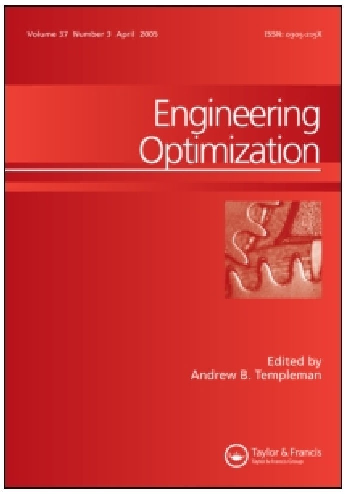

## Engineering Optimization

# Performance study of mode-pursuing sampling method

X. Duan [a]; G. G. Wang [b]; X. Kang [a]; Q. Niu [a]; G. Naterer [c]; Q. Peng [a]
[a] Department of Mechanical and Manufacturing Engineering, University of Manitoba, Winnipeg, MB, Canada [b] School of Engineering Science, Simon Fraser University, Surrey, BC, Canada [c] Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, Ontario

Online Publication Date: 01 January 2009

**To cite this Article** Duan, X., Wang, G. G., Kang, X., Niu, Q., Naterer, G. and Peng, Q.(2009)'Performance study of mode-pursuing sampling method',Engineering Optimization,41:1,1 — 21

**To link to this Article: DOI:** 10.1080/03052150802345995

**URL:** http://dx.doi.org/10.1080/03052150802345995

*Engineering Optimization*
Vol. 41, No. 1, January 2009, 1–21

# Performance study of mode-pursuing sampling method

X. Duan[a], G.G. Wang[b]*, X. Kang[a], Q. Niu[a], G. Naterer[c] and Q. Peng[a]

[a]*Department of Mechanical and Manufacturing Engineering, University of Manitoba, Winnipeg, MB, Canada R3T5V6;* [b]*School of Engineering Science, Simon Fraser University, 250-13450 102 Avenue, Surrey, BC, Canada V3T03A;* [c]*Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, Ontario*

(*Received 25 April 2008; final version received 14 July 2008*)

Since the publication of the authors' recently developed mode-pursing sampling method, questions have been asked about its performance as compared with traditional global optimization methods such as the genetic algorithm and when to use mode-pursing sampling as opposed to the genetic algorithm. This work aims to provide an answer to these questions. Similarities and distinctions between mode-pursing sampling and the genetic algorithm are presented. Then mode-pursing sampling and the genetic algorithm are compared via testing with benchmark functions and practical engineering design problems. These problems can be categorized from different perspectives such as dimensionality, continuous/discrete variables or the amount of computational time for evaluating the objective function. It is found that both mode-pursing sampling and the genetic algorithm demonstrate great effectiveness in identifying the global optimum. In general, mode-pursing sampling needs much fewer function evaluations and iterations than the genetic algorithm, which makes mode-pursing sampling suitable for expensive functions. However, the genetic algorithm is more efficient than mode-pursing sampling for inexpensive functions. In addition, mode-pursing sampling is limited by the computer memory when the total number of sample points reaches a certain extent. This work serves the purpose of positioning the new mode-pursing sampling method in the context of direct optimization and provides guidelines for users of mode-pursing sampling. It is also anticipated that the similarities in concepts, distinctions in philosophy and methodology and effectiveness as direct search methods for both mode-pursing sampling and the genetic algorithm will inspire the development of new direct optimization methods.

**Keywords:** mode-pursuing sampling; genetic algorithm; global optimization

## 1. Introduction

Practical engineering design problems are usually highly non-linear and involve many continuous and/or discrete variables. Often it is difficult to define a design problem or to express it as a mathematical model. In addition, the increasingly wide use of finite element analysis and computational fluid dynamics tools brings new challenges to optimization. Finite element analysis and computational fluid dynamics simulations involve a large number of simultaneous equations and therefore are considered computationally expensive and also 'black box' functions. The gradients computed

*Corresponding author. Email: gary_wang@sfu.ca

ISSN 0305-215X print/ISSN 1029-0273 online
© 2009 Taylor & Francis
DOI: 10.1080/03052150802345995
http://www.informaworld.com

from finite element analysis and computational fluid dynamics, which require extra computing resources, are often not reliable (Haftka *et al.* 1998). Therefore, it is difficult to apply traditional gradient-based optimization methods to these black box functions, thereby inhibiting solutions of many practical design problems. Meta-modelling-based design optimizations have emerged as a promising solution for expensive black box problems. Its essence is to use a computationally simpler model for approximating the original expensive black box model. This approximation is realized by sampling in the design space and performing model fitting for a chosen meta-model type. The meta-model can then be optimized. Current research in meta-modelling-based design optimizations focuses on developing better sampling methods, approximation models or the whole global optimization strategy (Wang and Shan 2007). Recently, Wang *et al.* (2004) developed a new mode-pursuing sampling (MPS)-based global optimization method for black box functions. Its discrete variable version is given in Sharif *et al.* (2008). MPS has been found to be effective and efficient in several applications (Wang *et al.* 2004, Liao and Wang 2008, Sharif *et al.* 2008).

The genetic algorithm (GA), on the other hand, has been widely used in engineering for global optimization (Goldberg 1989). The development of GAs was inspired by the principles of genetics and evolution. The GA employs the principal of 'survival of the fittest' in its search process for generating and selecting chromosomes (design solutions) composed of genes (design variables). Chromosomes that are more adaptive to their environment (design objectives/constraints) are more likely to be chosen. Over a number of generations (iterations), desirable traits (design characteristics) will evolve and remain in the population (set of design solutions generated at each iteration). The GA begins its search from a randomly generated population of design. Three operators are often used for propagating the populations from one generation to another to search for the optimum solution, namely selection, crossover and mutation. The GA can be used for problems that are not well defined, difficult to model mathematically or black box. It can also be used when the objective function is discontinuous, highly non-linear, stochastic or has unreliable or undefined derivatives. The limitation of the GA is that it usually demands a large number of function evaluations.

When analysing the performance of MPS, it is found that there are many similarities between the GA and MPS.

(1) The initial population is randomly created in the GA and the sampling points are randomly generated in MPS.
(2) The chromosomes are chosen by a stochastic process in the GA and the probability of a chromosome to be selected is determined by its fitness value, while in MPS the points are sampled according to a probability determined by its objective function value.
(3) Both the GA and MPS are based on a set of solutions or population and render themselves well suited for parallel computation.
(4) Both methods use no gradient information and, thus, are called derivative-free methods or direct methods.

The distinctions between the GA and MPS are also manifold. First they are based on different philosophies. The GA has its root in evolutionary processes, while MPS is based on 'discriminative' sampling. Second, the GA explores the space through operations of genes: it is intuitively a bottom-up approach from the perspective of exploring the entire design space. On the other hand, MPS is like a top-down approach as the entire space is explored at every iteration. Specific similarities and distinctions between the two methods are summarized in Table 1.

In order to understand the performance and the problems that are most amenable to MPS better, this work studies the performance of MPS in parallel with the GA. The selection of the GA is because of the similarities between the two algorithms and because GA is well known and widely used by practitioners, which makes the performance study of MPS more relevant to real practice.

