Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 434636, 12 pages
http://dx.doi.org/10.1155/2013/434636

Research Article

Antioptimisation of Trusses Using Two-Level Population-Based Incremental Learning

Phinit Tontragunrat and Sujin Bureerat

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Correspondence should be addressed to Sujin Bureerat; sujbur@kku.ac.th

Received 14 December 2012; Accepted 18 March 2013

Academic Editor: Xiaojun Wang

Practical optimum design of structures often involves parameters with uncertainties. There have been several ways to deal with such optimisation problems, and one of the approaches is an antioptimisation process. The task is to find the optimum solution of common design variables while simultaneously searching for the worst case scenario of those parameters with uncertainties. This paper proposed a metaheuristic based on population-based incremental learning (PBIL) for solving antioptimisation of trusses. The new algorithm is called two-level PBIL which consists of outer and inner loops. Five antioptimisation problems are posed to test the optimiser performance. The numerical results show that the concept of using PBIL probability vectors for handling the antioptimisation of truss is powerful and effective. The two-level PBIL can be considered a powerful optimiser for antioptimisation of trusses.

1. Introduction

A truss structure is one of the most widely used engineering structures due to its simplicity and low cost for construction and design. Such a structure can be used in many engineering applications such as bridges, roofs, and towers. Throughout its history, a great deal of research work on truss design optimisation has been investigated, for example, [1–6]. The design process can be categorised as topology, shape, and sizing optimisation. The optimisers used for tackling truss design problems could be the methods with and without using function derivatives. Some of the most popular optimisers are evolutionary algorithms (EAs), which can deal with truss design having single or multiple objective functions. Using EAs, although having low convergence rate and lacking of search consistency, is popular and advantageous since they are simple to use and robust and can deal with any kind of design problems particularly truss design with discrete design variables.

Nevertheless, practical design of trusses and other structures often has inevitable difficulties when uncertainties are involved. For example, it is always difficult to define certain applied loads on structures and material properties. This implies that the real world design problem of trusses will include parameters with uncertainties and it becomes reliability-based optimisation. The simplest way to handle this is by adding the factor of safety to structural constraints. Alternatively, multiobjective design problems which minimise cost and maximise a parameter indicating structural reliability (e.g., structural compliance and natural frequency) are posed and solved for a set of the Pareto optimum structures [6–8]. The more popular approach is the use of reliability-based optimisation techniques, which involve probabilistic design constraints. For example, the use of gradient-based optimisers [9] and evolutionary algorithms [10] for this design process has been reported.

Apart from the aforementioned approaches, a simple but effective way to deal with optimisation with uncertainties is to use the so-called antioptimisation [11]. In the antioptimisation process, there are two sets of design variables, common design variables and ones with uncertainties predefined in certain intervals. One optimisation run has two simultaneous tasks. The first task is to find the solution for the interval variables giving the worst case scenario for the structure,

Population 1				Population 2				Population 3			
0	0	1	1	1	1	1	0	0	0	0	1
1	1	1	0	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0	1

Probability vectors

[0.5, 0.5, 0.5, 0.5]	[0.5, 0.5, 0.5, 0.5]	[0.25, 0.5, 0, 0.75]

Figure 1: Probability vectors and their corresponding populations.

Initialisation $P_i = 0.5$, $k = 0$, $\mathbf{b} = \{\cdot\}$
Main procedure
(1) Generate a binary population $\mathbf{B}$ according to P_i.
(2) Perform function evaluations $f(\mathbf{B})$ and find the best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$.
(3) Update P_i using (5).
(4) Set P_i to $[0.1, 0.9]$ in cases that it is out of the interval.
(5) If the termination condition is met, stop the procedure; otherwise, set $k = k + 1$ and go to (1).

Algorithm 1: Population-based incremental learning.

while the second task is finding the optimum solution for the common design variables. Optimisers for this propose have been developed, for example, two-species genetic algorithm (GA) [12], and hybrid genetic algorithm (HybridGA) [13]. From the literature, it is found that using such evolutionary algorithms for solving an antioptimisation problem is somewhat time-consuming.

This paper is concerned with developing a metaheuristic (MH) for tackling antioptimisation of trusses. The work focuses on the metaheuristic because it can deal with discrete design variables which are usually employed in real-world truss design. Population-based incremental learning (PBIL) is chosen and adapted to deal with truss antioptimisation leading to five variants of PBIL. The five methods and HybridGA are then implemented on five antioptimisation problems of 2D trusses. Optimum results show that PBIL is a powerful optimiser for truss antioptimisation and it is superior to HybridGA.

2. Antioptimisation Problem

A typical single-objective truss optimisation is mass minimisation subject to structural design constraints, which can be expressed as

$$\begin{aligned} &\min_{\mathbf{x}} && f(\mathbf{x}), \\ &\text{subject to} && g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m, \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a vector sized $n_1 \times 1$ of common design variables, f is structural mass, and g_i are inequality constraints for safety requirements. When variables with uncertainties are included, the problem can be written as

$$\begin{aligned} &\min_{\mathbf{x}} && f(\mathbf{x}, \mathbf{y}), \\ &\text{subject to} && g_i(\mathbf{x}, \mathbf{y}) \le 0, \quad i = 1, \ldots, m, \end{aligned} \tag{2}$$

where $\mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u]$ is a vector sized $n_2 \times 1$ of variables with uncertainty having lower and upper bounds as $\mathbf{y}_l$ and $\mathbf{y}_u$, respectively.

For truss antioptimisation, the problem (2) is separated into two subproblems as the design problem (1) for searching for optimum design variables $\mathbf{x}$ and the design problem for finding the worst case $\mathbf{y}$ of a structure written as

$$\max_{\mathbf{y}} F(\mathbf{y}), \quad \mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u], \tag{3}$$

where F is a function indicating the maximum violation of structural constraints. For simplicity, F can be defined as

$$F(\mathbf{y}) = \max(g_i(\mathbf{x}, \mathbf{y})) \tag{4}$$

with the design point $\mathbf{x}$ being unchanged.

A traditional strategy for solving an antioptimisation problem is to use a two-level approach. The search process starts with initial solutions for $\mathbf{x}$ and $\mathbf{y}$. In one iteration, the method searches for the optimum solution of $\mathbf{x}$ based on the problem (1) by fixing the value of $\mathbf{y}$. Then, use the updated values of $\mathbf{x}$ for finding the worst solution of $\mathbf{y}$ based on the problem (3). The process is repeated until the termination conditions are fulfilled.

