Mode Pursuing Sampling Method for Discrete Variable Optimization on Expensive Black-Box Functions

Behnam Sharif

G. Gary Wang
e-mail: gary_wang@umanitoba.ca

Tarek Y. ElMekkawy

Department of Mechanical and Manufacturing Engineering,
University of Manitoba,
Winnipeg, Manitoba R3T 5V6, Canada

Based on previously developed Mode Pursuing Sampling (MPS) approach for continuous variables, a variation of MPS for discrete variable global optimization problems on expensive black-box functions is developed in this paper. The proposed method, namely, the discrete variable MPS (D-MPS) method, differs from its continuous variable version not only on sampling in a discrete space, but moreover, on a novel double-sphere strategy. The double-sphere strategy features two hyperspheres whose radii are dynamically enlarged or shrunk in control of, respectively, the degree of "exploration" and "exploitation" in the search of the optimum. Through testing and application to design problems, the proposed D-MPS method demonstrates excellent efficiency and accuracy as compared to the best results in literature on the test problems. The proposed method is believed a promising global optimization strategy for expensive black-box functions with discrete variables. The double-sphere strategy provides an original search control mechanism and has potential to be used in other search algorithms.
[DOI: 10.1115/1.2803251]

Keywords: discrete variable optimization, metamodeling, MPS, black-box functions

1 Introduction

Although most of the classic approaches of design optimization consider continuous variables, many practical problems deal with discrete or integer variables. For example, the diameters of a pipe, thickness of a structural member, or size of a screw are discrete design variables since they may have to be selected from commercially available or standard sizes. Also, many other design variables such as the number of bolts, number of teeth of a gear, or number of coils of a spring must be integers. If there is a discrete variable in an engineering optimization problem, it is often a mixed discrete-continuous nonlinear programming problem (MDNLP). Reference [1] reviewed different software tools that are currently used for solving MDNLP. It has been agreed on that the existence of any noncontinuous variable considerably increases the difficulty of finding the optimal solution [2]. This is due to the fact that MDNLP combines the difficulties of both of its subclasses: the combinatorial nature of mixed integer programs (MIPs) and the difficulty in solving nonconvex (and even convex) NLPs with nonlinear (or even linear) constraints [3]. Reference [4] categorized the mixed continuous-discrete variable problems into six types depending on the type of the variables, type of the objective function (differentiable, nondifferentiable, combinatorial), and whether or not the discrete variables can have nondiscrete values during the solution process. This categorization is based on the assumption that the objective function is presented explicitly. In most of the engineering design problems, this is not the case because engineers often deal with computer analysis or simulation processes, such as finite element analysis (FEA) and computational fluid dynamics (CFD) processes. These computational processes are usually referred as black-box functions, which can be nonconvex, nondifferentiable, and even discontinuous. Considering that only the input and output for these types of functions are available, characteristics of the function, such as the gradient and Hessian matrix, can be numerically approximated, but they are usually unreliable [5]. Although in most of the practical engineering problems, a suboptimal or even feasible solution is satisfactory, the global optimum is always preferred. This work addresses the challenge of discrete variable global optimization with expensive black-box functions such as FEA and CFD processes.

In general, global optimization approaches can be classified into two groups: deterministic and stochastic methods. Reference [6] gives a review of global optimization methods for MDNLP. Recently, metamodeling optimization techniques have been studied in the literature in support of engineering design optimization. Reference [7] gives a review of metamodeling techniques. Some of the deterministic, stochastic, and metamodeling methods will be discussed below.

When the objective/constraint functions are explicitly expressed and known, deterministic methods can be applied. These methods include the well-known branch and bound (B&B), sequential linear programming (SLP), cutting plane techniques, outer approximation (OA), Lagrange relaxation (duality) approaches, and so on. Another group of methods are called the rounding methods [8], which use simple dynamic rounding-off techniques. Most of these methods such as B&B [9] either use the closely related NLP to reach to the main MDNLP, or rely on the successive solution of the related mixed integer programming (MIP) such as OA [10].

For optimization problems with nonlinear constraints, linearly constrained Lagrangian (LCL) methods solve a sequence of subproblems that minimize an augmented Lagrangian function subject to linearization of constraints [11]. These deterministic approaches are based on explicitly expressed objective and constraint functions. Thus, they are not directly applicable to black-box functions, which is the main concern in this paper.

Some of the stochastic methods for global optimizations include the sequential random search, pure adaptive search (PAS) [12], and various evolutionary programming methods, such as simulated annealing (SA) [13] and genetic algorithms (GAs) [14]. Most of these stochastic methods do not need *a priori* knowledge about the objective function, which makes them good candidates for optimization on black-box functions. These methods, however, usually require a large number of function evaluations, often in the magnitude of 10,000 even for a two-variable problem, which

Contributed by the Design Automation Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received December 1, 2006; final manuscript received March 16, 2007; published online January 3, 2008. Review conducted by Timothy W. Simpson.

makes these methods impractical in modern design involving computational intensive processes. For example, it is reported that it takes Ford Motor Company about 36–160 h to execute one crash simulation [15]. Assuming only 50 crash simulations are needed to solve a two-variable optimization problem, the total computation time would be 75 days to 11 months, which is practically unacceptable. Several sequential random searches such as the PAS have been developed to ease the computational demand [16]. In addition, Ref. [17] introduced a Lipschitz global optimization algorithm called DIRECT for black-box functions, whose disadvantage is its slow convergence [18]. Different modifications of DIRECT have been proposed by Ref. [18] to solve the slow convergence problem and provide modifications for higher dimensions. Another approach for discrete variable optimization on black box function is to use soft-computing based approaches such as neural networks (NNs) as a modeling paradigm because of its universal approximation property. Moreover, combining NN with evolutionary algorithms and genetic quantum algorithms has also been studied in Refs. [19,20].

Metamodel based optimization algorithms are developed specifically for expensive black-box functions in order to reduce the number of function evaluations in stochastic and direct optimization. Most of these methods are based on the idea of sampling in the design space, building approximation models from these sample points, and then performing optimization on the approximation function. Detailed review on research in this area can be found in Refs. [7,21]. Reference [7] described different metamodeling techniques including model approximation, problem formulation, and design space exploration, which form a common supportive base for all types of optimization problems. As an example, Ref. [22] applied a Bayesian method to estimate a kriging model, and to gradually identify points in the space to update the model and perform the optimization. In their model, they assume a continuous objective function and a correlation function between sample points. Wang et al. [23] developed a mode pursuing sampling (MPS) method, which has been successfully applied to solve many benchmark as well as engineering design problems with continuous variables.

This paper adapts the MPS method to the discrete domain. The proposed method is called discrete variable MPS (D-MPS), distinguishing from the original MPS in Ref. [23], which is referred as continuous variable MPS (C-MPS). Due to the discrete variable nature, the convergence scheme of C-MPS is no longer applicable. A novel double-sphere strategy is developed to control the convergence of D-MPS. In Sec. 2, MPS in its original form [23,24] will be presented. In Sec. 3, the basic D-MPS method will be described. Section 4 is devoted to explaining D-MPS with the double-sphere strategy. Parameters of the D-MPS method will also be discussed in Sec. 4. Section 5 will show test problems and comparison with results from the literature. Section 6 discusses the generalization of the double-sphere strategy, and Sec. 7 is the conclusion.

