Applied Mathematics and Computation 218 (2012) 11042–11061

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A Modified Binary Particle Swarm Optimization for Knapsack Problems

Jagdish Chand Bansal [a,*], Kusum Deep [b]

[a] ABV-Indian Institute of Information Technology and Management Gwalior, Gwalior 474010, India
[b] Department of Mathematics Indian Institute of Technology Roorkee, Roorkee 247667, India

ARTICLE INFO

Keywords:
Binary Particle Swarm Optimization
Knapsack Problems
Sigmoid function

ABSTRACT

The Knapsack Problems (KPs) are classical NP-hard problems in Operations Research having a number of engineering applications. Several traditional as well as population based search algorithms are available in literature for the solution of these problems. In this paper, a new Modified Binary Particle Swarm Optimization (MBPSO) algorithm is proposed for solving KPs, particularly 0–1 Knapsack Problem (KP) and Multidimensional Knapsack Problem (MKP). Compared to the basic Binary Particle Swarm Optimization (BPSO), this improved algorithm introduces a new probability function which maintains the diversity in the swarm and makes it more explorative, effective and efficient in solving KPs. MBPSO is tested through computational experiments over benchmark problems and the results are compared with those of BPSO and a relatively recent modified version of BPSO namely Genotype–Phenotype Modified Binary Particle Swarm Optimization (GPMBPSO). To validate our idea and demonstrate the efficiency of the proposed algorithm for KPs, experiments are carried out with various data instances of KP and MKP and the results are compared with those of BPSO and GPMBPSO.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

Knapsack Problems (KPs) have been extensively studied since the pioneering work of Dantzig [1]. KPs have lot of immediate applications in industry, financial management. KPs frequently occur by relaxation of various integer programming problems.

The family of Knapsack Problems requires a subset of some given items to be chosen such that the corresponding profit sum is maximized without exceeding the capacity of the Knapsack(s). Different types of Knapsack Problems occur, depending upon the distribution of the items and knapsacks:

(a) 0–1 Knapsack Problem: each item may be chosen at most once.
(b) Bounded Knapsack Problem: if each item can be chosen multiple times.
(c) Multiple Choice Knapsack Problem: if the items are subdivided into some finite number of classes and exactly one item must be taken from each class.
(d) Multiple or Multidimensional Knapsack Problem: if we have n items and m knapsacks with capacities not necessarily same and knapsack are to be filled simultaneously.

* Corresponding author.
E-mail addresses: jcbansal@gmail.com, jcbansal@iiitm.ac.in (J.C. Bansal), kusumfma@iitr.ernet.in (K. Deep).

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.05.001

All the Knapsack Problems belongs to the family of *NP-hard*[1] problems. Despite of being NP-hard problems, many large instances of KPs can be solved in seconds. This is due to several years of research which have proposed many solution methodologies including exact as well as heuristic algorithms. 0–1 Knapsack and Multidimensional Knapsack Problems are solved using MBPSO in this paper. Therefore, only 0–1 Knapsack Problem (KP) and Multidimensional Knapsack Problem (MKP) are described here in detail.

1.1. 0–1 Knapsack Problem

0–1 Knapsack Problem (KP) is a typical NP-hard problem in operations research. The classical 0–1 Knapsack problem is defined as follows:

We are given a set of n items, each item i having an integer profit p_i and an integer weight w_i. The problem is to choose a subset of the items such that their total profit is maximized, while the total weight does not exceed a given capacity C. The problem may be formulated so as to maximize the total profit $f(x)$ as follows:

$$\left.\begin{array}{ll} \textit{Maximize} & f(x) = \sum_{i=1}^{n} p_i x_i, \\ \textit{Subject to} & \\ & \sum_{i=1}^{n} w_i x_i \leqslant C, \\ & x_i \in \{0,1\}, \quad i = 1,2,\ldots,n, \end{array}\right\} \tag{1}$$

where the binary decision variables x_i are used to indicate whether item i is included in the knapsack or not. Without loss of generality it may be assumed that all profits and weights are positive, that all weights are smaller than the capacity C so each item fits into the knapsack, and that the total weight of the items exceeds C to ensure a nontrivial problem.

KP has high theoretical and practical value; and there are very important applications in financial and industrial areas, such as investment decision, budget control, project choice, resources assignment, goods loading and so on. Many exact as well as heuristic techniques are available to solve the 0–1 Knapsack problems. Heuristic algorithms include simulated annealing [2], genetic algorithm [3–5], ant colony optimization [6,7], differential evolution [8], immune algorithm [9] and particle swarm optimization [10–14].

1.2. Multidimensional Knapsack Problem

The NP-hard 0–1 Multidimensional Knapsack Problem is a generalization of the 0–1 simple knapsack problem. It consists of selecting a subset of given objects (or items) in such a way that the total profit of the selected objects is maximized while a set of knapsack constraints are satisfied. More formally, the problem can be stated as follows:

$$\left.\begin{array}{ll} \textit{Maximize} & f(x) = \sum_{i=1}^{n} p_i x_i, \\ \textit{Subject to} & \\ & \sum_{i=1}^{n} w_{i,j} x_i \leqslant C_j \quad \forall j = 1,2,\ldots,m, \\ & w_{i,j} \geqslant 0, \quad C_j \geqslant 0, \\ & x_i \in \{0,1\}, \quad i = 1,2,\ldots,n, \end{array}\right\} \tag{2}$$

where n is the number of objects, m is the number of knapsack constraints with capacities C_j, p_i represents the benefit of the object i in the knapsack, x_i is a binary variable that indicates $x_i = 1$, if the object i has been stored in the knapsack and $x_i = 0$, if it remains out, and $w_{i,j}$ represents the entries of the knapsack's constraints matrix.