Table 1. Comparison of the features of the GA and MPS algorithms.

| | GA | MPS |
|---|---|---|
| **Similarities** | | |
| Mechanism | Generates more points around the current best point, uses probability in searching/sampling and statistically cover the entire design space | |
| Features | Random process, derivative free and population-based searches | |
| Capabilities | Performs a 'global' optimization and supporting parallel computation | |
| **Distinctions** | | |
| For discrete problem | Discrete in nature and easily handles continuous problems | Needs different treatments for continuous and discrete problems |
| Input parameters | Many parameters are set by the user, *e.g.* the population size, reproduction operators, crossover probability and mutation probability: the parameters are problem dependent and could be sensitive | Fewer parameters need to be changed and it is not very sensitive |
| Efficiency | Normally a large number of function evaluations needed | Developed specially for expensive functions to minimize the number of function evaluations |
| Form of function values | Usually coded form of the function values rather than the actual values | Actual function values |
| Robustness | Premature convergence may happen if the operators are not set properly | For the problems with a large quantity of local optima, it may be trapped in a local optimum |

The main purpose is thus to position MPS in the context of engineering global optimization in order to provide some guidelines for MPS users. The comparison between MPS and the GA is thus not to determine which algorithm is the winner, but rather to shed lights on characteristics and the performance behaviour of MPS in reference to the GA.

## 2. Overview of MPS

MPS entails its continuous variable version (C-MPS) (Wang *et al.* 2004) and its discrete variable version (D-MPS) (Sharif *et al.* 2008). This overview will focus on its original continuous variable version and provide a brief description of the new discrete-variable version.

### 2.1. *C-MPS method*

The essence of MPS is the integration of meta-modelling and a novel discriminative sampling method, which generates more sample points in the neighbourhood of the function mode (local optimal) and fewer points in other areas as guided by a special sampling guidance function.

Fundamental to MPS is Fu and Wang's (2002) algorithm, which is used for generating a sample of an asymptotic distribution of a given probability density function (PDF). Given a $d$-dimensional PDF $g(x)$ with compact support $S(g) \subset \Re^d$, Fu and Wang's (2002) algorithm 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 was shown in Fu and Wang (2002), 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 $K$.

For optimization, it is desired to minimize an $n$-dimensional black box function $f(x)$ over a compact set $S(f) \subset \Re^n$. Following the convention of engineering optimization, the minimum is referred to as the function mode. 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 $x \in S(f)$, a positive number can always be added to $f(x)$, so that it becomes positive on $S(f)$. Note that minimizing $f(x)$ is equivalent to maximizing $-f(x)$.

The MPS algorithm consists of the following four steps:

(1) Step 1. Generate $m$ initial points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ that are uniformly distributed on $S(f)$ ($m$ is usually small).
(2) Step 2. Use the $m$ function values $f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(m)})$ to fit a linear spline function

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

such that $\hat{f}(x^{(i)}) = f(x^{(i)}), i = 1, 2, \ldots, m$, where $\| \bullet \|$ stands for the Euclidean norm.
(3) 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 non-negative on $S(f)$, it can be viewed as a PDF, up to a normalizing constant, the modes of which 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 (2002) to draw a random sample $x^{(m+1)}, x^{(m+2)}, \ldots, x^{(2m)}$ 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 $\hat{f}(x)$.
(4) Step 4. Combine the sample points obtained in step 3 with the initial points in step 1 to form the set $x^{(1)}, x^{(2)}, \ldots, x^{(2m)}$ and repeat steps 2–3 until a certain stopping criterion is met.

For ease of understanding, the MPS method is illustrated with the well-known six-hump camel-back (SC) problem (Branin and Hoo 1972). The mathematical expression of the SC problem is

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

A contour plot of the SC function is shown in Figure 1, where the Hs 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 an equal function value $f_{\min} = -1.032$.

The first step of the MPS algorithm starts 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 Equation (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 Fu and Wang's (2002) algorithm is applied 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 the corresponding function values of $\hat{f}(x)$ is plotted in Figure 2(a), whereas the function $g(x)$ is plotted in Figure 2(b).

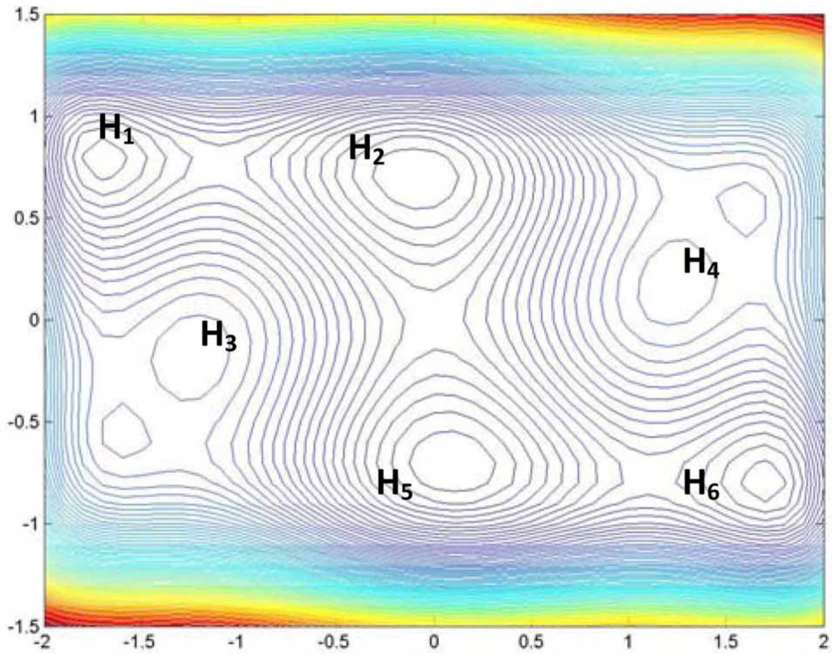

Figure 1. Contour plot of the SC function.

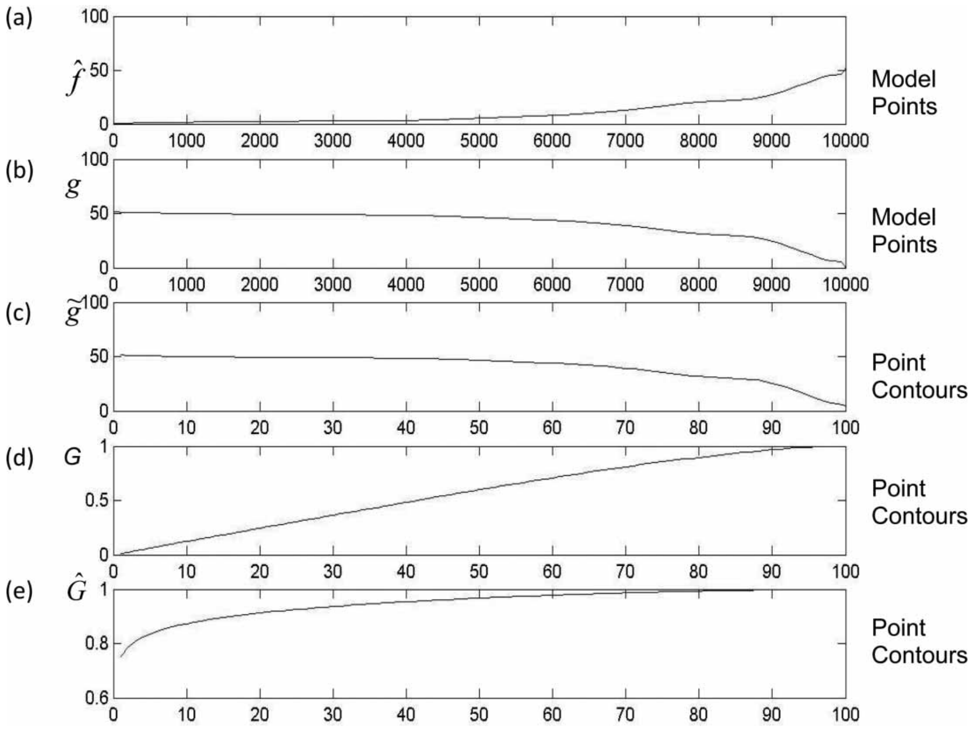

Figure 2. A screen shot of the ranked point distribution of $\hat{f}$, $g$, $\tilde{g}$, $G$ and $\widehat{G}$ for the SC problem.

