Metamodel-Based Optimization for Problems With Expensive Objective and Constraint Functions

Moslem Kazemi[1]
PhD Candidate
e-mail: moslemk@sfu.ca

G. Gary Wang
Associate Professor
Mem. ASME
e-mail: gary_wang@sfu.ca

School of Engineering Science,
Simon Fraser University,
Burnaby, BC, V5A 1S6, Canada

Shahryar Rahnamayan
Assistant Professor
Mem. ASME
Faculty of Engineering and Applied Science,
University of Ontario Institute of Technology,
Oshawa, ON, L1H 7K4, Canada
e-mail: shahryar.rahnamayan@uoit.ca

Kamal Gupta
Professor
School of Engineering Science,
Simon Fraser University,
Burnaby, BC, V5A 1S6, Canada
e-mail: kamal@cs.sfu.ca

Current metamodel-based design optimization methods rarely deal with problems of not only expensive objective functions but also expensive constraints. In this work, we propose a novel metamodel-based optimization method, which aims directly at reducing the number of evaluations for both objective function and constraints. The proposed method builds on existing mode pursuing sampling method and incorporates two intriguing strategies: (1) generating more sample points in the neighborhood of the promising regions, and (2) biasing the generation of sample points toward feasible regions determined by the constraints. The former is attained by a discriminative sampling strategy, which systematically generates more sample points in the neighborhood of the promising regions while statistically covering the entire space, and the latter is fulfilled by utilizing the information adaptively obtained about the constraints. As verified through a number of test benchmarks and design problems, the above two coupled strategies result in significantly low number of objective function evaluations and constraint checks and demonstrate superior performance compared with similar methods in the literature. To the best of our knowledge, this is the first metamodel-based global optimization method, which directly aims at reducing the number of evaluations for both objective function and constraints.
[DOI: 10.1115/1.4003035]

[1]Corresponding author.
Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received April 30, 2010; final manuscript received November 4, 2010; published online January 10, 2011. Assoc. Editor: Barnard Yannou.

1 Introduction

Due to the wide use of computation intensive tools in engineering design, metamodeling has become a popular approach in recent years [1,2]. Most researchers use metamodels as a surrogate during the expensive optimizations, directly or adaptively, for the reduction of computational costs. In these studies, the objective function of an optimization problem is assumed to be expensive. The constraints, either box or more complicated linear or nonlinear constraints, are usually assumed to be cheap (Refs. [3,4], to name a few).

In the field of metamodeling-based design optimization, there is little work on expensive constraints. Sasena et al. [5] studied constraint handling in the context of their efficient global optimization (EGO) algorithm. The authors compared the penalty method and the one that multiplies the expected improvement (of the objective) by the probability of the point being feasible. It was found that these two methods have distinct merits, depending on how strictly the constraints need to be satisfied. They tried to avoid sampling in infeasible areas, which indirectly reduces the computational costs for constraints. Yannou et al. [6,7] and Moghaddam et al. [8] used constraint programming with the assistance of metamodeling to reduce the search space. In these works, constraints are either expensive or inexpensive; the goal is to bring the optimization into a confined smaller space. Constraint programming, though promising, needs careful tuning and brings extra difficulties to the designer [9]. In general, the lack of study on expensive constraints is perhaps due to the following reasons. First, it is found that if the constraints are also approximated by surrogates, the obtained constrained optimum, which often rests on the boundary of the feasible space, may be quite far from the actual optimum because of the approximation errors in both the objective function and the constraints [10]. While there are still challenges to build an accurate surrogate for the objective, the constraints are then assumed inexpensive for convenience as well as for a better focus. Second, most of the time, researchers overlooked the challenge of expensive constraints, as we did before.

In general, for constrained optimization, there are some classic methods such as Lagrange multipliers, quadratic programming, steepest descent method, and penalty methods [11]. When both objective function and constraints are black-box, many of these methods are not applicable. Coello [12] gave a comprehensive review of constraint handling techniques in evolutionary computation in which the functions are also black-box. Besides, many algorithm-specific methods such as various chromosome representations and operators, the penalty methods are of special interests. In Ref. [12], the author reviewed six types of penalty methods, i.e., static penalty, dynamic penalty, annealing penalty, adaptive penalty, co-evolutionary penalty, and death penalty. A recent book [13] was also devoted to constraint handling for evolutionary optimization.

The present work was motivated from the application of the mode pursing sampling (MPS) method, a global optimization method originally developed in Ref. [14] for continuous variable optimization problems, which later on was extended to solve mixed-type variable problems [15]. Through testing its performance [16], it is found that MPS took an excessive amount of processing time for constraint handling, even for inexpensive constraints. This, in fact, brings down its performance for relatively high dimensional black-box problems ($n>10$; n is the number of design variables). Later on, the MPS method was applied for crashworthiness optimization where constraints were expensive; and the need for a technique to handle expensive constraint arose. This work thus aims at developing a constraint handling approach for optimization problems, involving both expensive objective function and constraints. As discussed before, using surrogates for both types of functions in optimization could yield erroneous results due to the metamodeling errors. New techniques are therefore needed. This work is based on the framework of the MPS method, which does not rely on accurate metamodels but rather on the use of metamodels as a sampling guide. The preliminary results obtained using our proposed approach has been presented in Ref. [17].