A comprehensive overview of practical and theoretical results for the MKP can be found in [15]. Many practical engineering design problems can be formulated as the MKP, for example, the capital budgeting problem, allocating processors, databases in a distributed computer system, cargo loading and development of pollution prevention and control strategies. MKP has been solved by many exact as well as heuristic methods.

Heuristic methods include Tabu Search [16–22], Genetic Algorithm (GA) [23–25,4,26–28], Ant Colony Optimization (ACO) [29,20,30,31], Differential Evolution (DE) [32], Simulated Annealing (SA) [33], Immune Inspired Algorithm [34], and Particle Swarm Optimization (PSO) [35–37], fast and effective heuristics [38], permutation based evolutionary algorithm [39].

In this paper, a new Modified Binary Particle Swarm Optimization method (MBPSO) is proposed. The method is based on replacing the sigmoid function by a linear probability function. Further, the efficiency of MBPSO is established by applying it to KP and MKP.

[1] A mathematical problem for which, even in theory, no shortcut or smart algorithm is possible that would lead to a simple or rapid solution. Instead, the only way to find an optimal solution is a computationally-intensive, exhaustive analysis in which all possible outcomes are tested.

Rest of the paper is organized as follows: In Section 2, BPSO is described. The details of proposed MBPSO are given in Section 3. The Section 4 presents experimental results of MBPSO and its comparison with original BPSO as well as Genotype–Phenotype MBPSO on test problems. In Section 5, numerical results of 0–1 Knapsack and Multidimensional Knapsack Problems, solved by BPSO and MBPSO are compared. Finally the conclusions, based on the results, are drawn in Section 6.

2. Binary Particle Swarm Optimization

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behaviour by Kennedy and Eberhart [40,41], solves problems in many fields, especially engineering and computer science. Only within a few years of its introduction PSO has gained wide popularity as a powerful global optimization tool and is competing with well-established population based search algorithms. The inspiration behind the development of PSO is the mechanism by which the birds in a flock and the fishes in a school cooperate while searching for food. In PSO, a group of active, dynamic and interactive members called swarm produces a very intelligent search behaviour using collaborative trial and error. Each member of the swarm called particle, represents a potential solution of the problem under consideration. Each particle in the swarm relies on its own experience as well as the experience of its best neighbour (in terms of fitness). Each particle has an associated fitness value. These particles move through search space with a specified velocity in search of optimal solution. Each particle maintains a memory which helps it in keeping the track of the best position it has achieved so far. This is called the particle's personal best position (pbest) and the best position the swarm has achieved so far is called global best position (gbest). The movement of the particles is influenced by two factors using information from iteration-to-iteration as well as particle-to-particle. As a result of iteration-to-iteration information, the particle stores in its memory the best solution visited so far, called *pbest*, and experiences an attraction towards this solution as it traverses through the solution search space. As a result of the particle-to-particle information, the particle stores in its memory the best solution visited by any particle, and experiences an attraction towards this solution, called *gbest*, as well. The first and second factors are called cognitive and social components, respectively. After each iteration, the *pbest* and *gbest* are updated for each particle if a better or more dominating solution (in terms of fitness) is found. This process continues, iteratively, until either the desired result is converged upon, or it is determined that an acceptable solution cannot be found within computational limits. Initially PSO was designed for continuous optimization problems, but later a wide variety of challenging engineering and scientific applications came into being. A survey of these recent advances can be found in [42–44]. In [45] the binary version of PSO (BPSO) was introduced. It is outlined as follows:

Suppose the search space is $S = \{0,1\}^D$, and the objective function f is to be maximized, i.e., max $f(x)$, then the ith particle of the swarm can be represented by a D – dimensional vector, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$, $x_{id} \in \{0,1\}, d = 1,2,\ldots,D$. The velocity (position change) of this particle can be represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^T$, $v_{id} \in [-V_{\max}, V_{\max}]$,

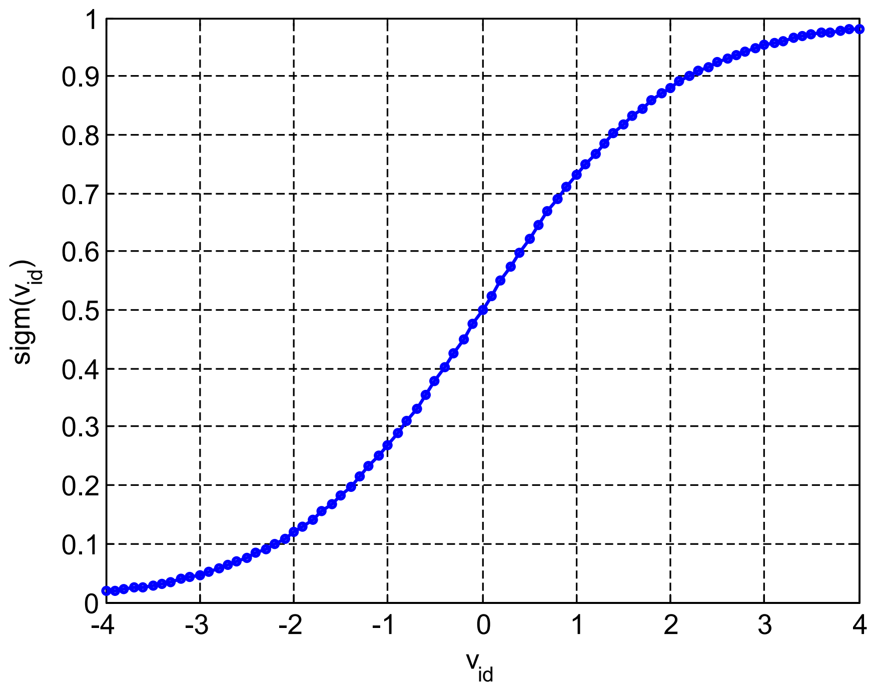

Fig. 1. Sigmoid function with $\lambda = 1$.

