metamodeling for high dimensional simulation-based design problems. *ASME Journal of Mechanical Design*, 132(5), 051009.

Shi, L., Rasheed, K., 2008. ASAGA: an adaptive surrogate-assisted genetic algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, 1049–1056.

Simpson, T.W., Mauery, T.M., Korte, J.J., Mistree, F., 2001. Kriging Metamodels for Global Approximation in Simulation-Based Multidisciplinary Design Optimization. *AIAA Journal*, 39(12), 2233–2241.

The Mathworks, Inc. 2009. *Matlab Optimization Toolbox: User's Guide, Version 4.* Natick, MA.

The Mathworks, Inc. 2010. *Matlab Global Optimization Toolbox: User's Guide.* Natick, MA.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelley, J., Marti, R., 2007. Scatter search and local NLP solvers: a multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3), 328–340.

Viana, F.A.C., Haftka, R.T., Watson, L.T., 2010. Why not run the efficient global optimization algorithm with multiple surrogates?. *51th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA 2010-3090, Orlando, USA.

Villemonteix, J., Vazquez, E., Walter, E., 2009. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4), 509–534.

Wang, G., Dong, Z., Aitchison, P., 2001. Adaptive Response Surface Method - A Global Optimization Scheme For Approximation-Based Design Problems. *Engineering Optimization*, 33(6), 707–733.

Wanner, E.F., Guimars, F.G., Takahashi, R.H., Saldanha, R.R., Fleming, P.J., 2005. Constraint quadratic approximation operator for treating equality constraints with genetic algorithms. *In*: *2005 IEEE Congress on Evolutionary Computation, CEC'2005, Vol. 3*, Edinburgh, Scotland: IEEE Press, 2255-2262.

Won, K.S., Ray, T., 2005. A framework for design optimization using surrogates. *Engineering Optimization*, 37(7), 685-703.

Yahyaie, F., Filizadeh, S., 2011. A surrogate-model based multi-modal optimization algorithm. *Engineering Optimization*, 43(7), 779–799.

Ye, Y., 1989. *SOLNP Users' Guide: a nonlinear optimization Program in MATLAB.*

7. Appendix

The appendix can be found in an online supplement that is available at http://people.sju.edu/∼rregis.