In Sec. 2, MPS will be briefly reviewed and its constraint handling strategy is explained. The proposed approach will be described in Sec. 3. Experimental verifications and comparison analysis will be presented in Sec. 4, and finally, the work will be concluded in Sec. 5.

2 Mode Pursuing Sampling Method: Review and Issues

The mode pursuing sampling method [14] integrates the technique of metamodeling and a novel discriminative sampling method proposed in Ref. [18] in an iterative scheme for global optimization of problems with black-box functions. It generates more sample points (candidate solutions) in the neighborhood of the function mode and fewer in other areas as guided by a special sampling guidance function. Moreover, by continuous sampling in the global search space, it avoids trapping in local minima.

Algorithm 1: MPS

input: $f(x)$ black-box function; $\mathcal{G}_f=\{g_k(x)\leq 0 \mid k=1,\ldots,K\}$ set of constraints; $\mathcal{D}_f\subset\mathbb{R}^n$ problem domain

output: $x_{\min}$ global minimum of $f(x)$, or NULL in case of failure

```
1  begin
2    X ← SampleWithConstraints (m);
3    V ← Evaluate (X, f);
4    iter = 1;
5    while iter ≤ MAX_ITERATION do
6      f̂ ← LinearSpline (X, V);
7      h ← c_0 − f̂
8      X_N ← SampleWithConstraints (N);
9      V_N ← Evaluate (X_N, h);
10     x_mod ← Mode (X_N, V_N);
11     X_m ← SampleTowardMode (m, X_N, V_N);
12     V_m ← Evaluate (X_m, f);
13     X ← X ∪ X_m;
14     V ← V ∪ V_m;
15     q ← (n+1)(n+2)/2+1;
16     [X_q, V_q] ← NeighborSamples (q, x_mod, X, V);
17     if QuadraticRegression (X_q, V_q) is accurate then
18       [x_min, v_min] ← DoLocalOptimization, (f, x_mod);
19       if x_min ∈ HyperCube (X_q) then
20         Return x_min
21       end
22     end
23     X ← X ∪ x_min;
24     V ← V ∪ v_min;
25     iter = iter + 1;
26   end
27   Return NULL
28 end
```

The MPS method for minimizing a black-box function $f:\mathbb{R}^n\rightarrow\mathbb{R}$ is given as a pseudocode in Algorithm 1. It takes the set of constraints $\mathcal{G}_f=\{g_k(x)\leq 0 \mid k=1,\ldots,K\}$ and the problem domain $\mathcal{D}_f\subset\mathbb{R}^n$ as inputs and returns the global minimum of $f(x)$ as output in case of success. The algorithm can be summarized in four steps as follows:

Step 1. (Initial sampling, lines 2 and 3): A set of m sample points $X=\{x_i\in\mathcal{D}_f \mid g_k(x_i)\leq 0, k=1,\ldots,K; i=1,\ldots,m\}$ is generated randomly at line 2 using function SampleWithConstraints () in the feasible region of the problem domain $D_f\subset\mathbb{R}^n$ where m is an arbitrary integer that usually increases as the dimension of the problem domain n increases. These m points are called *expensive* points since their function values are evaluated by the black-box function $f(x)$ at line 3.

Step 2. (Function approximation, lines 6–9): A piecewise-linear spline $\hat{f}(x)=\sum_{i=1}^{m}\alpha_i\|x-x_i\|$ is then fitted (line 6) to the set of expensive points X as a metamodel of the actual black-box function such that $\hat{f}(x_i)=f(x_i)$ for $i=1,\ldots,m$, with constant α_i. Then a sampling guidance function $h(x)=c_0-\hat{f}(x)$ is defined (line 7) where c_0 is a constant such that $c_0>\max(\hat{f}(x))$. The guidance function can be viewed as a probability density function (up to a normalizing constant) whose modes are located at those x_i's where the function values are the lowest among $f(x_i)$'s. N (usually large) number of valid sample points are then randomly generated within the feasible region of the problem domain, again by calling function SampleWithConstraints () at line 8. These points are called *cheap* points since their function values are evaluated by the linear guidance function $h(x)$, not the objective function, hence, their function values will be referred as *approximation* values.[2]

Step 3. (Mode pursuing sampling, lines 10–12): A discriminative sampling technique (see [18]) is then employed (function SampleTowardMode () at line 11) to draw another set of m sample points from the set of the cheap points obtained in step 2 according to $h(x)$ (please see [14] for implementation details). By construction, these sample points have the tendency to concentrate about the maximum (or mode) of $h(x)$, which corresponds to the minimum of $\hat{f}(x)$.