2 Review of Mode Pursuing Sampling Method

The MPS method as described in Ref. [23] is referred as the C-MPS to differentiate from this work. The C-MPS algorithm integrates the technique of metamodeling and a novel discriminative sampling method. It generates more sample points in the neighborhood of the function mode (optimal) and fewer points in other areas as guided by a special sampling guidance function. Basically, C-MPS consists of four steps. The first step is the initial random sampling in which m points are generated, where m is an arbitrary integer that usually increases as the dimension of the problem increases. These m points are called "expensive points" since their function value should be evaluated by the black-box function. Using the points generated in the first step, a linear spline function is fitted:

$$\hat{f}(x) = \sum_{i=1}^{m} c_i \|x - x^{(i)}\| \tag{1}$$

where $\hat{f}(x)$ is an approximation of the original black-box function $f(x)$ with coefficients c; $x^{(i)}$, $i=1,\ldots,m$ are sample points. Then another function is defined, $g(x)=c_0-\hat{f}(x)$, as the sampling guidance function, where c_0 is a constant. In the second step, N (usually large) number of points are randomly sampled and sorted in the descending order of their guidance function value. These points are called "cheap points" since their function values are evaluated by the linear guidance function, not the black-box function. Their function values will be referred as "approximation values." The third step is discriminative sampling of m points from the "cheap points." The inverse cumulative density function (CDF) sampling method is used for discriminative sampling, in which the CDF is constructed based on the approximation values. Hence, the new m sample points have the tendency to concentrate around the minimum of $\hat{f}(x)$. For better convergence, a speed-up factor is used in this step to increase the probability of sampling better points. The fourth step involves a quadratic regression in a subarea around the current best solution. When the approximation in the subarea is sufficiently accurate, local optimization can be performed in this subarea to obtain the optimum.

In short, C-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 optima. These mechanisms are based on the following approximations:

1. Approximation of the entire function. By fitting the metamodel described in Eq. (1) with all the previous expensive points, C-MPS tries to improve the metamodel (approximation) accuracy. The linear spline function is used for approximation due to its simplicity and its interpolation of existing points. Also because the metamodel is used only for guiding the sampling in MPS, the accuracy of the metamodel is not as critical as the cases when metamodels are used as surrogates in optimization. Meanwhile, MPS does not dictate the exclusive use of the linear spline; other metamodel types can be applied in lieu of the linear spline model.
2. Approximation around the attractive subareas. Due to the discriminative sampling, more and more points are generated around attractive regions, so the approximation accuracy of theses regions gradually increases.

For problems of discrete variables, the sampling and approximation will be performed only at valid discrete points. The quadratic regression in a subarea is no long applicable. Therefore, new methods are needed for optimization on expensive black-box functions with discrete variables.

3 Development of Basic Discrete Variable Mode Pursuing Sampling

This section describes the basic D-MPS without the double-sphere strategy.

3.1 Basic Discrete Variable Mode Pursuing Sampling Algorithm. Suppose an n-dimensional black-box function, $f(x)$, has to be minimized over a domain $S[f]$. As in most of the discrete engineering problems, it is assumed that each variable x_i has a predefined index set I_i. Without loss of generality and with the assumption that each set I_i has i_k members, $S[f]=\{X_1,X_2,\ldots,X_I\}$ can be written, where $I=\Pi_{k=1}^{n} i_k$ and X_i is a vector of length n, representing a point in the solution space.

Assume that $f(x)$ is positive on $S[f]$. In general, if $f(x)$ is negative for some $X \in S[f]$, then a positive number can be always added to $f(x)$, so that it becomes positive on $S[f]$. Note that mini-

mizing $f(x)$ is equivalent to maximizing $-f(x)$. The proposed algorithm consists of three main tasks: generating cheap points, approximation, and discriminative sampling.

1. Generating cheap points. $S[f]$ is the domain from which cheap points will be generated. In the first step, a discrete space $S_N[f] \subset S[f]$ is generated consisting of N uniformly distributed base points in $S[f]$. $S_N[f]=\{X_1,\ldots,X_N\}$ consists of the cheap points in the current iteration. This sampling is performed in the normalized index space defined by $I=\Pi_{k=1}^{n} i_k$. The sample points are mapped back to its original $S[f]$ to find its real variable values when function evaluation is performed. The normalization brings variables of the different unit and scale to the same [0 1] region. The sampling in the index space handles variables of nonuniformly distributed valid discrete values. Such a sampling strategy is used throughout the D-MPS method. After obtaining the sample points, the infeasible points are deleted before proceeding to the next step. In the proposed algorithm, the domain for generating $S_N[f]$ is not always $S[f]$. The double-sphere strategy, which will be introduced in Sec. 4, controls the domain for generating cheap points.
2. Approximation. The second task is the linear spline interpolation step. At the first step, m points are randomly generated. Then at each iteration, the interpolation is done based on all the existing expensive points. All of the expensive points form a set $E[f]$. Then, the linear spline function in Eq. (1) is used to fit the points and $\hat{f}(x)$ is derived. Note that $f(x)$ is defined over a discrete set $S[f]$, whereas $\hat{f}(x)$ is continuous. Therefore, there is a need to discretize $\hat{f}(x)$ before continuing to the next step. For all the discrete points in $S_N[f]$, their approximated function values are obtained as $p[k]=\hat{f}(X_k)$ for $k=1,2,\ldots,N$.
3. Discriminative sampling. First, all the points in $S_N[f]$ are sorted by their approximated function values $p[k]$. Because of the desire to raise the sampling probability for points of lower function value for the purpose of minimization, $g[k]=c_0-p[k]$ is defined as the sampling guidance function, where c_0 is a constant larger than or equal to the maximum of $p[k]$ to ensure that $g[k]$ is always positive. Then, $g[k]$ for each point is divided by the sum of all $g[k]$ values to generate a value analogous to a probability density. Then, these "probability density" values are added up to generate an analogous CDF with respect to the sorted sample points named as $G[k]$. At the end, m random numbers inside the interval [0, 1] are generated, and m new samples are selected at which the corresponding CDF values equal to these random numbers along the sorted sample. As a result, more points of lower f value will be selected due to the convex and monotonically increasing shape of CDF. All the new samples are then added to the $E[f]$ with their actual function values. In the next iteration, these new sample points will be used in addition to all existing expensive points for interpolation. The next section explains the basic D-MPS algorithm with an example.

3.2 Sampling Example for Basic Discrete Variable Mode Pursuing Sampling. For the ease of understanding, the basic D-MPS (without the double-sphere strategy) is illustrated with the well known six-hump camel back (SC) function, which is known having six local minima [11]. The mathematical expression of SC is

$$f_{SC}(x) = 4x_1^2 - \frac{21}{10}x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \tag{2}$$

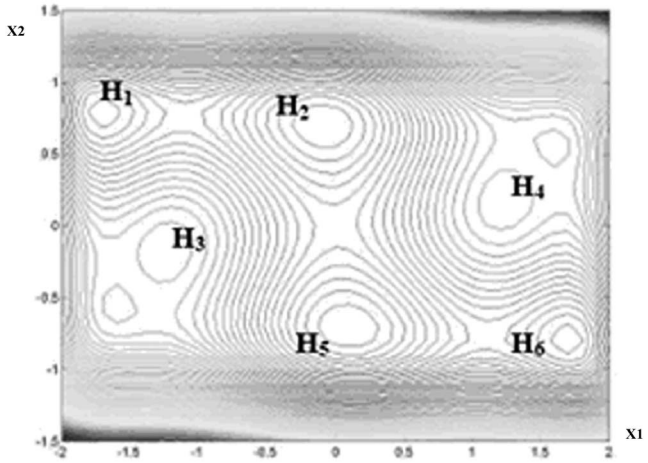

Fig. 1 Contour plot of the SC function

Although this function in its original form has two continuous variables, we discretize the variables with "0.01" distance between each index as follows: $x_1,x_2 \in \{-2,-1.99,\ldots,2\}$. A contour plot of the continuous SC function is shown in Fig. 1, where the H2 and H5 indicate the two global optima at (−0.09, 0.71) and (0.09, −0.71), respectively, with $f_{min}=-1.0316$. With discretization, the domain set $S[f]=\{(-2,-2),(-1.99,-2),\ldots,(2,2)\}$ can be defined. The three main steps of D-MPS algorithm are described below:

