Mode-Pursuing Sampling Method for Global Optimization on Expensive Black-box Functions

Liqun Wang[1] Songqing Shan[2] G. Gary Wang[2*]

Abstract

The presence of black-box functions in engineering design, which are usually computation-intensive, demands efficient global optimization methods. This paper proposes a new global optimization method for black-box functions. The global optimization method is based on a novel mode-pursuing sampling (MPS) method which systematically generates more sample points in the neighbourhood of the function mode while statistically covering the entire search space. Quadratic regression is performed to detect the region containing the global optimum. The sampling and detection process iterates until the global optimum is obtained. Through intensive testing, this method is found to be effective, efficient, robust, and applicable to both continuous and discontinuous functions. It supports simultaneous computation and applies to both unconstrained and constrained optimization problems. Because it does not call any existing global optimization tool, it can be used as a standalone global optimization method for inexpensive problems as well. Limitations of the method are also identified and discussed.

Keywords: Computation-intensive design; Simultaneous computation; Constrained Optimization; Black-box function, Discretization-based sampling; Local quadratic fitting.

Introduction

In today's engineering design, as computer modeling capabilities increase dramatically, product behaviours are modeled and analyzed using finite element analysis (FEA) and computational fluid dynamics (CFD) techniques. These analysis and simulation processes are usually computationally expensive. Design optimization based on these computation-intensive processes is challenging in several respects.

[1] Dept. of Statistics, The University of Manitoba, Winnipeg, MB, Canada, R3T 2N2
[2] Dept. of Mech. & Indus. Engineering, The University of Manitoba, Winnipeg, MB, Canada, R3T 5V6
[*] Corresponding author, Tel: 204-474-9463 Fax: 204-275-7507, Email: gary_wang@umanitoba.ca

- Firstly, the overall optimization time should be acceptable to an increasingly impatient manufacturing industry. The optimization time relates to two issues, the total number of expensive function evaluations and the amount of simultaneous computation. The term *simultaneous computation* is carefully chosen to be distinct from *parallel computation*. S*imultaneous computation* means the possibility of having multiple simultaneous computing processes, not necessarily involving interactions between these computing processes, which is the characteristic of *parallel computation*.
- Secondly, for FEA or CFD, an explicit expression of optimization objective and constraint functions with respect to design variables is not available. Also, gradients computed from these processes are usually unreliable or expensive [1]. To a design engineer, these processes are like a black-box, i.e., only the input and output can be obtained without *a priori* knowledge about the function.
- Thirdly, a global design optimum is always preferred to a local optimum if the computation cost is acceptable.

Numerous global optimization methods can be found in the literature. A recent survey is given in Ref. [2]. However, few of them are suitable for the above expensive black-box function problems. Current global optimization methods can be classified into two groups: deterministic and stochastic. Deterministic methods require *a priori* knowledge of the objective function, e.g., its shape, expression, gradient, and so on. Thus, they are not directly applicable to black-box functions. Most stochastic methods, such as simulated annealing, genetic algorithms, tabu search, multistart (including clustering), and many others, require a large number of function evaluations even for a simple 2-dimensional design problem, though they do not demand *a priori* knowledge of the objective function. In addition, most of these methods are not developed to maximize the amount of simultaneous computation. Therefore, most existing stochastic methods are not efficient in saving the total optimization time for expensive functions.

In the emerging area of metamodeling-based optimization, methods have been developed to address the computational efficiency problem. These methods are based on the idea of sampling in the design space, building approximation models from these sampling points, and then performing optimization on the approximation function. Research focuses on developing better sampling strategies, approximation models, or methods dealing with the sampling, modeling, and

optimization as a whole. Detailed surveys on research in this direction can be found in the third author's previous work [3; 4]. Metamodeling-based optimization methods entail many attractive ideas for optimizing expensive black-box problems, such as the idea of sampling and approximation. However, in general these methods rely on the structure of the approximation model. Furthermore, the choice of the best approximation model is problem dependent. So far, few methods have been developed for global optimization. One successful development is in Refs. [5; 6], where the authors apply a Bayesian method to estimate a kriging model, and to gradually identify points in the space to update the model and perform the optimization. Their method, however, presumes a continuous objective function and a correlation structure between sample points. Identification of a new point requires a complicated optimization process; moreover, construction of the kriging model usually requires a global optimization process.

The third author of this paper and his colleagues have developed a number of global optimization strategies for expensive black-box functions [3; 4; 7]. These methods focus on strategies to gradually reduce the search space. In the reduced final region, an existing global optimization method is applied on the approximation model to locate the optimum. In this paper, a new global optimization method for expensive black-box functions is proposed, assuming the design space cannot be confidently reduced. In contrast to the methods in Ref. [5; 6], this method does not assume any properties of the black-box function; it works for both continuous and discontinuous functions; and it does not call any existing global optimization process or tool in its optimization process.

Before the global optimization strategy is discussed, a new mode-pursuing sampling method is introduced, as it forms the core of the proposed global optimization method.

Mode-Pursuing Sampling Method (MPS)

In this section we introduce the so-called mode-pursuing sampling algorithm. It is an extension of the random-discretization based sampling method of Fu and Wang [8], which is a general-purpose algorithm to draw a random sample from any given multivariate probability distribution. This algorithm only requires knowledge of the probability density function up to a normalizing constant. This sampling method has been successfully implemented in many high-dimensional random sampling and numerical integration problems. This section will first describe the

proposed MPS algorithm, which will be elaborated and explained with a sampling example. Then, the properties of the MPS method will be given and proved.

Algorithm of the MPS Method

Suppose we are given a d-dimensional probability density function $g(x)$ with compact support $S(g) \subset \Re^d$. Fu and Wang's algorithm [8] consists of three steps. In the first step, the discretization step, a discrete space $S_N(g)$ is generated consisting of N uniformly distributed base points in $S(g)$. Usually N is large and should be larger if the dimension of $g(x)$, d, is higher. These uniform base points may be generated using either deterministic or stochastic procedures. In the second step, the contourization step, the base points of $S_N(g)$ are grouped into K contours $\{E_1, E_2, \ldots, E_K\}$ with equal size according to the relative height of the function $g(x)$. For example, the first contour E_1 contains the $[N/K]$ points having the highest function values among all base points, whereas the last contour E_K contains the $[N/K]$ points having the lowest function values. Also in this step, a discrete distribution $\{P_1, P_2, \ldots, P_K\}$ over the K contours is constructed, which is proportional to the average function values of the contours. Finally, a sample is drawn from the set of all base points $S_N(g)$ according to the discrete distribution $\{P_1, P_2, \ldots, P_K\}$ and the discrete uniform distribution within each contour. As has been shown in Fu and Wang [8], the sample drawn according to their algorithm is independent and has an asymptotic distribution $g(x)$. The approximation gets better for larger values of N and K.

In this paper, we incorporate Fu and Wang's algorithm as a component of our new sampling method for optimization problems. Following the convention of engineering optimization, we refer to the minimum as the function mode.

Concretely, we wish to minimize an n-dimensional black-box function $f(x)$ over a compact set $S(f) \subset \Re^n$. To simplify notation, assume that $S(f) = [a, b]^n$, where $-\infty < a < b < \infty$ are known, and $f(x)$ is positive on $S(f)$ and continuous in a neighbourhood of the global minimum. In general, if $f(x)$ is negative for some $x \in S(f)$, then we can always add a positive number to $f(x)$, so that it becomes positive on $S(f)$. Note that minimizing $f(x)$ is equivalent maximizing $-f(x)$. The proposed mode-pursuing sampling algorithm consists of the following four steps:

Step 1. Generate m initial points $x^{(1)}$, $x^{(2)}$, …, $x^{(m)}$ that are uniformly distributed on $S(f)$ (m is usually small).