Step 4. (Quadratic regression and local optimization, lines 15–22): The fourth step involves a quadratic regression in a subarea around the current minimum of $\hat{f}(x)$ (or mode of $h(x)$) according to the discriminative sampling in step 3. If the approximation in the subarea is sufficiently accurate, local optimization is performed in this subarea to obtain the minimum $x_{\min}$. The $x_{\min}$ is returned as the global minimum of $f(x)$ if it is located inside the identified subarea around the mode of $h(x)$. Otherwise, it is added to the set of expensive points and the algorithm restarts from step 2.

In short, the MPS is an algorithm, which uses discriminative sampling as its engine and has an intelligent mechanism to use the information from past iterations to lead the search toward the global optimal. At each iteration of the MPS, two types of approximations are used: (1) approximation of the entire function by fitting the metamodel (given as a linear spline) to all expensive points (line 6) and (2) quadratic regression around the attractive subregions (line 17). The first approximation uses a piecewise-linear spline as the metamodel because of its simplicity. One should note that the accuracy of the metamodel is not very critical here compared to the cases where metamodels are used as surrogates since it is only used to guide the search toward the function mode. Nonetheless, the MPS method does not dictate the exclusive use of the linear functions, and other types of metamodels can be applied in lieu of the linear model. The accuracy of the quadratic regression (second approximation) around the attractive areas is increased at each iteration due to the discriminative sampling, which generates more and more sample points around attractive regions.

2.1 Limitations of MPS for Constrained Optimization Problems. Several simulations and design examples have shown the effectiveness, robustness, and applicability of the MPS method

[2]Throughout this paper, *cheap* samples refer to those points evaluated by a spline approximation of the objective function, while *expensive* samples denote the points evaluated by the objective/constraint function itself, which is usually more time consuming.

in both continuous [14] and discontinuous [15] optimization. Some of the limitations of the MPS method were discussed in Ref. [14] and a study was conducted in Ref. [16] comparing the performance of the MPS with other global optimization methods such as the genetic algorithms in solving global optimization of both expensive and inexpensive objective functions.

Similar to many algorithms in this field, one of the main issues, which unfortunately has been overlooked is the performance of the MPS method in the presence of *expensive* constraints. In the examples provided in the early works mentioned above, the cost of checking constraints when generating random sample points (specifically in function SampleWithConstraints () called at line 8 of Algorithm 1) is not considered in the total cost of the optimization. Due to the large number of constraint checks that the MPS method relies on, this cost could be a determinant factor for constrained optimization problems especially when the design problem consists of expensive constraints. This can be explained by taking a closer look at the strategy that MPS algorithm utilizes to generate sample points, particularly in function SampleWithConstraints () where a large number of *cheap* random points need to be generated within the *feasible* region of the problem domain. To ensure that each sample point falls into the feasible region of the problem domain, it is checked against the set of all the constraints. If the sample point satisfies all the constraints, then it is added to the set of valid samples. Otherwise, it is discarded and a new sample is randomly generated and checked for the constraints. The generation of random samples continues based on the above scheme until the required numbers of valid sample points are obtained.

Apparently, in constrained optimization problems, the above strategy might result in a large number of constraint checks considering the relative size of the forbidden and feasible regions. Moreover, in the above strategy, the invalid samples are discarded and the information obtained through the constraint check is not used in the overall MPS optimization algorithm.

To overcome the above shortcomings of the MPS technique, we present a novel sampling technique, which systematically biases the generation of sample points toward feasible regions of the problem domain using the information that is incrementally obtained about the constraints, hence, the name *constraint-importance mode pursuing sampling* (CiMPS). The proposed CiMPS strategy is explained in more details in the next section.

3 Constraint-Importance Mode Pursuing Sampling (CiMPS)

The crux of our proposed CiMPS sampling method is to utilize the information obtained about the constraints to bias the generation of samples toward feasible region of the problem domain, hence, resulting in a more efficient sampling of the space and substantially less number of constraint checks compared to the MPS method proposed in Ref. [14]. It is worthwhile to mention that the CiMPS technique still benefits from the advantages of original MPS technique by biasing the samples toward the mode of the objective function and, hence, the number of objective function evaluation is kept relatively low due to fast convergence of the optimization process toward the mode of the function. Thus, in summary, the CiMPS method provides efficient sampling of the problem domain by accounting for the information incrementally obtained on both the objective function and constraints. This strategy is shown to result in a substantially low number of both function evaluations and constraint checks as we explain next.

