International Journal of Computational Intelligence Systems, Vol.3, No. 6 (December, 2010), 832-842

Comprehensive Learning Particle Swarm Optimizer for Constrained Mixed-Variable Optimization Problems

Lei Gao
School of Agricultural and Resource Economics, University of Western Australia, 35 Stirling Highway, Crawley Perth, 6009, Western Australia
E-mail: dr.leigao@gmail.com

Atakelty Hailu
School of Agricultural and Resource Economics, University of Western Australia, 35 Stirling Highway, Crawley Perth, 6009, Western Australia
E-mail: atakelty.hailu@uwa.edu.au

Received: 07-05-2010
Accepted: 05-10-2010

Abstract

This paper presents an improved particle swarm optimizer (PSO) for solving multimodal optimization problems with problem-specific constraints and mixed variables. The standard PSO is extended by employing a comprehensive learning strategy, different particle updating approaches, and a feasibility-based rule method. The experiment results show the algorithm located the global optima in all tested problems, and even found a better solution than those previously reported in the literature. In some cases, it outperforms other methods in terms of both solution accuracy and computational cost.

Keywords: Particle swarm optimization, mixed variables, feasibility-based rules, constrained optimization, evolutionary algorithms, comprehensive learning strategy

1. Introduction

Many real-world optimization problems are hard to solve because they are: (1) computationally intensive and multimodal (i.e. have many local optima); (2) heavily constrained; and/or (3) contain mixtures of continuous, integer, discrete, and/or binary variables, and are often referred to as mixed-variable nonlinear optimization problems.

In the past few decades, evolutionary algorithms (EAs)[1–5], such as genetic algorithms, evolutionary programming, and evolutionary strategies, have been successfully applied to real-world optimization problems. The main advantage of these algorithms, relative to most conventional optimization methods (e.g., Newton-based techniques, linear programming, and interior point methods) lies in that they do not apply mathematical assumptions to the optimization problems and have better global search capabilities. A relatively new EA, particle swarm optimization (PSO), was first proposed by Kennedy and Eberhart[6,7]. PSO is an algorithm inspired by the social behavior of animals, such as bird flocking and fish schooling. It is attractive because of its simplicity of implementation and its ability to quickly converge to a reasonably good solution[7,8]. PSO has been successfully applied in a variety of fields mainly for unconstrained continuous optimization problems. However, it may get trapped in a local optimum when solving complex multimodal problems. In this paper, the standard PSO algorithm is extended to improve PSO's performance on complex

multimodal problems by using a comprehensive learning strategy.

In real life, many practical optimization problems involve continuous as well as discrete, integer and binary variables. To solve those problems, some solutions based on conventional methods are proposed. For example, Sandgren[9] and Hajela and Shih[10] proposed nonlinear branch and bound algorithms, which are modified versions of the most widely used methods in integer programming. Fu et al.[11] developed an interior penalty approach to impose penalties on integer and/or discrete violations on the objective function to force the search to converge upon standard values. Loh and Papalambros introduced a sequential linearization approach for solving mixed-discrete nonlinear optimization problems[12]. Although most of the EAs were created to handle continuous variables, some of them, such as genetic algorithm[13], evolutionary programming[14], and ant colony optimization[15], have been extended to handle mixed variables. The PSO was also originally proposed for continuous variable problems. Some PSO variants employ simple ways of dealing with mixed variables, for example, simply truncating the real values to integers[16,17]. Here, the standard PSO is extended to handle mixed-variable nonlinear optimization problems more effectively.

Many optimization problems are hard to solve using conventional optimization algorithms or EAs because they involve a number of constrains. In their basic form, EAs are unconstrained optimization techniques, and thus are not able to handle constrained optimization problems directly[18]. To address these constrains, many different approaches have been proposed in the literature. A survey of constraint-handling techniques suitable for EAs can be found in Ref. 19. PSO algorithms have been applied to constrained optimization problems. There are three main approaches incorporated into PSO for solving constrained optimization problems: penalty function method[16,20], constraint-preserving method[17] and feasibility-based methods[21]. Other attempts include applying a multi-objective optimization technique to handle constrains[22]. However, as in the case of constrained optimization problems, relatively fewer studies have employed the PSO algorithm as opposed to other kinds of EAs. The penalty function method requires careful tuning of the penalty parameters, which turns out to be a difficult optimization itself[23]. Constraint-preserving methods consume a lot of time as these methods require an initialization of all particles inside the feasible region. In this paper, a feasibility-based rule is incorporated into the PSO method to better handle constraints.

The paper is organized as follows. The next section describes the mathematical formulation of constrained mixed-variable optimization problems. Section 3 introduces the standard PSO algorithm. A comprehensive learning PSO algorithm is proposed to address multimodal optimization problems with constrains and mixed variables in Section 4. In Section 5, three numerical examples are used to investigate the performance of our proposed PSO algorithm and results for these problems are compared to those obtained from other methods. Experimental results and discussions are given in this section. The paper is summarized and some conclusions are drawn in Section 6.

2. Mathematical Formulation of Constrained Mixed-Variable Optimization Problems

Generally, a constrained mixed-variable optimization problem can be described as follows,

$$\min f(X) \tag{1}$$

Subject to

$$g_i(X) \le 0\ ,\ i = 1,2,...,n_g \tag{2}$$

$$h_j(X) = 0\ ,\ j = 1,2,...,n_h \tag{3}$$

where $f(X)$ is the scalar objective function, $g_i(X)$ and $h_j(X)$ are the inequality and equality constrains, respectively, n_g is the number of inequality constraints, n_h is the number of equality constraints (in both cases, constraints could be linear or non-linear), and X is the vector of solution variables consisting of continuous, binary, integer, and discrete variables.