Step 2. Use the m function values $f(x^{(1)})$, $f(x^{(2)})$,…, $f(x^{(m)})$ to fit a linear spline function

$$\hat{f}(x) = \sum_{i=1}^{m} a_i \| x - x^{(i)} \|, \qquad \textbf{(1)}$$

such that $\hat{f}(x^{(i)}) = f(x^{(i)})$, $i = 1, 2, \ldots, m$, where $\| \bullet \|$ stands for the Euclidean norm.

Step 3. Define $g(x) = c_0 - \hat{f}(x)$, where c_0 is any constant such that $c_0 \geq \hat{f}(x)$, for all x in $S(f)$. Since $g(x)$ is nonnegative on $S(f)$, it 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 $\{f(x^{(i)})\}$. Then apply the sampling algorithm of Fu and Wang [8] to draw a random sample $x^{(m+1)}$, $x^{(m+2)}$, …, $x^{(2m)}$ from $S(f)$ according to $g(x)$. These sample points have the tendency to concentrate about the maximum of $g(x)$, which corresponds to the minimum of $\hat{f}(x)$.

Step 4. Combine the sample points obtained in Step 3 with the initial points in Step 1 to form the set $x^{(1)}$, $x^{(2)}$, …, $x^{(2m)}$ and repeat Steps 2–3 until a certain stopping criterion is met.

Remark. In Step 2 above, a linear spline function is used to fit the expensive points because:

1. the linear spline function, or radial basis function (RBF), is the simplest function that passes through all the expensive points;
2. the linear spline function also prevents unnecessary "curvature" added to the unknown surface, in contrast to other models such as kriging;
3. moreover, it preserves the minimum among the expensive points, i.e., $\min \hat{f}(x) = \min\{ f(x^{(i)}),\ i = 1, \cdots, m\}$ due to linearity. This feature ensures that more expensive points are generated around the current minimum of $f(x)$, rather than being biased because of the approximation model $\hat{f}(x)$.

This algorithm is illustrated with a sampling example.

A Sampling Example

For ease of understanding, the mode-pursuing sampling method is illustrated with the well-known six-hump camel-back (SC) problem [9]. The mathematical expression of SC is

$$f_{sc}(x) = 4x_1^2 - \frac{21}{10}x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, (x_1, x_2) \in [-2,\ 2]^2. \quad (2)$$

A contour plot of the SC function is shown in Figure 1, where the H's represent local optima. H_2 and H_5 are two global optima at points (-0.090, 0.713) and (0.090, -0.713), respectively, with equal function value $f_{\min} = -1.032$.

Figure 1 here

In the first step of the MPS algorithm, we start with $m = 6$ initial random points $x^{(1)}, x^{(2)}, \ldots, x^{(6)} \in [-2,\ 2]^2$. Then $\hat{f}(x)$ is computed by fitting Eq. 1 to $f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(6)})$. Further, the function $g(x)$ is obtained by using the maximum of $\{f(x^{(i)}), i=1, \ldots, 6\}$ as c_0.

Now we apply Fu and Wang's algorithm to draw a sample as follows. First, $N = 10^4$ uniformly distributed base points are generated to form $S_N(g)$, the discretized version of the sample space $[-2,\ 2]^2$. Note that the base points in $S_N(g)$ are cheap points, in contrast to the original $m = 6$ expensive points used to build $\hat{f}(x)$. Further, without loss of generality, suppose the points in $S_N(g)$ are sorted in ascending order of the values of function $\hat{f}(x)$. The sequence of corresponding function values of $\hat{f}(x)$ is plotted in Figure 2(a), whereas the function $g(x)$ is plotted in Figure 2(b).

Figure 2 here

According to Fu and Wang's [8] method, the ordered 10^4 base points are grouped into $K = 10^2$ contours $\{E_1, E_2, \ldots, E_{100}\}$, with each having $N/K = 100$ points. For example, the first contour E_1 contains the 100 points at which the values of function $\hat{f}(x)$ are the lowest, whereas the last contour E_{100} contains the 100 points at which the values of $\hat{f}(x)$ are the highest. Let $\tilde{g}(i)$ be the average of $g(x)$ over E_i, $i = 1, 2, \ldots, 100$. The function $\tilde{g}(i)$, $i = 1, 2, \ldots, 100$ is plotted in Figure 2(c) and its cumulative distribution function $G(i)$ is displayed in Figure 2(d).

Finally, m = 6 contours are drawn with replacement according to distribution $\{G(i)\}$ and, if the contour E_i occurs $m_i > 0$ times in these draws, then m_i points are randomly drawn from E_i. All such points form the new sample $x^{(m+1)}$, $x^{(m+2)}$, …, $x^{(2m)}$.

As one can see from Figure 2(d), the contours from E_{80}~E_{100} (corresponding to high $\hat{f}$ values) have lower selection probabilities for further sampling than other contours, since the G curve is relatively flat in this area. However, such a probability for each contour is always larger than zero. On the other hand, it is generally desired to increase the probability of the first few contours as they correspond to low $\hat{f}$ values. To better control the sampling process, a speed control factor is introduced, which will be discussed later in the Speed Control Factor subsection. Figure 2(e) shows $\{\widehat{G}(i)\}$, which is obtained by applying the speed control factor to $\{G(i)\}$ in Figure 2(d). From Figure 2(e), one can see that the first few contours have high selection probabilities for next-step sampling, while the contours from E_{40} ~E_{100} have low probabilities. This curve shows an aggressive sampling step, as many more new sample points be close to the current minimum of $f(x)$ as compared to the sampling based on Figure 2(d).

The whole procedure is repeated eight times, so that a total of 48 sample points are generated. Figure 3 shows these 48 sample points, where the circles indicate attractive design points having a function value less than –0.5. Even with only 48 sample points, many attractive points have already shown up around H_2 and H_5. It can also be seen that points spread out in the design space with a high density around function mode H_2 (global minimum).

Figure 3 here

Properties of the MPS Method

From the above construction, it is easy to see that this sampling procedure has the following two properties: firstly, every point in $S(f)$ has a positive probability of being drawn, so that the probability of excluding the global optimum is zero; and secondly, as the iteration process continues, more and more sample points will concentrate around the modes of function $g(x)$, which in turn pursue the modes of function $f(x)$. Thus, the algorithm automatically pursues the modes of $f(x)$. It can be proved that by neglecting the stopping criterion in Step 4, the method can identify the global optimum of $f(x)$.

Suppose the stopping rule in Step 4 is not applied, so that the sampling algorithm iterates to infinity. For any integer $k > 0$, the minimum function value obtained after k-th iteration is clearly

$$f_k = \min \{f(x^{(i)}), i = 1, 2, \ldots, km\}.$$

The following theorem shows that, under fairly general conditions, the sequence f_k converges to the global minimum, as k increases to infinity.

Theorem 1. Suppose an objective function $f(x)$ is continuous in a neighbourhood of its global minimum on a compact subset $S(f) \subset \Re^n$. Then, as $k \to \infty$, f_k converges to the global minimum $f_0 = \inf_{x \in S(f)} f(x)$.

Proof: Suppose the global minimum is attained at $x_0 \in S(f)$, so that $f(x_0) = f_0$. By construction, the set of all sampled points is extended by m new points after each iteration. It follows that the sequence $\{f_k\}$ is nonincreasing. Since $f(x)$ is continuous in a neighbourhood of the global minimum, for any $\varepsilon > 0$, there exists $\eta > 0$, such that $0 < f(x) - f(x_0) < \varepsilon$, for all $\|x - x_0\| < \delta$. Because in each iteration the density $g(x)$ is positive on $S(f)$, there exists an integer $K > 0$, such that after K-th iteration, a point satisfying $\| x_1 - x_0 \| < \delta$ can be sampled, which implies that $0 < f(x_1) - f(x_0) < \varepsilon$. It follows that $0 < f_k - f_0 < \varepsilon$ for all $k > K$. The proof is completed.