Algorithm 2: CiMPS
input: $f(x)$ black-box function; $\mathcal{G}_f=\{g_k(x)\leq 0|k=1,\ldots,K\}$ set of constraints; $\mathcal{D}_f\subset\mathbb{R}^n$ problem domain
output: $x_{\min}$ global minimum of $f(x)$, or NULL in case of failure
1 **begin**
2 $X\leftarrow$ SampleWithConstraints (m);
3 $V\leftarrow$ Evaluate (X,f);
4 iter=1;
5 **while** iter $\leq$ MAX_ITERATION **do**
6 $\hat{f}\leftarrow$ LinearSpline (X,V);
7 $h\leftarrow c_0-\hat{f}$
8 $X_N\leftarrow$ Sample (N);
9 $V_N\leftarrow$ Evaluate (X_N,h);
10 $x_{\text{mod}}\leftarrow$ Mode (X_N,V_N);
11 $X_m\leftarrow$ SampleTowardMode (m,X_N,V_N);
12 $v_{\text{penalty}}\leftarrow$ Max (V);
13 $V_m\leftarrow$ EvaluateWithConstraints (X_m,v_{penalty});
14 $X\leftarrow X\cup X_m$;
15 $V\leftarrow V\cup V_m$;
16 $q\leftarrow(n+1)(n+2)/2+1$;
17 $[X_q,V_q]\leftarrow$ NeighborSamples (q,x_{mod},X,V);
18 **if** QuadraticRegression (X_q,V_q) *is accurate* **then**
19 $[x_{\min},v_{\min}]\leftarrow$ DoLocalOptimization (f,x_{mod});
20 **if** $x_{\min}\in$ HyperCube (X_q) **then**
21 Return $x_{\min}$
22 **end**
23 **end**
24 $X\leftarrow X\cup x_{\min}$;
25 $V\leftarrow V\cup v_{\min}$;
26 iter=iter+1;
27 **end**
28 Return NULL
29 **end**

The CiMPS algorithm (see Algorithm 2) generally follows the four steps similar to the MPS algorithm explained in Sec. 2. To bias the sample points toward the mode of the objective function, the sampling technique proposed in Ref. [18] is employed as the core of the discriminative sampling in CiMPS method (via function SampleTowardMode () at line 11) where a set of m sample points is systematically selected from a large number of cheap points biased toward the function mode according to their guidance function values.

3.1 Relaxing Constraint Checks. As we mentioned earlier in Sec. 2, in the MPS method a large number of feasible cheap sample points are generated at line 2 by calling function SampleWithConstraints () in which each sample point is checked against all the constraints and in case of no constraint violation, it is added to the set of cheap sample points. In the CiMPS algorithm, this condition is relaxed at line 8 by calling function Sample () in which the cheap sample points are generated randomly within the problem domain without being checked against the constraints. Hence, some of the samples might fall into forbidden regions defined by the constraints. Please note that the objective function may be undefined in these regions; however, the cheap sample points are supposed to be evaluated by the guidance function $h(x)$ (line 9, Algorithm 2), which is defined everywhere in the problem domain even in the forbidden regions. Nonetheless, the function values obtained for these infeasible samples may not properly represent the underlying objective function. If they are not treated appropriately they would result in improper sampling of the forbidden regions and eventually yields an invalid global minimum. As we see in the next section, in our proposed sampling technique, the information obtained through these invalid samples are utilized to bias the sampling away from the constraints and toward the feasible regions of the problem domain.

The set of cheap sample points generated as explained above is then sampled by function SampleTowardMode () at line 11 using the sampling technique in Ref. [18] to obtain a set of *m expensive* sample points, which are biased toward the mode of the guidance function. This step follows the implementation of the MPS technique proposed in Ref. [14].

3.2 Constraint-Importance Sampling. The expensive sample points obtained using function SampleTowardMode () are used next to perform a quadratic regression (line 18, algorithm 2) around the function mode, and later are added to the set of expensive sample points (lines 24 and 25) for further improvement of the spline approximation of the black-box function at line 6. Therefore, to provide more accurate approximations, these samples should be evaluated by the objective function to obtain their exact function values. However, as mentioned above, some of these samples might fall into forbidden regions where the objective function may be undefined.

Algorithm 3: EvaluateWithConstraints (X, v_{penalty})
input: X, set of sample points to be evaluated; v_{penalty}, the penalty value of infeasible samples;
output: V, value functions of points in X, i.e. $V=\{v \mid v = f(x), \forall x \in X\}$

1 **begin**
2 $V=\{\}$;
3 **foreach** *x in the set X* **do**
4 **if** *x satisfies all constraints* **then** $v \leftarrow f(x)$;
5 **else**
6 **if** *variable penalty is selected* **then** $v \leftarrow v_{\text{penalty}}$;
7 **else** $v \leftarrow$ USER_DEFINED_PENALTY;
8 **end**
9 $V \leftarrow V \cup \{v\}$;
10 **end**
11 Return V;
12 **end**

Therefore, these sample points are especially treated at line 13 by calling function EvaluateWithConstraints () in which each sample is first checked against all the constraints (see Algorithm 3). If a sample point satisfies all constraints, then its actual function value is evaluated by the objective function. Otherwise, an appropriate penalty value is assigned as its function value (lines 6–7, Algorithm 3), without calling the expensive objective function. Two schemes are proposed for selecting the penalty assigned to invalid sample points: *static* or *dynamic* penalty, which are described in Secs. 3.2.1 and 3.2.2.