According to Fu and Wang's (2002) 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 $x^{(m+1)}, x^{(m+2)}, \ldots, x^{(2m)}$.

As can be seen from Figure 2(d), the contours from $E_{80} \sim E_{100}$ (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. 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 $E_{40} \sim E_{100}$ have low probabilities. This curve shows an aggressive sampling step, as many more new sample points are 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). In the MPS step, every point has

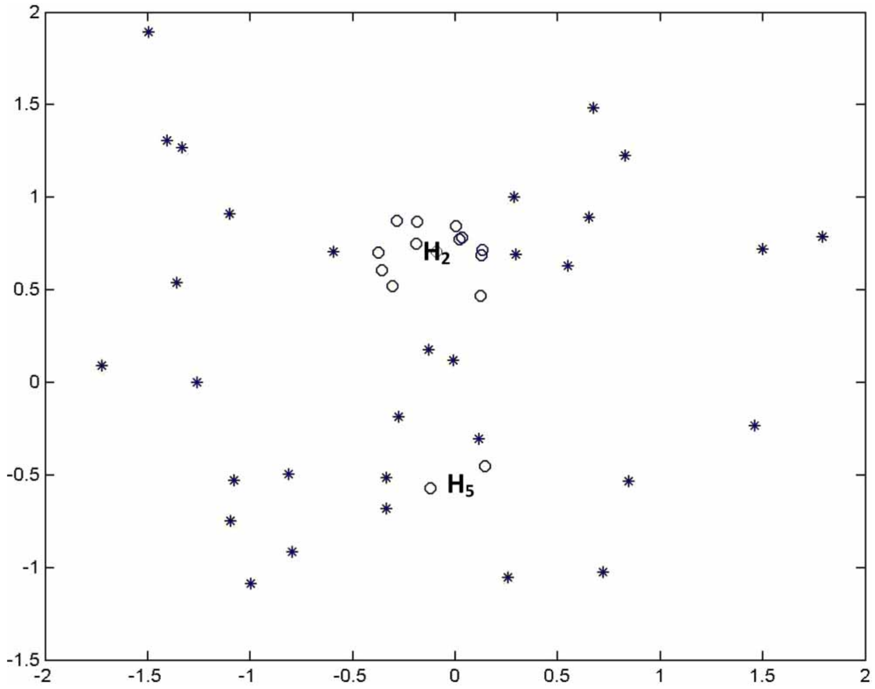

Figure 3. Sample points of the SC problem generated by the MPS method, where 'o' indicates its function value less than $-0.5$ and H2 and H5 are the locations of two global optima.

a positive probability of being drawn, so that the probability of excluding the global optimum is zero. As the iteration process continues, more and more sample points will be generated around the minimum of function $f(x)$.

### 2.2. *D-MPS method*

D-MPS inherits the meta-modelling and discriminative sampling ideas from C-MPS. For the discrete variable space, sample points are mapped from a continuous space to the discrete variable space. The main difference between C-MPS and D-MPS lies on their convergence strategies.

In C-MPS (Wang *et al.* 2004) a local quadratic meta-model is employed for identifying a sub-area adaptively, with which a local optimization is called to search for the local optimum. For a discrete variable space there is no continuous sub-area to be approximated. Even if one could build a continuous function in a discrete space, the local optimum on a continuous function might not be a valid or optimal solution in the discrete space. Therefore, local quadratic meta-modelling does not apply to discrete variable optimization problems.

A 'double sphere' method is used for discrete problems. This method includes two areas (or 'spheres') of dynamically changing radius. One sphere controls the 'exploration' and the other controls 'exploitation'. Recall that the C-MPS is composed of three main steps, *i.e.* generating cheap points, approximation and discriminative sampling. In 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 hyper-sphere and $D_2$ is the domain between the smaller hyper-sphere and bigger hyper-sphere. The three main steps of D-MPS are performed on both $D_1$ and $D_2$. The discriminative sampling is performed independently in the two spheres until the optimum is found or the maximum number of iterations is reached. D-MPS thus does not call any existing local optimization routine and there is no local meta-modelling. The optimum is found by sampling alone (Sharif *et al.* 2008).

## 3. Testing functions and problems

Given the focus on examining the performance of MPS, in this work a real-value GA is employed. The implementation from Houck *et al.* (1995) is selected for a number of reasons. First, this implementation has commonly used operators and it demonstrates good performance. Second, it is based on Matlab™, which renders a common basis for computational cost (CPU time) comparisons because MPS is also based on Matlab™. Last, since a large number of GA implementations and algorithms exist, this work is not intended to draw any general conclusion with regard to the GA as a whole. It is intended to obtain certain qualitative insights into MPS and provide guidelines to users of MPS, which should largely be insensitive to the choice of specific GA implementation.

The selected implementation (Houck *et al.* 1995) forces the design variable to only take values within its upper and lower bounds to ensure the feasibility of solutions. It uses three selection operators, namely the normGeom selection (a ranking selection based on the normalized geometric distribution), the roulette selection (a traditional selection with the probability of surviving equal to the fitness of an individual over the sum of the fitness of all) and the tournament selection (choosing the winner of the tournaments as the new population). It also has three crossover operators, namely the simple crossover, the heuristic crossover and the arithmetic crossover. Finally, it uses four optional mutation operators called the boundary mutation, multi-non-uniform mutation, non-uniform mutation and uniform mutation. Details of these operators are in Houck *et al.* (1995). Two termination criteria are used in the GA implementation. The first is based on the maximum number of generations, which is suitable for problems without knowing the analytical optimum *a priori*. The second termination criterion occurs when the analytical optimum is reached. The GA optimization process will be terminated whenever one of these two criteria is met. For constrained optimization problems, the discard method was used in order to satisfy the constraints. Three constraint checks are performed on the initial population, with new children generated from crossover, as well as from mutation operations at each iteration. Any chromosome that does not pass the constraint check will be discarded from the population. For discrete variable problems, if all of the variables are integers, a simple rounding function is used for rounding a number to its closest integer. If the discrete variable values are picked from a set, for example, $x \in [16.2, 17.1, 18.5, 19.3]$, the possible values, *e.g.* index $= \{1, 2, 3, 4\}$, are first indexed and then a random number is generated and rounded to the closest index. For example, if the index is 3, then $x = 18.5$ is used. For mixed variable problems only the discrete variables are handled as described above.

Nine problems were tested for a performance comparison of the chosen GA implementation and MPS optimization algorithms. The characteristics of these problems are summarized in Table 2. In Table 2 only optimization problems with constraints other than bounds are referred to as constrained problems.

As can be seen from Table 2, the test functions and problems present different challenges to the optimization algorithms. This section briefly describes each test problem.

### 3.1. *The SC problem*

The SC problem is a well-known benchmark test problem. It appeared in Branin and Hoo (1972) and has been used by many other researchers. The mathematical expression is shown in Equation (2). The SC function has six local optima. A contour plot of the SC function is shown in Figure 1.