The variable vector X represents a set of variables which can be written as

$$X = \begin{Bmatrix} x_{1_c},...,x_{n_c}, \\ x_{1_b},...,x_{n_b}, \\ x_{1_i},...,x_{n_i}, \\ x_{1_d},...,x_{n_d} \end{Bmatrix} \tag{4}$$

where $x_{L_c} \le x_{1_c},...,x_{n_c} \le x_{U_c}$; $x_{1_b},...,x_{n_b} \in \{x_{L_b}, x_{U_b}\}$; $x_{L_i} \le x_{1_i},...,x_{n_i} \le x_{U_i}$; $x_{L_c} \le x_{1_c},...,x_{n_c} \le x_{U_c}$; x_{L_c} , x_{L_b} , x_{L_i} , and x_{L_d} are lower bounds of continuous, binary, integer, and discrete variables, respectively; x_{U_c} , x_{U_b} , x_{U_i} , and x_{U_d} are upper bounds of continuous, binary, integer, and discrete variables, respectively; and n_c , n_b , n_i , and n_d are the number of continuous, binary, integer, and discrete variables, respectively. The number of total independent variables is

$$D = n_c + n_b + n_i + n_d \tag{5}$$

3. Particle Swarm Optimizer

PSO is a population-based optimization algorithm. The population of solution candidates is called a "swarm", while each candidate is called a "particle". The current position in the *D*-dimensional search space of a particle represents a potential solution. The particles have memory and each particle keeps track of its previous best position, called *pbest* and the corresponding fitness value. The swarm remembers another value denoted *gbest* , which is the best position discovered so far by the swarm. The trajectory of each particle in the search space is dynamically adjusted by updating its velocity, according to its *pbest* and *gbest* . Therefore, PSO combines the local search technique (from the particle's own experience) and the global search method (from the neighborhood experience) to balance well the exploration and exploitation search aspects and move towards the global optimum.

Here is how a PSO specifically works. Let a swarm of *n* particles be considered. Each particle is fully described by a position and a velocity vector. In every generation of particle population, the velocity V_i^d and the position X_i^d of the *d*-th dimension of the *i*-th particle are updated as follows, using information on its historical velocity as well as its distance from the global and local best solution proposals:

$$V_t^d(t+1) = w(t)\cdot V_i^d(t) + c_1 \cdot rand1_i^d \cdot (pbest_i^d(t) - X_i^d(t)) + c_2 \cdot rand2_i^d \cdot (gbest^d(t) - X_i^d(t)) \tag{6}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \tag{7}$$

where t indicates a pseudo time (generation increment), $X_i = (X_i^1, X_i^2,..., X_i^D)$ is the position of the *i*-th particle; $V_i = (V_i^1, V_i^2,..., V_i^D)$ represents velocity of particle *i* ; $pbest_i = (pbest_i^1, pbest_i^2,..., pbest_i^D)$ is the best previous position yielding the best fitness value for the *i*-th particle; and $gbest = (gbest_i^1, gbest_i^2,..., gbest_i^D)$ is the best position in the whole swarm population. w is called inertia factor. c_1 and c_2 are constants called acceleration coefficients, which reflect the weighting of stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions, respectively. $rand1_i^d$ and $rand2_i^d$ are two random numbers in the range $[0,1]$.

4. A Comprehensive Learning Particle Swarm Optimizer for Constrained Mixed-Variable Optimization Problems

As mentioned in the introduction, the difficulties in using EAs to solve many real-world optimization problems arise because these problems are multimodal, heavily constrained, and involve mixed variables. There have been very few studies that solved problem-specific constraints and mixed variables. In this section, PSO techniques for handling both mixed variables and constraints are proposed. A comprehensive learning strategy is employed to improved the PSO's capability to deal with complex multimodal problems.

4.1. *Mixed-variable handling method*

In our algorithm, different types of variables are dealt with different methods when updating position dimensions of a particle. The algorithm for updating mixed-variable position dimensions is given in Table 1.

Table 1. Pseudo-code for updating mixed-variable position dimensions in the proposed algorithm.

Algorithm 1: Updating mixed-variable position dimensions	
01.	**For** each dimensional variable d in total length of dimension D
02.	**If** (d is continuous)
03.	$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1)$
04.	**End If**
05.	**If** (d is binary)
06.	The sigmoid function of that velocity is calculated as $sigmoid = [1 + e^{-V_i^d(t+1)}]^{-1}$
07.	Generate a random number *rand3* from a uniform distribution between 0 and 1
08.	**If** ($rand3 < sigmoid$)
09.	$X_i^d(t+1) = 1$
10.	**Else**

Table 1. (Continued)

```
11.      X_i^d(t+1) = 0
12.    End If
13.    If ( d is integer)
14.        If ( V_i^d(t+1) > 0 )
15.            X_i^d(t+1) = X_i^d(t+1) + 1
16.        End If
17.        If ( V_i^d(t+1) < 0 )
18.            X_i^d(t+1) = X_i^d(t+1) - 1
19.        End If
20.    End If
21.    If ( d is discrete)
22.        If ( V_i^d(t+1) > 0 )
23.            X_i^d(t+1) = d[j+1]  /*suppose X_i^d(t) = d[j] */
24.        End If
25.        If ( V_i^d(t+1) < 0 )
26.            X_i^d(t+1) = d[j-1]  /*suppose X_i^d(t) = d[j] */
27.        End If
28.    End If
29.  End For
```

For continuous variables, the updating procedure is the standard procedure shown in Eq. (7). For binary variables, the updating procedure found in Ref. 7 is followed. In this procedure, a sigmoid function, $sigmoid = [1+e^{-V_i^d(t+1)}]^{-1}$, is used to generate the probability that a particle might change its dimensional position. For integer variables, the dimensional velocity of the particle will determine if the new dimensional position is forward to plus one or back to minus one position from current location. For discrete variables, the algorithm selects the indices of the set of discrete variables. For this purpose, we first sort the set of discrete variables in ascending/descending order as $d = \{d[1],...,d[j],...,d[n_d]\}$, then the index value j of the discrete variable $d[j]$ is optimized instead of the discrete value of the variable directly.