1. Generating cheap points. By assuming $N=100$, $S_N[f]$ is uniformly constructed by selecting N members of $S[f]$. By assuming $m=6$, $X_1,X_2,\ldots,X_6$ are then randomly selected from $S_N(f)$. These points will be added to $E(f)=\{(X_1,f(X_1)),\cdots,(X_6,f(X_6))\}$. $E[f]$ is the set for all expensive points.
2. Approximation. In this step, $\hat{f}$ is constructed by fitting Eq. (1) to the m sample points.
3. Discriminative sampling. Calculate all $p[k]=\hat{f}(X_k)$ for $k=1,2,\ldots,N$. Then a discrete function $g[k]=c_0p[k]$ for $k=1,2,\ldots,N$ is constructed for all of the members of $S_N[f]$. c_0 is set to be the maximum of $p[k]$ so that $g[k]$ will be positive. Sorting the points in $S_N[f]$ in the descending order of the values of $g[k]$, the sequence of corresponding function values of $g[k]$ is plotted in Fig. 2(*a*). Next, the $g[k]/\Sigma_{k=1}^{N}g[k]$ will be calculated, which result in $\overline{g[k]}$ or "probability" for each cheap point. By using the sorted $S_N[f]$, the $G[k]$ for X_k will be calculated, which is the "cumulative probability density" for each point, as shown in Fig. 2(*b*). Then $m=6$ points are drawn with replacement according to the distribution of $G[k]$. All such points form the new sample points $X_7,X_8,\ldots,X_{12}$. These new samples will be passed to the black-box function and receive the corresponding function values. Accordingly, the number of function evaluation is increased by m and the set of expensive points will be updated as $E(f)=\{(X_1,f(X_1)),\cdots,(X_{12},f(X_{12}))\}$. Continuing the process until convergence, the final set of sample points are plotted in Fig. 5(*a*), which are compared with the complete D-MPS (with the double-sphere strategy.) The convergence criterion is set to be the maximum number of iterations.

4 Development of Complete Discrete Variable Mode Pursuing Sampling With Double-Sphere Strategy

In C-MPS, Wang et al. [23] used quadratic fitting with a speed control factor to control the convergence of MPS. For D-MPS,

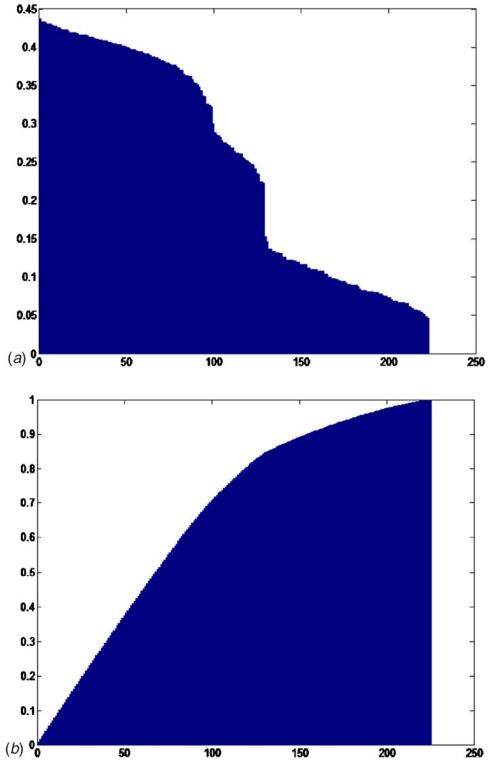

Fig. 2 Plot of the ranked point distributions of (*a*) $g[k]$ and (*b*) $G[k]$

since the optimization region is not continuous, the quadratic fitting approach no longer applies. Moreover, it is more difficult to reach the exact global optimum in a discrete domain than in a continuous domain, for which the optimum can be closely approached. In this paper, a novel double-sphere strategy has been developed for D-MPS. To distinguish from the basic D-MPS, D-MPS with the double-sphere strategy is called the complete D-MPS.

4.1 Double-Sphere Concept. Double-sphere consists of two hyperspheres with the center on the current optimum (the best solution, which has been found in the algorithm at the current iteration) and their radii are dynamically changing based on the information from previous iterations. Sampling should be done inside these hyperspheres. The two hyperspheres dynamically control the "exploration" and "exploitation" functions of the search algorithm. Description of the two hyperspheres is as follows.

4.1.1 Hypersphere S. This hypersphere with a small initial radius shrinks to a new radius $R_s^* \alpha$ if there is no improvement after n_α iterations, where R_s is the current radius of S and α is a coefficient within $[0, 1]$ for size control. If there is an improvement in the current iteration, i.e., a better solution is found by the algorithm, hypersphere S increases in size with a new radius R_s/α.

4.1.2 Hyper-Sphere B. This hypersphere starts with a large initial radius R_b and acts as a dual of hypersphere S. It grows to a new radius R_b/α if there is no improvement after n_α iterations and shrinks to $R_b^* \alpha$ if there is an improvement in the current iteration. At iteration i of the double-sphere strategy, neither R_s nor R_b can be larger than the original design space, $R_{b,0}$.

The two hyperspheres, B and S, are designed in order to achieve the balance between exploration and exploitation in the search process. Exploration aims to extend the search scope, while exploitation intensifies the search in a local area thus speeds up the search. There is no definite task assignment of the two hyperspheres as which controls exploration or exploitation, as one might expect. In an iteration of the search, S could have a bigger radius than B or vice versa, and they may even have equal radii. In the case when the search yields marginal or no improvement, hypersphere S shrinks for exploitation in the attractive region, while hypersphere B enlarges for exploration of new promising regions. On the other hand, if the search procedure is inside an attractive region in which consecutive improvements occur, then S tends to grow to avoid being trapped intro a local optimum, i.e., exploration, and B tends to shrink for exploitation. The dynamic coupled behavior of the two hyperspheres is the core of the double-sphere strategy, which will be further discussed in following sections.

4.2 Discrete Variable Mode Pursuing Sampling With Double-Sphere Strategy. In the first iteration, the initial radii for B and S are set as $R_{b,0}$ and $R_{s,0}$, respectively, where $R_{b,0} \geqslant R_{s,0} \geqslant R_{\min}$. $R_{s,0}$ is one of the parameters of the proposed algorithm, which denotes the initial radius of the smaller hypersphere. $R_{\min}$ is set to be the smallest radius that encloses m discrete points in $S[f]$, noting for discrete variables there are only a finite number of valid points in a given region. Therefore, when a discrete variable space is given, $R_{\min}$ can then be determined. $R_{b,0}$ is set to be the smallest real number that contains all the points inside the feasible region $S[f]$.

Recall that the basic D-MPS is composed of three main steps, i.e., generating cheap points, approximation, and discriminative sampling. In the complete D-MPS, these main steps will be performed on the domains provided by the double sphere. For the objective function $f(x)$ on domain $S[f]$, the double-sphere strategy dynamically provides a domain $D_1 \cup D_2 \subset S[f]$; D_1 is the domain inside the smaller hypersphere, and D_2 is the domain between the smaller hypersphere and bigger hypersphere. The three main steps of D-MPS are performed on both D_1 and D_2. In other words, generating cheap points, approximation, and discriminative sampling are called two times at each iteration of the algorithm. Therefore, at each iteration, one will have $m/2$ new samples or expensive points from D_1, and $m/2$ new samples or expensive points from D_2. Although the number of expensive points from each subset can also be dynamically changed, it is fixed in this work for the simplicity of algorithm parameter setting. These two groups of sample points are generated on the basis of two approximations:

- Approximation inside the smaller hypersphere (D_1). This approximation will in general be of higher quality than the one in the original space, since it is performed inside a smaller hypersphere. This approximation provides rich information about the black-box function in the attractive region.
- Approximation for the space between the smaller hypersphere and bigger hypersphere (D_2). This approximation results in an approximation of a solution space inside the bigger sphere excluding the smaller hypersphere.

The double-sphere strategy uses a Boolean input $b_{\text{iter}}=0$ or 1 to indicate if the new sample point in the current iteration has a better objective function value than the previously found optimum. Steps of the sampling procedure of the complete D-MPS algorithm are as follows. Figure 3 shows the flowchart of the proposed method.

The following are the inputs:

- $S[f]=\{X_1, X_2, \ldots, X_I\}$: Domain or the discrete set over which $f(x)$ is to be minimized.
- N: Number of cheap points to be generated in each iteration.
- m: Number of expensive points or sample points to be generated in each iteration.