3.2.1 Static Penalty Scheme. If the user has some information regarding the maximum value (for a minimization problem) that the objective function can achieve over the problem domain, then a value equal to or greater than the maximum can be selected as a (fixed) user defined penalty and is returned as the function value of infeasible samples (line 7, Algorithm 3). This information can be obtained by the user at the beginning by examining the objective function.

3.2.2 Dynamic Penalty Scheme. The second scheme modifies the penalty value as follows. At each iteration, the penalty value would be set equal to or greater than the maximum function value (for a minimization problem) of the valid samples, which have been already identified up to the current iteration (line 12, Algorithm 2). This value v_{penalty} is passed to the function EvaluateWithConstraints, where it is returned as the function value of infeasible samples (line 6, Algorithm 3). This approach is more practical and less demanding since usually global information about the black-box function is not available beforehand and the fixed penalty scheme may not be applicable.

Interestingly, assigning an appropriate penalty obtained using either of above two schemes imposes a high function value to the approximated objective function $\hat{f}(x)$ in forbidden regions represented by infeasible samples. This, in turn, results in low values for the guidance function $h(x)$ in forbidden region. Hence, the generation of sample points in function SampleTowardMode () will be biased away from the forbidden regions because, according to the sampling technique in Ref. [18], function SampleTowardMode generates more sample points in the neighborhood of the function mode (regions with better fitness values) and generates less sample points in areas with worse fitness.

4 Performance Evaluations and Comparisons

Intensive experiments and performance comparisons have been performed to evaluate the performance of the proposed CiMPS method on constrained benchmark and design problems. The results are presented in Secs. 4.1 and 4.2. Please note that in all of the following experiments, the termination criterion for the CiMPS method (and similarly for the MPS technique) is determined by checking if the current local minimum falls in the hypercube defined by the neighbor samples around the current mode of the set of sample points (lines 17–23, Algorithm 2).

4.1 Constrained Benchmark Problems and Results. The performance of the CiMPS method has been compared with the original MPS method on seven well-known constrained optimization problems selected from a suite of test benchmarks compiled in Ref. [19]. The specifications of these problems (and their original references) along with our results have been summarized in Table 1.

For each of these problems, 30 runs have been carried out using both the MPS and CiMPS methods and their performances are compared based on two cost indicators: (1) number of objective function evaluations (nfe) and (2) number of constraint checks (ncc). Both CiMPS and MPS methods applied to each problem share the same run settings as given in Table 1. For a detailed explanation of these settings and their effects, one can refer to Ref. [14].

The simulation results obtained using the MPS and CiMPS methods for above problems are summarized in Table 1. As shown, for all test examples, the number of constraint checks (ncc) by the CiMPS method is significantly lower compared with the corresponding number of constraint checks by the MPS method. Moreover, for all benchmarks, the CiMPS performs better in terms of number of function evaluations (nfe) as well. This is because that the optimization process is directed away from infeasible regions and converges earlier than the MPS.

It is also noteworthy that the optimization of problems with equality constraints (e.g., g03, g05, g11, and g13 in Ref. [19]) is a quite challenging and time consuming task for sampling-based optimization techniques including MPS and its extended version CiMPS. This is due to the fact that the probability of generating feasible random samples which guarantee the equality of constraints is quite low. Hence, for such problems sampling-based techniques fail to generate the required number of feasible samples in a timely manner. These techniques also yield a large number of infeasible samples for problems with tightly constrained domains whose optimum solutions are located in narrow feasible regions. This can be observed from the results obtained for the problems g06, g07, g09, and g10 (see Table 1) for which a relatively large number of constraint checks has been performed to obtain the required number of feasible samples.

4.2 Constrained Design Problems. Two well-known engineering design problems are used to evaluate the performance of the proposed method: (1) minimization of weight of the spring and (2) pressure vessel design. Both are constrained optimization problems consisting of several expensive constraints. Each problem is described with its corresponding constraints, bounds, and objective function in Secs. 4.2.1 and 4.2.2.

4.2.1 Minimization of Weight of the Spring. This problem has been used as a test benchmark in literature, e.g., Refs. [11,24–26]. In its standard form [11], it consists of designing a tension/compression spring (see Fig. 1) to carry a given axial load.