$d = 1,2,\ldots,D$ and $V_{\max}$ is the maximum velocity. Previously visited best position of the ith particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^T$, $p_{id} \in \{0,1\}$, $d = 1,2,\ldots,D$. Define g as the index of best performer in the swarm and p_{gd} as the swarm best, then the swarm is manipulated according to the following two equations:

Velocity Update Equation:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}). \quad (3)$$

Position Update Equation:

$$x_{id} = \begin{cases} 1 & \text{if } U(0,1) < sigm(v_{id}), \\ 0 & \text{otherwise}, \end{cases} \quad (4)$$

where $d = 1,2\ldots D$; $i = 1,2\ldots N$, and N is the size of the swarm; c_1 and c_2 are constants, called cognitive and social scaling parameters respectively; r_1, r_2 are random numbers, uniformly distributed in $[0,1]$. $U(a,b)$ is a symbol for uniformly distributed random number between 0 and 1. $Sigm(v_{id})$ is a sigmoid limiting transformation having an "S" shape as shown in Fig. 1 and defined as $sigm(v_{id}) = \frac{1}{1+\exp(-\lambda v_{id})}$; where λ controls the steepness of the sigmoid function. If steepness $\lambda = 1$ then Eqs. (3) and (4) constitute the BPSO algorithm [45].

The pseudo code of BPSO for maximization of $f(X)$ is shown below:

```
Create and initialize a D-dimensional swarm, N
  Loop
    For i = 1 to N
      if f(X_i) > f(P_i) then do
        For d = 1 to D
          p_id = x_id
        Next d
      End do
      g = i
      For j = 1 to N
          if f(P_j) > f(P_g) then g = j
      Next j
      For d = 1 to D
          Apply Eq. (3)
            v_id ∈ [−V_max, V_max]
          Apply Eq. (4)
      Next d
    Next i
  Until stopping criterion is true
Return (P_g, f(P_g))
```

A drawback observed with BPSO is the non-monotonic shape of the changing probability function (sigmoid function) of a bit (from 0 to 1 or vice versa). The sigmoid function has a concave shape that for some bigger v_{id} values the changing probability will decrease (i.e., for bigger values of velocities BPSO produces low exploration). Thus for more diversified search in BPSO some improvements are possible. This motivates authors to introduce a new probability function in BPSO with property of large exploration capability even in the case of large velocity values. This paper proposes a new Modified Binary Particle Swarm Optimization method (MBPSO) and its application to 0–1 KP and MKP.

3. The proposed Modified Binary Particle Swarm Optimization (MBPSO)

3.1. Motivation

In BPSO velocities v_{id} are restricted to be in the range $[0,1]$ to be interpreted as a probability for selection of 1. Sigmoid function is applied to normalize the velocity of a particle such that $v_{id} \in [0,1]$. For BPSO, the velocities will increase in their absolute value, until the V_{max} bounds are reached, at which point BPSO has little exploration. It can happen very quickly that velocities approach V_{max} and when it happens, there is a very small probability of 0.018 (for $V_{max} = 4$) that a bit will change. Now we consider the cases when the steepness λ of the sigmoid function in BPSO, is varied.

3.1.1. Case I: When steepness λ is close to 0:

Then the sigmoid function tends to a straight line parallel to the horizontal axis as λ tends to zero (Refer Fig. 2). This provides a probability close to 0.5 and BPSO starts to behave like a random search algorithm. For example, for $\lambda = 0.1$, probability lies between 0.4 and 0.6 approximately and for $\lambda = 0.2$, probability lies between 0.3 and 0.7 approximately. In this way, the

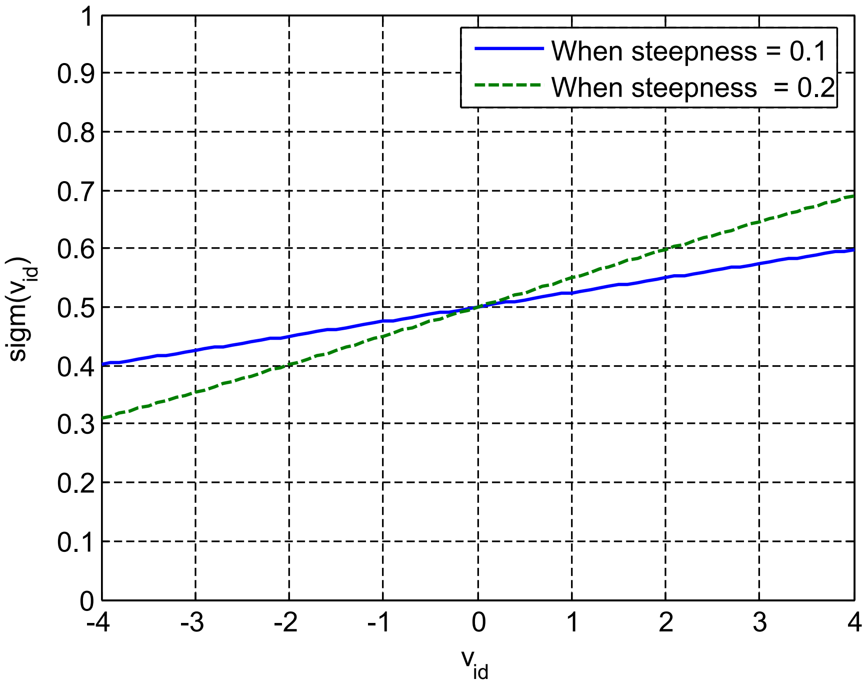

Fig. 2. Sigmoid function when steepness λ is close to 0.

algorithm will converge very slowly and will be very prone to provide local optima only. Thus, we can conclude that for steepness close to zero the method will converge slowly.