### 3.2. *The Corana function*

The Corana function is a well-known benchmark test function (Corana *et al.* 1987). It was also tested in Humphrey and Wilson (2000). The Corana function is a paraboloid with axes parallel to the coordinate direction with the exception of a set of open disjoint rectangular flat ‘pockets’. Let the domain of the function $f(x)$in an $n$-dimensional space be

$$D_f \equiv \{x \in R^n : -a_i \leq x_i \leq a_i; a_i \in R^n_+, i = 1, 2, \ldots, n\}$$

Table 2. Characteristics of the test functions and problems.

| Function | Characteristics |
|---|---|
| SC problem ($n = 2$) | Multiple local optima |
| Corana function ($n = 2$) | Flat bottom multiple local optima |
| Hartman function ($n = 6$) | Multiple local optima |
| Gear train problem ($n = 4$) | Discrete multiple local optima |
| F16 function ($n = 16$) | High dimension |
| R10 function (R10) ($n = 10$) | Flat bottom relatively high-dimension multiple local optima |
| Insulation layer design ($n = 2$) | Constrained engineering problem |
| Pressure vessel design ($n = 4$) | Constrained engineering problem |
| FJP design ($n = 4$) | Constrained expensive black box function |

Note: $n$ is the number of variables.

Let $D_m$ represent the set of 'pockets' in $D_f$:

$$d_{k_1,\ldots,k_n} = \left\{x \in D_f : k_i s_i - t_i < x_i < k_i s_i + t_i; k_i \in \mathrm{Z}; t_i, s_i \in R^n_+; t_i < \frac{s_i}{2}, i = 1, 2, \ldots, n\right\}$$

$$D_m = \bigcup_{k1,\ldots,kn} d_{k_1\cdots,k_n} - d_{0,0,\ldots,0.}$$

The Corana function can then be defined as

$$f(x) = \begin{cases} \sum_{1}^{n} d_i x_i^2, x \in D_{\mathrm{f}} - D_m, d_i \in R^n_+ \\ c_r \sum d_i z_i^2, x \in d_{k_1,\ldots,k_n}, (k_1, \ldots, k_n) \neq 0 \end{cases} \tag{3}$$

where

$$z_i = \begin{cases} k_i s_i + t_i, & k_i < 0 \\ 0, & k_i = 0 \\ k_i s_i - t_i, & k_i > 0 \end{cases}$$

The parameters in this test are set to $n = 4$, $a_i = 100$, $s_i = 0.2$, $t_i = 0.05$, $c_r = 0.15$ and $d_i = e^{(i-1)}$. A less extreme weight $d_i$ is used as compared to the standard representation, $10^{(i-1)}$. The searching space is $x_i = [-100, 100]$. The four-dimensional Corana function has its global minimum $f^* = 0$ at $x^* = (0, 0, 0, 0)$. A difficulty arises from the fact that the optimum is located in a deep valley and several deep flat bottoms exist along the space. Figure 4 shows a two-dimensional Corana function in the range of $[-0.5, 0.5]$.

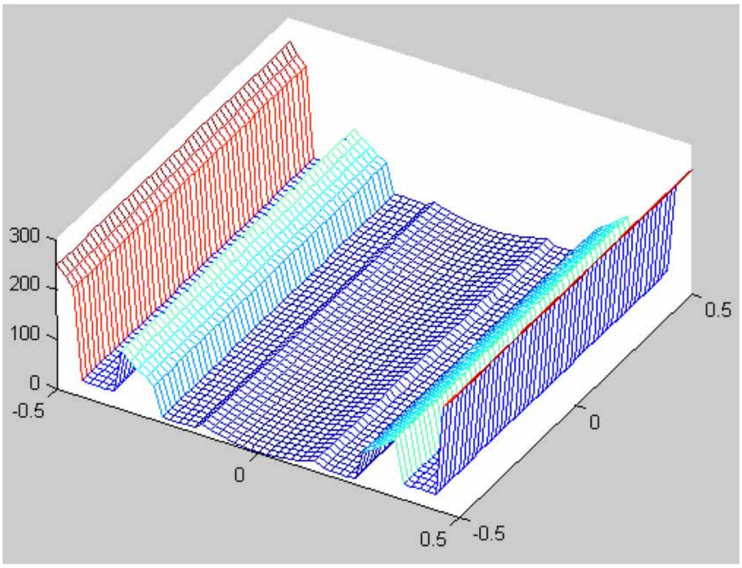

Figure 4. Two-dimensional Corana function in $[-0.5, 0.5]$.

### 3.3. *The Hartman function*

The Hartman function with $n = 6$ was tested in Wang *et al.* (2004). It can be expressed as

$$f_{HN}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} \alpha_{ij}(x_j - p_{ij})^2\right], \quad x_i \in [0, 1], \quad i = 1, \ldots, n \tag{4}$$

where $\alpha_{ij}$ and $p_{ij}$ are listed in Table 3.

### 3.4. *The compound gear train design problem*

This is a discrete variant problem involving a compound gear train design (Fu *et al.* 1991, Cai and Thierauf 1996), as shown in Figure 5. It is desired to produce a gear ratio as close as possible to 1:6.931. For each gear, the number of teeth must be an integer between 14 and 60. The integer design variables are the numbers of teeth $x = [T_\mathrm{d}, T_\mathrm{b}, T_\mathrm{a}, T_\mathrm{f}]^T = [x_1, x_2, x_3, x_4]^T$. The optimization problem is formulated as:

$$\text{Minimize } GT(X) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)^2 \tag{5}$$

subject to $14 \leq x_i \leq 60$ with $i = 1, 2, 3, 4$.

Table 3. Coefficients for the Hartman function.

| | $j$ | | | | | | |
|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | $c_i$ |
| $\alpha_{ij}$ | | | | | | | |
| 1 | 10 | 3 | 17 | 3.5 | 1.7 | 8 | 1 |
| 2 | 0.05 | 10 | 17 | 0.1 | 8 | 14 | 1.2 |
| 3 | 3 | 3.5 | 1.7 | 10 | 17 | 8 | 3 |
| 4 | 17 | 8 | 0.05 | 10 | 0.1 | 14 | 3.2 |
| $p_{ij}$, | | | | | | | |
| 1 | 0.1312 | 0.1696 | 0.5569 | 0.0124 | 0.8283 | 0.5886 | |
| 2 | 0.2329 | 0.4135 | 0.8307 | 0.3736 | 0.1004 | 0.9991 | |
| 3 | 0.2348 | 0.1451 | 0.3522 | 0.2883 | 0.3047 | 0.6650 | |
| 4 | 0.4047 | 0.8828 | 0.8732 | 0.5743 | 0.1091 | 0.0381 | |

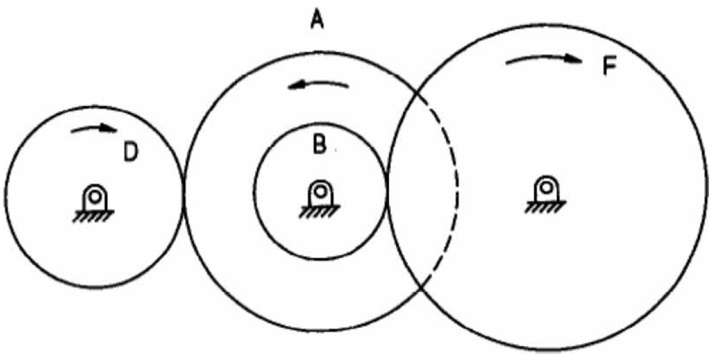

Figure 5. Compound gear train.