3. PBIL for Antioptimisation

Over the years, there have been a significant number of metaheuristic algorithms developed for a wide variety of real-world applications, for example, [14–18]. PBIL is chosen for this work because it is a simple but powerful MH and suits well with the concept of antioptimisation. PBIL was first proposed by Baluja in 1994 [19]. The method is based upon a simple estimation of distribution algorithm and searches for an optimum using binary strings. Unlike GA which keeps a set of binary design solutions (better known as a population), PBIL uses the so-called probability vector to represent a binary population. Figure 1 shows some probability row vectors which will be used to produce binary population matrices where one row of the matrix represents a binary design solution. The element P_i of the vector determines the probability of having the string "1" on the ith column of the binary population. From the figure, it can be seen that one probability vector can produce various binary populations. As the optimisation proceeds, the probability vector is updated iteratively leading to an optimum solution. A typical computational procedure of PBIL is given in Algorithm 1. Initially, the probability vector is set as $P_i = 0.5$. Then, the corresponding population is generated where the best binary

Initialisation $P_{x,i} = 0.5$, $P_{y,i} = 0.5$, and find $\mathbf{b} = \{\mathbf{b}_x \quad \mathbf{b}_y\}$ and $\mathbf{w} = \{\mathbf{w}_x \quad \mathbf{w}_y\}$ from an initial population.
Main procedure
(1) For $k = 1$ to N_O (outer loop)
(2) Generate a binary population $\mathbf{B}_x$ according to $P_{x,i}$ and assign $\mathbf{w}_y$ as the second part of every solution in $\mathbf{B}_x$. The population is $\mathbf{B} = [\mathbf{B} \quad \mathbf{w}_y]$.
(3) Perform function evaluations $f(\mathbf{B})$.
(4) Find the new best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$ based on the objective function in (1).
(5) Update $P_{x,i}$ with $\mathbf{b}_x$ using (5).
(6) Set $P_{x,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.
(7) For $j = 1$ to N_I (inner loop)
(8) Generate a binary population $\mathbf{W}_y$ according to $P_{y,i}$ and assign $\mathbf{w}_x$ as the first part of every solution in $\mathbf{W}_y$. The population is $\mathbf{W} = [\mathbf{w}_x \quad \mathbf{W}_y]$.
(9) Perform function evaluations $f(\mathbf{W})$
(10) Find the new worst solution $\mathbf{w}$ from $\mathbf{W} \cup \mathbf{w}$ based on the objective function in (3).
(11) Update $P_{y,i}$ with $\mathbf{w}_y$ using (5).
(12) Set $P_{y,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.
(13) Next j
(14) Next k

Algorithm 2: Population-based incremental learning for antioptimisation.

solution $\mathbf{b}$ is found. The probability vector is then updated based upon the best solution as

$$P_i^{\text{new}} = (1 - L_R) P_i^{\text{old}} + L_R b_i, \tag{5}$$

where P_i^{old} is the ith element of the probability vector from the previous iteration, P_i^{new} is the updated ith element of the probability vector, and b_i is the ith element of the best binary solution. L_R is called a learning rate which herein is set to be a uniform random number in the range of $[0.4, 0.6]$ generated anew when the probability vector update takes place. In this work, mutation is not used to modify the probability vector, but the value of P_i after being updated will be set to the range of $[0.1, 0.9]$ (if the updated value of P_i is lower than 0.1 or higher than 0.9, it will be set as 0.1 and 0.9, resp.) in order to prevent a premature convergence.

When dealing with an antioptimisation problem, the method is modified as shown in Algorithm 2. As it involves two levels of optimisation search, the developed PBIL will be named two-level population-based incremental learning. The algorithm starts with generating an initial binary population where each binary solution contains strings for design variables $\mathbf{x}$ and variables with uncertainties $\mathbf{y}$. Then, the best binary solution $\mathbf{b}$ containing two parts as $\mathbf{b}_x$ for $\mathbf{x}$ and $\mathbf{b}_y$ for $\mathbf{y}$ is found based on the design problem (1). On the other hand, the worst binary solution $\mathbf{w} = \{\mathbf{w}_x \quad \mathbf{w}_y\}$ is obtained based on the problem (3). The method uses two probability vectors as $\mathbf{P}_x$ and $\mathbf{P}_y$ for representing binary populations of $\mathbf{x}$ and $\mathbf{y}$, respectively. Afterwards, a set of binary design solutions for $\mathbf{x}$ (denoted as $\mathbf{B}_x$) are created based on $\mathbf{P}_x$, while each solution in the population has its second part of a binary string for $\mathbf{y}$ as $\mathbf{w}_y$. The new best solution $\mathbf{b}$ is obtained from sorting the current population and the best solution from the previous iteration. The probability vector $\mathbf{P}_x$ is updated using the first part of the new best solution $\mathbf{b}_x$. This is the first level of the process which is said to be outer loops.

For inner loops (steps (7)–(13)) which are the search for the worst case scenario, a set of binary solutions for $\mathbf{y}$ (denoted as $\mathbf{W}_y$) is generated from the probability vector $\mathbf{P}_y$ where each solution has the first part of a binary string for $\mathbf{x}$ as $\mathbf{w}_x$ which is the first part of the current worst solution. The new worst solution $\mathbf{w}$ is then obtained based on the problem (3) by sorting the current population and the previous worst solution. The probability $\mathbf{P}_y$ is updated by using $\mathbf{w}_y$, the second part of $\mathbf{w}$. The inner loop procedure is stopped after N_I loops. Similarly, the outer loop search is terminated after N_O loops. This implies that if the population size for both inner and outer loops is N_P, the total number of function evaluations for each simulation run is $N_O(N_I + 1)N_P$. It should be noted that we use the worst solution for finding the worst case structures because it is expected to give the worst objective function values which may help accelerate PBIL to the worst case. In the case studies, we will examine the use of best and worst solutions for searching the worst case scenario.

4. Truss Design Case Studies

Five antioptimisation problems (AOPs) of three truss structures will be used to test the proposed algorithms. The first truss is called a 2-bar truss shown in Figure 2. The structure is subject to a vertical load F with uncertainty at node 3. The second structure is named a 10-bar truss [20] shown in Figure 3. The structure consists of 6 nodes and 10 bars where nodes 1 and 2 are fixed and nodes 4 and 6 are subject to external forces F_1 and F_2, respectively. The third structure is called a 25-bar truss as displayed in Figure 3. The truss has 12 nodes and 25 bars where nodes 1 and 12 are fixed. The structure is applied by loads F_1, F_2, F_3, F_2, and F_1 at nodes 2, 4, 6, 8, and 10, respectively. The design problems can be detailed as follows.

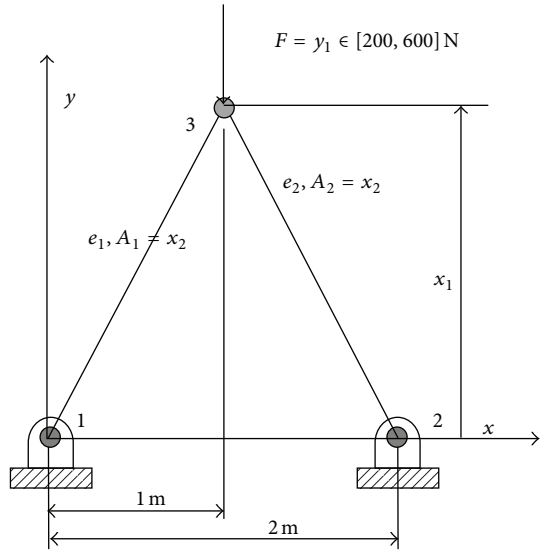

Figure 2: 2-bar truss.