3.1.2. Case II: when steepness λ is increased:

The curve no longer remains a straight line, but instead starts taking the shape of English alphabet S. This leads to the drawback of sigmoid function, i.e., provides low diversity and low exploration. Refer Fig. 3, wherein the sigmoid curves with steepness 0.7, 1, and 2 are drawn. Since steepness is problem dependent, hence an extensive study needs to be carried out in order to fine tune the steepness for a problem under consideration. In other words, the normalization of velocity is problem dependent and therefore, instead of using sigmoid function for this purpose one can use other approaches for better exploration as suggested in [46]. In this paper, a linear normalization function is proposed to replace the sigmoid function of BPSO to make the search process more explorative and efficient.

3.2. Modified Binary Particle Swarm Optimization

In BPSO, there is no role of particle's previous position after updating velocity while in MBPSO, position update equation is an explicit function of previous velocity and previous position. Swarm is manipulated according to the following equations:

Velocity Update Equation:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}). \tag{5}$$

Position Update Equation:

The basic idea of proposed position update equation for MBPSO is taken from position update equation of PSO for continuous optimization. The position update equation of continuous PSO is:

$$x_{id} = x_{id} + v_{id}.$$

If velocity bounds are $-V_{max}$ and V_{max}, then since x_{id} can take values 0 or 1, the term $x_{id} + v_{id}$ is bounded between $(0 - V_{\max} = -V_{\max})$ and $(1 + V_{\max})$. Now the proposed position update equation for MBPSO is

$$x_{id} = \begin{cases} 1 & \textit{if } U(-V_{\max}, 1 + V_{\max}) < (x_{id} + v_{id}) \\ 0 & \textit{otherwise} \end{cases} \tag{6}$$

Since $U(0,1) = \frac{U(-V_{\max},1+V_{\max})+V_{\max}}{(1+2V_{\max})}$, so we can rewrite (6) as

$$x_{id} = \begin{cases} 1 & \text{if } U(0,1) < \frac{x_{id}+v_{id}+V_{\max}}{(1+2V_{\max})}, \\ 0 & \text{otherwise}. \end{cases}$$

Let $p(x_{id}, v_{id}) = \frac{x_{id}+v_{id}+V_{\max}}{(1+2V_{\max})}$, then the position update equation for MBPSO becomes

$$x_{id} = \begin{cases} 1 & \text{if } U(0,1) < p(x_{id}, v_{id}), \\ 0 & \text{otherwise}. \end{cases} \quad (7)$$

Symbols have their usual meaning as in Section 2. In MBPSO, the term $p(x_{id}, v_{id})$ gives a probability of selection of 1. This is similar to the BPSO where the term $\frac{1}{1+\exp(-\lambda v_{id})}$ gives this probability but the difference is that MBPSO allows for better exploration. To illustrate this point, one has to look at these two probability functions, which are illustrated in Fig. 4. In this Figure the straight line with small dots represents the case when x_{id} at previous time step was 1 and the straight line with dashes, when $x_{id} = 0$.

From Fig. 4, it is clear that MBPSO provides better exploration. For example, if $v_{id} = 2$, then according to BPSO (when steepness $\lambda = 1$) there is a 0.8808 probability that x_{id} will be bit 1, and a 0.1192 probability for it to be bit 0. Now according to MBPSO, if Vmax = 4 and assuming that $x_{id} = 1$, the probability of producing bit 1 is 0.7778 and for bit 0 it is 0.2222. Now if $x_{id} = 0$ then the probability of producing bit 1 is 0.6667 and for bit 0 is 0.3333. These smaller probabilities for MBPSO allow more exploration.

The pseudo code of MBPSO for maximization of $f(X)$ is shown below:

```
Create and initialize a D-dimensional swarm, N
  Loop
    For i = 1 to N
      if f(X_i) > f(P_i) then do
        For d = 1 to D
          p_id = x_id
        Next d
      End do
      g = i
      For j = N
        if f(P_j) > f(P_g) then g = j
      Next j
      For d = 1 to D
        Apply Eq. (5)
        v_id ∈ [−V_max, V_max]
        Apply Eq. (7)
      Next d
    Next i
  Until stopping criterion is true
Return (P_g, f(P_g))
```

The next section presents experimental results of MBPSO and its comparison with original BPSO as well as Genotype–Phenotype MBPSO.

4. Results and discussions

From (7), it is evident that the proposed MBPSO is highly dependent on the Vmax, the constant maximum velocity. Therefore, in the next subsection fine tuning (the process of obtaining Vmax which provides the best results) of Vmax is carried out.

4.1. Fine Tuning of Vmax

In the proposed MBPSO, The position update equation shows that in the search process, behavior of the particles is highly dependent on Vmax (i.e., the proposed MBPSO is sensitive with the choice of parameter Vmax). Therefore, experiments are carried out to find the most suitable value of Vmax. This fine tuning is performed for the first five test problems (i.e., problem 1 to 5) given in Table 1.