## 3.5. *A function of 16 variables (F16)*

This high-dimensional problem was described in Wang *et al.* (2004). It can be expressed by

$$f_{F16}(x) = \sum_{i=1}^{16}\sum_{j=1}^{16} a_{ij}(x_i^2 + x_i + 1)(x_j^2 + x_j + 1), i, j = 1, 2, \ldots, 16, \tag{6}$$

$$[a_{ij}]_{\text{row } 1-8} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$[a_{ij}]_{\text{row } 9-16} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

## 3.6. *The ten-dimensional Rosenbrock function (R10)*

The Rosenbrock function is a widely used benchmark problem for testing optimization algorithms such as those in Bouvry *et al.* (2000) and Augugliaro *et al.* (2002). It can be expressed as

$$f_{RO}(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{7}$$

The Rosenbrock function has its global minimum $f_{\min} = 0$ at $x^* = (1, 1, \ldots, 1)$. While attempting to find a global minimum, a difficulty arises from the fact that the optimum is located in a deep parabolic valley with a flat bottom. In the present study, the 10-dimensional Rosenbrock function will be used as another high-dimensional benchmark problem, in addition to F16. The searching range for each variable is $[-5, 5]$.

## 3.7. *The two layer insulation design problem*

This is a problem involving the design of a two-layer insulated steam pipe line, as shown in Figure 6. The objective is to design $r_1$ and $r_2$ to minimize the total cost per unit length of pipe (\$/m) in one year of operation. The total cost function includes the cost of heat loss, the cost of insulation materials and the cost of fixed charges including interest and depreciation (with a rate of 15%). The parameters in the design are as follows.

(1) An inner radius of $r_0 = 273\,\text{mm}$.
(2) A steam temperature of $T_f = 400\,^\circ\text{C}$ and a temperature of the ambient air of $T_a = 25\,^\circ\text{C}$.
(3) The conductive resistance through the wall is neglected.

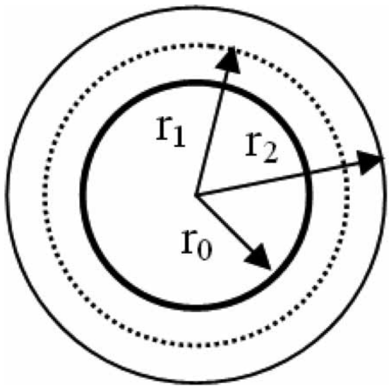

Figure 6. A steam pipe with two-layer insulation.

(4) The convective heat transfer rate at the inner surface has $h_f = 55\,\mathrm{W/m^2K}$.
(5) The first layer of insulation is rock wool, with a thermal conductivity $k_1 = 0.06\,\mathrm{W/mK}$ and thickness $\delta_1 = r_1 - r_0$.
(6) The second insulation layer is calcium silicate with $k_2 = 0.051\,\mathrm{W/mK}$ and thickness of $\delta_2 = r_2 - r_1$.
(7) The outside convective heat transfer coefficient is $h_0 = 10\,\mathrm{W/m^2K}$.
(8) The cost of heat is $C_h = 0.02\,\$/\mathrm{kWh}$, the cost of rock wool is $C_{i1} = 146.7\,\$/\mathrm{m}^3$ and the cost of calcium silicate is $C_{i2} = 336\,\$/\mathrm{m}^3$.

The constraints are determined by the insulation requirement and other practical considerations, which are described in Zaki and Al-Turki (2000).

The total cost per unit length of pipe can be expressed as

$$\begin{aligned}
&\text{Minimize}\\
&f_{IL}(r_1, r_2)\\
&\quad = \tau C_h \frac{0.001(T_f - T_a)}{(1/2\pi r_0 h_f) + (\ln(r_1/r_0)/2\pi k_1) + (\ln(r_2/r_1)/2\pi k_2) + (1/2\pi r_2 h_o)}\\
&\qquad + f\pi[C_{i_1}(r_1^2 - r_0^2) + C_{i_2}(r_2^2 - r_1^2)] \qquad (8)
\end{aligned}$$

Subject to $38\,\mathrm{mm} < \delta < 5r_0;\ r_1 > r_0;\ r_2 > r_1;$

$$T_a + \frac{(T_f - T_a)}{(1/2\pi r_0 h_f) + (\ln(r_1/r_0)/2\pi k_1) + (\ln(r_2/r_1)/2\pi k_2) + (1/2\pi r_2 h_o)} \frac{1}{2\pi r_2 h_o} < 60 \qquad (9)$$

The characteristic of this problem is its complexity in both the objective function and the last constraint. The optimum solution is $f^* = 5.5362\,\$/\mathrm{m}$ at $r_1 = 0.311\,\mathrm{m}$ and $r_2 = 0.349\,\mathrm{m}$.

## 3.8. *Pressure vessel design problem*

The design of a pressure vessel was used as a test problem in Wang *et al.* (2004) and is shown in Figure 7. 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. They 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 $0.625 \le T_h \le 1.0$.

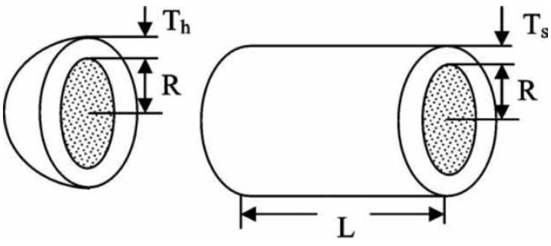

Figure 7. Pressure vessel (adapted from Wang *et al.* 2004).

The design objective is to minimize the total system cost, which is a combination of welding, material and forming costs. The optimization model is then expressed as

$$\text{Minimize } f_{PV}(R, T_s, T_h, L) = 0.622T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R \quad (10)$$

$$\text{Subject to } Ts - 0.0193 \geq 0;\ Th - 0.00954R \geq 0;\ \pi R^2L + \frac{4}{3}\pi R^3 - 1.296E6 \geq 0 \quad (11)$$

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.

## 3.9. *Fixture and joining positions (FJP) optimization problem*

Liao and Wang (2008) studied simultaneous optimization of fixture and joining positions for a non-rigid sheet metal assembly. This is a black box function problem involving finite element analysis, which makes it computationally expensive. The optimization problem can be described as follows. In the presence of part variation and fixture variation, as well as the constraints from the assembly process and designed function requirements, the best locations of fixtures and joining points are found so that the non-rigid sheet metal assembly can achieve the minimal assembly variation. The assembly of two identical flat sheet metal components by lap joints shown in Figure 8 is optimized. Assuming that these two components are manufactured under the same conditions, their fabrication variations are expected to be the same. The size of each flat sheet metal part is 100 mm × 100 mm × 1 mm, with Young's modulus $E = 2.62e + 9\,\text{N/mm}^2$ and Poisson's ratio $\nu = 0.3$. The finite element computational model of the assembly is created in ANSYS™.

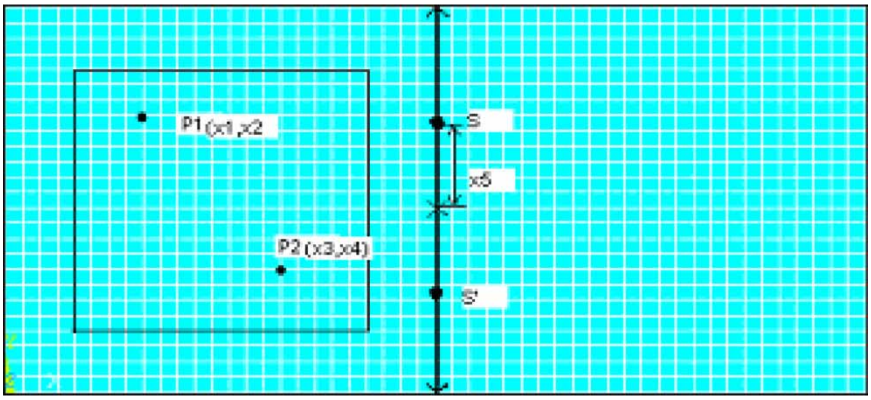

Figure 8. An assembly of two sheet metal parts (note that the symbol 'p' indicates the fixture location and 's' indicates the joint positions).