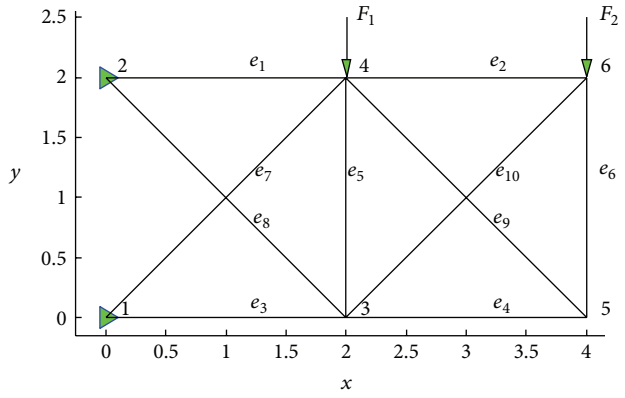

Figure 3: 10-bar truss.

AOP1: Optimisation of the 2-Bar Truss. The design problem can be written as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}) \\ \text{Subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{\text{yt}} \le 0, \ \text{N/m}^2, \\ & P_i - P_{\text{cr}} \le 0, \ \text{N}, \\ & x_1 \in [0.5, 1.5], \ \text{m}, \\ & x_2 = A_1 = A_2 \in [0.001, 0.01], \ \text{m}, \\ & 200 \le y_1 = F \le 600, \ \text{N}, \end{aligned} \tag{6}$$

where A_i is the cross-sectional area of the ith bar element. $\sigma_{\max}$ is the maximum stress on truss elements. P_i is the axial load acting on the ith element, while P_{cr} is the critical buckling load for the element. The structures for all 5 test problems are made up of material with the Young modulus $E = 200 \times 10^9$ N/m^2, yield stress $\sigma_{\text{yt}} = 235 \times 10^6$ N/m^2, and density $\rho = 7800$ kg/m^3. The common design variables are shape and sizing variables. The external force F is set to be variables with uncertainties. The task is to find the external force that gives the worst case scenario for the structure while simultaneously trying to perform mass minimisation.

AOP2: Sizing Optimisation of the 10-Bar Truss. The second design problem can be written as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}), \\ \text{subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{\text{yt}} \le 0, \ \text{N/m}^2, \\ & P_i - P_{\text{cr}} \le 0, \ \text{N}, \\ & U_{\max} - 0.005 \le 0, \ \text{m}, \\ & x_i = d_i \in \{0.04, 0.05, 0.06, 0.07, 0.08, 0.09, \\ & \qquad 0.10, 0.20\}, \ \text{m}, \\ & \{75, 150\}^T \le \mathbf{y} = \{F_1, F_2\}^T \le \{80, 175\}^T, \ \text{N}, \end{aligned} \tag{7}$$

where d_i is the diameter of the ith bar element. $U_{\max}$ is the maximum y-direction displacement of the structure. The common design variables are set to be discrete so that it is close to a practical truss design process. The external forces F_1 and F_2 are set to be variables with uncertainties. The task is to find the external forces that give the worst case scenario for the structure while, at the same time, trying to perform mass minimisation with sizing design variables.

AOP3: Sizing Optimisation of the 25-Bar Truss. The design problem can be written as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}), \\ \text{subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{\text{yt}} \le 0, \ \text{N/m}^2, \\ & P_i - P_{\text{cr}} \le 0, \ \text{N}, \\ & U_{\max} - 0.005 \le 0, \ \text{m}, \\ & x_i \in \{0.06, 0.07, 0.08, 0.09, 0.10, 0.15, \\ & \qquad 0.20, 0.25\} \ \text{m}; \quad i = 1, \ldots, 13, \\ & \left\{150, 150, 150, 199.00 \times 10^9\right\}^T \\ & \qquad \le \mathbf{y} = \{F_1, F_2, F_3, E\}^T \\ & \qquad \le \{300, 300, 300, 201.00 \times 109\}^T. \end{aligned} \tag{8}$$

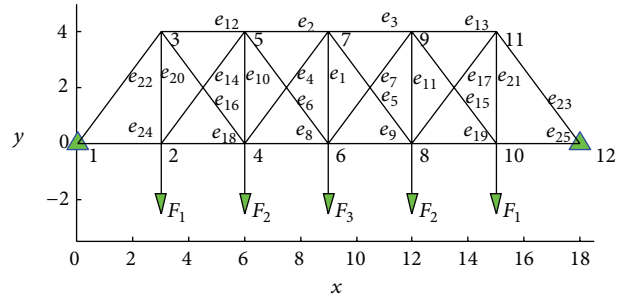

Figure 4: 25-bar truss.

Table 1: Optimum results of AOP1.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.5048	0.0242	0.5613	0.1011	0.6247	0.1176	0.6333	0.1587	0.6344	0.1558	0.5914	0.1302
x_2	0.0029	0.0001	0.0038	0.0003	0.0037	0.0004	0.0038	0.0005	0.0038	0.0004	0.0038	0.0004
y_1	200.0000	0.0000	399.1398	130.0818	370.3226	142.7784	400.8602	164.3949	405.1613	157.4620	424.0860	151.7861
$f(\mathbf{x}, \mathbf{y})$	49.9355	1.6789	67.2216	4.8174	68.2111	5.5543	70.2185	5.9409	69.9382	4.6033	69.4287	5.3346
$\max g_i$	−0.0018	0.0024	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000

In this case, there are uncertainties in applied loads and material property. The sizing design variables are set to have a symmetric structure. Thus, we have

(i) x_1 for d_1,

(ii) x_2 for d_2 and d_3,

(iii) x_3 for d_4 and d_5,

(iv) x_4 for d_6 and d_7,

(v) x_5 for d_8 and d_9,

(vi) x_6 for d_{10} and d_{11},

(vii) x_7 for d_{12} and d_{13},

(viii) x_8 for d_{14} and d_{15},

(ix) x_9 for d_{16} and d_{17},

(x) x_{10} for d_{18} and d_{19},

(xi) x_{11} for d_{20} and d_{21},

(xii) x_{12} for d_{22} and d_{23},

(xiii) x_{13} for d_{24} and d_{25}.

AOP4: Shape and Sizing Optimisation of the 10-Bar Truss. The fourth design problem is the same problem as AOP2 with additional 4 shape design variables for $\mathbf{x}$. The shape variables determine the positions of nodes 3 and 5 in both x and y directions. The design variables are used to modify the original positions (shown in Figure 3) of nodes 3 and 5. The bound constraints are set as $-0.75\,\text{m} \le x_i \le 0.75\,\text{m}$ for $i = 11, \ldots, 14$.