Since MBPSO is a binary optimization algorithm and test problems of Table 1 are continuous optimization problems therefore, it is necessary to have a routine to convert binary representation into real values. In this paper, first a swarm of N particles is created. Each particle is a vector of D bit strings, each of length L. A simple routine which converts each bit string into an integer is used:

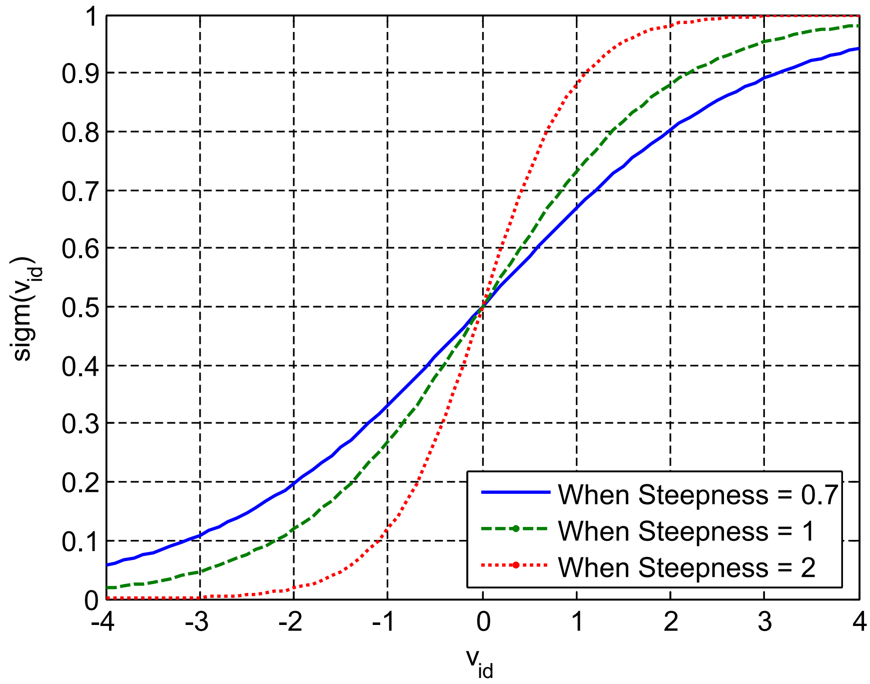

Fig. 3. Sigmoid function when steepness λ is increased.

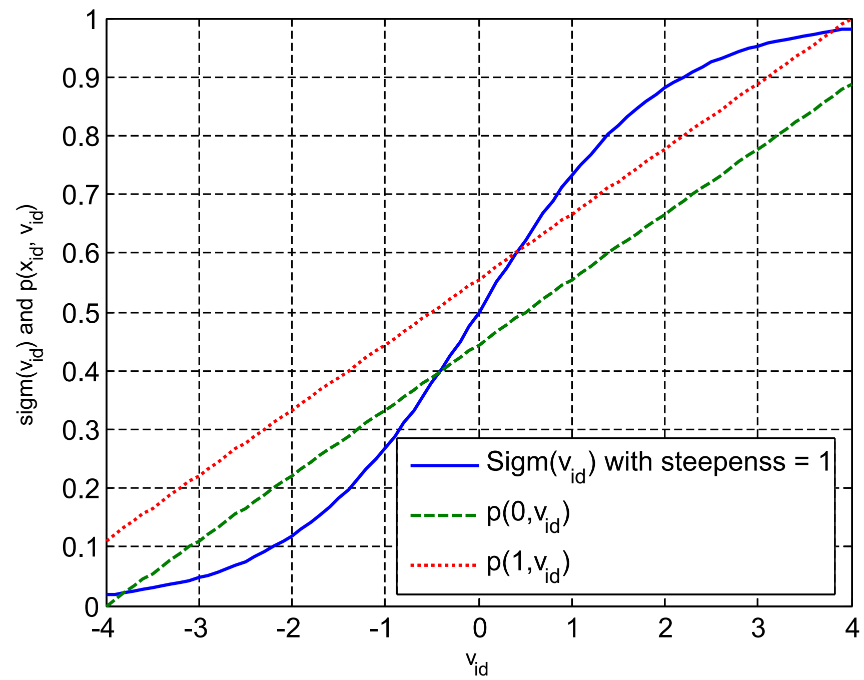

Fig. 4. Comparison between sigmoid function and proposed function $p(x_{id}, v_{id})$.

$$\text{X\_Integer} = \text{Convert\_From\_Binary\_ To\_Integer}(\text{Binary\_Representation of X}).$$

Here, X represents any coordinate of a particle.

The routine, Convert_From_Binary_To_Integer works using following formula:

$X_{integer} = \sum_{i=0}^{L}(X_i \times 2^i)$; Here, it is assumed that ith bit in the binary representation of X is X_i.

Table 1
Test problems.

Problem no.	Function name	Expression	Search space	Objective function value
1.	Sphere	$\text{Min } f(x) = \sum_{i=1}^{n} x_i^2$	$-5.12 \leqslant x_i \leqslant 5.12$	0
2.	Griewank	$\text{Min } f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-600 \leqslant x_i \leqslant 600$	0
3.	Rosenbrock	$Min\ f(x) = \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i^- 1)^2\right)$	$-30 \leqslant x_i \leqslant 30$	0
4.	Rastrigin	$\text{Min } f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$-5.12 \leqslant x_i \leqslant 5.12$	0
5.	Ellipsoidal	$Min\ f(x) = \sum_{i=1}^{n}(x_i - i)^2.$	$-n \leqslant x_i \leqslant n$	0
6.	Cosine Mixture	$\text{Min } f(x) = -0.1\sum_{i=1}^{n} cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2 + 0.1n$	$-1 \leqslant x_i \leqslant 1$	0
7.	Exponential	$\text{Min } f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right) + 1$	$-1 \leqslant x_i \leqslant 1$	0
8.	Zakharov's	$\text{Min } f(x) = \sum_{i=1}^{n} x_i^2 + \left[\sum_{i=1}^{n}\left(\frac{i}{2}\right)x_i\right]^2 + \left[\sum_{i=1}^{n}\left(\frac{i}{2}\right)x_i\right]^4$	$-5.12 \leqslant x_i \leqslant 5.12$	0
9.	Cigar	$\text{Min } f(x) = x_i^2 + 100000\sum_{i=2}^{n} x_i^2$	$-10 \leqslant x_i \leqslant 10$	0
10.	Brown 3	$\text{Min } f(x) = \sum_{i=1}^{n-1}\left[(x_i^2)^{(x_{i+1}^2+1)} + \left(x_{i+1}^2\right)^{(x_i^2+1)}\right]$	$-1 \leqslant x_i \leqslant 4$	0