In summary, because MPS does not presume any properties of the function, it applies to general black-box functions, which can be either continuous or discontinuous across the entire design space. If the objective function is continuous in a neighbourhood of the global optimum, then the MPS systematically converges to the global optimum.

Global Optimization Strategy

From the above sections, we can see that the MPS method has the property of statistically sampling more points near the minimum (mode) while still covering the entire design space. For the purpose of optimization, one may iterate this sampling procedure until a maximum number of function evaluations has been reached. Such an approach is a legitimate global optimization method. However, such a crude approach can be slow to converge, when $f(x)$ is relatively flat around the global optimum. In such cases, more "intelligence" needs to be built in. A natural method, as seen in some existing global optimization methods, is to use a threshold of sample

density or similar measures to shrink the design space to a small area, in which a local optimization can be performed. No matter what measure is used, whether it is a sample density or a hyper-sphere with fixed diameter, it would be too difficult to decide the value of the measure, and be too rigid as it is to be applied to all types of problems. On the other hand, the idea of integrating a global technique with a local technique has been recognized as a plausible approach in designing a global optimization method [10]. In this paper, we design a special approach to guide the optimization process towards identifying an attractive local area without using a rigid measure.

Concept of the Strategy

As the MPS progresses, more sample points will be generated around the current minimum point, with chances of finding a better minimum. If a better point is found, the mode shifts to the new point and more sample points will be generated about that point. If the current minimum point is the global optimum, more and more points are generated in the small region that contains the optimum, with a small chance that this point can be exactly reached. On the other hand, according to Taylor's theorem, any smooth function can be locally, accurately approximated by a quadratic function. Therefore, a good fit of a quadratic model to the points in a sufficiently small neighbourhood of the current minimum is an indication that the "valley" containing the global minimum may have been reached. Given such a quadratic model, local optimization can be performed without expensive function evaluation to locate the global minimum.

An *n*-dimensional generic quadratic model is usually expressed by

$$y = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^{\,2} + \sum_{i<j} \sum_{j=1}^{n} b_{ij} x_i x_j, \qquad \textbf{(3)}$$

where b_i, b_{ii}, and $\hat{a}_{ij}$ represent regression coefficients, $x_i, (i = 1, \cdots, n)$ are design variables, and *y* is the response. Note that this model follows the standard formula used in response surface methodology (RSM) [11]. To assess the model's goodness of fit, usually the coefficient of determination

$$R^2 = \frac{\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad \textbf{(4)}$$

is used, where $\hat{y}_i$ are the fitted function values, y_i are the function values at the sample points and $\bar{y}$ is the mean of y_i. In general, $0 \le R^2 \le 1$ and the closer the R^2 value to 1, the better the modeling accuracy [11].

Now the question is how to determine the "neighbourhood" of the current minimum for model fitting. First, to fit the quadratic model in Eq. 3, at least $(n+1)(n+2)/2$ points are needed to obtain the estimates of the same number of coefficients. But if exactly $(n+1)(n+2)/2$ points are used to fit the model, then R^2 will reach its maximum value 1, making it impossible to evaluate the fitting accuracy. In order to avoid this situation, and at the same time to keep the number of function evaluations low, we propose to use $(n+1)(n+2)/2+1$ points to fit the quadratic model at the beginning of the sampling procedure. Along with the iteration of the sampling procedure, there will be enough points, which are close to the current minimum. These points will determine a sub-region (a hyper-box), which is defined by the minimum and the maximum of each coordinate. At the beginning of the sampling procedure, the size of this sub-region will be almost the same as the original design space, because the total number of points generated to that point is small and sparse. If the objective function $f(x)$ is quadratic, a perfect fit will be obtained and the global optimum can be quickly secured. If $f(x)$ is multi-modal, quadratic model fitting won't be perfect; even if by chance it fits well, it is not likely to pass the model validation stage, to be discussed later. In this case, the MPS process will concentrate less on the current optimum but instead spread out more in the rest of the space by tuning a speed control factor r (to be discussed later). As the optimization process iterates, and more and more points are generated, the sub-region will shrink gradually to a small region containing the global optimum. As one can see, in the proposed method, the size of the sub-region adapts to the complexity of the objective function, although it is almost the same as the original space at the beginning of the sampling procedure. In this way, a rigid measure of identifying the local area, such as a sample density or a hyper-sphere with fixed diameter, is avoided.

Quadratic Model Detection

In this paper, a quadratic function within a sub-region is detected by a two-stage approach. At the first stage, $(n+1)(n+2)/2+1$ points are used to fit a quadratic model and the R^2 value is obtained. If R^2 is not close to 1, it means that the function values are highly polynomial. If R^2 is close to 1, then a second stage testing is used. At the second stage, $[n/2]$ new expensive points are randomly generated within the sub-region, where $[n/2]$ stands for the integer part of $n/2$. Then the quadratic model is fitted again using these new points and the previous $(n+1)(n+2)/2+1$ points combined. If the new R_{new}^2 is close to 1, and if all predicted values are close to their corresponding function values, then it is most likely that the function is locally quadratic. Precisely, the criterion for second stage testing is

$$\begin{aligned} &1 - R_{New}^2 < e_R \quad \text{and} \\ &Diff = \max\{|\, f_m^{(i)} - f^{(i)} \,|,\ i = 1, \cdots,\ j \equiv (n+1)(n+2)/2 + 1 + [n/2]\} < e_d, \end{aligned} \tag{5}$$

where f_m is the fitted quadratic model and f is the objective function, e_R and e_d are small positive numbers for the two criteria, respectively. In practice, e_R is often taken to be 10^{-5}. It is difficult, however, to select e_d because it is problem dependent. This paper defines e_d as

$$e_d = c_d\,[\max(f^{(i)}) - \min(f^{(i)})], \tag{6}$$

where c_d is a constant in the interval [0, 1] and is to be specified by the user. Generally, the smaller the c_d, the more accurate result one will get and the more function evaluations are needed. In this paper, the default value of c_d is 10^{-2}. Note that second stage testing is necessary because R^2 can be close to 1 even though the function may not be quadratic, due to the fact that only $(n+1)(n+2)/2+1$ points are used in model fitting.

Algorithm of the Proposed Method

The flowchart of the algorithm is shown in Figure 4. Detailed description of the algorithm is given below. The speed control factor r is discussed in the next section.

Figure 4 here

Description of the Algorithm

Input: n: the number of design variables; x: desgin varaible vector; $f(x)$: objective black-box function.

Output: x^*: the obtained optimum; $f(x^*)$: the function value at x^*; *nit*: the number of optimization iterations; *nfe*: the number of expensive function evaluations.

BEGIN:

1) Randomly sample [(n+1)(n+2)/2+1-n_p] points, where, n_p is an arbitrary integer; usually set as $n_p = n$.
2) Sample n_p points using the MPS procedure; after which [(n+1)(n+2)/2+1] points are available, enough to fit a quadratic model.
3) Find [(n+1)(n+2)/2+1] points around the current mode, along with the sub-region defined by these points.
4) In the sub-region, fit a quadratic model to the [(n+1)(n+2)/2+1] points and compute the R^2 value.
5) **If** $1 - R^2 < e_R$, **then** generate [n/2] expensive points within the sub-region; fit the quadratic model again using all points in the sub-region and compute the R_{new}^2 and *Diff* values; add the new points to the point set;

   Else update the speed control factor r and go back to Step 2).
6) **If** $1 - R_{New}^2 < e_R$ and $Diff < e_d$, **then** perform local optimization on the fitted quadratic model to find the optimum x_t^*.
7) **If** x_t^* is in the sub-region, **then** the program stops with all the outputs and $x^* = x_t^*$; **Else** obtain its function value and add this point (x_t^*, $f(x_t^*)$) and the points generated in Step 5) to the original set of expensive points, update the speed factor r and go back to Step 2).