AOP5: Shape and Sizing Optimisation of the 25-Bar Truss. The fifth design problem is the same problem as AOP2 with additional 3 shape design variables for $\mathbf{x}$. The shape variables determine the y-direction positions of nodes 3, 5, 7, 9, and 11, where x_{14} is set for the positions of nodes 3 and 11, x_{15} is set for the positions of nodes 5 and 9, and x_{16} is set for the positions of node 7. The design variables are used to modify the original positions (shown in Figure 4) of those nodes. The bound constraints are set as $-3.00\,\text{m} \le x_i \le 3.00\,\text{m}$ for $i = 14, \ldots, 16$.

By defining a few set of design parameter settings, various versions of PBIL can be obtained as follows.

(i) PBILb5: this optimiser uses $\mathbf{b}_x$ from the best solution $\mathbf{b}$ instead of $\mathbf{w}_x$ from the worst solution $\mathbf{w}$ in step (8) of Algorithm 2. N_I is set to be 5.

(ii) PBILw5: this is PBIL detailed in Algorithm 2, where N_I is set to be 5.

(iii) PBILw5m: this is PBILw5 with GA mutation. Having generated binary populations in steps (2) and (8), The GA mutation is applied to the populations, where each solution has the probability of being mutated as 0.25.

(iv) PBILw1: this is PBIL detailed in Algorithm 2, where N_I is set to be 1.

(v) PBILw10: this is PBIL detailed in Algorithm 2, where N_I is set to be 10.

For AOP1, the population size is set to be 40, whereas the total number of function evaluations for stopping the search procedure is 2,400. For other design problems, the population size is set to be 100, whereas the total number of function evaluations for stopping the search procedure is 15,000. PBIL with various N_I values is set to examine the effect of the inner and outer loops on the search performance. The GA mutation is employed under the assumption that it may help improving PBIL convergence rate. Each optimiser is used to solve the antioptimisation problems 30 runs. Since PBIL cannot

Table 2: Optimum results of AOP2.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0453	0.0073	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0503	0.0018
x_2	0.0463	0.0096	0.0493	0.0025	0.0500	0.0026	0.0497	0.0032	0.0477	0.0043	0.0543	0.0063
x_3	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0807	0.0025
x_4	0.0610	0.0040	0.0603	0.0018	0.0600	0.0000	0.0603	0.0018	0.0600	0.0000	0.0650	0.0068
x_5	0.0500	0.0087	0.0493	0.0025	0.0500	0.0026	0.0500	0.0000	0.0477	0.0043	0.0520	0.0041
x_6	0.0607	0.0025	0.0613	0.0051	0.0610	0.0040	0.0607	0.0025	0.0600	0.0000	0.0627	0.0064
x_7	0.0810	0.0031	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0803	0.0018
x_8	0.0517	0.0099	0.0503	0.0018	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0517	0.0038
x_9	0.0427	0.0052	0.0493	0.0025	0.0500	0.0000	0.0500	0.0000	0.0483	0.0038	0.0493	0.0025
x_{10}	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0803	0.0018
y_1	74.7849	1.1783	77.5938	1.1768	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000
y_2	171.5082	13.5158	165.3552	7.5152	175.0000	0.0000	175.0000	0.0000	174.9984	0.0089	175.0000	0.0000
$f(\mathbf{x}, \mathbf{y})$	558.7238	30.9474	567.8900	9.9080	568.7844	7.3917	568.2535	6.2552	559.2016	7.1018	594.2857	20.6724
$\max g_i$	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000

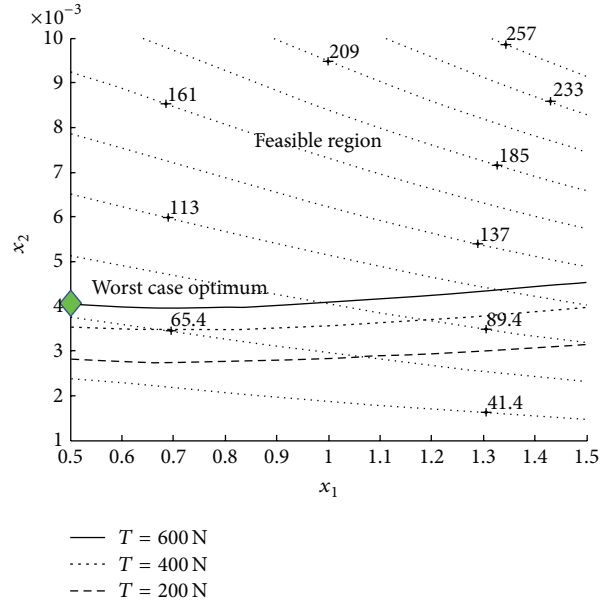

Figure 5: Optimum of AOP1.

deal with design constraints directly, the penalty function technique based on the fuzzy set theory [21] is employed. It should be noted that we have tested using several penalty function techniques earlier, and the fuzzy set theory gives the best results. The metaheuristics employed do not have reliable termination criterion, therefore, their search will be stopped based on the maximum number of function evaluations.

Furthermore, in order to verify the performance of the proposed algorithms, an existing evolutionary algorithm for antioptimisation called a hybrid genetic algorithm [13] will be employed to solve the test problems and compared with the various PBILs. For this performance test, HybridGA uses real codes for design variables. The method of Hooke and Jeeves is used for searching the worst case. The number of good solutions taken for mutation is set to be 10. The roulette wheel selection method is used to select design solutions for crossover operation. The optimiser is terminated when the maximum number of function evaluations set for PBILs in the previous paragraph is reached.

5. Optimum Results

The optimum results of AOP1 are given in Table 1. Figure 5 displays the contour of the objective function and the boundaries of buckling constraints with $F = 200$ N, 400 N, and 600 N. It should be noted that the stress constraints do not affect the feasible region for this design case. From the figure, it is seen that the buckling constraint where $F = 600$ N