Table 1 Simulation results obtained for the test benchmarks using both the MPS and CiMPS methods (30 runs for each algorithm per problem); problem specifications: *n* is the number of variables and *K* is the number of constrains; run settings [14]: *N* is the number of cheap points generated at each iteration, *l* is the number of contours by which the cheap points are grouped, and *m* is number of expensive sample points generated at each iteration; results: nfe is the number of function evaluations and ncc is the number of constraint checks

								Optimum found		nfe		ncc	
$f(x)$	n	K	Known optimum	N	m	l	Method	Best	Mean	Mean	Median	Mean	Median
g01 [20]	13	9	−15	50	10	5	MPS	−14.209	−13.780	219	218	26,712	25,891
							CiMPS	−14.587	−14.212	203	202	17,181	16,919
g04 [21]	5	6	−30665.539	50	10	5	MPS	−30665.539	−30665.539	75	74	635	617
							CiMPS	−30665.539	−30665.539	60	59	266	266
g06 [20]	2	2	−6961.81388	50	10	5	MPS	−6961.81388	−6961.81388	30	29	405,740	386,985
							CiMPS	−6961.81388	−6961.81388	15	14	29,706	29,661
g07 [22]	10	8	24.30621	50	10	5	MPS	24.30621	24.30621	136	130	187,217	172,872
							CiMPS	24.30621	24.30621	126	126	107,135	106,535
g08 [23]	2	2	0.095825	100	10	10	MPS	0.095825	0.095825	115	107	41,630	27,216
							CiMPS	0.095825	0.095825	103	103	11,811	11,485
g09 [22]	7	4	680.630057	100	10	10	MPS	680.630057	680.630057	142	133	82,695	67,616
							CiMPS	680.630057	680.630057	113	110	18,803	18,794
g10 [22]	8	6	7049.3307	50	5	10	MPS	7049.2480	7049.2480	72	72	592,518	584,525
							CiMPS	7049.2480	7049.2480	56	56	194,860	202,131

The objective is to minimize the weight of the spring f_{WS} as

$$f_{WS}(x_1,x_2,x_3) = (x_3+2)x_2x_1^2 \tag{1}$$

where $x_1=d$ is the wire diameter (inch), $x_2=D$ is the mean coil diameter (inch), and $x_3=N$ is the number of active coils, subject to the following four constraints:

$$g_1(x) = 1.0 - \frac{x_2^3x_3}{71875x_1^4} \le 0$$

$$g_2(x) = \frac{x_2(4x_2-x_1)}{12566x_1^3(x_2-x_1)} + \frac{2.46}{12566x_1^2} - 1.0 \le 0$$

$$g_3(x) = 1.0 - \frac{140.54x_1}{x_2^2x_3} \le 0$$

Fig. 1 A tension/compression coil spring

$$g_4(x) = \frac{x_2+x_1}{1.5} - 1.0 \le 0 \tag{2}$$

and $0.05 \le x_1 \le 0.20$, $0.25 \le x_2 \le 1.30$, and $2 \le x_3 \le 15$.

Table 2 summarizes and compares the results obtained by applying both the MPS and CiMPS methods (30 runs for each algorithm). The number of cheap points generated at each iteration is $N=100$ and the number of contours used is five for all runs. As it can be seen, the CiMPS method results in significantly lower number of constraint checks (ncc) and also less number of function evaluations (nfe).

This problem has been solved by a number of researchers: by Arora [11] using a numerical technique called constraint correction at constant cost (CCC), by Coello and Mezura-Montes [26] using a Genetic Algorithm (GA)-based approach, by Mahdavi et al. [25] using an improved variation of harmony search (IHS) algorithm [27], and by Belegundu and Arora [24] using eight numerical techniques with the best solution obtained using M-4 method, which is a variation of a Lagrange multipliers code based on Powell's algorithm [28]. The above solutions are compared with the best solution found using our proposed CiMPS technique and the original MPS method in Table 2. As it can be seen, the solution obtained using our proposed CiMPS method (and the MPS) is better than the ones obtained by other techniques. More-

Table 2 Best solution obtained for the minimization of weight of the spring problem using the CiMPS method compared with the best solutions reported by other works (N/A: not available)

	Method					
Design	CiMPS (this work)	MPS [14]	CCC [11]	GA-based [26]	M-4 [24]	IHS [25]
x_1	0.05156	0.05154	0.053396	0.051989	0.0500	0.05115438
x_2	0.35363	0.35322	0.399180	0.363965	0.3176	0.34987116
x_3	11.47221	11.49692	9.185400	10.890522	14.027	12.0764321
$f_{WS}(x)$	0.012665	0.012664	0.012730	0.012681	0.01272	0.0126706
nfe	21	33	18	80,000	1605	30,000
ncc	1911	20,994	N/A	N/A	N/A	N/A

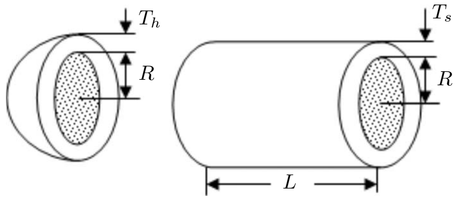

Fig. 2 A pressure vessel

over, other than the CCC technique presented in Ref. [11], the CiMPS maintains a superior performance in terms of number of function evaluations (nfe) compared with others. The CCC technique utilizes the gradient information of the objective function, which increases the convergence rate of the optimization process and, hence results in lower number of function evaluations. Unfortunately, other than in the MPS technique, the number of constraint checks has been neglected (or is not reported) in the above previous works and we are not able to provide a performance comparison based on that criterion. The cost of constraint checking has been paid almost no attention in optimization literature. To the best of our knowledge the proposed CiMPS approach is the first metamodel-based technique, which directly aims at reducing the number of constraint checks when it comes to optimization of problems with expensive black-box constraints.