4.2. *Constraint handling method*

Motivated by Ref. 24, we use a feasibility-based rule to handle constraints. The rule can be described as follows: (1) Any feasible solution is preferred to any infeasible solution; (2) Given two feasible solutions, the one with a better objective function value is preferred; and (3) Given two infeasible solutions, the one having smaller constraint violation value is preferred.

The rule listed above aims at obtaining good feasible solutions. Objective function and constraint violation information pieces are considered separately. In the first and the third cases, the search tends to the feasible region rather than the infeasible region, and in the second case the search tends to the feasible region with good solutions. However, our algorithm differs from Ref. 24 where an additional fitness function was designed to evaluate solutions. In our paper we show that it is unnecessary to design the additional fitness function because the rule can be incorporated into PSO.

In our algorithm, the constraint violation value of an infeasible solution is calculated as follows:

$$violation(X) = \sum_{i=0}^{n_g} \max(g_i(X),0) + \sum_{j=0}^{n_h} \max(|h_j(X)|,0) \quad (8)$$

Suppose that $pbest_i(t)$ represents the best previous position yielding the best fitness value for the *i*-th particle at generation t and $X_i(t+1)$ represents the newly generated position of that particle at generation $t+1$. In terms of a feasibility-based rule, $pbest_i(t)$ will be replaced by $X_i(t+1)$ in any of the following cases: (1) $pbest_i(t)$ is infeasible, but $X_i(t+1)$ is feasible; (2) Both $pbest_i(t)$ and $X_i(t+1)$ are feasible, but $f(pbest_i(t)) > f(X_i(t+1))$; and (3) Both $pbest_i(t)$ and $X_i(t+1)$ are infeasible, but $violation(pbest_i(t)) > violation(X_i(t+1))$.

Similarly, *gbest* can be updated based on the rule at every generation.

4.3. *Comprehensive learning strategy*

We employ a comprehensive learning strategy described in Ref. 25. In this learning strategy, all particles' *pbests* in the population can potentially be used as exemplars to guide a particle's flying direction, while the original PSO[6] only uses particle's own *pbest* and *gbest* as the exemplars. In addition, instead of learning from the same exemplar particle for all dimensions in the original PSO, in the new strategy each dimension of a particle may learn from the corresponding dimension of a different particle's *pbest*. To ensure that a particle learns from good exemplars and to minimize the time wasted on poor directions, this strategy does not allow the particle to learn from exemplars across all generations. Only if the particle ceases improving for a certain number of generations, called the refreshing gap m, is the particle permitted to learn. The strategy is demonstrated in Table 2.

Table 2. Pseudo-code for updating *pbest* after a particle ceases improving for the refreshing gap *m*.

Algorithm 2: Updating *pbest* **after a particle ceases improving for** *m*	
01.	**For** each dimensional variable d in total length of dimension D of particle i
02.	Generate a random number *rand4* from a uniform distribution between 0 and 1
03.	Update Pc_i according to Eq. (9)
04.	**If** ($rand4 < Pc_i$)
05.	$index1 = \lceil rand5 \cdot n \rceil$ /* $\lceil \ \rceil$ is a ceiling operator and n is the population size */
06.	$index2 = \lceil rand6 \cdot n \rceil$
07.	**If** (particle *index1* is better than particle *index*2 based on feasibility-based rule)
08.	$pbest_i^d(t) = pbest_{index1}^d(t)$
09.	**else**
10.	$pbest_i^d(t) = pbest_{index2}^d(t)$
11.	**End If**
12.	**End If**
13.	**End For**

In the learning strategy, each particle learns potentially from all particles' *pbests* in the swarm. During the search process, each dimension of a particle has an equal chance to learn from other particles. For each particle, some dimensions of other particles' *pbests* are randomly chosen according to a probability *Pc* , called learning probability. Each particle has its own Pc_i , which could be different from that of other particles. Here, an empirically developed Pc_i formulation from Ref. 25 is used; see Eq. (9).

$$Pc_i = 0.05 + 0.45 \cdot \frac{\exp(\frac{10 \cdot (i-1)}{n-1} - 1)}{\exp(10) - 1} \qquad (9)$$

where n is the population size of the swarm and i is the particle's id.

For each dimension of a particle i , a random number *rand4* is generated from a uniform distribution between 0 and 1. If $rand4 > Pc_i$, the corresponding dimension will learn from its own *pbest* ; otherwise it will learn from another particle's *pbest* . When a dimension of one particle has to learn from other particles, choice of source is made using a tournament selection procedure is employed as follows: (1) two particles are randomly chosen out of the population, which excludes the particle being updated; (2) these two particles' *pbests* are compared in terms of feasibility-based rule described in Section 4.2; and (3) then the winner's *pbest* is used as the exemplar for that dimension. If all exemplars of a particular particle are its own *pbest* , then one dimension is randomly selected to learn compulsorily from other particles' *pbest* .

4.4. *Implementation of search bounds and maximum velocities*

In many practical problems, there are bounds on the variable ranges. The search range for a problem is $[X_{\min}, X_{\max}]$. In order to prevent particles moving out of the search bounds, some researchers use the equation $X_i^d = \min(X_{\max}^d, \max(X_{\min}^d, X_i^d))$. But our algorithm uses a different method: a particle will choose a random value in the search bounds if the particle moves out of $[X_{\min}, X_{\max}]$. This choice was made because we found choosing a random value lead to better performance.