Now the integer so obtained is converted and bounded in the continuous interval [a, b] as follows:

$$\mathrm{X_{Real}} = \mathrm{a} + \mathrm{X_{Integer}} \times \frac{(\mathrm{b} - \mathrm{a})}{2^{\mathrm{L}}};$$

The bit string length L is set to be 10, in this paper.

The most common values of c_1 and $c_2 (c_1 = 2 = c_2)$ are chosen for experiments. Swarm Size is set to be 5 times the number of decision variables. The algorithm terminates if either maximum number of function evaluations which is set to be $3000 \times$ Number of decision variables, is reached or optimal solution is found. For different values of Vmax (Vmax is varied from 1 to 11 with step size 1.), Success Rate (SR), Average Number of Function Evaluations (AFE), and Average Error (AE) are recorded. SR, AFE, and AE for test problems 1–5 and for different values of Vmax are tabulated in Table 2(a–c) respectively. If any entry in Table 2 is less than 10^{-8}, it is rounded to 0. It is clear that mean of SR, AFE, and AE over the chosen set of test problems is best for Vmax $\in [2, 5]$. Therefore, based on these experiments, the most suitable value of Vmax, for this study is set to 4.

4.2. Comparative Study

In order to verify the feasibility and effectiveness of the proposed MBPSO method for optimization problems, MBPSO is tested on 10 well known benchmark problems, listed in Table 1. The parameter setting is same as suggested in fine tuning of Vmax. Results obtained by MBPSO are also compared with those of original BPSO and Genotype–Phenotype

Table 2
Fine Tuning of Vmax.

Function	Vmax										
	1	2	3	4	5	6	7	8	9	10	11
2(a): *Fine Tuning of Vmax based on Success Rate (SR)*											
Sphere	100	100	100	100	100	100	100	100	100	100	100
Griewank	100	100	100	97	100	100	100	100	100	100	100
Rosenbrock	10	17	3	10	17	23	27	13	10	0	3
Rastrigin	100	100	100	100	100	100	100	100	100	100	100
Ellipsoidal	43	40	70	70	43	20	10	17	0	0	0
Mean	70.67	71.33	74.67	75.33	72	68.67	67.33	66	62	60	60.67
2(b): *Fine Tuning of Vmax based on Average Number of Function Evaluations (AFE)*											
Sphere	19327	20673	16567	17480	22633	24533	26313	28433	29940	31967	33260
Griewank	25587	27667	27107	24813	30207	30760	32967	36313	37120	39567	42433
Rosenbrock	48113	48600	48000	49680	49133	48680	48240	49553	49887	50000	49960
Rastrigin	24520	26013	24640	24187	27120	28467	30093	32280	33653	35207	36933
Ellipsoidal	46927	47200	42027	43327	47227	49293	49700	49453	50000	50000	50000
Mean	32895	34031	31668	31897	35264	36347	37463	39207	40120	41348	42517
2(c): *Fine Tuning of Vmax based on Average Error (AE)*											
Sphere	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Griewank	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rosenbrock	130.87	114.90	64.47	54.43	56.70	92.73	95.30	92.10	89.23	87.87	133.90
Rastrigin	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ellipsoidal	0.30	0.33	0.83	0.97	1.17	0.60	0.73	0.60	1.40	1.67	1.87
Mean	26.23	23.05	13.06	11.08	11.57	18.67	19.21	18.54	18.13	17.91	27.15

Modified Binary Particle Swarm Optimization (GPMBPSO) [52]. For a fair comparison parameters of BPSO are taken same as MBPSO. The routine which converts binary representation into real, discussed in subsection 4.1 is used for BPSO and GPMBPSO also. For GPMBPSO all parameters are same as given in the original paper [52] except swarm size and stopping criteria. Swarm size and stopping criteria for GPMBPSO are same as those of BPSO and MBPSO. All results are based on the 100 simulations of BPSO, GPMBPSO and MBPSO. Number of decision variables for all problems are taken to be 10.

From Table 3, it is clear that except for problem 8 and 9, MBPSO performs better than original BPSO and GPMBPSO [52], in terms of reliability i.e., success rate. Efficiency (due to AFE) of MBPSO is also superior to both other versions except for problem 4. In terms of accuracy (due to AE), MBPSO is again better than BPSO and GPMBPSO except for problems 7, 8 and 9.

To observe the consolidated effect of success rate, average number of function evaluations and average error on BPSO, GPMBPSO, and MBPSO, a Performance Index (PI) is used as given in [53]. The relative performance of an algorithm using this PI is calculated as:

$$PI = \frac{1}{N_P}\sum_{i=1}^{N_p}\left(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i\right),$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$,

$$\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases} \text{and}$$

$$\alpha_3^i = \begin{cases} \frac{Me^i}{Ae^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases}$$

$$i = 1, 2, \ldots, N_p.$$

Table 3
Comparative results of BPSO, GPMBPSO and proposed MBPSO on Test Problems.