4.2.2 Pressure Vessel Design. The second problem is to minimize the total cost, including the cost of material, forming and welding of a cylindrical vessel, which is capped at both ends by hemispherical heads as shown in Fig. 2. The total cost $f_{PV}(x)$ is given as

$$f_{PV}(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{3}$$

where $x_1 = T_s$ is the thickness of the shell, $x_2 = T_h$ is the thickness of the head, $x_3 = R$ is the inner radius, and $x_4 = L$ is the length of the cylindrical section, subject to the following six constraints:

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(x) = -x2 + 0.00954x_3 \leq 0$$

$$g_3(x) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1,296,000 \leq 0$$

$$g_4(x) = x_4 - 240 \leq 0$$

$$g_5(x) = 1.1 - x_1 \leq 0$$

$$g_6(x) = 0.6 - x_2 \leq 0 \tag{4}$$

with the bounds $1.0 \leq x_1 \leq 1.375$, $0.625 \leq x_2 \leq 1.0$, $25 \leq x_3 \leq 150$, and $25 \leq x_4 \leq 240$.

Table 3 summarizes and compares the results obtained by applying both the MPS and CiMPS methods (30 runs for each algorithm). The number of cheap points generated at each iteration is $N = 200$ and the number of contours used is 20 for all runs. As it is seen, the CiMPS method results in significantly lower number of constraint checks (ncc) and also less number of function evaluations (nfe).

This problem has been solved by Ref. [29] using a GA-based approach, by [30] using harmony search (HS) algorithm [27], by [25] using an improved harmony search (IHS) algorithm, and by [31] using branch and bound (BB) method. The best solutions obtained using the above techniques are compared with the best solutions found using our proposed CiMPS technique and the original MPS method in Table 3. As it can be seen, the solution obtained using our proposed CiMPS method (and MPS) is better than the ones obtained by other techniques. Unfortunately, none of the referred works have reported on the number of function evaluations, constraint checks, or iterations. These works are mostly focused on finding the best solutions and as our results show for this problem, both MPS and CiMPS techniques have yielded better solutions than the other mentioned techniques.

5 Conclusion Remarks

When we deal with expensive optimization problems, our focus of attention is mostly on objective function(s), and we assume that the constraints are inexpensive. That is why in the majority of literature on computer experiments, just the average number of function evolutions are measured and compared and the number of constraint evaluations is simply ignored. In real-world applications, unlike benchmark functions, expensive mixed equality and inequality constraints are commonly met. In this work, a new method called CiMPS is developed for optimization problems involving both expensive objective function and constraints. This method inherits characteristics of MPS and furthermore enhances MPS by steering the sampling away from infeasible regions, saving the number of evaluations for both constraints and the objective.

The performance of the CiMPS method was experimentally verified through two test suites, namely, seven constrained benchmark problems and two real design problems. The CiMPS and its parent algorithm (MPS) was compared on both test suites, also both competed with four other well-known optimization methods on design problems. The reported results clearly confirmed that the CiMPS outperforms the MPS. The CiMPS also yields better optimum for the two design problems with close to the least number of function evaluations compared with four well-established algorithms. It thus opens a promising direction to tackle with expensive constrained optimization problems.

Developing a mixed-variable version of CiMPS and enhancing it for large-scale problems are our directions for future work.

Table 3 Best solution obtained for the pressure vessel design problem using the CiMPS method compared with the best solutions reported by other works (N/A: not available)

	Method					
Design	CiMPS (this work)	MPS [14]	GA-based [29]	HS [30]	IHS [25]	BB [31]
x_1	1.10000	1.10000	1.125	1.125	1.125	1.125
x_2	0.625	0.625	0.625	0.625	0.625	0.625
x_3	56.99482	56.99482	58.1978	58.2789	58.29015	48.97
x_4	51.00125	51.00125	44.2930	43.7549	43.69268	106.72
$f_{PV}(x)$	7163.739	7163.739	7207.494	7198.433	7197.730	7980.894
nfe	37	62	N/A	N/A	N/A	N/A
ncc	335	4565	N/A	N/A	N/A	N/A

References

[1] Simpson, T., Peplinski, J., Koch, P., and Allen, J., 2001, "Metamodels for Computer-Based Engineering Design: Survey and Recommendations," Eng. Comput., **17**(2), pp. 129–150.

[2] Wang, G., and Shan, S., 2007, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," ASME J. Mech. Des., **129**(4), pp. 370–380.

[3] Schonlau, M., Welch, W. J., and Jones, D. R., 1998, "Global Versus Local Search in Constrained Optimization of Computer Models," Lecture Notes—Monograph Series, **34**, pp. 11–25.

[4] Regis, R. G., and Shoemaker, C. A., 2005, "Constrained Global Optimization of Expensive Black Box Functions Using Radial Basis Functions," J. Global Optim., **31**(1), pp. 153–171.