The element type is SHELL63. The number of elements and the number of nodes are 1250 and 1352, respectively.

The mathematical optimization model for this specific example can be written as follows:

$$\text{Minimize } \Theta(x) = abs(U_z^1(x)) + abs(U_z^2(x)) \tag{12}$$

$$\text{Subject to } (x_1 - x_3)^2 + (x_2 - x_4) \geq 100;\ 20 \leq x_1, x_2, x_3, x_4, \leq 80;\ 10 \leq x_5 \leq 40 \tag{13}$$

where $U$ is the deformation of critical points in an assembly. It is obtained by modelling the assembly deformation using the finite element analysis through ANSYS™. In this study, ANSYS™ and Matlab™ are integrated to implement the optimization on a finite element analysis process for both MPS and the GA.

## 4. Results and discussion

### 4.1. *Performance criteria*

Two main performance criteria are used for evaluating the two algorithms, namely effectiveness and efficiency. The effectiveness includes the robustness of the algorithm and the accuracy of the identified global optimum. For robustness, 10 independent runs are carried out for each problem and each algorithm, except for the expensive FJP problem, where only five runs are performed. The range of variation and median value of the optima are recorded and compared against the analytical or known optimum. If the known solution of a problem is not zero, the accuracy of an algorithm is quantified by

$$Q_{sol} = 1 - \left| \frac{solution - known\ solution}{known\ solution} \right| \tag{14}$$

If the known solution is zero, the deviation of the solution from zero is examined. When there is no analytical or known optimum, the optima found by the GA and MPS are compared by their values. The second criterion for comparison is efficiency. In this study, the number of iterations and number of function evaluations was used for evaluating the efficiency of an algorithm in solving a problem, as well as the CPU time required for solving the problem. Again, the average (arithmetic mean) and median values of these values in 10 (or five) runs were used.

### 4.2. *Effects of tuning parameters*

As discussed previously, there are several optimization operators to be set in the GA program. In testing the same problems, these factors were found to have significant effects on the efficiency. Figure 9 shows the variation in the GA's performance under different crossover rates ($Pc$) when solving the SC function (with a population size of 100 and a mutation rate of $Pm = 0.01$). A general trend can be observed in Figure 9, whereby an increasing $Pc$ implies that the number of function evaluations increases and the number of iterations decreases, while the computation time remains at a similar level. With a small $Pc$, the GA needs more generations to establish the optimum and, therefore, more iterations are needed. When the value of $Pc$ is set to be very large, the number of function evaluations increases sharply because almost all parents are replaced and few good solutions are inherited.

Similarly, the effect of the mutation rate $Pm$ on the GA's performance in solving the SC problem (with a population size of 100 and crossover rate of $Pc = 0.6$) is studied. It is observed that the

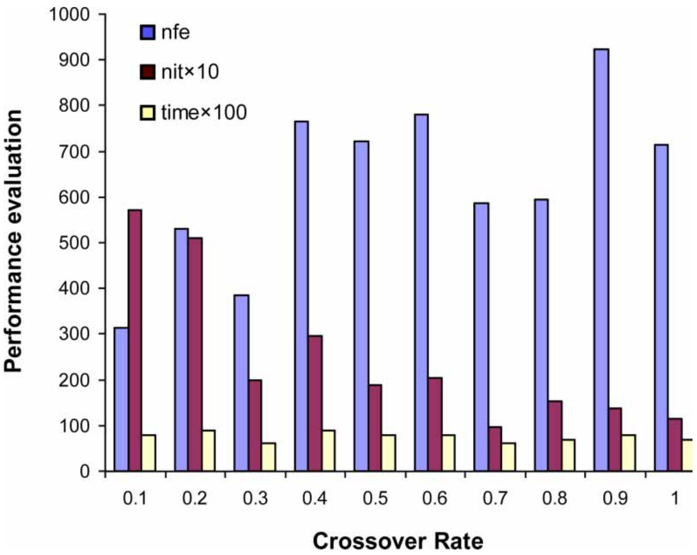

Figure 9. Performance evaluation with different crossover rates in solving the SC problem.

minimum of number of function evaluations occurs at $Pm = 0.1$ and the number of function evaluations increases with $Pm > 0.1$.

In the MPS program there is a tuneable parameter (the difference coefficient $k$), which was also found to have a noticeable effect on the performance of the MPS. The difference coefficient $k$ (denoted as $c_d$ in Wang *et al.* (2004)) is used for a convergence check in quadratic model detection. A smaller difference coefficient makes the MPS have a more accurate solution, but it may cause more computational expense, as shown in Table 4. By examining the MPS performance in several problems in this study, it is found that $k = 0.01$ is a good choice for most problems. However, for the discrete gear train problem and the R10 problem, a larger coefficient ($k > 0.5$) must be used for establishing an acceptable solution.

Both the random characteristic and the effects of their tuning parameters necessitate the method of using results of multiple runs in this comparison study. Crossover rates of 0.6 or 0.7 and mutation rates of 0.05 or 0.1 for the GA were used specifically for the comparison in this work. A difference coefficient of 0.01 was used for most problems for MPS, but a coefficient larger than 0.5 was used for the gear train and R10 problems. Different combinations of parameters for a problem were first tested for both MPS and GA and the set of parameters that yielded the best solution was then used for comparison. Hence, for each test problem the best results of the GA were compared with the best results of MPS.

Table 4. Efficiency of MPS with different k values on the SC and pressure vessel problems.

| Coefficient (k) | Average CPU time (s) | Average of the number of iterations | Average of the number of function evaluations |
|---|---|---|---|
| SC problem | | | |
| 0.001 | 15.3531 | 17.9 | 67.1 |
| 0.01 | 4.6312 | 9.6 | 34.7 |
| 0.1 | 3.5311 | 6.9 | 27.8 |
| Pressure vessel problem | | | |
| 0.001 | 46.6891 | 8.8 | 51.8 |
| 0.01 | 26.2938 | 5.2 | 37.2 |
| 0.1 | 31.2279 | 5.8 | 39.9 |

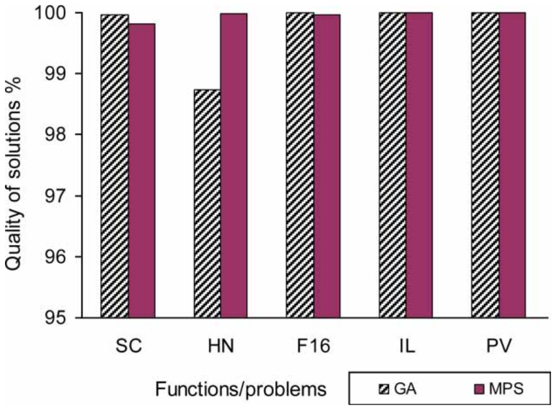

Figure 10. Mean quality ratings of solutions obtained by the GA and MPS for five testing problems.

## 4.3. *Effectiveness comparison of the GA and MPS*

For those problems with known optima of non-zero (the SC, Hartman, F16, insulation layer and pressure vessel problems), the qualities of the solutions (using the median solution) were compared based on Equation (14). The results are shown in Figure 10. It can be seen that, for all of these five problems, both the GA and MPS methods can find solutions of high quality close to 100%. A lower quality of 98.7% was only obtained by the GA on the Hartman problem.