A particle's velocity on each dimension is clamped to a maximum magnitude $V_{\max}$. If $|V_i^d|$ exceeds a positive constant value $V_{\max}^d$ specified by the user, then the velocity of that dimension is limited to $V_{\max}^d$. In our algorithm, for binary variables, $V_{\max}^d$ is set as 4; and for other variables, $V_{\max}^d$ is set as $0.25 \cdot (X_{\max}^d - X_{\min}^d)$. Eq. (10) is used to restrict V_i^d :

$$V_i^d = \min(V_{\max}^d, \max(-V_{\max}^d, V_i^d)) . \qquad (10)$$

4.5. *Proposed particle swarm optimizer algorithm*

The proposed algorithm is given in Table 3.

Table 3. Pseudo-code for the proposed PSO algorithm.

Algorithm 3: Proposed PSO algorithm for constrained mixed-variable optimization problems	
01.	Initialize a swarm S with n particles
02.	**For** each particle i in S
03.	Initialize random position and velocity
04.	Initialize $pbest_i$ with a copy of the position for each particle
05.	**End For**
06.	**While** the termination conditions are not met
07.	Update *gbest* according to the feasibility-based rule
08.	**For** each particle i in S
09.	Update inertia factor $w(t)$ using a linearly decreasing function in Eq. (11)
10.	**If** (the particle ceases improving for the refreshing gap m)
11.	Update $pbest_i$ using comprehensive learning strategy /* see Table 2 */
12.	**End If**
12.	**For** each dimension d in total length of dimension D
13.	Update the velocity in terms of Eq. (6)
14.	Restrict velocity if it exceeds the range specified

Table 3. (Continued)

15.	Update mixed-variable position dimension using Algorithm 2/* see Table 1 */
16.	**End For**
17.	Update $pbest_i$ according to the feasibility-based rule
18.	**End For**
19.	**End While**

Following Refs. 7 and 25, c_1 and c_2 are both set as 2; m is set as 7; the weighting function $w(t)$ for t-th iteration is determined by Eq. (11):

$$w(t) = w_{\max} - \frac{(w_{\max} - w_{\min}) \cdot t}{t_{\max}}. \quad (11)$$

where $w_{\max}$ is set as 0.9, $w_{\min}$ is set as 0.4, and $t_{\max}$ is the maximum number of iterations. As t approaches the maximum number of iterations, $w(t)$ approaches $w_{\min}$ reducing the speed of the particle.

5. Numerical Tests and Analysis

In this section, three numerical examples are used to investigate the performance of our algorithm. All these problems have mixed variables, as well as linear and nonlinear constraints. They have been widely used in the literature for benchmarking of algorithms and have been investigated by various EAs or traditional techniques. For each problem, 100 independent runs were carried out, in order to statistically assess the performance of our algorithm.

5.1. *Example 1: a Pressure Vessel Design Problem*

The objective of this problem is to minimize the total cost of materials for forming and welding of a pressure vessel. As shown in Figure 1, there are four design variables: shell thickness $T_s = x_1$, thickness of the head $T_h = x_2$, inner radius $R = x_3$, and length of the cylindrical section of the vessel $L = x_4$. Variables $T_s = x_1$ and $T_h = x_2$ are integer multiples of 0.0625 in., in accordance with the available thickness of rolled steel plates, while variables $R = x_3$ and $L = x_4$ are continuous.

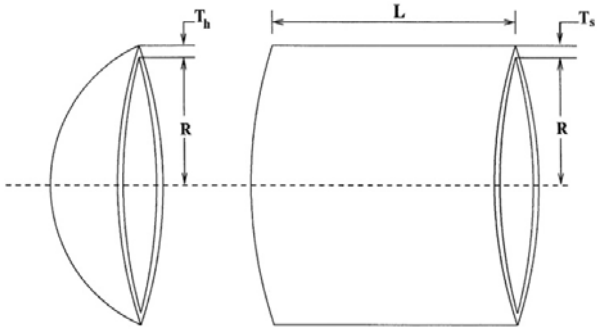

Figure 1. Design of a pressure vessel.

The optimization problem is stated as

$$\min f(X) = 0.6224 \cdot x_1 \cdot x_3 \cdot x_4 + 1.7781 \cdot x_2 \cdot x_3^2 + 3.1661 \cdot x_1^2 \cdot x_4 + 19.84 \cdot x_1^2 \cdot x_3 \quad (12)$$

subject to:

$$g_1(X) = 0.0193 \cdot x_3 - x_1 \le 0 \quad (13)$$

$$g_2(X) = 0.00954 \cdot x_3 - x_2 \le 0 \quad (14)$$

$$g_3(X) = 1296000 - \pi \cdot x_3^2 \cdot x_4 - \frac{4}{3} \cdot \pi \cdot x_3^3 \le 0 \quad (15)$$

$$g_4(X) = x_4 - 240 \le 0 \quad (16)$$

The ranges for the design variables are

$$0.0625 \le x_1, x_2 \le 6.1875, \; 10 \le x_3, x_4 \le 200 \quad (17)$$

This problem is introduced by Sandgren[9] and has been solved using the following approaches: genetic adaptive search[26], an augmented Lagrange multiplier approach[27], a branch and bound technique[9], a GA-based co-evolution model[28], a GA through the use of dominance-based tournament selection[29], a socio-behavioural simulation model[30], and some variants of PSO[16,17,20,21]. Dimopoulos pointed out if the variable x_4 has an upper limit of 200, the fourth constraint is automatically satisfied[16]. So in his study, the upper limit of variable x_4 was extended to 240. For convenience, we designate the problem formulation in Refs. 9, 17, 20, 21, 26–30 as "Ex1-FormuA" and the one in Ref. 16 as "Ex1-FormuB".