END

Speed Control Factor

The speed control factor r adjusts the "greediness" of the MPS procedure (see Figure 5). Referring to Figure 2, recall that $\{G(i), i = 1,2,..., K\}$ denotes the cumulative distribution based on the average heights $\{\tilde{g}(i), i = 1,2,..., K\}$, which is the discrete distribution $\{P_1, P_2, \ldots, P_K\}$ in the algorithm of Fu and Wang [8]. Note that $G(i) \in [0, 1]$. Define $G_{\min} = \min\{G(i)\}$ and $G_{\max} = \max\{G(i)\}$. It is easy to see that $G_{\max} = 1$, while $G_{\min}$ is the probability of sampling the points around the current mode of $\hat{f}(x)$. For each $i = 2, \cdots, K$, the difference $G(i) - G(i-1)$ represents the probability of sampling the points whose function values are between $\tilde{g}(i-1)$ and $\tilde{g}(i)$. As one can see from Figure 2, by increasing the value of $G_{\min}$ one can force the MPS to generate more "local" points around the current mode; and by decreasing $G_{\min}$ to generate more "exploratory"

points in other area of the design space. Thus, a speed control factor *r* can be introduced to allow the user to adjust the balance between "local" and "exploratory" points and, thus, to tune the "greediness" of the MPS process. In this paper, the tuning is done through the transformation $\hat{G}(i) = G(i)^{1/r}, i = 1,2,...,K$. The sampling in the next iteration is then based on the $\hat{G}$-curve instead of the *G*-curve. Theoretically, $r \in [1, \infty]$. If *r=1*, then the original discrete distribution is used. A higher *r>1* increases the probability of sampling around the current mode and decreases the chance of spreading out in the design space. Figure 5 is a screen shot of $G^{1/r}$-curves for the SC problem, where the curve becomes flatter as *r* increases. We can also see that the factor *r* controls the speed of the MPS process: when $r = 1$, the original discrete distribution is used and the convergence speed doesn't change, while the convergence speed of the MPS process increases as *r* increases. During the optimization process, however, *r* does not increase monotonically, because it updates dynamically from iteration to iteration.

Figure 5 here

The next question is how to adjust the value of the speed control factor in practice. Generally speaking, if a function is detected to be complex (non-quadratic), then *r* should be reduced to generate more "exploratory" points; otherwise, *r* should be increased to generate more "local" points. In practice, a value of $G_{\min} = 0.75$ represents a very aggressive sampling scheme and, therefore, we choose this value as a reference point to determine $r_{\max}$. Given the values of $g(i)$, we can compute the actual $G_{\min}$. By setting $G_{\min}^{\ 1/r_{\max}} = 0.75$, we obtain $r_{\max} = \log G_{\min} / \log(0.75)$. Therefore, the range of the speed control factor is determined as $r \in [1, \ \log G_{\min} / \log(0.75)]$. In this paper, the information R^2 is used to guide the "greediness" of the MPS. In many practical problems, fitting the model Eq. 3 with only [(*n*+1)(*n*+2)/2+1] sample points usually results in a $R^2 \in (.8, 1]$. Therefore we propose to set *r=1* to allow more "exploratory" points when $R^2 \le 0.8$. When $R^2 \in (.8, 1]$, we propose to determine the *r*-R^2 relationship using an elliptical curve, as shown in Figure 6. The origin of the ellipse is (.8, $r_{\max}$), with the length of the short axis 0.2 and the long axis $r_{\max} - 1$. The origin of the ellipse and the long axis are automatically adjusted by the value of $G_{\min}$ calculated from $\hat{f}(x)$, which is obtained from the available expensive sample points. Figure 6 shows the r-R^2 curve of the last iteration in the SC example. Thus, the value of *r* is controlled by the value of R^2 from the previous iteration. It is easy to see

that the speed control factor does not undermine the global sampling property of the MPS strategy. This can be seen from Figure 2(e) and Figure 5, where $G(i)^{1/r} \to 1$ as $r \to \infty$.

Figure 6 here

Note that there is a risk associated with using R^2 as a feedback for speed control of the MPS. Since the sampling process controlled by factor r tends to be more aggressive, the optimization process may converge to a local optimum before enough exploratory points have been generated. It would be ideal to have an indicator of whether the current mode is the global optimum or not. Such an indicator will be theoretically better than R^2. However, without knowing the global optimum or properties of the expensive function *a priori*, it is difficult to develop such an indicator. The R^2 value only provides a necessary condition for a global optimum, not a sufficient condition. A similar use of R^2 is found in Ref. [12]. Nevertheless, the proposed method works well for most of the test problems. It also maintains the possibility of mode shifting because even with aggressive sampling, new exploratory sample points will still be generated.

Constrained Optimization Problems

The algorithm of the proposed method in the previous section applies to unconstrained optimization problems. For constrained optimization problems, we only consider the problems with inexpensive constraints. Specifically, we are interested in the problem

$$\textbf{\textit{Minimize}} \qquad f(x) \tag{7}$$

$$\textbf{\textit{Subject to}} \qquad g_k(x) \le 0, \quad k = 1, \cdots q \tag{8}$$

where $f(x)$ is the expensive objective function and $g_k(x)$ are the inexpensive equality or inequality constraints, which may include variable bounds. Metamodeling for constrained optimization problems has been studied in the literature ([3; 13; 14]).

With minor modifications, the optimization procedure of the previous sections for the unconstrained problems applies to the constrained problem Eqs. 7-8 as well. The flowchart for the constrained problem is the same as that in Figure 4. The modifications are:

1. In all sampling steps, including "Initial random sampling [$(n+1)(n+2)/2+1-n_p$] points", "MPS of n_p points" and "Randomly generate [$n/2$] points", all points that do not satisfy the constraints of Eq. 8 are discarded before evaluating the objective function.
2. In the step "Perform local optimization", the constraints $g_k(x)$ are included in the optimization.

In this paper, the Matlab^{TM} Optimization Toolbox is used for the constrained local optimization.

Tests of the Approach

The proposed method has been tested on a number of well-known test problems. They are described as follows:

(1) A simple quadratic function (QF) with $n = 2$:

$$f_{QF} = (x_1 + 1)^2 + (x_2 - 1)^2, \quad x_1, x_2 \in [-3,\ 3]. \tag{9}$$

(2) Six-hump camel-back function (SC) with $n = 2$, as defined by Eq. 1.

(3) Golden-Price function (GP) with $n = 2$:

$$\begin{aligned} f_{GP}(x_{1,}x_2) &= [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)] \\ &*[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)], \end{aligned} \tag{10}$$

where, $x_1, x_2 \in [-2,\ 2]$.