Comparison Criterion	Function Serial No.	BPSO	GPMBPSO	MBPSO
SR	1.	100	100	100
	2.	97	89	100
	3.	0	0	18
	4.	100	100	100
	5.	0	2	63
	6.	3	10	14
	7.	2	13	15
	8.	99	96	98
	9.	100	100	99
	10.	94	100	100
AFE	1.	24588	20510	16804
	2.	39216	39246	27558
	3.	50000	50000	47638
	4.	23990	20930	25236
	5.	50000	49920	41640
	6.	49944	49714	49428
	7.	49972	49526	49150
	8.	38076	32504	30084
	9.	30166	25884	22204
	10.	38924	35316	34680
AE	1.	0	0	0
	2.	0.000899	0.003981	0
	3.	202.87	170.6	115.45
	4.	0	0	0
	5.	1.74	1.55	0.41
	6.	2.568	1.656	2.088
	7.	0.644258	0.460274	0.485219
	8.	0.06	0.643125	1.51435
	9.	0	0	1
	10.	0.21	0	0

Sr^i = Number of successful runs of ith problem
Tr^i = Total number of runs of ith problem
Mf^i = Minimum of average number of function evaluations of successful runs used by all algorithms in obtaining the solution of ith problem
Af^i = Average number of function evaluations of successful runs used by an algorithm in obtaining the solution of ith problem
Me^i = Minimum of average error produced by all the algorithms in obtaining the solution of ith problem
Ae^i = Average error produced by an algorithm in obtaining the solution of ith problem
N_p = Total number of problems analyzed.

k_1,k_2 and k_3($k_1 + k_2 + k_3 = 1$ and $0 \leqslant k_1, k_2, k_3 \leqslant 1$) are the weights assigned to percentage of success, average number of function evaluations and average error of successful runs, respectively. From above definition it is clear that PI is a function of k_1,k_2 and k_3. Since $k_1 + k_2 + k_3 = 1$, one of k_i, $i = 1,2,3$ could be eliminated to reduce the number of dependent variables from the expression of PI. Equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

(i) $k_1 = W,\ k_2 = k_3 = \frac{1-W}{2},\ 0 \leqslant W \leqslant 1$
(ii) $k_2 = W,\ k_1 = k_3 = \frac{1-W}{2},\ 0 \leqslant W \leqslant 1$
(iii) $k_3 = W,\ k_1 = k_2 = \frac{1-W}{2}, 0 \leqslant W \leqslant 1$

In each case, performance indices are obtained for BPSO, GPMBPSO, and MBPSO and are shown in Figs. 5–7. It can be observed that in each case MBPSO performs better than GPMBPSO and BPSO while GPMBPSO is always better than BPSO. Thus, overall MBPSO is the best performer on the set of test problems considered in this paper.

5. Application of MBPSO to 0–1 KP and MKP

Now in order to verify the feasibility and effectiveness of the proposed MBPSO method for solving some NP-complete problems having a number of engineering applications, MBPSO is tested on 0–1 Knapsack and Multidimensional Knapsack problems. Instances are picked from OR-Library [48] available and other online sources. Results obtained by MBPSO are compared with GPMBPSO and BPSO. The parameters of GPMBPSO, BPSO and MBPSO are set as in Section 4. Static penalty function approach is applied for handling knapsack constraints. All results are based on the 100 simulations (runs) of GPMBPSO, BPSO and MBPSO.

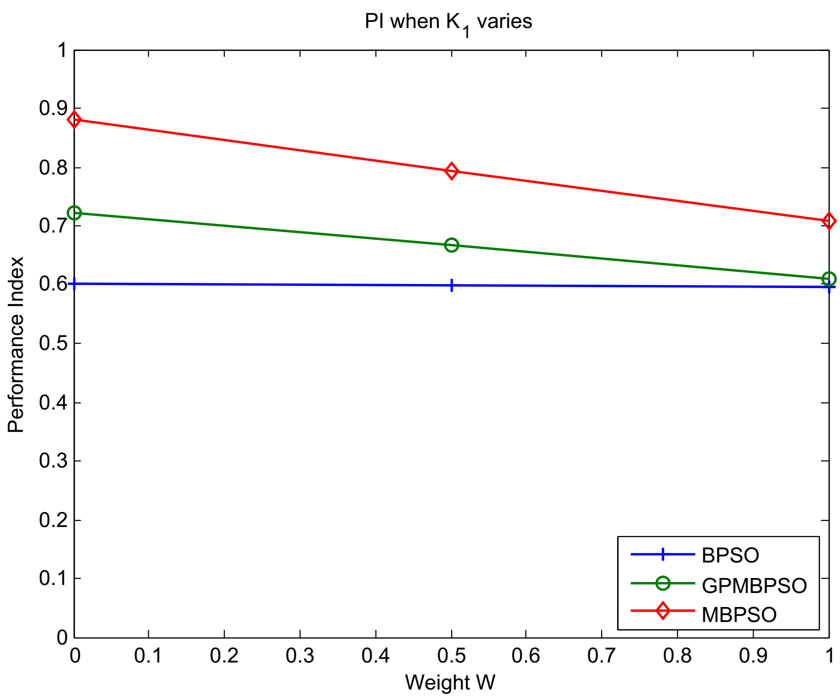

Fig. 5. Performance Index when weight to success rate $k_1$varies.