However, the two algorithms show great differences in solving the other problems. For the Corana problem (analytical optimum of zero), the GA finds very good solutions ranging from 0 to 0.003, while MPS only finds solutions from 34.25 to 5337.44. For the 10-dimensional R10 function (analytical optimum of zero), the GA also outperforms MPS. The solutions obtained by the GA ranged from 0 to 0.013, while those obtained by MPS ranged from 73.92 to 128.79. It was found while solving the R10 function that MPS reaches the computer memory limit when the number of function evaluations reaches about 2500. This is because all of the 2500 points participate in the construction of the meta-model defined by Equation (1). The GA, on the other hand, can afford around 3,000,000 function evaluations. In a later comparison, both the GA and MPS were allowed to run for 2500 points and it was found that the results were similar to those of MPS, as shown in Table 5.

A particular interest in this study was to compare the effectiveness of the GA and MPS in solving discrete problems. In this study, a well-known gear train problem was solved and the solutions are compared. Table 6 shows the solutions of the gear train problem with different optimization algorithms. The MPS outperforms most of the other approaches, including the GA. Table 7 shows

Table 5. Comparison of the effectiveness of the GA and MPS for the R10 and Corana problems.

| | | | Minimum | | | |
|---|---|---|---|---|---|---|
| | | | GA | | MPS | |
| Function | $n$ | Space | Range of variation | Median | Range of variation | Median |
| Corana | 4 | [−100,100] | [0.000, 0.003] | 0 | [34.256, 5337.445] | 306.019 |
| R10 | 10 | [−5,5] | [0, 0.013] | 0.003 | [73.926, 128.794] | 118.341 |
| R10 | 10 | [−5,5] | [63.641,207.665]* | 130.889* | [73.926, 128.794] | 118.341 |

Note: an asterisk represents solutions of the GA for 2500 number of function evaluations.

Table 6. Solutions for the gear train design problem: best results.

| Approach | Optimization solution | $GT_{\min}$ |
|---|---|---|
| Penalty function approach (PFA) (Cai and Thierauf, 1996) | 14, 29, 47, 59 | $4.5 \times 10^{-6}$ |
| Two-membered evolution strategy (2-ES) (Bouvry *et al.*, 2000) | 18, 15, 32, 58 | $1.4 \times 10^{-6}$ |
| GA | 20, 20, 47, 59 | $9.75 \times 10^{-10}$ |
| MPS | 16, 19, 43, 49 | $2.7 \times 10^{-12}$ |

Table 7. Solutions of the gear train design problem: solutions of 10 runs.

| | Solutions | | |
|---|---|---|---|
| Algorithm | Range of variation | Median | Average |
| GA | $[9.745 \times 10^{-10}, 1.683 \times 10^{-6}]$ | $4.615 \times 10^{-9}$ | $4.033 \times 10^{-7}$ |
| MPS | $[2.701 \times 10^{-12}, 1.827 \times 10^{-8}]$ | $5.544 \times 10^{-10}$ | $2.423\text{e} \times 10^{-9}$ |

Table 8. Comparison of the optimal solutions of the FJP problem by the GA and MPS with five runs.

| | Solutions | | |
|---|---|---|---|
| Algorithm | Range of variation | Median | Average |
| GA | [0.2177, 0.3451] | 0.2468 | 0.2714 |
| MPS | [0.1515, 0.4104] | 0.2187 | 0.2453 |

Table 9. Comparison of the optimal solutions of the FJP problem by the GA and MPS: best results.

| Algorithm | Optimization solution | $\Theta_{\min}$ |
|---|---|---|
| MPS | $P_1 = (75.153, 31.709)$, $P_2 = (68.117, 59.831)$, $S = 37.473$ | 0.1515 |
| GA | $P_1 = (63.944, 65.459)$, $P_2 = (20.043, 27.741)$, $S = 18.297$ | 0.2177 |

the solutions with 10 runs of the both GA and MPS. The MPS outperforms the GA for the discrete problem.

Another particular problem, the FJP problem, also deserves more explanation. It is an expensive black box function problem that involves finite element analysis. When testing this problem with the GA, the maximum number of generations was set as 60 and the population size was 100 with the normGeomSelect parameter, as well as the three crossover ($Pc = 0.7$) and four mutation methods ($Pm = 0.1$), as described in section 2. For MPS, $k = 0.01$. The results are shown in Tables 8 and 9. Both the GA and MPS gave good solutions in all five independent runs. However, MPS provided a wider range of variation in the solutions and it gave a better optimum of 0.1515.

## 4.4. *Efficiency comparison of the GA and MPS*

The efficiencies are compared in Figure 11 in terms of the number of function evaluations, the number of iterations and the CPU times used for solving the problems. The number of function evaluations and iterations needed by the GA were dramatically larger than those needed by MPS for most problems, except the Hartman problem. For better illustration, Figure 11(a) scales the

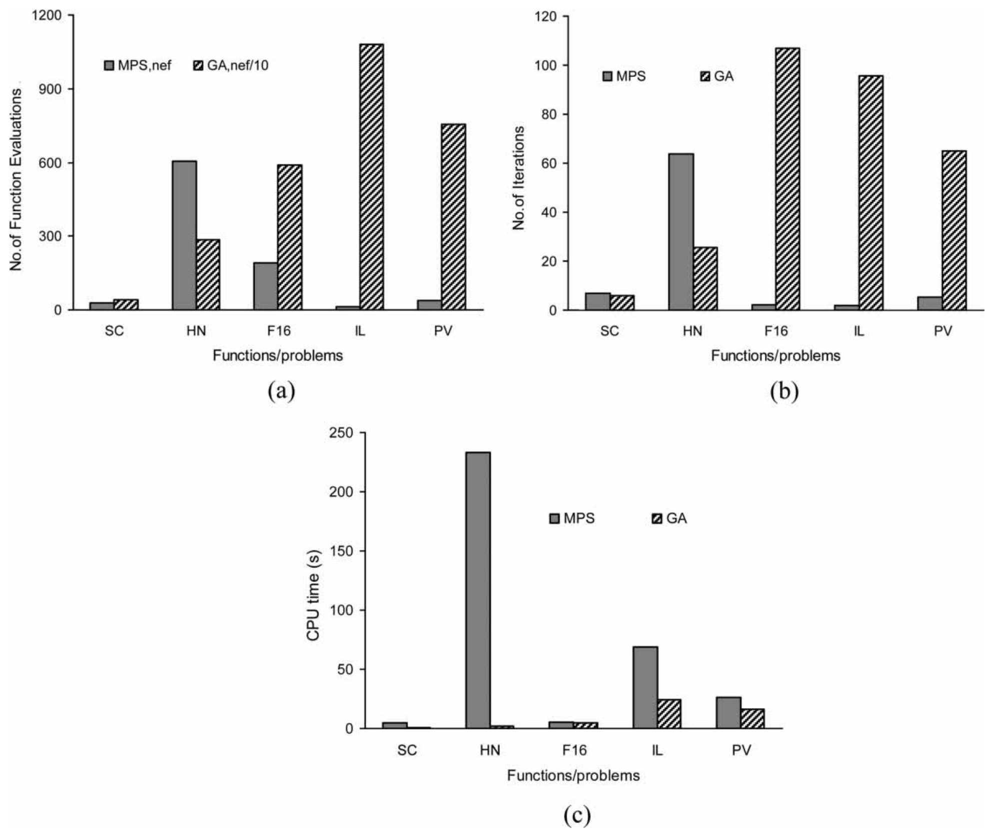

Figure 11. Computational expenses of MPS and GA for the solution of five testing problems (a) numbers of function evaluations, (b) numbers of iterations, and (c) CPU time.

number of function evaluations of the GA by 0.1. It can also be found that the GA needed more function evaluations and iterations for high dimensional problems and complex design problems, such as the F16, pressure vessel and insulation layer problems. The lower number of function evaluations of MPS occurred because of its discriminative sampling and meta-modelling formulation in the algorithms. Only the expensive points were calculated using the objective function, while the cheap points were calculated using an approximate linear spline function or quadratic function. Since only several expensive points were generated in MPS, the number of function evaluations was very low. However, for the GA, each individual in the population was calculated using the objective function, so the number of function evaluations was very high.