(4) Hartman function (HN) with $n = 6$:

$$f_{HN}(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2],\ x_i \in [0, 1],\ i = 1, \cdots, n, \tag{11}$$

where

i	$a_{ij},\ j = 1, \cdots, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	.05	10	0.1	14	3.2

i	$p_{ij},\ j=1,\cdots,6$					
1	.1312	.1696	.5569	.0124	.8283	.5886
2	.2329	.4135	.8307	.3736	.1004	.9991
3	.2348	.1451	.3522	.2883	.3047	.6650
4	.4047	.8828	.8732	.5743	.1091	.0381

(5) A function of 16 variables (F16) with n = 16:

$$f_{F16}(x)=\sum_{i=1}^{16}\sum_{j=1}^{16}a_{ij}(x_i^2+x_i+1)(x_j^2+x_j+1),\quad i,j=1,2,\cdots,16, \tag{12}$$

$$[a_{ij}]_{row1-8}=\begin{bmatrix}1&0&0&1&0&0&1&1&0&0&0&0&0&0&0&1\\0&1&1&0&0&0&1&0&0&1&0&0&0&0&0&0\\0&0&1&0&0&0&1&0&1&1&0&0&0&1&0&0\\0&0&0&1&0&0&1&0&0&0&1&0&0&0&1&0\\0&0&0&0&1&1&0&0&0&1&0&1&0&0&0&1\\0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0\end{bmatrix},\ [a_{ij}]_{row9-16}=\begin{bmatrix}0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1\\0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\end{bmatrix}.$$

(6) Griewank function (GN) with n = 2:

$$f_{GN}(x)=\sum_{i=1}^{n}x_i^2/200-\prod_{i=1}^{n}\cos(x_i/\sqrt{i})+1,\quad x_1,x_2\in[-100,100]. \tag{13}$$

For each problem, 10 runs are carried out to reduce random variation in the numerical results. The average (arithmetic mean) number of function evaluations, *nfe*, and the number of iterations, *nit*, are used as an indication of the time and resources required in the computation. The medians are also given for reference. For the solution, the minimum and the maximum for the 10 runs are recorded, along with the average. The results are summarized in Table 1.

Table1 here

As one can see from Table 1, the proposed method successfully captures the global optimum for all test problems except for GN. The total number of function evaluations is modest. Over the 10 runs for each case, the range of the minimum objective function value is very small. Variations among the *nfe* and *nit* are also small, as shown by the average and median of these numbers. This indicates that the proposed method is very robust, though being random in nature. The only exception is GN: although in some runs the results are very good (obtained f=0.003

with 9 function evaluations), the overall performance is not stable, as demonstrated by the large solution variation and differences between the average and median *nfe* and *nit* numbers. As a matter of fact, function GN has 500 local optima in the design space and, hence, the optimization process tends to be trapped in one of these local optima, as is shown in Table 2. Apparently this happened because the MPS process was terminated prematurely when there were not enough "exploratory" points to present the entire design space. To confirm this speculation and also to provide an example for Theorem 1, the GN problem is solved by applying the MPS method alone with a stopping criterion of $|f-0|<1e-3$, since we know the analytical minimum is zero. In other words, we let the MPS process continue until the analytical minimum is found. The results are summarized in Table 2, where x^* stands for the optimum point and f^* is the obtained function minimum.

Table 2 here

As one can see from Table 2, the global optimum is obtained for all runs. The number of function evaluations, however, differs dramatically. The test on GN confirms the theoretical result of Theorem 1 that the proposed MPS procedure converges systematically to the global optimum. The overall optimization strategy, which is based on the MPS method, might converge prematurely for problems having a large quantity of local optima such as GN.

Overall, from both the accuracy and efficiency perspectives, the proposed optimization method demonstrates satisfactory performance. The solutions are generally robust. In addition, assuming the availability of computers, the total time needed by the optimization will be represented by the number of iterations, rather than the number of total function evaluations, because function evaluations can be done independently and simultaneously at all sample points. Thus the proposed method also supports simultaneous computation, which leads to significant time savings.

Design Examples & Results

Two engineering test problems are used to test the utility of the proposed method. Both are constrained optimization problems. Each problem is described with its corresponding constraints, bounds, and objective function. We assume their objective functions are expensive black-box functions and thus the two design optimization problems are of the form described by Eqs. 7-8.

Design of a two-member frame

This example has been used in [15] and is the design of a two-member frame subjected to the out-of-plane load, P, as is shown in Figure 7. Besides L = 100 inches, there are three design variables: the frame width (d), height (h), and wall thickness (t), all of which are in inches and have the following ranges of interest: $2.5 \le d \le 10$, $2.5 \le h \le 10$ and $0.1 \le t \le 1.0$. Figure 7 here

The objective is to minimize the volume of the frame subject to stress constraints and size limitations. The optimization problem can be written as

$$\begin{aligned} &\text{minimize } V = 2L(2dt + 2ht - 4t^2) \\ &\text{Subject to } g_1 = (s_1^2 + 3t^2)^{1/2} \le 40{,}000 \\ &\qquad\qquad g_2 = (s_2^2 + 3t^2)^{1/2} \le 40{,}000 \\ &\qquad\qquad d, h \in [2.5, 10] \quad t \in [0.1\ 1] \end{aligned} \tag{14}$$

where s_1, s_2 and t are respectively the bending stresses at point (1) (also point (3)) and (2) and the torsion stress of each member. They are defined by the following equations:

$$s_1 = M_1 h/(2I) \qquad s_2 = M_2 h/(2I) \qquad t = T/(2At)$$

where

$$M_1 = 2EI(-3U_1 + U_2 L)/L^2 \qquad M_2 = 2EI(-3U_1 + 2U_2 L)/L^2 \qquad T = -GJU_3/L$$

$$I = (1/12)[dh^3 - (d-2t)(h-2t)^3] \qquad J = 2t[(d-t)^2(h-t)^2]/(d+h-2t) \qquad A = (d-t)(h-t)$$

Given the constants E=3.0E7 psi, G=1.154E7 psi and the load P = -10,000 lbs, the displacements U_1 (vertical displacement at point (2)), U_2 (rotation about line (3)-(2)), and U_3 (rotation about line (1)-(2)) are calculated by using the finite element method.

$$\frac{EI}{L}\begin{bmatrix} 24 & -6L & 6L \\ -6L & (4L^2 + \frac{GJ}{EI}L^2) & 0 \\ 6L & 0 & (4L^2 + \frac{GJ}{EI}L^2) \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix}.$$

The optimum found in [15] is V = 703.916 in^3, occurring at: d^* = 7.798 in, h^* = 10.00 in, and t^* = 0.10 in.

Design of a pressure vessel

The design of a pressure vessel was first documented in [16]. The cylindrical pressure vessel is shown in Figure 8. The shell is made in two halves of rolled steel plate that are joined by two

longitudinal welds. Available rolling equipment limits the length of the shell to 20 ft. The end caps are hemispherical, forged and welded to the shell. All welds are single-welded butt joints with a backing strip. The material is carbon steel ASME SA 203 grade B. The pressure vessel should store 750 ft^3 of compressed air at a pressure of 3,000 psi. There are four design variables: radius (R) and length (L) of the cylindrical shell, shell thickness (T_s) and spherical head thickness (T_h), all of which are in inches and have the following ranges of interest: $25 \le R \le 150$, $1.0 \le T_s \le 1.375$, $25 \le L \le 240$ and $0.625 \le T_h \le 1.0$.

Figure 8 here

The design objective is to minimize total system cost, which is a combination of welding, material and forming costs. The constraints include ASME boiler and pressure code for wall thickness T_s and T_h, as well as tank volume. The optimization model is therefore formulated as:

$$\begin{aligned} &\min\ F = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_s^2 R \\ &\text{Subject to}\ \ g_1 = T_s - 0.0193R \ge 0 \\ &\qquad g_2 = T_h - 0.00954R \ge 0 \\ &\qquad g_3 = pR^2 L + (4/3)pR^3 - 1.296E6 \ge 0 \\ &\qquad R \in [25,\ 150],\ T_s \in [1.0,\ 1.375],\ L \in [25,\ 240],\ T_h \in [0.625,\ 1.0] \end{aligned} \qquad \textbf{(15)}$$

The optimum continuous solution is F = 7006.8, occurring at R^* = 51.814 in., T_s^* = 1.0 in., L^* = 84.579 in., and T_h^* = 0.625 in.