Table 3: Optimum results of AOP3.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0667	0.0106	0.0753	0.0078	0.0707	0.0037	0.0810	0.0243	0.0697	0.0032	0.0817	0.0172
x_2	0.1017	0.0091	0.1533	0.0127	0.1550	0.0201	0.1483	0.0207	0.1500	0.0000	0.1600	0.0242
x_3	0.0800	0.0037	0.0820	0.0076	0.0847	0.0073	0.0850	0.0078	0.0817	0.0046	0.0883	0.0079
x_4	0.0690	0.0109	0.0727	0.0052	0.0750	0.0078	0.0747	0.0082	0.0700	0.0037	0.0800	0.0164
x_5	0.0690	0.0109	0.0770	0.0160	0.0770	0.0084	0.0843	0.0355	0.0750	0.0051	0.0867	0.0121
x_6	0.0660	0.0104	0.0690	0.0061	0.0713	0.0035	0.0737	0.0156	0.0687	0.0043	0.0750	0.0073
x_7	0.1000	0.0000	0.1283	0.0252	0.1500	0.0131	0.1517	0.0207	0.1500	0.0000	0.1500	0.0186
x_8	0.0923	0.0043	0.1027	0.0131	0.0997	0.0018	0.1053	0.0229	0.0993	0.0025	0.1050	0.0185
x_9	0.0680	0.0092	0.0730	0.0065	0.0727	0.0069	0.0767	0.0103	0.0720	0.0081	0.0843	0.0155
x_{10}	0.0680	0.0096	0.0750	0.0078	0.0760	0.0097	0.0793	0.0083	0.0710	0.0031	0.0827	0.0108
x_{11}	0.0657	0.0068	0.0770	0.0099	0.0727	0.0074	0.0770	0.0158	0.0697	0.0032	0.0820	0.0167
x_{12}	0.1500	0.0000	0.1500	0.0000	0.1517	0.0091	0.1517	0.0091	0.1500	0.0000	0.1583	0.0190
x_{13}	0.0777	0.0057	0.0847	0.0090	0.0873	0.0078	0.0907	0.0087	0.0840	0.0056	0.0913	0.0221
y_1	150.0000	0.0000	279.9365	13.8161	297.8886	11.5648	293.5239	21.5257	300.0000	0.0000	299.7019	1.6330
y_2	150.0000	0.0000	278.6706	15.4757	296.4614	19.3818	289.2913	33.1890	300.0000	0.0000	298.5484	7.9508
y_3	150.0000	0.0000	252.3558	33.5946	297.3412	14.5631	289.8240	31.2534	300.0000	0.0000	297.0088	16.3835
y_4	$2.009E+11$	$5.409E+07$	$2.001E+11$	$5.567E+08$	$1.990E+11$	$1.820E+07$	$1.991E+11$	$3.734E+08$	$1.990E+11$	$4.828E+07$	$1.990E+11$	$8.067E+07$
$f(\mathbf{x},\mathbf{y})$	4751.2034	230.5610	5974.5867	363.5549	6222.8573	456.2727	6566.8994	1086.3438	5912.7947	92.7879	7045.4299	1006.9112
$\max g_i$	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000

Table 4: Optimum results of AOP4.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0467	0.0084	0.0517	0.0053	0.0520	0.0041	0.0513	0.0043	0.0490	0.0031	0.0533	0.0084
x_2	0.0477	0.0107	0.0510	0.0040	0.0503	0.0018	0.0513	0.0035	0.0507	0.0045	0.0543	0.0082
x_3	0.0833	0.0048	0.0867	0.0055	0.0863	0.0067	0.0867	0.0048	0.0860	0.0050	0.0873	0.0069
x_4	0.0603	0.0049	0.0577	0.0082	0.0607	0.0098	0.0577	0.0068	0.0563	0.0076	0.0610	0.0099
x_5	0.0570	0.0070	0.0543	0.0068	0.0590	0.0092	0.0533	0.0071	0.0553	0.0078	0.0583	0.0105
x_6	0.0613	0.0043	0.0603	0.0072	0.0617	0.0079	0.0603	0.0093	0.0577	0.0043	0.0653	0.0107
x_7	0.0720	0.0130	0.0783	0.0046	0.0773	0.0058	0.0797	0.0056	0.0760	0.0081	0.0793	0.0087
x_8	0.0457	0.0082	0.0507	0.0037	0.0500	0.0037	0.0507	0.0037	0.0490	0.0031	0.0523	0.0082
x_9	0.0440	0.0056	0.0510	0.0031	0.0507	0.0025	0.0497	0.0032	0.0483	0.0038	0.0507	0.0045
x_{10}	0.0787	0.0035	0.0787	0.0051	0.0797	0.0041	0.0770	0.0047	0.0780	0.0048	0.0810	0.0055
x_{11}	0.0518	0.1773	0.3132	0.2705	0.1478	0.3773	0.3319	0.3448	0.2337	0.3527	0.2345	0.4409
x_{12}	0.5991	0.1766	0.5610	0.1850	0.6211	0.1079	0.5072	0.2017	0.6345	0.1444	0.5174	0.1896
x_{13}	−0.6919	0.1381	−0.5828	0.1442	−0.5369	0.2001	−0.4271	0.2176	−0.6712	0.1039	−0.4691	0.2160
x_{14}	0.6804	0.1490	0.5697	0.1723	0.5550	0.1935	0.5924	0.1489	0.6494	0.1410	0.5549	0.1754
y_1	75.3793	1.2182	78.1703	1.5065	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000
y_2	170.9637	7.2167	166.1746	7.4693	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000
$f(\mathbf{x}, \mathbf{y})$	479.4000	34.7275	529.8728	21.6285	535.8320	21.6881	534.5283	22.3252	501.9159	11.7035	574.2029	37.1328
$\max g_i$	−0.0047	0.0009	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000

gives the most narrow feasible region and therefore leads to the worst case results. The optimum for the worst case scenario is $\mathbf{x} = \{0.5, 0.0041\}^T$ and $y = 600$ N. According to the results obtained from the various optimisers, HybridGA cannot capture the worst case for all runs, while PBILs can find the worst case for some optimisation runs. The best solutions obtained by using five versions of PBIL are the same, that is, $\mathbf{x} = \{0.5, 0.0042\}^T$ and $y = 600$ N, which can be classified as a near optimum of the design problem. Based on the worst cases being explored, PBILw10 gives the best results, while the second best is PBILw1.

The optimum results of AOP2 obtained from the various optimisers are given in Table 2. For this design problem, it is obvious that the worst possible case for the 10-bar truss is when $\{F_1, F_2\}^T = \{80, 175\}^T$ N. From the results, it can be seen that HybridGA and PBILb5 fail to explore the worst case, while other optimiser which are based on using $\mathbf{w}_x$ in step (8) of Algorithm 2 can find the worst case for almost every optimisation run. According to the mean value of objective function from 30 runs, PBILw1 gives the best results. However, it is found that PBILw1 cannot find the worst case for all 30 runs, although the values are not significant. This is due to the lower number of inner loops used to explore the worst case. In contrast, PBILw10 gives the worst results among PBILw variants because it spends less time in searching for the optimum mass. By comparing between PBILw5 and PBILw5m, the GA mutation gives insignificant improvement for PBIL.

The optimum results of AOP3 obtained from the various PBILs are given in Table 3. The worst case is $\{F_1, F_2, F_3, E\}^T = \{1000, 1000, 1000, 199.0 \times 10^9\}^T$. Similar conclusions as with the AOP2 case can be drawn except that PBILw5 is slightly better than PBILw5m. Table 4 shows the results of AOP4. Similar to the first two cases, HybridGA and PBILb5 fail to explore the worst case scenario, while, for this case, all the versions of PBILw can capture the worst case for all runs. PBILw1 gives the best results, while the worst among PBILw versions is PBILw10. For the fifth design case as shown in Table 5, PBILw optimisers can capture the worst case scenario for almost 30 runs, while PBILb5 and HybridGA fail to find out the worst case solutions. The best optimiser is PBILw1 whereas PBILw5m is slightly better than PBILw5.