In Table 4, the best solutions from our proposed algorithms for the two formulations of the problem as well as the best ones obtained by previous approaches are shown. As shown in Table 4, the proposed algorithm was able to efficiently locate the global optimums both of "Ex1-FormuA" and "Ex1-FormuB". For "Ex1-FormuA", the optimum solution (6059.7143) is also found by the work reported in Refs. 17 and 21.

Table 4. Comparison of the best solution for the pressure vessel design problem.

Method	x_1	x_2	x_3	x_4	$f(x)$
Ex1-FormuA					
Sandgren[9]	1.1250	0.6250	47.7000	117.7010	8129.8000
Kannan[27]	1.1250	0.6250	58.2910	43.6900	7198.0428
Deb[26]	0.9375	0.5000	48.3290	112.6790	6410.3811
Coello[28]	0.8125	0.4375	40.3239	200.0000	6288.7445
Akhtar[30]	0.8125	0.4375	41.9768	182.2845	6171.0000
He and Wang[20]	0.8125	0.4375	42.0913	176.7465	6061.0777
Coello and Montes[29]	0.8125	0.4375	42.0974	176.6540	6059.9463
He et al.[17]	0.8125	0.4375	42.0984	176.6366	6059.7143
He and Wang[21]	0.8125	0.4375	42.0984	176.6366	6059.7143
This paper	**0.8125**	**0.4375**	**42.0984**	**176.6366**	**6059.7143**
Ex1-FormuB					
Dimopoulos[16]	0.7500	0.3750	38.8601	221.36549	5850.3804
This paper	**0.7500**	**0.3750**	**38.8601**	**221.36547**	**5850.3831**

The performance results are depicted in Table 5, where N represents the size of the population and FFE stands for the maximum fitness function evaluations.

Table 5. Statistical results of different methods for the pressure vessel design problem.

Method	N	FFE	Best	Mean	Std.
Ex1-FormuA					
Sandgren[9]	N/A	N/A	8129.8000	N/A	N/A
Kannan[27]	N/A	N/A	7198.0428	N/A	N/A
Deb[26]	N/A	N/A	6410.3811	N/A	N/A
Coello[28]	90	N/A	6288.7445	6293.8432	7.4133
Akhtar[30]	100	20,000	6171.0000	6335.0500	N/A
He and Wang[20]	70	200,000	6061.0777	6147.1332	86.4545
Coello and Montes[29]	200	80,000	6059.9463	6177.2533	130.9297
He et al.[17]	30	30,000	6059.7143	6289.9288	305.7800
He and Wang[21]	250	81,000	6059.7143	6099.9323	86.2022
This paper	**30**	**60,000**	**6059.7143**	**6066.0311**	**12.2718**
Ex1-FormuB					
Dimopoulos[16]	100	100,000	5850.3804	6272.5745	538.3703
This paper	**30**	**60,000**	**5850.3831**	**5923.1568**	**105.1191**

As shown in Table 5, the mean fitness value was 6066.0311 with a standard deviation of 12.2718, which is significantly superior to those of other methods. The mean fitness value was 6119.3708 with a standard variation of 107.7036, even when we reduce the maximum fitness function evaluations (FFEs) to 30,000. For the formulation of the problem modified by Dimopoulos, compared with the work in Ref. 16, our algorithm required considerably lower FFEs (60,000) to improve the searching quality significantly (the mean fitness value was 5923.1568 with a standard variation of 105.1191), as demonstrated in Table 5.

5.2. *Example 2: a welded beam design problem*

The following problem is taken from Ref. 28. As shown in Figure 2, a welded beam is designed for minimum cost of fabrication subject to constraints on shear stress (τ), bending stress (σ), end deflection (δ) in the beam, buckling load on the bar (P_c) and side constraints. The problem involves four design variables: thickness of the weld $h = x_1$, length of the welded joint $l = x_2$, width of the beam $t = x_3$ and thickness of the beam $b = x_4$. Independent variables x_1 and x_2 are integer multiples of 0.0065. The welded beam design problem is stated as follows:

$$\min f(X) = 1.10471 \cdot x_1^2 \cdot x_2 + 0.04811 \cdot x_3 \cdot x_4 \cdot (14.0 + x_2) \quad (18)$$

subject to:

$$g_1(X) = \tau(X) - \tau_{\max} \le 0 \quad (19)$$

$$g_2(X) = \sigma(X) - \sigma_{\max} \le 0 \quad (20)$$

$$g_3(X) = x_1 - x_4 \le 0 \quad (21)$$

$$g_4(X) = 0.10471 \cdot x_1^2 + 0.04811 \cdot x_3 \cdot x_4 \cdot (14.0 + x_2) - 5.0 \le 0 \quad (22)$$

$$g_5(X) = 0.125 - x_1 \le 0 \quad (23)$$

$$g_6(X) = \delta(X) - \delta_{\max} \le 0 \quad (24)$$

$$g_7(X) = P - P_c(X) \le 0 \quad (25)$$

where

$$\tau(X) = \sqrt{(\tau')^2 + 2 \cdot \tau' \cdot \tau'' \cdot \frac{x_2}{2 \cdot R} + (\tau'')^2} \quad (26)$$

$$\tau' = \frac{P}{\sqrt{2} \cdot x_1 \cdot x_2}, \ \tau'' = \frac{M \cdot R}{J}, \ M = P \cdot (L + \frac{x_2}{2}) \quad (27)$$

$$J = 2 \cdot \left\{ \sqrt{2} \cdot x_1 \cdot x_2 \cdot [\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2] \right\} \quad (28)$$

$$\sigma(X) = \frac{6 \cdot P \cdot L}{x_4 \cdot x_3^2}, \ \delta(X) = \frac{4 \cdot P \cdot L^3}{E \cdot x_3^3 \cdot x_4} \quad (29)$$

$$P_c(X) = \frac{4.103 \cdot E \cdot \sqrt{\frac{x_3^2 \cdot x_4^6}{36}}}{L^2} \cdot (1 - \frac{x_3}{2 \cdot L} \cdot \sqrt{\frac{E}{4 \cdot G}}) \quad (30)$$

with $P = 6000$, $L = 14$, $\delta_{\max} = 0.25$, $E = 30 \times 10^6$, $G = 12 \times 10^6$, $\tau_{\max} = 13{,}600$, $\sigma_{\max} = 30{,}000$.