Results

For each of the two design problems, 10 runs are executed. For the solution, the minimum and the maximum of the 10 runs are recorded along with the average. The results are shown in Table 3, where the abbreviations have the same meaning as those in Table 1. Because the obtained optima from the 10 runs for each problem are practically the same as the analytical optimum, the difference between the analytical solution and the one found by the proposed method should be due to numerical rounding errors. In addition, the number of function evaluations *nfe* is small for both cases.

Table 3 here

Discussion

As shown in Table 1, the proposed method finds the optimum of the QF function with an average of only 9.6 function evaluations. In fact, it is true that for any *n*-dimensional quadratic function, the number of function evaluations needed is between nfe_l and nfe_u, defined as

$$nfe_l = (n+1)(n+2)/2 + 1 + [n/2], \qquad nfe_u = nfe_l + 1 + n + [n/2]. \quad \textbf{(16)}$$

nfe_l is required when the algorithm converges at the first iteration, where $[(n+1)(n+2)/2 + 1]$ is the number of points used in the first quadratic model fitting, and $[n/2]$ is the number of points generated for model validation. According to the algorithm, the real function value at the model minimum is obtained if the model minimum is outside the sub-region. Then the second iteration starts with n new sample points based on the MPS method, and uses another $[n/2]$ points for model validation. Since the function is locally approximately quadratic, the model minimum at the first iteration should be the real optimum. Thus, the process terminates after the second iteration with the maximum number of function evaluations $nfe_u = nfe_l + 1 + n + [n/2]$. For $n = 2$, $nfe_l = 8$ and $nfe_u = 12$, which were exactly observed for the QF function. Generally, it can be seen by construction that this algorithm requires $O(n^2)$ number of sample points. The number of points can be reduced if the quadratic model in Eq. 3 is further simplified.

The proposed approach shares some similarities with a mature clustering method called the Topographical Global Optimization (TGO) method in [17]. The TGO method randomly generates N sample points in a given space (N is usually big). Then it finds points better than a set of k prescribed neighbouring points. Local optimization is performed from all such points to find q local optima. It keeps the selected points and re-samples (N-q) points randomly. The process iterates until a prescribed number of iterations is reached. Our method differs from the TGO method in the following aspects:

(1) Only one-time random sampling is used. The initial sample size is small. Subsequent sampling follows the mode-pursuing sampling procedure.
(2) Our method only finds the best sample point with its neighbouring points.
(3) Our method approximates the local region and performs local optimization only when the local region is close to quadratic.
(4) Our method stops automatically and no prescribed *nit* is used.
(5) Our method does not require the user to prescribe N and k.
(6) The overall number of function evaluations is low.

Conclusions

In this paper, a novel mode-pursuing sampling (MPS) method has been developed, which systematically converges to the global optimum if the objective function is continuous in a

neighbourhood of the optimum. Based on MPS, a global optimization method was developed for expensive black-box functions. This optimization strategy applies local quadratic fitting to adaptively identify an attractive sub-region in which local optimization can be performed. The MPS is then adjusted by a speed control factor to balance the "local" search and global "exploration." This optimization method has been tested with both low (n=2) and high (n=16) dimensional test problems. It has also been applied to two constrained design optimization problems. Overall, the proposed method has been found to have the following advantages:

1. In general, it is an effective and efficient global optimization method for expensive black-box objective functions, which can be either continuous or discontinuous. The MPS method converges to the global optimum if the objective function is continuous in a neighbourhood of the global optimum.
2. The method supports simultaneous computation. The extent of simultaneous computation can be controlled by the parameter n_p.
3. Although random in nature, the method is robust and requires little parameter tuning. For a given problem, only one optimization parameter c_d might need user specification, which is however not mandatory.
4. It works for both unconstrained and constrained problems with expensive objective functions and inexpensive constraints.
5. Unlike other metamodeling-based global optimization methods such as in Refs. [3; 4], it does not call any existing global optimization procedure. Therefore, it can work as a standalone global optimization method even for inexpensive functions.

Through the test examples, it has also been found that, for problems with a large quantity of local optima, the proposed optimization method may be trapped in a local optimum. This is due to the lack of a rigorous indicator of the global optimum. In such a case, the MPS may be applied alone as it guarantees convergence to the global optimum, as long as the function is continuous in a neighbourhood of the global optimum.

Acknowledgement

Financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

References

1. Haftka, R.T., Scott, E.P. and Cruz, J.R. (1998). Optimization and Experiments: A Survey. Applied Mechanics Review, 51(7): pp. 435-448.

2. Neumaier, A. (2001). Chapter 4: Constrained Global Optimization, In: Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art, COCONUT project.

3. Wang, G.G. and Simpson, T.W. (2002). Fuzzy Clustering Based Hierarchical Metamodeling for Design Optimization. 9th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia.

4. Wang, G.G. (2003). Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points. Transactions of ASME, Journal of Mechanical Design, 125: pp. 210-220.

5. Jones, D.R., Schonlau, M. and Welch, W.J. (1998). Efficient Global Optimization of Expensive Black Box Functions. Journal of Global Optimization, 13: pp. 455-492.

6. Scholau, M.S., Welch, W.J. and Jones, D.R. (1998). Global Versus Local Search in Constrained Optimization of Computer Models. In: N. Flournoy, W.F. Rosenberger and W.K. Wong (Editors), New Development and Applications in Experimental Design, Institute of Mathematical Statistics, pp. 11-25.

7. Wang, G.G., Dong, Z. and Aitchison, P. (2001). Adaptive Response Surface Method - A Global Optimization Scheme for Computation-intensive Design Problems. Journal of Engineering Optimization, 33(6): pp. 707-734.

8. Fu, J.C. and Wang, L. (2002). A random-discretization based Monte Carlo sampling method and its applications. Methodology and Computing in Applied Probability, 4: pp. 5-25.

9. Branin, F.H. and Hoo, S.K. (1972). A method for finding multiple extrema of a function of n variables. In: F. Lootsma (Editor), Numerical methods for non-linear optimization. Academic Press, New York, pp. 231-237.

10. Törn, A. (1998). Course notes on Probabilistic Algorithms, Department of Computer Science, Åbo Akademi University, Turku, Finland.

11. Montgomery, D. (1991). Design and Analysis of Experiments. John Wiley and Sons, New York.

12. Hacker, K., Eddy, J. and Lewis, K. (2001). Tuning a Hybrid Optimization Algorithm by Determining the Modality of the Design Space. ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Pittsburgh, PA.

13. Sasena, M., Papalambros, P. and Goovaerts, P. (2002). Global Optimization of Problems with Disconnected Feasible Regions Via Surrogate Modeling. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia.

14. Frits, A.P. and Mavris, D.N. (2002). A Screening Method For Customizating Designs Around Non-Convergent Region of Design Spaces. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia.

15. Arora, J.S. (1989). Introduction to Optimum Design. McGraw-Hill Higher Education, New York.

16. Wilde, D. (1978). Globally Optimal Design. Wiley, New York.

17. Törn, A. and Viitanen, S. (1996). Iterative Topographical Global Optimization. In: C.A. Floudas and P.M. Pardalos (Editors), State of the Art in Global Optimization. Kluwer Academic Publishers, pp. 353-363.

List of Figures

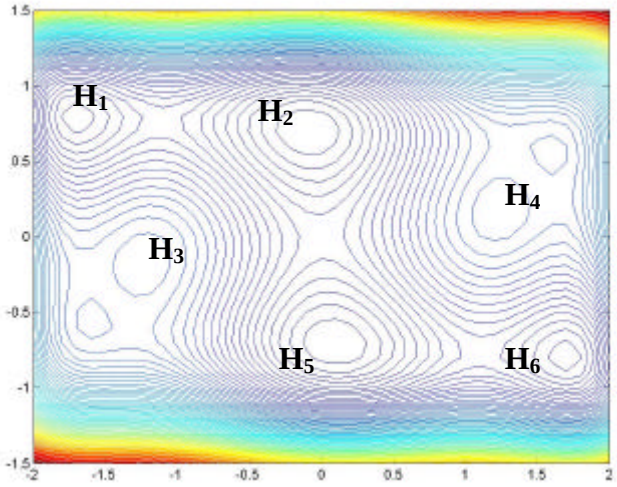

Figure 1 Contour plot of the SC function.