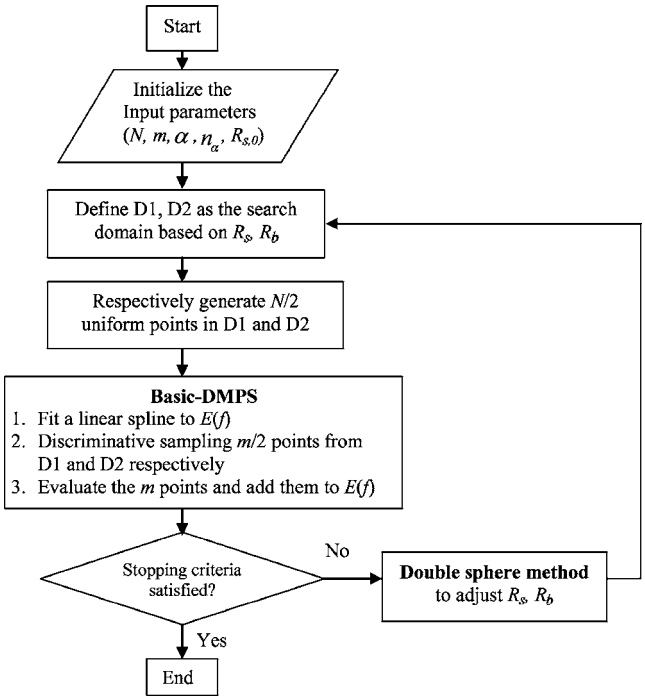

Fig. 3 Flowchart of the proposed complete DMPS algorithm

- α: The coefficient for increasing/decreasing the radius of hyperspheres.
- n_α: The number of consecutive iterations to activate the double-sphere strategy.
- $R_{s,0}$: The initial radius of hyper-sphere S.
- $f(x)$: The (expensive) black-box objective function.

The following are the outputs:

- The optimum X^* and $f(X^*)$ from the Set $E[f]$, which has all the expensive points with their function values.
- nEval: number of expensive function evaluations.

Step 1. Initialize the five parameters N, m, α, n_α, and $R_{s,0}$; define D_1, D_2, and set Counter=0, where "Counter" counts the number of consecutive iterations at which no improvement is made.

Step 2. Generate a uniform distribution (in the index space) of $N/2$ points from D_1 and D_2, respectively, to form discrete space sets $S_{N,1}[f]$ and $S_{N,2}[f]$. Delete the infeasible points.

Step 3. Perform basic D-MPS in D_1 and D_2, respectively, to generate $m/2$ points from $S_{N,1}[f]$ and $S_{N,2}[f]$, respectively. Evaluate the m points.

Step 4. Add the m points and their function values to $E[f]$; $E(f)=\{(X_1,f(X_1)),\ldots,(X_{\text{iter}^*m},f(X_{\text{iter}^*m}))\}$, where "iter" is the number of iterations thus far. Find the lowest function value and update X^*_{iter} as the best solution. If the stopping criterion is satisfied, i.e., the maximum number of iterations is reached, go to Step 7. Otherwise, if there is an improvement, set $b_{\text{iter}}=1$; else $b_{\text{iter}}=0$.

Step 5. Double-sphere strategy.

Step 5.1. If $b_{\text{iter}}=1$ go to Step 5.2, else Counter=Counter+1, go to Step 5.3.

Step 5.2:

- Move the center of both hyper spheres to the current optimum
- Increase the radius of smaller hypersphere to R_s/α
- Reduce the bigger hypersphere to αR_b

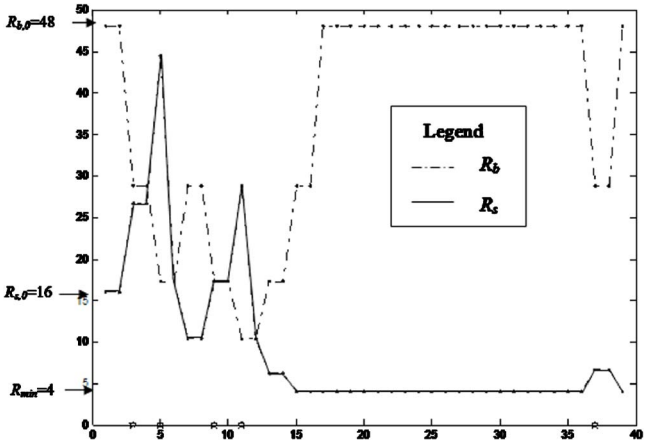

Fig. 4 Hypersphere radii over iterations

- Check if the inequalities $R_{\min}<=R_b$ and $R_s<=R_{b,0}$ holds, if not, set them to the closest bound ($R_{\min}$ or $R_{b,0}$).

Step 5.3. If Counter$<n_\alpha$, go to Step 6.

If Counter$=n_\alpha$
If ($R_s>R_{\min}$ or $R_b<R_{b,0}$)

- Reduce the smaller radius to $R_s*\alpha$
- Increase the bigger radius to R_b/α
- Check for the bounds as in Step 5.2
- Set "Counter"=0

If ($R_s=R_{\min}$ and $R_b=R_{b,0}$)

- Set $R_b=R_{\min}$
- Set "Counter"=0

Step 6. iter=iter+1; define D_1,D_2 based on the new radii (R_s,R_b); generate a uniform distribution of N points from D_1,D_2 to form a discrete space set $S_{N,1}[f]$, $S_{N,2}[f]$. Delete the infeasible points. Go to step 3.

Step 7. Report the X^*_{iter}, and its corresponding variable as the global minimum. Report the number of function evaluations nEval=iter$^*m$.

During an optimization process, if there are consecutive improvements, the double-sphere method may lead to a situation at which $R_b=R_{\min}$ and $R_s=R_{b,0}$. This situation usually happens at the beginning of the search when it is desirable to exploit regions that lead to improvements. On the other hand, if there are no consecutive improvements, R_s may reach its minimum $R_{\min}$ and R_b may reach its maximum $R_{b,0}$. This situation is called a stable state. The stable state happens usually at last iterations of the D-MPS algorithm. Figure 4 shows the variation of both radii versus the iterations for the gear train test problem (to be discussed in Sec. 5). The stars on the horizontal line indicate the iterations when an improvement is observed.

From Fig. 4, one can see that the search reaches the stable condition at which $R_b=R_{b,0}$ and $R_s=R_{\min}$ at Iteration 15 and continues to Iteration 37, largely due to the fact that further shrinking S or enlarging B is not possible in these iterations. Therefore in Step 5.3 of the complete D-MPS algorithm, a "bouncing-back" feature is designed. That is, when the stable condition is reached ($R_b=R_{b,0}$ and $R_s=R_{\min}$ and Counter=n_α), hypersphere B is forced to be of the same radius of $R_{\min}$. As it continues with no improvement, hypersphere B will enlarge gradually to $R_{b,0}$ (while S takes $R_{\min}$) so that the entire space is searched again in a layer-by-layer manner for potential improvements.