Based on the standard deviation, the variants of PBILw are rather consistent. From the constraint violation values in Tables 1–5, it can be said that the fuzzy set theory is efficient and effective for constrained truss design. Table 6 shows the total number of worst cases captured by the optimisers having solved the design problems for 30 runs. It can be seen that PBILw is effective for exploring the worst case scenarios for AOP2-5. PBILw10 fails to capture the worst case once for AOP3 and AOP5, although it spends more time to search for the worst case scenario. This can possibly be due to a premature convergence of the algorithm. For the first design problem, all the optimisers have more difficulty to search for the worst case.

The best structures obtained from running PBILw5, PBILw5m, PBILw1, and PBILw10 for AOP2, AOP3, AOP4, and AOP5 are illustrated in Figures 6, 7, 8, and 9, respectively. The structures obtained from the 4 optimisers for each design case are rather similar in size and shape to each other. In cases of the design problems AOP2-5, with 15,000 function evaluations and population size of 100, PBILw1 has a number of outer loops as $N_O = 75$ loops, while PBILw5 and PBILw10 have only 25 and 14 loops, respectively. In the test problems, the number of inner loops $N_I = 1$ is adequate to search

Table 5: Optimum results of AOP5.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0787	0.0157	0.0787	0.0082	0.0813	0.0111	0.0800	0.0105	0.0730	0.0065	0.0870	0.0115
x_2	0.1233	0.0286	0.1533	0.0127	0.1483	0.0091	0.1500	0.0131	0.1500	0.0000	0.1577	0.0311
x_3	0.0830	0.0174	0.0780	0.0119	0.0833	0.0175	0.0810	0.0171	0.0717	0.0087	0.0903	0.0227
x_4	0.0770	0.0124	0.0810	0.0212	0.0787	0.0111	0.0747	0.0104	0.0727	0.0087	0.0883	0.0261
x_5	0.0807	0.0141	0.0770	0.0079	0.0803	0.0096	0.0803	0.0110	0.0717	0.0065	0.0847	0.0117
x_6	0.0793	0.0131	0.0740	0.0062	0.0783	0.0099	0.0770	0.0088	0.0710	0.0031	0.0857	0.0205
x_7	0.1200	0.0249	0.1483	0.0091	0.1533	0.0183	0.1567	0.0217	0.1517	0.0091	0.1557	0.0310
x_8	0.0893	0.0187	0.0857	0.0176	0.0840	0.0214	0.0840	0.0181	0.0750	0.0107	0.0963	0.0318
x_9	0.0780	0.0140	0.0807	0.0098	0.0827	0.0120	0.0777	0.0094	0.0747	0.0094	0.0833	0.0163
x_{10}	0.0810	0.0106	0.0783	0.0095	0.0783	0.0105	0.0777	0.0101	0.0713	0.0035	0.0817	0.0162
x_{11}	0.0817	0.0109	0.0773	0.0098	0.0823	0.0205	0.0767	0.0152	0.0707	0.0052	0.0850	0.0242
x_{12}	0.1500	0.0000	0.1533	0.0127	0.1500	0.0000	0.1500	0.0000	0.1500	0.0000	0.1540	0.0234
x_{13}	0.0857	0.0161	0.0823	0.0168	0.0860	0.0203	0.0823	0.0165	0.0737	0.0089	0.0940	0.0302
x_{14}	−1.3764	0.8037	−1.9818	0.7962	−1.8035	0.7824	−1.5169	0.9604	−1.8596	0.7010	−1.5277	1.0779
x_{15}	−0.6342	0.7655	−0.9292	1.0364	−0.6835	1.0393	−0.5185	0.9608	−0.7484	0.9619	−0.5791	1.3451
x_{16}	−0.4087	0.7922	−0.3812	1.1555	−0.3243	1.1576	−0.1742	1.0618	−0.4516	0.9760	−0.3193	1.3226
y_1	194.0161	59.2515	238.8221	48.0552	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	299.5406	2.5164
y_2	168.5592	35.6225	251.6178	31.7033	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	296.8964	16.9992
y_3	239.6624	57.7826	231.6178	47.0528	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	299.4868	2.8109
y_4	$1.995E+11$	$6.119E+08$	$1.999E+11$	$5.890E+08$	$1.990E+11$	$0.000E+00$	$1.990E+11$	$0.000E+00$	$1.990E+11$	$0.000E+00$	$1.990E+11$	$1.902E+08$
$f(\mathbf{x},\mathbf{y})$	5222.3215	813.5479	5440.5355	524.4721	5616.7374	429.4775	5679.4132	654.9014	4938.2933	266.2513	6585.0710	1474.1376
$\max g_i$	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	−0.0050	0.0000	68.9978	377.9440