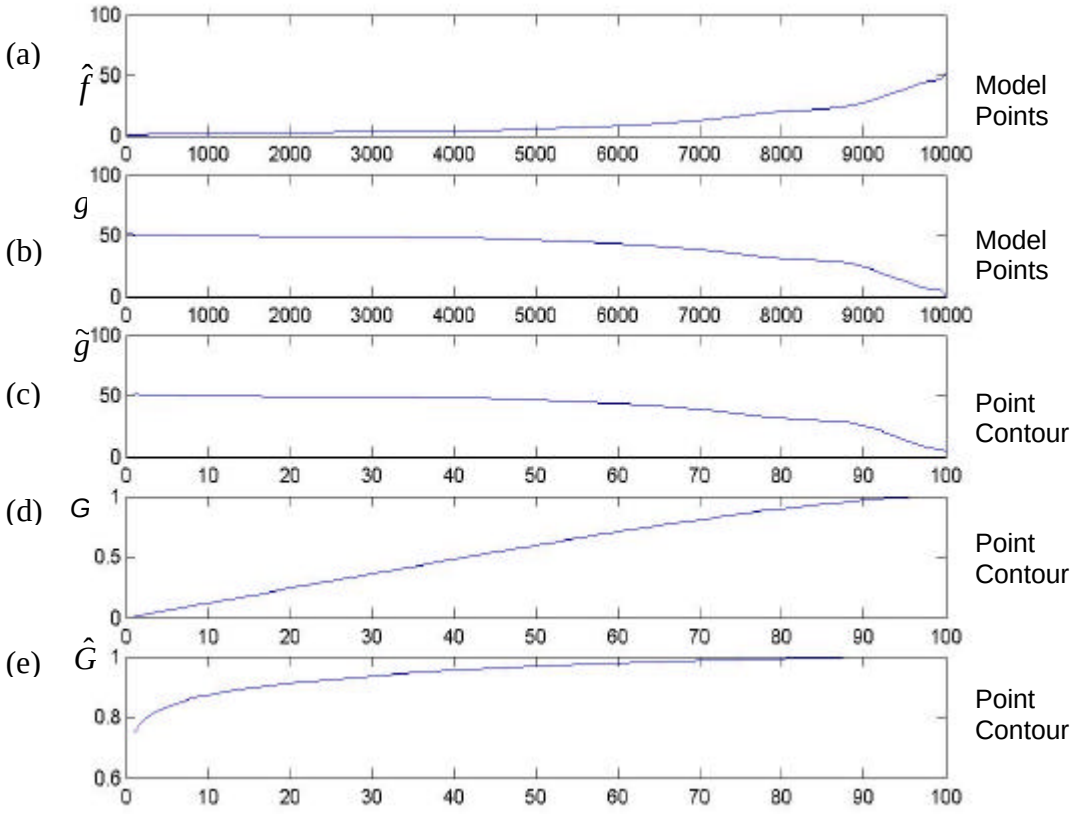

Figure 2 A screen shot of ranked point distribution of $\hat{f}$, g, $\tilde{g}$, G and $\hat{G}$ functions for the SC problem.

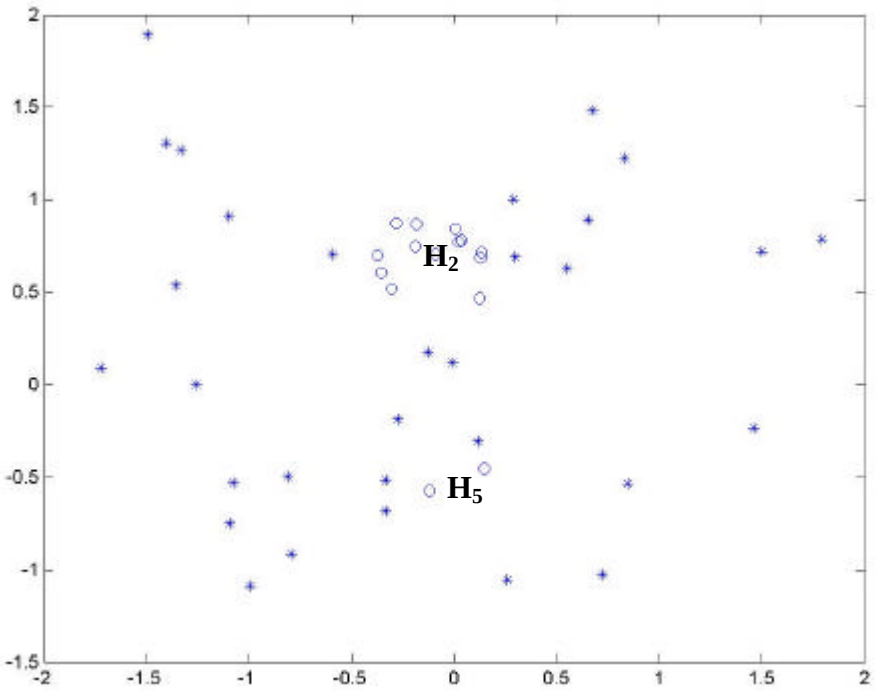

Figure 3 Sample points of the SC problem generated by the MPS method, where "o" indicates its function value less than –0.5; and H_2 and H_5 are the locations of two global optima.