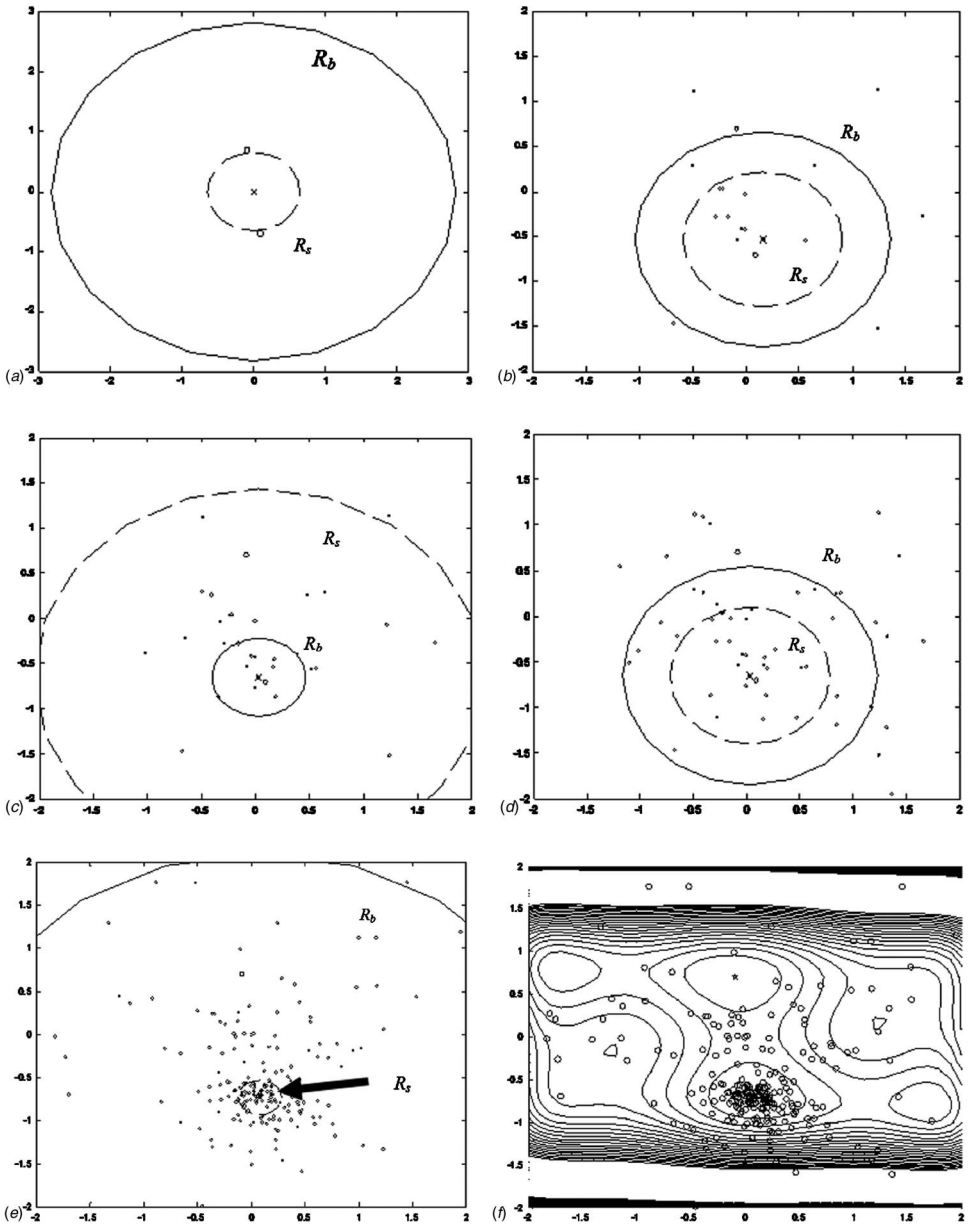

Fig. 5 Example for D-MPS with the double-sphere strategy for SC function

4.3 Example for the Complete Discrete Variable Mode Pursuing Sampling. Consider the SC function defined in Eq. (2) in Sec. 3.2. The parameters of D-MPS are as follows:

- The number of expensive points at each iteration, $m=4$.
- The number of cheap points generated at each iteration, $N=32$.
- The radius of the small hypersphere, $R_{s,0}=0.5$.
- The coefficient for increasing/decreasing the radius, $\alpha=0.7$.
- The number of steps for the double-sphere strategy, $n_\alpha=2$.

The solution space for SC function is $[-2,2]^2$; the center for both hyperspheres S and B is (0, 0), and $R_{b,0}=\sqrt{8}$. Since both variables are discretized to an interval of size 0.01, $R_{\min}=0.01\sqrt{2}/2$. The initial state is shown in Fig. 5(*a*), where the circles show the twin global optima and the cross shows the center for hyperspheres B and S. Hypersphere S is shown with the dotted line and hypersphere B is shown with the solid line.

The D-MPS algorithm runs for 30 iterations. Hyperspheres at Iterations 2, 4, 8, and 20 are shown in Figs. 5(*b*)–5(*f*), respectively. In the second iteration of D-MPS, as shown in Fig. 5(*b*), a new best solution improves the current optima, and the center of both hyperspheres moves to the new optimum. In addition, the radius of hypersphere S is increased to avoid being trapped to a local optimum, while B is decreased for potential exploitation. In the next iterations, the algorithm continues to find better solutions and hypersphere B becomes smaller than S, as it can be noted from Fig. 5(*c*). From the fourth iteration to the eighth iteration, no improvement is made. Therefore, hypersphere S starts to shrink for exploitation around the region of current optima, while B starts to grow to explore other regions in the search space. As a result, S becomes smaller than B, as shown in Fig. 5(*d*). At the Iteration 20, as it is shown in Fig. 5(*e*), most of the sample or expensive points are around the mode of the function, while in the meantime the entire solution space is explored. Figure 5(*f*) shows the contour plot of the SC function with all the sample points generated by the D-MPS algorithm.

The basic D-MPS algorithm (without the double-sphere strategy) is compared with the complete D-MPS (with the double-sphere strategy) for the SC problem. Figure 6 shows the expensive

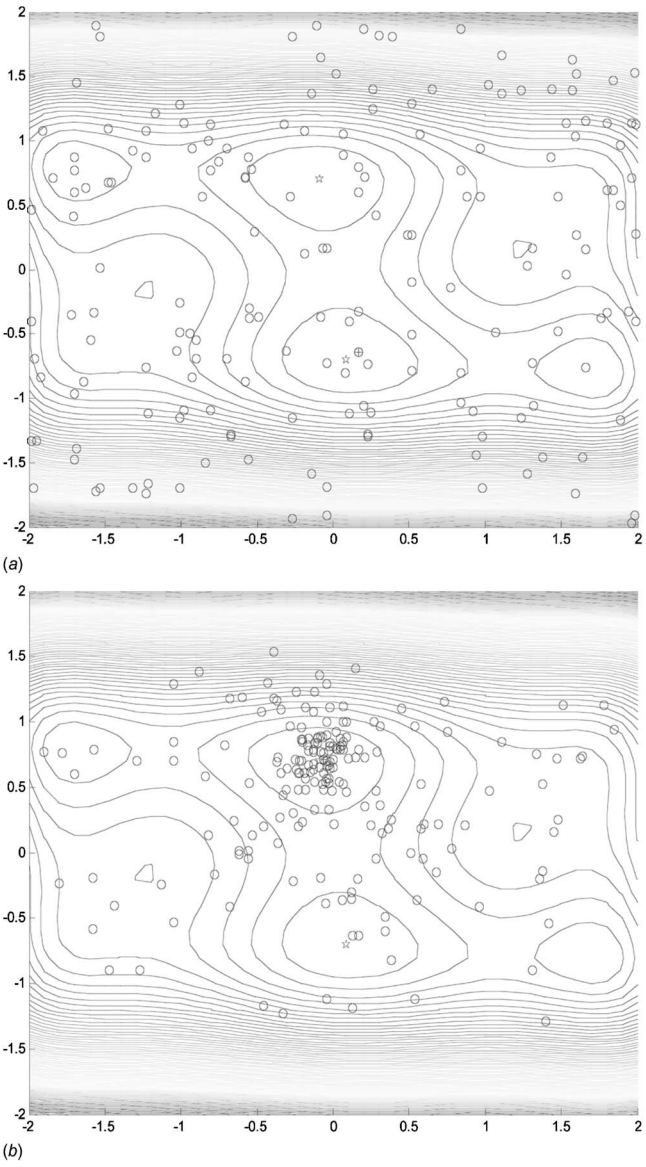

Fig. 6 Result comparison between (*a*) the basic D-MPS and (*b*) complete D-MPS

points with small circles. In both cases, 150 expensive points have been generated. Figure 6(*b*) generated by the complete D-MPS shows a concentration of points around one of the global optima, while no obvious pattern is observed in Fig. 6(*a*) for the basic D-MPS.

4.4 Analysis of Variance Study of Discrete Variable Mode Pursuing Sampling Parameters. As discussed in Sec. 4.1, there are five parameters that have to be set for the proposed method. In this section, the result of analysis of variance (ANOVA) study of the parameters on the SC function will be discussed. As it can be seen from Table 1, each parameter has three different levels, and ten runs are executed for each setting. For a full factorial design, 2430 runs are executed in total. Table 1 shows the different levels for the variables. For a two-variable problem, N is taken as the square of the number of indices for each variable, N_d.