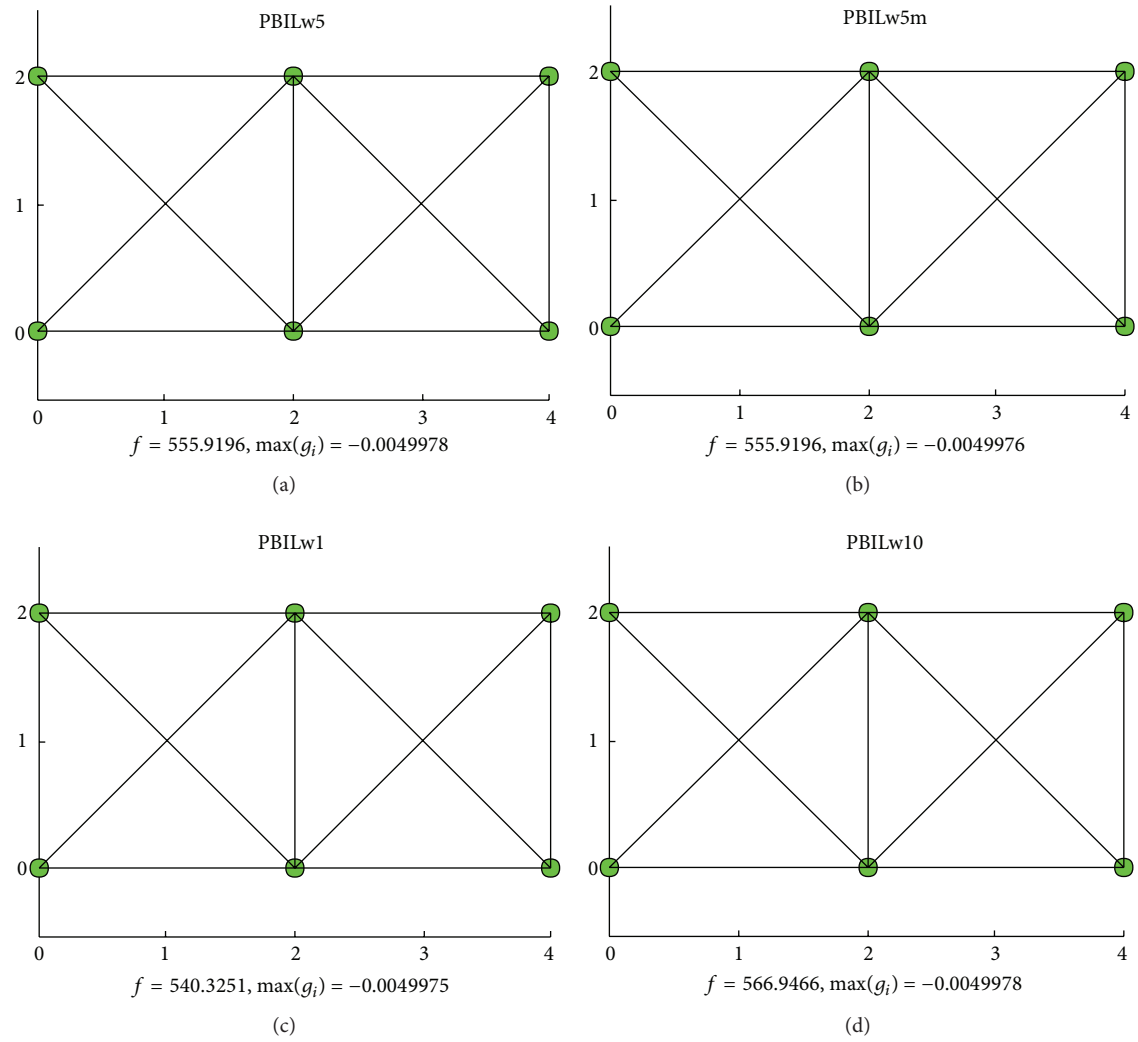

Figure 6: Optimum structures of AOP2 from various optimisers.

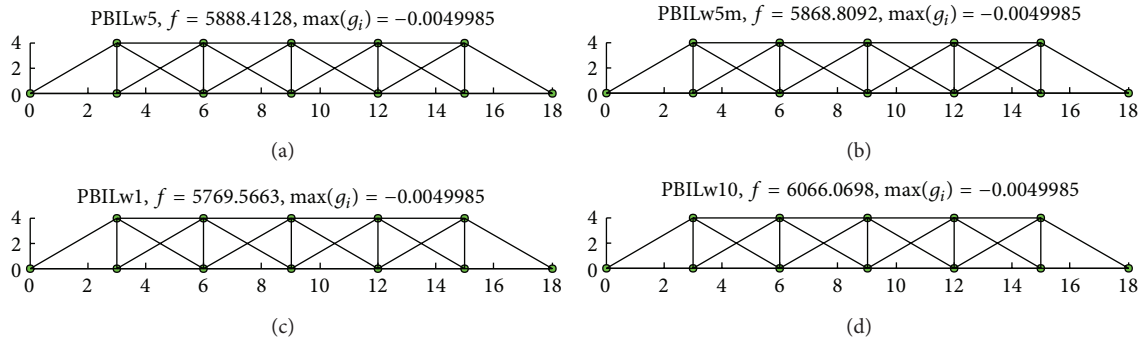

Figure 7: Optimum structures of AOP3 from various optimisers.

for the worst cases for all design cases; thus, PBILw1 which spends more time to search for the optimum solution of $\mathbf{x}$ gives the best results for all design cases. In order to use the proposed PBIL for antioptimisation of other engineering systems, predefining the proper value of N_I is important since it can affect the search performance. This relies on several factors such as the size of variables $\mathbf{x}$ and $\mathbf{y}$. The hybrid genetic algorithm, on the other hand, is not efficient for the truss design problems. However, optimisation parameter settings can affect its search performance.

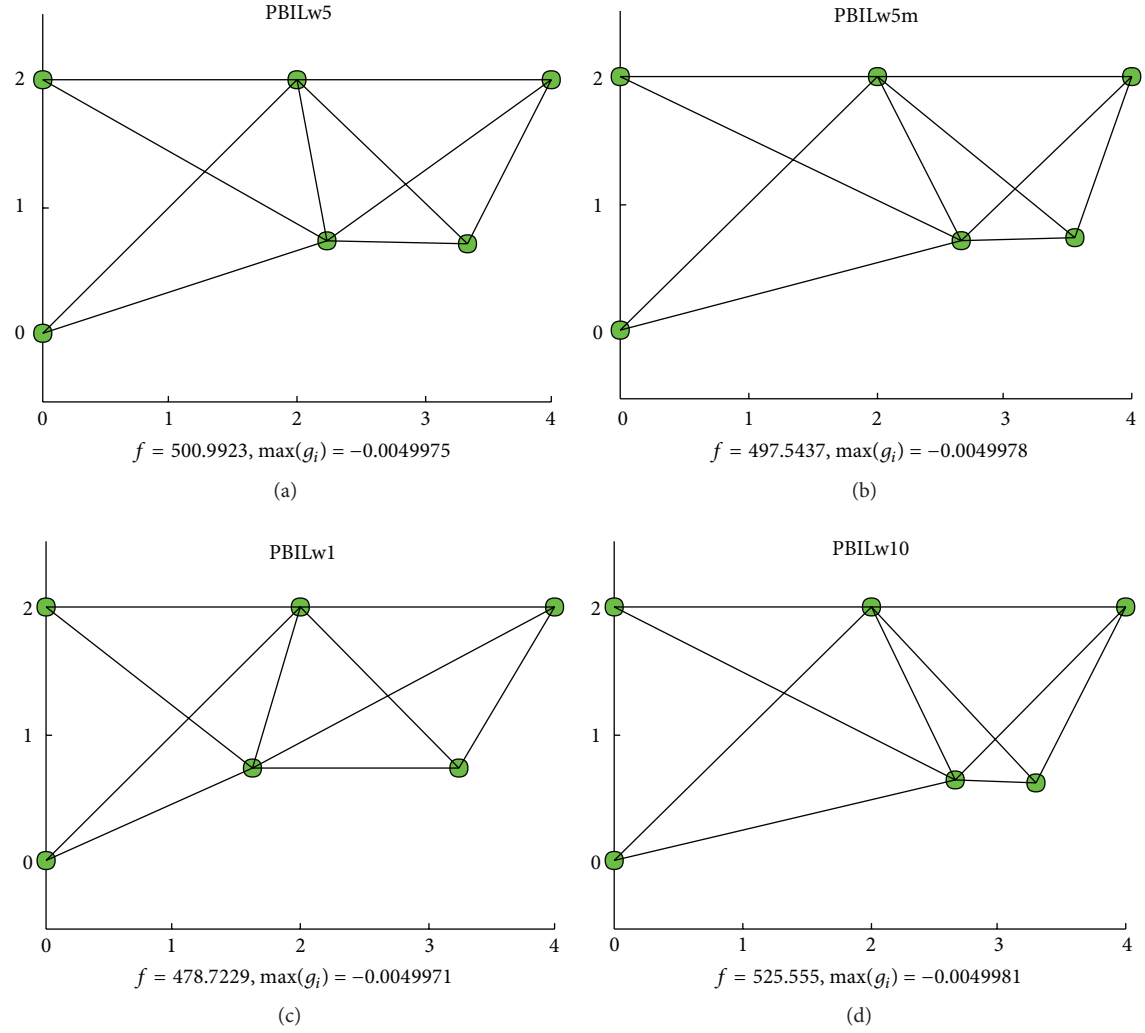

Figure 8: Optimum structures of AOP4 from various optimisers.