The ranges for the design variables are

$$0.1 \le x_1, x_4 \le 2\,, 0.1 \le x_2, x_3 \le 10 \qquad (31)$$

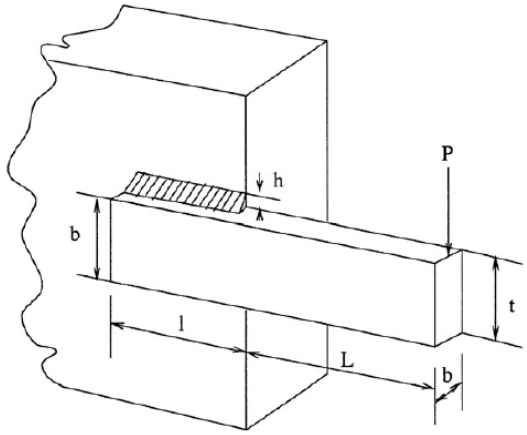

Figure 2. The welded beam design problem.

The problem was previously investigated by the following approaches: a real parameter GA[24], An optimization algorithm based on the simulation of social behavior[31], a GA-based co-evolutionary model[28], a GA through the use of dominance-based tournament selection[29], a domain knowledge-based cultural algorithm[32], and some variants of PSO[16,17,20,21]. However, the formulations of the problem vary slightly across these methods. In References 17, 24, 31, Eq. (28) and (30) are replaced by Eqs. (32) and (33), respectively.

$$J = 2 \cdot \left\{ \frac{x_1 \cdot x_2}{\sqrt{2}} \cdot [\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2] \right\} \qquad (32)$$

$$P_c(X) = \frac{4.103 \cdot \sqrt{\frac{E \cdot G \cdot x_3^2 \cdot x_4^6}{36}}}{L^2} \cdot (1 - \frac{x_3}{2 \cdot L} \cdot \sqrt{\frac{E}{4 \cdot G}}) \qquad (33)$$

Also, References 17, 20, 21, 24, 28, 29, 31, and 32 found their optimums by treating independent variables x_1 and x_2 as real numbers. Only the solution in Ref. 16 treated these variables as integer multiples of 0.0065. For convenience, we designate the formulation of the problem in Refs. 17, 24, and 31 as "Ex2-FormuA", the formulation in Refs. 20, 21, 28, 29, and 32 as "Ex2-FormuB" and the that in Ref. 16 as "Ex2-FormuC". The best solutions obtained using the above mentioned approaches are depicted in Table 6, where the best solutions from different formulations of the problem obtained using the PSO algorithm presented in this paper are also reported. The performance results of the different algorithms are shown in Table 7.

Table 6. Comparison of the best solution for the welded beam design problem.

Method	x_1	x_2	x_3	x_4	$f(x)$
Ex2-FormuA					
Ray and Liew[31]	0.244438	6.237967	8.288576	0.244566	2.385435
Deb[24]	N/A	N/A	N/A	N/A	2.381190
He et al.[17]	0.244369	6.217520	8.291471	0.244369	2.380957
This paper	**0.244369**	**6.217520**	**8.291471**	**0.244369**	**2.380957**
Ex2-FormuB					
Coello[28]	0.208800	3.420500	8.997500	0.210000	1.748309
Coello and Montes[29]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang[20]	0.202369	3.544214	9.048210	0.205723	1.728024
Coello and Becerra[32]	0.205700	3.470500	9.036600	0.205700	1.724852
He and Wang[21]	0.205730	3.470489	9.036624	0.205730	1.724852
This paper	**0.205730**	**3.470489**	**9.036624**	**0.205730**	**1.724852**
Ex2-FormuC					
Dimopoulos[16]	0.2015	3.5620	9.041398	0.205706	1.731186
This paper	**0.2015**	**3.5620**	**9.041398**	**0.205706**	**1.731186**

Table 7. Statistical results of different methods for the welded beam design problem.

Method	N	FFE	Best	Mean	Std.
Ex2-FormuA					
Ray and Liew[31]	40	33,095	2.385435	3.255137	0.959078
Deb[24]	50	40,080	2.381190	2.392890	N/A
He et al.[17]	30	30,000	2.380957	2.381900	0.005200
This paper	**30**	**30,000**	**2.380957**	**2.384111**	**0.004256**
Ex2-FormuB					
Coello[28]	90	2,100	1.748309	1.771973	0.011220
Coello and Montes[29]	200	80,000	1.728226	1.792654	0.074713
He and Wang[20]	70	200,000	1.728024	1.748831	0.012926
Coello and Becerra[32]	20	50,000	1.724852	1.971809	0.443131
He and Wang[21]	250	81,000	1.724852	1.749040	0.040049
This paper	**30**	**60,000**	**1.724852**	**1.728180**	**0.005324**
Ex2-FormuC					
Dimopoulos[16]	100	100,000	1.731186	1.762200	0.065900
This paper	**100**	**100,000**	**1.731186**	**1.737459**	**0.017577**

It is shown in Table 6 that the proposed PSO algorithm can obtain the best known solution. From Table 7, it can be found for the formulation "Ex2-FormuA", that our algorithm performs much better than those reported in Refs. 24 and 31, but a bit worse than

the one reported in Ref. 17. However, the average CPU time required for execution of the algorithm in Ref. 17 for a single run was 10.2 seconds (on a Pentium 4, 2-GHz machine), which is even more than the execution time of our algorithm for 100 runs – 9 seconds only (on a Pentium Dual, 2.16-GHz notebook). For the formulations "Ex2-FormuB" and "Ex2-FormuC", the average searching quality of our proposed algorithm is far superior to those of other methods. The mean values and the standard deviations of results by the proposed algorithm for these two formulations are also very small.