The total number of function evaluations needed to reach the known global optimum is the output factor of concern. The average function value for every ten runs is used for the ANOVA study. The ANOVA results are listed in Table 2 and the parameter interactions are plotted in Fig. 7. The first column of Table 2 shows the source of variability with some interaction terms omitted due to negligible effects. The second shows the mean squares due to each source; the third is the F statistics, and the fourth is the p value for the F statistics. If p is sufficiently small (<0.01), the associated factor has a nontrivial effect on the result.

Table 1 Variables for ANOVA study

Factor	No. Levels	Values
N	3	16^2, 32^2, 64^2
m	3	4, 8, 10
n_α	3	1, 4, 8
α	3	0.5, 0.75, 0.95
$R_{s,0}$	3	0.2, 0.7, 1.4

It can be seen from Table 2 that N is not sensitive with respect to the number of function evaluations (nEval). The interaction effect between N and other parameters is also negligible (see Fig. 7). It thus suggests that N can be set to a low level to decrease the computation cost. Parameter α is the most sensitive variable. When α is high (i.e., little changes in radii), the interactions between α and other parameters, especially R_s and n_α, have high influence on the number of function evaluations. This is understood that when there is little change in radii of the hyperspheres, the double-sphere strategy does not function actively and therefore D-MPS performs as a basic D-MPS, for which R_s and n_α play a more important role in algorithm control. Therefore, parameter α is recommended at a low setting, e.g., 0.5. If α is too low, however, the algorithm will be too aggressive and may converge prematurely. From Fig. 7, R_s, n_α, and m can also be chosen at a

Table 2 ANOVA results for all the parameters

Source	Mean square	F	p
N	1681	0.34	0.716
m	561406	112.81	0.000
n_α	1461420	293.66	0.000
α	2694001	541.34	0.000
$R_{s,0}$	1280365	257.28	0.000
m^*n_α	40663	8.17	0.000
$m^*\alpha$	79712	16.02	0.000
$m^*R_{s,0}$	71000	14.27	0.000
$n_\alpha^*\alpha$	241748	48.58	0.000
$n_\alpha^*R_{s,0}$	165153	33.19	0.000
$\alpha^*R_{s,0}$	571855	114.91	0.000
$m^*\alpha^*R_{s,0}$	39489	7.94	0.000
$n_\alpha^*\alpha^*R_{s,0}$	67093	13.48	0.000

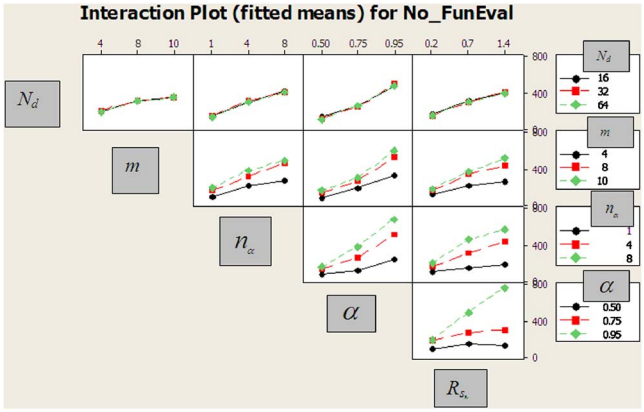

Fig. 7 Parameter interaction plot (mean versus *n*Eval)

Table 3 Results of D-MPS on SC

Run No.	*n*Iter	*n*Eval	X1*	X2*	Y*
1	16	64	0.1	−0.72	−1.0309
2	16	64	0.11	−0.72	−1.0298
3	8	32	0.09	−0.7	−1.0303
4	12	48	−0.07	0.7	−1.0291
5	16	64	0.07	−0.7	−1.0291
6	22	88	−0.08	0.7	−1.0301
7	15	60	−0.07	0.7	−1.0291
8	19	76	0.1	−0.71	−1.0311
9	15	60	0.1	−0.7	−1.0298
10	11	44	0.08	−0.71	−1.0312
Avg.	15	60	⋯	⋯	−1.0301

low level to reduce the total number of function evaluations. One is cautioned that when parallel computation is possible and the goal is to reduce the total computing time at the expense of more function evaluations, these parameters can be set at higher levels. Although the parameter study is based on the SC function, it helps one understand the influence of the parameters and their interactions, and provides some guidelines for application as in the test cases in Sec. 5.

5 Test of Discrete Variable Mode Pursuing Sampling

Three test problems have been solved using the proposed D-MPS algorithm. These problems are (1) discrete six-hump camel back (SC) function [11], (2) gear train problem [4], and (3) pressure vessel problem [25]. In the following, these problems will be discussed and D-MPS will be compared with other methods from the literature on the results. As described in Refs. [21,26] for black-box functions, the number of function evaluations is a more appropriate indicator of computation efficiency than CPU time. Therefore, this work also uses the number of function evaluations for efficiency comparison.

5.1 SC Problem. The six-hump camel back (SC) function is defined in Eq. (2) and discussed in Sec. 3.2. The variables are assumed to be discretized as follows: $[-2,-1.99,\ldots,1.99,2]$. Parameters are set to $(N,m,R_{s,0},\alpha,n_\alpha)=(256,4,0.2,0.5,1)$. Ten runs have been performed. Results of D-MPS are given in Table 3, where *n*Iter stands for the number of iterations and *n*Eval stands for the number of function evaluations. The known analytical optima for the discretized problem are (−0.09, 0.71) and (0.09, −0.71) with the function value $f^*=-1.0316$. From Table 3, D-MPS consistently find a close-to-optimum solution with a modest number of function evaluations.

5.2 Gear Train Problem. The objective of the GT problem is to optimize the gear ratio for the compound GT, as shown in Fig. 8. The gear ratio for a reduction GT is defined as the ratio of the angular velocity of the output shaft to that of the input shaft (Table 4). In order to produce the desired gear ratio, the compound GT is constructed out of two pairs of the gear wheels, "d−a" and "b-f." The gear ratio i_{tot} between the input and output shafts can be expressed as

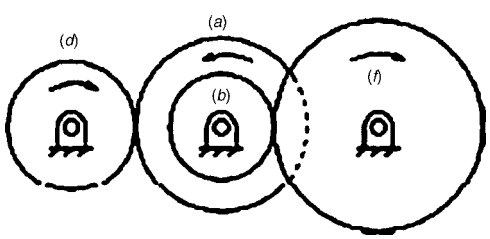

Fig. 8 GT problem

Table 4 Results of D-MPS on GT

Run No.	*n*Iter	*n*Eval	Function value	X^*
1	500	2000	$2.3078E-011$	15 26 53 51
2	215	860	$2.7009E-012$	19 16 49 43
3	21	84	$2.7009E-012$	19 16 49 43
4	500	2000	$1.8274E-008$	14 16 42 37
5	500	2000	$9.9216E-010$	24 13 46 47
6	500	2000	$1.0936E-009$	17 15 57 31
7	290	1160	$2.7009E-012$	16 19 43 49
8	500	2000	$1.3616E-009$	17 14 55 30
9	500	2000	$1.1661E-010$	17 22 54 48
10	500	2000	$2.3576E-009$	15 18 39 48
Avg./ Median	Avg: 402	Avg: 1610	Median: $5.5439E-010$	

$$i_{\text{tot}}=\frac{w_o}{w_i}=\frac{z_d z_b}{z_a z_f} \tag{3}$$

where w_o, w_i are the angular velocities of the output and input shafts, respectively, and z denotes the number of teeth on each gear wheel. It is desirable to produce a gear ratio as close as possible to 1/6.931, which is given as a standard parameter from the designer [27]. For each gear, the number of teeth must be from 12 to 60. The design variables are denoted by a vector $\mathbf{X}=[x_1,x_2,x_3,x_4]=[z_d,z_b,z_a,z_f]$. Hence, x_1, x_2, x_3, and x_4 are the numbers of teeth of Gears d, b, a, and f, respectively, which must be integers. The optimization problem is expressed below with a known optimum of zero:

$$\min f=\left(\frac{1}{6.931}-\frac{x_1x_2}{x_3x_4}\right)^2$$

$$\text{s.t. } 12\leqslant x_i\leqslant 60 i=1,\ldots,4 \tag{4}$$

For the GT problem, ten runs have been executed with the following parameter setting, $m=4$, $N=1250$, $\alpha=0.7$, $n_\alpha=7$, and $R_{s,0}=0.33$ (all variables are normalized to [0, 1]). The Table 5 compares the best solution of D-MPS with those from gradient-based approaches and different evolutionary algorithms.