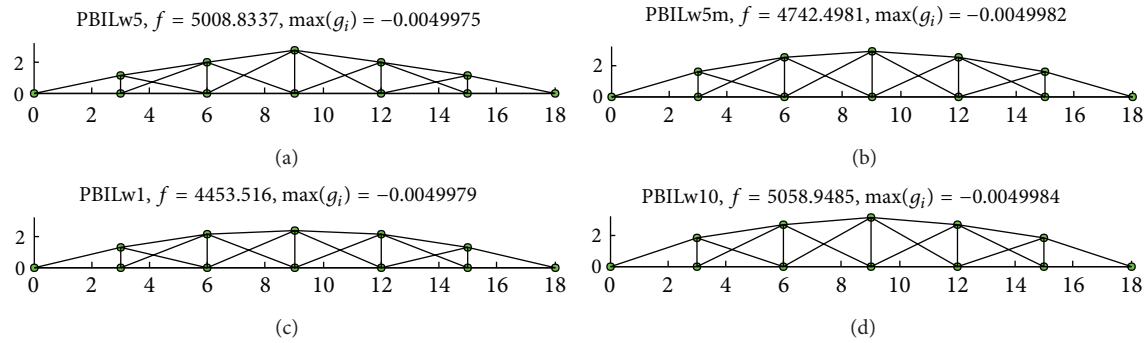

Figure 9: Optimum structures of AOP5 from various optimisers.

6. Conclusions and Discussion

A metaheuristic based on PBIL for antioptimisation of trusses is developed, and it is termed two-level PBIL. The method consists of two levels for two searching tasks. The first level is an outer loop used to explore the optimum of common design variables, while the inner loop is used to find the worst case scenario of the variables with uncertainties. It is shown that when searching for the worst case, the function evaluation based on the current worst solution gives better search results than using the current best solution. The GA mutation applied to a binary population of PBIL does not have significantly impact on PBILw search performance. The results show that the proposed PBILw is an efficient optimiser

Table 6: Number of worst cases found by the optimisers.

MHs	AOP1	AOP2	AOP3	AOP4	AOP5
HybridGA	0	0	0	0	0
PBILb5	1	0	0	0	0
PBILw5	5	30	29	30	30
PBILw5m	8	30	27	30	30
PBILw1	8	29	24	30	30
PBILw10	9	30	29	30	29

for tackling a truss antioptimisation problem. Moreover, it is superior to the hybridGA. The performance of PBILw, to a great extent, relies on the predefined number of inner loops for searching the worst case. It can be concluded that the concept of using probability vectors to represent binary populations in PBIL is powerful for dealing with antioptimisation. For future work, the hybrid approach of PBIL will be studied. Also, the applications of the two-level PBIL will be employed to tackle other antioptimisation problems.

Acknowledgments

The authors are grateful for the support from The Royal Golden Jubilee Ph. D. Program (Grant no. PHD/0145/2553) and the Thailand Research Fund (Grant no. BRG5580017).

References

[1] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267–283, 2009.

[2] P. B. Thanedar, J. S. Arora, C. H. Tseng, O. K. Lim, and G. J. Park, "Performance of some SQP algorithms on structural design problems," *International Journal for Numerical Methods in Engineering*, vol. 23, no. 12, pp. 2187–2203, 1986.

[3] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267–283, 2009.

[4] H. M. Gomes, "Truss optimization with dynamic constraints using a particle swarm algorithm," *Expert Systems with Applications*, vol. 38, no. 1, pp. 957–968, 2011.

[5] W. A. Bennage and A. K. Dhingra, "Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing," *International Journal for Numerical Methods in Engineering*, vol. 38, no. 16, pp. 2753–2773, 1995.

[6] N. Pholdee and S. Bureerat, "Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient," *Computers and Structures*, vol. 106-107, pp. 115–124, 2012.

[7] C. Noilublao and S. Bureerat, "Topology and sizing optimization of trusses with adaptive ground finite elements using multiobjective PBIL," *Advanced Materials Research*, vol. 308–310, pp. 1116–1121, 2011.

[8] N. Noilublao and S. Bureerat, "Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms," *Computers and Structures*, vol. 89, pp. 2531–2538, 2011.

[9] B. M. Adams, M. S. Eldred, and J. W. Wittwer, "Reliability-based design optimization for shape design of compliant micro-electro-mechanical systems," in *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pp. 1042–1056, AIAA, September 2006.

[10] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, "Reliability-based optimization using evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054–1074, 2009.

[11] M. Lombardi and R. T. Haftka, "Anti-optimization technique for structural design under load uncertainties," *Computer Methods in Applied Mechanics and Engineering*, vol. 157, no. 1-2, pp. 19–31, 1998.

[12] G. Venter and R. T. Haftka, "Two-species genetic algorithm for design under worst case conditions," *Evolutionary Optimization*, vol. 2, no. 1, pp. 1–19, 2000.

[13] N. Wang and Y. Yang, "Optimization of structures under load uncertainties based on hybrid genetic algorithm," in *Evolutionary Computation*, W. P. dos Santos, Ed., pp. 321–340, I-Tech, Vienna, Austria, 2009.

[14] A. R. Yıldız, "A novel hybrid immune algorithm for global optimization in design and manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 261–270, 2009.

[15] I. Durgun and A. R. Yıldız, "Structural design optimization of vehicle components using cuckoo search algorithm," *Materials Testing*, vol. 54, no. 3, pp. 185–188, 2012.

[16] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *Journal of Applied Mathematics*, vol. 2012, Article ID 467631, 12 pages, 2012.

[17] X. Cai, S. Fan, and Y. Tan, "Light responsive curve selection for photosynthesis operator of APOA," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 373–379, 2012.

[18] Z. Cui, F. Gao, Z. Cui, and J. Qu, "A second nearest-neighbor embedded atom method interatomic potential for Li-Si Alloys," *Journal of Power Sources*, vol. 207, p. 150, 2012.

[19] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Tech. Rep. CMU_CS_95_163, Carnegie Mellon University, 1994.

[20] Z. Qiu and X. Wang, "Structural anti-optimization with interval design parameters," *Structural and Multidisciplinary Optimization*, vol. 41, no. 3, pp. 397–406, 2010.

[21] F. Y. Cheng and D. Li, "Fuzzy set theory with genetic algorithm in constrained structural optimization," in *Proceedings of the 1st US-Japan Joint Seminar on Structural Optimization*, pp. 55–66, Advances in Structural optimization, April 1997.