5.3. *Example 3: the second variation of welded beam design problem*

This problem is taken from Deb and Goyal[33] and is a variation of that in Example 2. Example 2 is extended to include two types of welded joint configurations (as depicted in Figure 3) and four possible beam materials (as described in Table 8). There are six independent variables in the design problem: thickness of the weld $h = x_1$, length of the welded joint $l = x_2$, width of the beam $t = x_3$, thickness of the beam $b = x_4$, material $= x_5$ and joint type $= x_6$. Variables x_1, x_3 and x_4 are now integer multiples of 0.0625 in., variable x_2 is continuous, variable x_5 is an integer ranging from 1 to 4 representing the material (1 stands for "Steel", 2 represents "Cast Iron", 3 is "Aluminium", and 4 means "Brass"), and variable x_6 is binary representing the joint type (0 stands for two sided welded joint and 1 represents four sided welded joint).

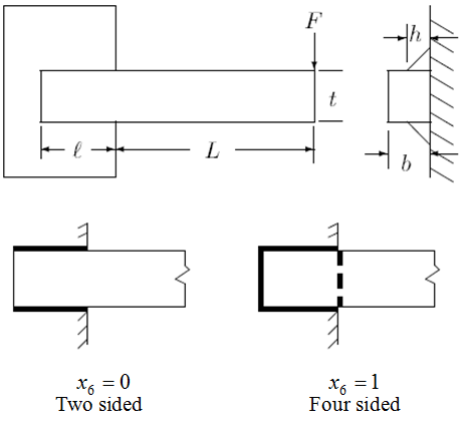

Figure 3. Welded joint configurations of the 2nd variation of welded beam design problem.

Table 8. Material properties for the 2nd variation of welded beam design problem.

Material x_5	S (10^3 psi)	E (10^6 psi)	G (10^6 psi)	c_1	c_2
1 = Steel	30	30	12	0.1047	0.0481
2 = Cast iron	8	14	6	0.0489	0.0224
3 = Aluminium	5	10	4	0.5235	0.2405
4 = Brass	8	16	6	0.5584	0.2566

This variation of the welded beam design problem is stated:

$$\min f(X) = (1 + c_1) \cdot x_1^2 \cdot x_2 + c_2 \cdot x_3 \cdot x_4 \cdot (14.0 + x_2). \quad (34)$$

subject to the constrains of Eqs. (19)–(25), with the relations demonstrated in Eqs. (26), (27), (29), and (30) still valid. However, Eq. (28) is replaced by

$$J = \begin{cases} 2 \cdot \sqrt{2} \cdot x_1 \cdot x_2 \cdot \left[\dfrac{x_2^2}{12} + \left(\dfrac{x_1 + x_3}{2} \right)^2 \right], & x_6 = 0 \\ 2 \cdot \sqrt{2} \cdot x_1 \cdot \left[\dfrac{(x_1 + x_2 + x_3)^3}{12} \right], & x_6 = 1 \end{cases} \quad (35)$$

The values of parameters to material are given in Table 8. Also, the value of $\tau_{\max}$ is changed to:

$$\tau_{\max} = 0.577 \cdot S \quad (36)$$

The bounds of variables x_1, x_2, x_3, and x_4 remain the same as in Eq. (31).

The only other two solutions to this problem (known to the authors) are by Deb and Goyal[33] and Dimopoulos[16]. The best solutions obtained by these two solutions as well as from our proposed algorithm in this paper are listed in Table 9, and their statistical simulation results are shown in Table 10.

Table 9. Comparison of the best solution for the 2nd variation of welded beam design problem.

Method	x_1	x_2	x_3	x_4	x_5	x_6	$f(x)$
Deb and Goyal[33]	0.1875	1.6849	8.25	0.25	1	1	1.9422
Dimopoulos[16]	0.2500	2.2219	8.25	0.25	1	0	1.7631
This paper	**0.2500**	**1.1412**	**8.25**	**0.25**	**1**	**1**	**1.5809**

Table 10. Statistical results of different methods for the 2nd variation of welded beam design.

Method	*N*	FFE	Best	Mean	Std.
Deb and Goyal[33]	50	5,000	1.9422	N/A	N/A
Dimopoulos[16]	100	100,000	1.7631	1.7694	0.0192
This paper	**30**	**60,000**	**1.5809**	**1.7405**	**0.2109**

From Table 9, it can be seen that our PSO algorithm yielded a new optimum which is far below the ones obtained by the only other two solutions. It can be found in Table 10 that the average searching quality of the proposed algorithm is better than the one reported by Dimopoulos. It is worth mentioning that this improved performance is obtained under a smaller (60,000) FFEs.

6. Conclusions

This paper extends the standard PSO to address multimodal and constrained mixed-variable optimization problems. A comprehensive learning strategy is employed to improve PSO's performance on complex multimodal problems. In this strategy, other particles' previous best positions are exemplars to be learned from by any particle and different dimensions of a particle can potentially learn from different exemplars. Different types of variables are handled using different approaches. Constraint handling is based on a feasibility-based rule, which provides an effective alternative to overcome the weakness of penalty function methods and constraint-preserving methods.