[5] Sasena, M. J., Papalambros, P., and Goovaerts, P., 2002, "Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization," Eng. Optimiz., **34**, pp. 263–278.

[6] Yannou, B., Simpson, T. W., and Barton, R., 2005, "Towards a Conceptual Design Explorer Using Metamodeling Approaches and Constraint Programming," ASME Paper No. DETC2003/DAC-48766.

[7] Yannou, B., Moreno, F., Thevenot, H., and Simpson, T., 2005, "Faster Generation of Feasible Design Points," ASME Paper No. DETC2005/DAC-85449.

[8] Moghaddam, R., Wang, G., Yannou, B., and Wu, C., 2006, "Applying Constraint Programming for Design Space Reduction in Metamodeling Based Optimization," 16th International Institution for Production Engineering Research (CIRP) International Design Seminar, Paper No. 10081.

[9] Yannou, B., and Harmel, G., 2006, "Use of Constraint Programming for Design," *Advances in Design*, Springer, London, UK, pp. 145–157.

[10] Wang, G., 2003, "Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points," ASME J. Mech. Des., **125**(2), pp. 210–220.

[11] Arora, J., 2004, *Introduction to Optimum Design*, Elsevier Academic, New York.

[12] Coello, C., 2002, "Theoretical and Numerical Constraint-Handling Techniques Used With Evolutionary Algorithms: A Survey of the State of the Art," Comput. Methods Appl. Mech. Eng., **191**(11–12), pp. 1245–1287.

[13] 2009, *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montes, ed., Springer, New York.

[14] Wang, L., Shan, S., and Wang, G., 2004, "Mode-Pursuing Sampling Method for Global Optimization of Expensive Black-Box Functions," Eng. Optimiz., **36**(4), pp. 419–438.

[15] Sharif, B., Wang, G., and ElMekkawy, T., 2008, "Mode Pursuing Sampling Method for Discrete Variable Optimization on Expensive Black-Box Functions," ASME J. Mech. Des., **130**(2), p. 021402.

[16] Duan, X., Wang, G., Kang, X., Niu, Q., Naterer, G., and Peng, Q., 2009, "Performance Study of Mode-Pursuing Sampling Method," Eng. Optimiz., **41**(1), pp. 1–21.

[17] Kazemi, M., Wang, G. G., Rahnamayan, S., and Gupta, K., 2010, "Constraint Importance Mode Pursuing Sampling for Continuous Global Optimization," ASME Paper No. DETC2010-28355.

[18] Fu, J., and Wang, L., 2002, "A Random-Discretization Based Monte Carlo Sampling Method and Its Applications," Methodol. Comput. Appl. Probab., **4**(1), pp. 5–25.

[19] Runarsson, T. P., and Yao, X., 2000, "Stochastic Ranking for Constrained Evolutionary Optimization," IEEE Trans. Evol. Comput., **4**(3), pp. 284–294.

[20] Floudas, C. A., and Pardalos, P. M., 1990, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, New York.

[21] Himmelblau, D. M., 1972, *Applied Nonlinear Programming*, McGraw-Hill, New York.

[22] Hock, W., and Schittkowski, K., 1981, *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Secaucus, NJ.

[23] Koziel, S., and Michalewicz, Z., 1999, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," Evol. Comput., **7**(1), pp. 19–44.

[24] Belegundu, A. D., and Arora, J. S., 1985, "A Study of Mathematical Programming Methods for Structural Optimization. Part II: Numerical Results," Int. J. Numer. Methods Eng., **21**(9), pp. 1601–1623.

[25] Mahdavi, M., Fesanghary, M., and Damangir, E., 2007, "An Improved Harmony Search Algorithm for Solving Optimization Problems," Appl. Math. Comput., **188**(2), pp. 1567–1579.

[26] Coello, C. A. C., and Mezura-Montes, E., 2002, "Constraint-Handling in Genetic Algorithms Through the Use of Dominance-Based Tournament Selection," Adv. Eng. Inf., **16**(3), pp. 193–203.

[27] Geem, Z., Kim, J., and Loganathan, G., 2001, "A New Heuristic Optimization Algorithm: Harmony Search," Simulation, **76**(2), pp. 60–68.

[28] Powell, M., 1978, "Algorithms for Nonlinear Constraints That Use Lagrangian Functions," Math. Program., **14**(1), pp. 224–248.

[29] Wu, S., and Chow, P., 1995, "Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems via Meta-Genetic Parameter Optimization," Eng. Optimiz., **24**(2), pp. 137–159.

[30] Lee, K., and Geem, Z., 2005, "A New Meta-Heuristic Algorithm for Continues Engineering Optimization: Harmony Search Theory and Practice," Comput. Methods Appl. Mech. Eng., **194**(36–38), pp. 3902–3933.

[31] Sandgren, E., 1990, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization," ASME J. Mech. Des., **112**(2), pp. 223–229.