The methods that have been compared with D-MPS in Table 5 are as follows: mixed integer branch and bound using the sequential quadratic programming algorithm (MIBBSQP) [28], integer-discrete-continuous nonlinear programming algorithm (IDCN-LPC) [29], SA [30], mixed-variable evolutionary programming (MVEP) [25], mixed-integer hybrid evolutionary algorithm (MI-HDE) [28]. All the results from references are based on 50,000 function evaluations, as compared to 1610 needed by D-MPS.

Authors in Ref. [31] tested different ant algorithms for the GT problem. Their results are compared to the D-MPS in Table 6. Their results are based on an average of 30 runs with 10,000 function evaluations per run. The methods which have been compared to D-MPS in Table 6 include ant system (AS), ant colony system (ACS), max-min ant system (MMAS), rank based ant system (RBAS), and best-worst ant system (BWAS) [31]. As can be seen from Tables 5 and 6, D-MPS leads to better solutions than most of the methods, while the number of function evaluations is much lower.

5.3 Design of a Pressure Vessel. The pressure vessel problem is to design a compressed air storage tank with a working pressure of 3000 psi and a minimum volume of 750 ft^3. The schematic of a pressure vessel is shown in Fig. 9. The cylindrical pressure vessel is capped at both ends by hemispherical heads. Using rolled steel plate, the shell is to be made in two halves that are joined by two longitudinal welds to form a cylinder. Each head is forged and then welded to the shell. Let the design variables be denoted by

Table 5 Performance comparison between D-MPS and other algorithms on GT

Items	MIBB-SQP [27]	IDCN-LP [28]	SA [29]	MVEP [25]	MIHDE [27]	D-MPS
x_1	18	14	30	30	19	16
x_2	22	29	15	15	16	19
x_3	45	47	52	52	43	43
x_4	60	59	60	60	49	49
$f(X)$	5.7×10^{-6}	4.5×10^{-6}	2.36×10^{-9}	2.36×10^{-9}	2.7×10^{-12}	2.7×10^{-12}
Gear ratio	0.1466	0.1464	0.1442	0.1442	0.1443	0.1443
Error (%)	1.65	1.47	0.033	0.033	0.00047	0.00047
*n*Eval	50,000	50,000	50,000	50,000	50,000	1610

Table 6 Performance comparison between D-MPS and ant algorithms on GT

	AS	ACS	MMAS	RBAS	BWAS	D-MPS
Best (10^{-12})	2358	2.71	245970	2.71	27267	2.71
Avg. (10^{-4})	6.45	5.56	1070	45.71	321.02	$5.54E-6$
Worst (10^{-2})	1.92	1.61	148.77	6.82	25.09	$1.83E-6$
*n*Eval	10^4	10^4	10^4	10^4	10^4	1610

the vector $\mathbf{X}=[x_1,x_2,x_3,x_4]$. x_1 is the spherical head thickness, x_2 is the shell thickness, x_3 and x_4 are the radius and length of the shell, respectively.

The objective function is to reduce the combined costs of materials, forming and welding of the pressure vessel. The constraints are set in accordance to ASME codes. The mathematical model of the problem is:

$$\min f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$\text{s.t. } g_1(X) = 0.0193x_3 - x_1 \leqslant 0$$

$$g_2(X) = 0.00954x_3 - x_2 \leqslant 0$$

$$g_3(X) = 750 \times 1728 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leqslant 0$$

$$g_4(X) = x_4 - 240 \leqslant 0 \tag{5}$$

where the design variables x_3 and x_4, are continuous and x_1, x_2 are integer multiplies of 0.0625. We consider continuous variables as discrete ones with 0.1 increment. All the variables are in the following ranges: $x_1 \in [1.0, 1.375], x_2 \in [0.625, 1.0], x_3 \in [25, 150]$, $x_4 \in [25, 240]$ Table 7 shows the result of D-MPS on pressure vessel for ten runs. For this test, different parameters are set for each run so that the performance of D-MPS can be observed for a wide scope of parameter settings ($n_\alpha \in [1,3], R_S \in [0.6, 0.7], \alpha \in [0.5, 0.6]$). The best results from different methods are compared with that of D-MPS for this problem, as listed in Table 8. As it can be noted, D-MPS leads to better quality results when compared with other methods. The number of function evaluations required by D-MPS on the problem is only 400, in contrast to 50,000 in other methods shown in Table 8.

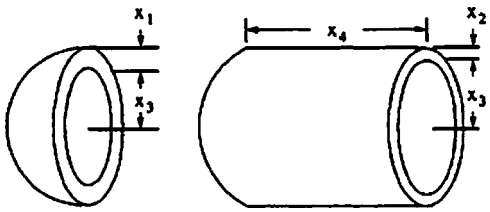

Fig. 9 Pressure vessel

6 Generalization of Double-Sphere Strategy

In the current D-MPS algorithm for discrete variables, the discrete indices are normalized into the domain [0, 1]. The normalization handles problems having variables in different units and scales. The double-sphere strategy, however, may encounter difficulties when there is a significant difference in the number of indices for different variables, even with normalization. For example, for handling binary variables as well as continues variable in a mixed variable problem, binary variables have only two indices, while a continuous variable can be represented as a discrete variable with infinite number of indices. In this situation, a d-dimensional hyper sphere that is symmetrical with respect to all d variables is no longer a good measure for providing a domain for optimization. Therefore, a generalization of the double-sphere strategy is needed.

In the original form of double-sphere strategy, for a discrete d-dimensional function f with support $S[f] \subset I_1 I_2, \ldots, I_d$, where I_j $(j=1,\ldots,d)$ is the index set for variable x_j, the hypersphere B provides a domain at the ith iteration,

Table 7 Results of D-MPS for the pressure vessel problem

Run No.	*n*Iter	*n*Eval	Function Value	X^*
1	100	400	7197.6932	1, 0.625, 49.5, 102.8
2	100	400	7249.9739	1.0625,0.625,55,66.5
3	100	400	7140.2116	1,0.625,50.9,93.3
4	100	400	7351.9418	1.0625,0.625,52.4,81.8
5	100	400	7279.9775	1.0625,0.625,54.8,68.2
6	100	400	7178.2013	1, 0.625, 51.3, 92.2
7	100	400	7241.1644	1.0625,0.625,53.7, 72.5
8	100	400	7372.8138	1.0625,0.625,54,74.4
9	100	400	7155.836	1,0.625,51,93.2
10	100	400	7072.9156	1,0.625,51.3,89.2
Avg./ Median	⋯	⋯	7224.073/ 7219.429	

Table 8 Comparison of D-MPS and other methods for the pressure vessel problem

Items	MIBB-SQP [27]	IDCNLP [28]	SA [29]	MVEP [25]	D-MPS
x_1	1.125	1.125	1.125	1.000	1.000
x_2	0.625	0.625	0.625	0.625	0.625
x_3	47.70	48.38	58.29	51.20	51.3
x_4	117.70	111.74	43.69	90.78	89.2
$f(X)$	8129.8	8048.62	7197.7	7108.62	7072.92
nEval	50,000	50,000	50,000	50,000	400

$$S_1(i) = x | x \in S[f], \quad \sum_{j=1}^{d} (X(j) - X_c(j))^2 \leq R_{b,(i-1)}^2 \tag{6}$$

which X_c represents the current optima (a d-dimensional vector). A generalization of Eq. (6) would be

$$S_2(i) = x | x \in S[f], \quad \sum_{j=1}^{d} \frac{i_j}{i^*} (X(j) - X_c(j))^2 \leq R_{b,(i-1)}^2 \tag{7}$$

where $i^* = \max\{i_1, i_2, \ldots, i_d\}$, and i_j represents the cardinal number of the set I_j. If there exists a continuous variable, i^* can be considered as the solution precision of the continuous variable. In other words, instead of having a hypersphere, we will have a hyperellipse. Thus, in this generalized double-sphere strategy with Eq. (7), variables' search domain varies in proportion to the number of indices of the variable. If the cardinal numbers of the set I_j are similar, i.e., $i_j \approx i^*$, Eq. (7) becomes Eq. (6) and the double-sphere strategy will be performed in its original form. In cases that all variables are binary, D-MPS as a general method may not be as effective as dedicated combinatorial optimization algorithms.