The advantages of the proposed algorithm are illustrated by solving four mechanical design optimization problems. The numerical results and comparisons with solutions from other methods show that our proposed algorithm improves performances in terms of search quality, efficiency, and robustness. The numerical results obtained by the proposed algorithm are better than or equal to those obtained from other existing methods.

An issue for future work is to improve the feasibility-based rule because it is not completely reasonable that feasible solutions are always considered better than infeasible ones in the rule. This could lead to overpressure from selecting feasible solutions and lead to premature convergence. For example, the constraint boundary handling method has not been taken into account, while many real-world constrained optimization problems have optimum solutions in or near the boundary of the feasible region.

Acknowledgements

This work was supported in part by the Ningaloo Collaboration Cluster, CSIRO Wealth from Oceans Flagship Program. The authors wish to acknowledge the support from School of Agricultural and Resource Economics, University of Western Australia.

References

1. C. A. C. Coello, G. B. Lamont and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems* (Springer-Verlag, New York, 2007).
2. Y. B. Liu and J. Huang, A novel fast multi-objective evolutionary algorithm for QoS multicast routing in MANET. *Int. J. Comput. Int. Sys.* **2**(3) (2009) 288–297.
3. E. F. Golen, B. Yuan and N. Shenoy, An evolutionary approach to underwater sensor deployment. *Int. J. Comput. Int. Sys.* **2**(3) (2009) 184–201.
4. C. Kahraman, O. Engin, I. Kaya and M. K. Yilmaz, An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *Int. J. Comput. Int. Sys.* **1**(2) (2008) 134–147.
5. L. Gao, Y. S. Ding and H. Ying, An adaptive social network-inspired approach to resource discovery for the complex grid systems. *Int. J. Gen. Syst.* **35**(3) (2006) 347–360.
6. J. Kennedy and R. C. Eberhart, Particle swarm optimization, in *IEEE International Conference on Neural Networks*, the University of Western Australia, Perth, Australia, 1995, pp. 1942–1948.
7. J. Kennedy and R. C. Eberhart, *Swarm Intelligence* (Morgan Kaufmann, San Mateo, CA, 2001).
8. L. Hu, X. Che and X. C. Cheng, Bandwidth prediction based on nu-support vector regression and parallel hybrid particle swarm optimization. *Int. J. Comput. Int. Sys.* **3**(1) (2010) 70–83.
9. E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization. *J. Mech. Design* **112**(2) (1990) 223–229.
10. P. Hajela and C. J. Shih, Multiobjective optimum design in mixed integer and discrete design variable problems. *AIAA J.* **28**(4) (1990) 670–675.
11. J. F. Fu, R. G. Fenton and W. L. Cleghorn, A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Eng. Optimiz.* **17**(4) (1991) 263–280.
12. H. T. Loh and P. Y. Papalambros, A sequential linearization approach for solving mixed-discrete

nonlinear design optimization problems. *J. Mech. Design* **113**(3) (1991) 325–334.
13. J. G. Ndiritu and T. M. Daniell, An improved genetic algorithm for continuous and mixed discrete-continuous optimization. *Eng. Optimiz.* **31**(5) (1999) 589–614.
14. Y. J. Cao, L. Jiang and Q. H. Wu, An evolutionary programming approach to mixed-variable optimization problems. *Appl. Math. Model.* **24**(12) (2000) 931–942.
15. K. Socha, ACO for continuous and mixed-variable optimization. *Lect. Notes in Comput. Sci.* **3172** (2004) 53–61.
16. G. G. Dimopoulos, Mixed-variable engineering optimization based on evolutionary and social metaphors. *Comput. Method Appl. M.* **196**(4-6) (2007) 803–817.
17. S. He, E. Prempain and Q. H. Wu, An improved particle swarm optimizer for mechanical design optimization problems. *Eng. Optimiz.* **36**(5) (2004) 585–605.
18. M. J. Tahk and B. C. Sun, Coevolutionary augmented Lagrangian methods for constrained optimization. *IEEE T. Evolut. Comput.* **4**(2) (2000) 114–124.
19. C. A. C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput. Method Appl. M.* **191**(11-12) (2002) 1245–1287.
20. Q. He and L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intel.* **20**(1) (2007) 89–99.
21. Q. He and L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Appl. Math. Comput.* **186**(2) (2007) 1407–1422.
22. T. Ray and K. M. Liew, A swarm metaphor for multiobjective design optimization. *Eng. Optimiz.* **34**(2) (2002) 141–153.
23. P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization. *IEEE T. Evolut. Comput.* **4**(3) (2000) 284–294.
24. K. Deb, An efficient constraint handling method for genetic algorithms. *Comput. Method Appl. M.* **186**(2-4) (2000) 311–338.
25. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE T. Evolut. Comput.* **10**(3) (2006) 281–295.
26. K. Deb, *GeneAS: A robust optimal design technique for mechanical component design* (Springer-Verlag, Berlin, 1997).
27. B. K. Kannan and S. N. Kramer, An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *J. Mech. Design* **116**(2) (1994) 405–411.
28. C. A. C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems. *Comput. Ind.* **41**(2) (2000) 113–127.
29. C. A. C. Coello and E. M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inform.* **16**(3) (2002) 193–203.
30. S. Akhtar, K. Tai and T. Ray, A socio-behavioural simulation model for engineering design optimization. *Eng. Optimiz.* **34**(4) (2002) 341–354.
31. T. Ray and K. M. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE T. Evolut. Comput.* **7**(4) (2003) 386–396.
32. C. A. C. Coello and R. L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm. *Eng. Optimiz.* **36**(2) (2004) 219–236.
33. K. Deb and M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design. *Comput. Sci. and Inform.* **26**(4) (1996) 30–45.