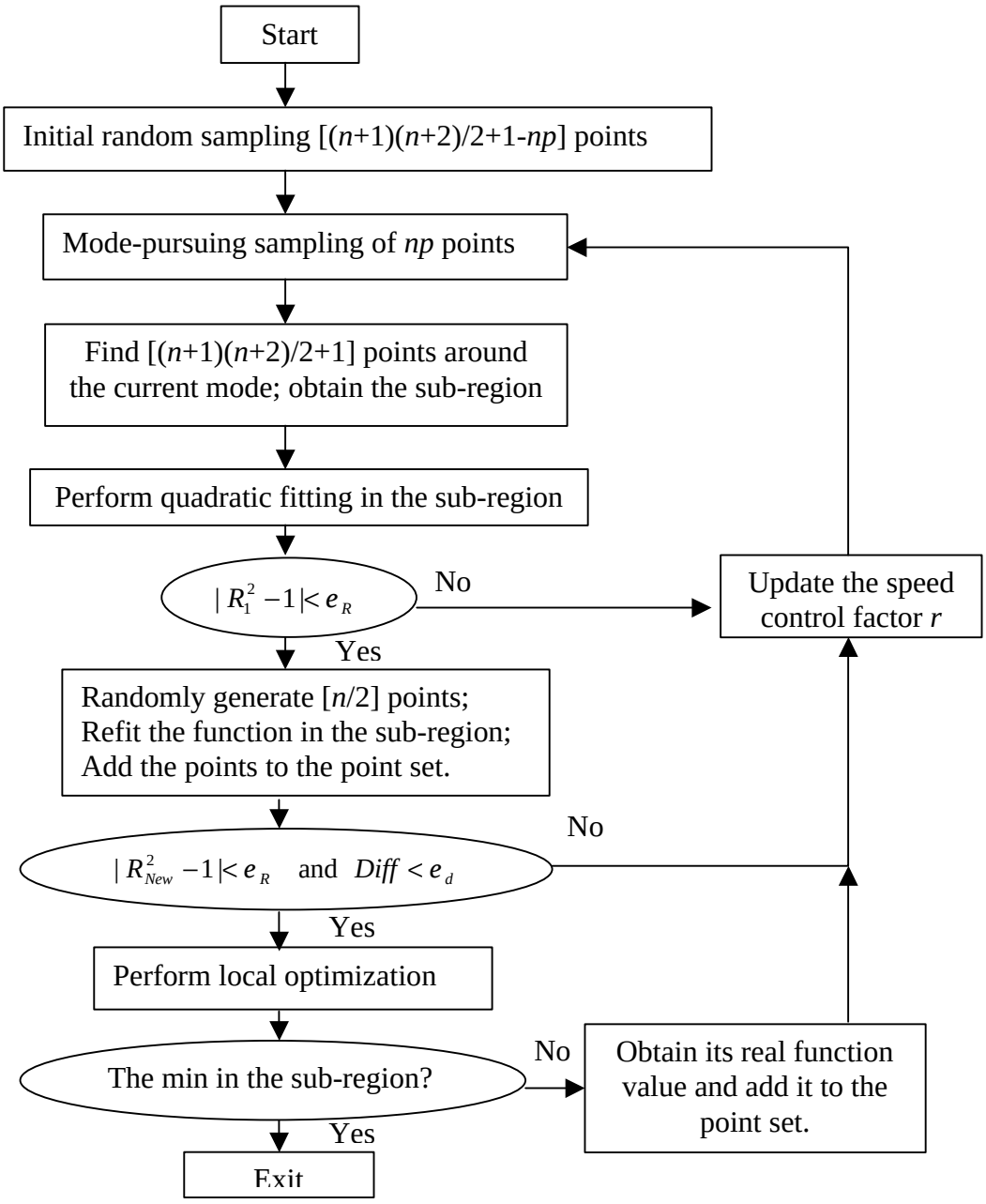

Figure 4 Flowchart of the proposed global optimization method.

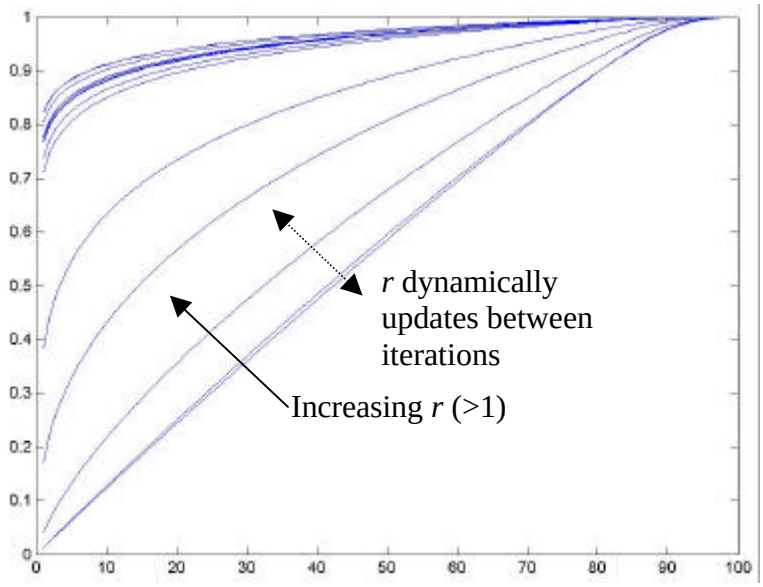

Figure 5 A screen shot of $G(i)^{1/r}$ curves for the SC problem.

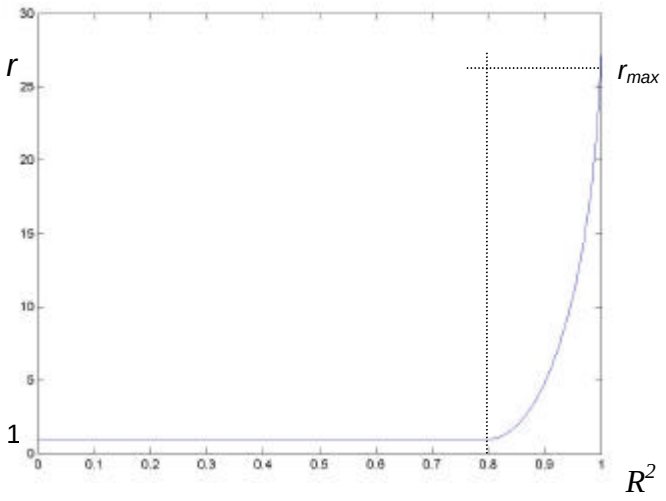

Figure 6 The last r-R^2 curve for the SC problem.

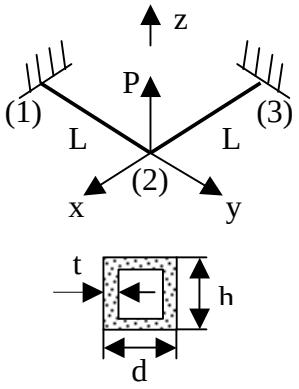

Figure 7 The two-member frame.

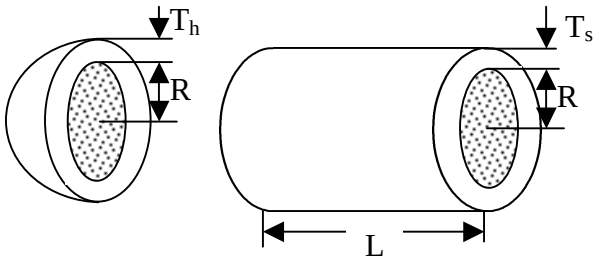

Figure 8 Pressure Vessel.

List of Tables

Table 1 Summary of test results on the proposed method.

Func.	*n*	Space	Analy. Min.	Minimum		*Nfe*		*nit*	
				Range of Variation	Median	Avg.	Median	Avg.	Median
QF	2	$x_{i\in}$ [-3 3]	0	[0, 0]	0	9.6	8	1.4	1
SC	2	$x_{i\in}$ [-2 2]	-1.032	[-1.031, -1.014]	-1.030	37.8	30.5	9	7
GP	2	$x_{i\in}$ [-2 2]	3.000	[3.000, 3.216]	3.005	138	134	32.9	34
HN	6	$x_{i\in}$ [0 1]	-3.322	[-3.322, –3.148]	-3.305	592.1	576	49.6	47
F16	16	$x_{i\in}$ [-1 0]	25.875	[25.880, 25.915]	25.885	254.8	250	3.8	3.5
GN	2	$x_{i\in}$ [-100 100]	0	[0.003, 1.367]	0.1469	371	43	123.8	12

Table 2 Optimization results on GN by applying the mode-pursuing sampling method alone.

Run No.	x^*	f^*	*nfe*	*nit*
1	(0.0000, -0.0070)	0	149	45
2	(0.0005, 0.00047)	0	1742	756
3	(0.0010, 0.0135)	0	1241	315
4	(0.0017, -0.0044)	0	1475	714
5	(0.0000, -0.0001)	0	390	173
6	(-0.0020, 0.0004)	0	1846	627
7	(0.0014, 0.0005)	0	2015	990
8	(0.0057, 0.0045)	0	218	93
9	(0.0058, -0.0070)	0	59	17
10	(-0.0003, -0.0005)	0	1291	627

Table 3 Summary of test results on the constrained designed problems.

Func.	*n*	Space	Analy. Min.	Minimum		*nfe*		*nit*	
				Range of Variation	Median	Avg.	Median	Avg.	Median
FD	3	[2.5 10] [2.5 10] [.1 1.0]	703.916	[703.947, 703.947]	703.947	20	20	2	2
VD	4	[25 150] [25 240] [1.0 1.375] [.625 1.0]	7006.8	[7006.8, 7007.9]	7006.8	44.7	46	6.7	7