7 Conclusion

In this paper, an algorithm for optimization on expensive black-box functions with discrete variables is proposed. The proposed approach can be considered as an extension to the MPS method for continuous variables [23]. Moreover, this work proposes a novel double-sphere method that dynamically controls the two often contradictory "exploration" and "exploitation" behaviors of an optimization process. The double-sphere method is found effective and efficient in improving the D-MPS algorithm. In a more general sense, the double-sphere method provides an original concept for any direct optimization method. Compared to the well-known Trust Region method, the double-sphere method provides more control and flexibility over a search space. It is believed that the double-sphere concept can be applied to other search processes, or design visualization applications.

Numerical results for three test cases show that D-MPS yields high quality results with a modest number of function evaluations. Because of the double-sphere strategy and approximation only in controlled domains, D-MPS has potential for high-dimensional problems, although this needs to be further studied. It is found in practice that D-MPS often finds a high quality region that contains global optima in a few steps. This is due to the coupling effect of the MPS' discriminative sampling strategy and the double-sphere strategy. Future research will focus on examining and developing D-MPS for high dimensional expensive black-box functions.

Acknowledgment

The authors are grateful to the financial support from Canadian Natural Science and Engineering Research Council (NSERC).

References

[1] Khompatraporn, C., Pinter, J. D., and Zabinsky, Z. B., 2005, "Comparative Assessment of Algorithms and Software for Global Optimization," J. Global Optim., **31**, pp. 613–633.

[2] Lampinen, J., and Zelinka, I., 1999, "Mixed Integer-Discrete-Continuous Optimization By Differential Evolution, Part 2: A Practical Example," *Proceedings of MENDEL '99, 5th International Mendel Conference on Soft Computing*, Brno, Czech Republic, Jun. 9–12.

[3] Bussieck, R., Drud, S., and Meeraus, A., 2003, "MINLPLib-A Collection of Test Models for Mixed-Integer Nonlinear Programming," INFORMS J. Comput., **15**(1), pp. 114–119.

[4] Huang, M., and Arora, J. S., 1997, "Optimal Design Discrete Variables: Some Numerical Experiments," Int. J. Numer. Methods Eng., **40**, 165–188.

[5] Zabinsky, Z., 1998, "Stochastic Methods for Practical Global Optimization," J. Global Optim., **13**, 433–444.

[6] Horst, R., and Pardalos, P. M., 1995, *Handbook of Global Optimization*, Kluwer Academic, Dordrecht.

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

[8] Arora, J. S., 1989, *Introduction to Optimum Design*, McGraw-Hill Higher Education, New York.

[9] Gupta, O. K., and Ravindran, A., 1985, "Branch and Bound Experiments in Convex Nonlinear Integer Programming," Manage. Sci., **31**(12), pp. 1533–1546.

[10] Fletcher, R., and Leyffer, S., 1994, "Solving Mixed Integer Programs by Outer Approximation," Math. Program., **66**, 327–349.

[11] Branin, F. H., and Hoo, S. K., 1972, "A Method for Finding Multiple Extrema of a Function of N Variables," *Numerical Methods for Non-Linear Optimization*, F. Lootsma, Academic, New York, pp. 231–237.

[12] Reaume, J., Romeijn, H., and Smith, R., 2001, "Implementing Pure Adaptive Search for Global Optimization Using Markov Chain Sampling," J. Global Optim., **20**, pp. 33–47.

[13] Hedar, A.-R., and Fukushima, M., 2004, "Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization," J. Global Optim., **35**(4), pp. 521–549.

[14] Lemonge, A. C. C., and Barbosa, H. J. C., 2004, "An Adaptive Penalty Scheme for Genetic Algorithms in Structural Optimization," Int. J. Numer. Methods Eng., **59**, pp. 703–736.

[15] Gu, L., 2001, "A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis," *2001 ASME Design Engineering Technical Conferences—Design Automation Conference*, A. Diaz, ed., ASME, Pittsburgh, PA, Sept. 9–12, DAC–21063.

[16] Kiatsupaibul, S., Smith, R. L., and Zabinsky, Z. B., 2001, *Discrete Hit-and-Run Algorithm for Generating Samples from General Discrete Multivariate Distribution*, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI.

[17] Jones, D. R., Perttunen, C. D., and Stuckman, B. E., 1993, "Lipschitzian Optimization without the Lipschitz Constant,," J. Optim. Theory Appl., **79**, pp. 157–181.

[18] Sergeyev, Y., and Kvasov, D., 2006, "Global Search Based on Efficient Diagonal Partitions and Set of Lipschiz Constants," SIAM J. Optim., **16**(3), pp. 910–937.

[19] Zhang, L., and Subbarayan, G., 2002, "An Evaluation of Back Propagation Neural Networks for the Optimal Design of Structural Systems: Part I. Numerical Evaluation," Comput. Methods Appl. Mech. Eng., **191**, pp. 2887–2904.

[20] Akbarzadeh, M. R., and Khorsand, R., 2005, "Evolutionary Quantum Algorithms for Structural Design," *Proceedings of IEEE International Conference on Systems, Men, and Cybernetics*, Oct. 10–12, pp. 3077–3082.

[21] Wang, G. G., and Simpson, T. W., 2004, "Fuzzy Clustering Based Hierarchical Metamodeling for Space Reduction and Design Optimization," Eng. Optimiz., **36**(3), pp. 313–335.

[22] Jones, D. R., Schonlau, M., and Welch, W. J., 1998, "Efficient Global Optimization of Expensive Black Box Functions," J. Global Optim., **13**, pp. 455–492.

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

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

[25] Cao, Y. J., and Wu, Q. H., 1997, "Mechanical Design Optimization by Mixed-Variable Evolutionary Programming," *Proceedings of IEEE Conference on*

Evolutionary Computation, ICEC, Indianapolis, IN, Apr. 13–16.
[26] Droste, S., Jansen, T., and Wegener, I., 2001, *A New Framework for the Valuation of Algorithms for Black-Box Optimization*, University of Dortmund, Dortmund, Germany, CI-118/01.
[27] Sandgren, E., 1990, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization," ASME J. Mech. Des., **112**, pp. 223–229.
[28] Lin, Y. C., Wang, F. S., and Hwang, K., 1999, "A Hybrid Method of Evolutionary Algorithms for Mixed-Integer Nonlinear Optimization Problems," *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, IEEE, Washington, Jul. 6–9, Vol. 3. Cat. No. 99TH8406.
[29] Fu, J. F., Fenton, R. G., and Cleghorn, W. L., 1991, "A Mixed Integer Discrete-Continuous Programming Method and its Application to Engineering Design Optimization," Eng. Optimiz., **17**(3), pp. 263–280.
[30] Zhang, C., and Wang, H. P., 1993, "Mixed-Discrete Nonlinear Optimization With Simulated Annealing," Eng. Optimiz., **21**(44), pp. 277–291.
[31] Capriles, P. V. S. Z., da Fonseca, L. G., Barbosa, H. J. C., and Lemonge, A. C. C., 2005, "Ant Colony Algorithms Applied to Discrete Optimization Problems," *XXVIII Congress Nacional de Matemátical Aplicada e Computacional (CNMAC)*, Sao Paulo, Brazil, Sept. 12–15.