As for the CPU time used for solving the problems, the MPS method used more CPU time than the GA in all of the five problems, especially the Hartman problem, where MPS spent 100 times more CPU time than the GA, as shown in Figure 11(c). The reason is that MPS needs to generate 10,000 points per iteration and calculate the function value of each point from the meta-model, thereby leading to a large computing load and high CPU time. However, this was not counted as a function evaluation since the fundamental assumption of MPS is that the objective function evaluation is at least one magnitude more expensive than evaluation with a meta-model in MPS.

The computational expenses by the GA and MPS for the other functions/problems are listed in Table 10. Generally, the GA performs better than MPS on the high-dimensional problems with inexpensive functions. The efficiency advantage of MPS shows in the expensive function problem, the FJP problem, in which MPS uses only half of the computational efforts of the GA and finds better solutions.

Table 10. Efficiency of the GA and MPS algorithms in some optimization problems.

| | Number of function evaluations | | | | Number of iterations | | | | CPU time (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GA | | MPS | | GA | | MPS | | GA | | MPS | |
| Function or problem | Average | Median | Average | Median | Average | Median | Average | Median | Average | Median | Average | Median |
| SC | 418.8 | 408 | 27.8 | 27 | 5.9 | 6 | 6.9 | 7 | 0.5 | 0.5 | 4.6 | 3.9 |
| Hartman | 2837.1 | 2193.5 | 603.4 | 535 | 25.7 | 23.5 | 63.9 | 56.5 | 1.9 | 1.5 | 232.8 | 79.9 |
| F16 | 5887 | 5889 | 191.8 | 163 | 106.8 | 107.5 | 2.2 | 4 | 4.5 | 4.5 | 5.1 | 3.9 |
| Insulation layer | 10821.2 | 10812.2 | 12.7 | 13 | 95.7 | 97 | 2 | 2 | 24.3 | 24.2 | 68.6 | 68.7 |
| Pressure vessel | 7566 | 4643.5 | 37.2 | 34 | 65.1 | 39.5 | 5.2 | 4.5 | 16.1 | 10.1 | 26.3 | 23.3 |
| Corana | 2854.7 | 2807.5 | 20.3 | 19 | 65.2 | 65.5 | 1.2 | 1 | 2.5 | 2.5 | 1.0 | 0.6 |
| Gear train | 291.7 | 283 | 1610 | 2000 | 4.5 | 4.5 | 402 | 500 | 0.5 | 0.5 | 3847.6 | 4846.3 |
| R10 | 2928774.5 | 2928511 | 2231 | 2434 | 4963.8 | 4972.5 | 158.5 | 173 | 1648.9 | 1652.4 | 23830.9 | 26043.7 |
| FJP | 3421 | 3752 | 1371 | 1591 | 37 | 42 | 234.8 | 291 | 27595.9 | 11766.5 | 12393.2 | 4864.5 |

## 5. Summary and remarks

This article studied the performance of the MPS method with reference to the widely used GA. The following observations were made based on qualitative analysis of MPS and the GA and quantitative comparisons on the test problems.

(1) MPS can robustly and accurately identify the global optimum for a majority of test problems, including both continuous and discrete variable problems. However, it meets its limitation when the number of function evaluations required for convergence is larger than a certain value that exceeds the computer memory. In this regard, MPS is best suited for low-dimensional problems and high-dimensional problems with simple functions. For high-dimensional complex or expensive functions, large-memory computers are needed or a better memory-management strategy needs to be developed for MPS.
(2) MPS is recommended for global optimization of expensive functions. MPS is developed for expensive functions and it therefore does not bear advantages over the GA for global optimization on inexpensive functions.
(3) The difference coefficient $k$ is a sensitive parameter for MPS. It is recommended to set $k = 0.01$ if the user has no *a priori* knowledge of the optimization problem.
(4) Common features of MPS and the GA, such as the group-based (or population-based) sampling and selective generation of new samples, may be found in other recognized global optimization methods, *e.g.* simulated annealing, ant colony optimization, particle swarm optimization, etc. The unique philosophy behind MPS, namely, the top-down exploration and discriminative sampling, may inspire the development of future algorithms.

Future research on MPS will enhance its capability for high-dimensional problems. One possible method is to employ a more economical meta-modelling method in order to avoid using all of the evaluated points in model fitting while still providing an overall guide for discriminative sampling.

## References

Augugliaro, A., Dusonchet, L., and Sanseverino, E.R., 2002. An evolutionary parallel Tabu search approach for distribution systems reinforcement planning. *Advanced Engineering Informatics*, 16, 205–215.

Bouvry, P., Arbab, F., and Seredynski, F., 2000. Distributed evolutionary optimization in manifold: the Rosenbrock's function case study. *Information Sciences*, 122, 141–159.

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

Cai, J. and Thierauf, G., 1996. Evolution strategies for solving discrete optimization problems. *Advances in Engineering Software*, 25, 177–183.

Corana, A., Marchesi, M., Martini, C., and Ridella, S., 1987. Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. *ACM Transactions on Mathematical Software*, 13, 262–280.

Fu, J., Fenton, R.G., and Cleghorn, W.L., 1991. A mixed integer discrete–continuous programming method and its application to engineering design optimization. *Engineering Optimization*, 17, 263–280.

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, 5–25.

Goldberg, G., 1989. *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Haftka, R.T., Scott, E.P., and Cruz, J.R., 1998. Optimization and experiments: a survey. *Applied Mechanics Review*, 51 (7), 435–448.

Houck, C., Joines, J., and Kay, M., 1995. *A genetic algorithm for function optimization: a Matlab implementation*. Technical Report NCSU-IE-TR-95-09, North Carolina State University, Raleigh, NC.

Humphrey, D.G. and Wilson, J.R., 2000. A revised simplex search procedure for stochastic simulation response surface optimization. *INFORMS Journal on Computing*, 12 (4), 272–283.

Liao, X. and Wang, G.G., 2008. Simultaneous optimization of fixture and joint positions for non-rigid sheet metal assembly. *International Journal of Advanced Manufacturing Technology*, 36, 386–394.

Sharif, B., Wang, G.G., and ElMekkawy, T., 2008. Mode pursuing sampling method for discrete variable optimization on expensive black-box functions. *Transactions of ASME, Journal of Mechanical Design*, 130 (2), 021402-1-11.

Wang, G.G. and Shan, S., 2007. Review of metamodeling techniques in support of engineering design optimization. *Transactions of ASME, Journal of Mechanical Design*, 129 (4), 370–380.

Wang, L., Shan, S., and Wang, G.G., 2004. Mode-pursuing sampling method for global optimization on expensive black-box functions. *Journal of Engineering Optimization*, 36 (4), 419–438.

Zaki, G.M. and Al-Turki, A.M., 2000. Optimization of multilayer thermal insulation for pipelines. *Heat Transfer Engineering*, 21, 63–70.