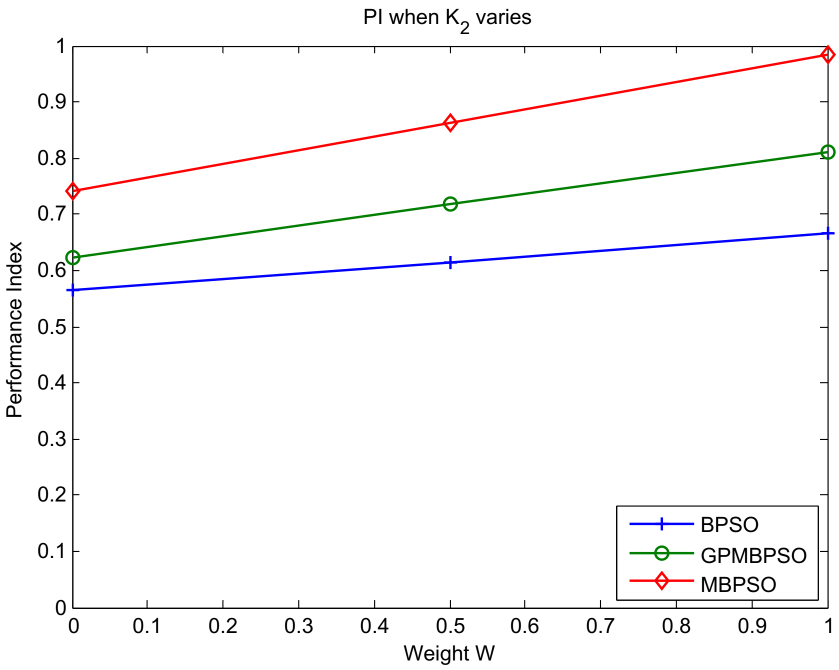

Fig. 6. Performance Index when weight to AFE k_2 varies.

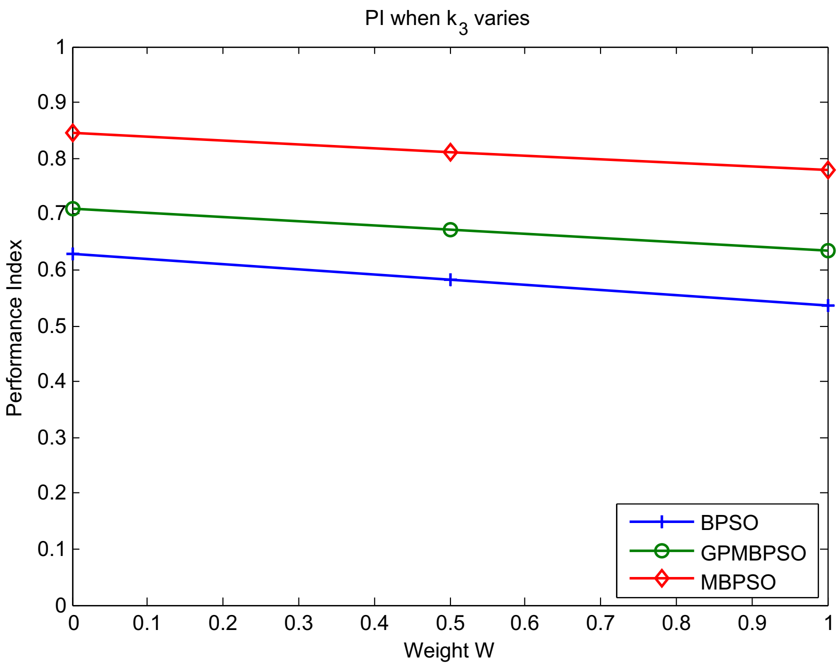

Fig. 7. Performance Index when weight to AE k_3 varies.

5.1. 0–1 Knapsack Problem

Two sets of knapsack problems are considered here to test the efficacy of MBPSO. First set of KP instances which contains 25 instances is taken from http://www.math.mtu.edu/~kreher/cages/Data.html. The instances are used in [47] also. Number of items in these instances ranging between 8 and 24. Since the number of items in this set is relatively small therefore a major difference among the performance of BPSO, GPMBPSO, and MBPSO performance is not expected. Also since the

Table 4
Comparative results of problem Set I of 0–1 Knapsack Problem.

Example	0–1 Knapsack Problem	No. of Items	Method	AVPFT	MAXPFT	WHTGP
1	ks_8a	8	BPSO	3921857.19	3924400	1.99
			GPMBPSO	3922251.98	3924400	1.99
			MBPSO	3924400	3924400	1.99
2	ks_8b	8	BPSO	3807911.86	3813669	0.7189
			GPMBPSO	3807671.43	3813669	0.7189
			MBPSO	3813669	3813669	0.7189
3	ks_8c	8	BPSO	3328608.71	3347452	0.6540
			GPMBPSO	3326300.19	3347452	0.6540
			MBPSO	3347452	3347452	0.6540
4	ks_8d	8	BPSO	4186088.27	4187707	2.9984
			GPMBPSO	4184469.54	4187707	2.9984
			MBPSO	4187707	4187707	2.9984
5	ks_8e	8	BPSO	4932737.28	4955555	2.0509
			GPMBPSO	4921758.82	4955555	2.0509
			MBPSO	4954571.72	4955555	2.0509
6	ks_12a	12	BPSO	5683694.29	5688887	0.2557
			GPMBPSO	5678227.28	5688887	0.2557
			MBPSO	5688552.41	5688887	0.2557
7	ks_12b	12	BPSO	6478582.96	6498597	0.0636
			GPMBPSO	6476487.08	6498597	0.0636
			MBPSO	6493130.57	6498597	0.0636
8	ks_12c	12	BPSO	5166957.08	5170626	0.7633
			GPMBPSO	5162237.91	5170626	0.7633
			MBPSO	5170493.3	5170626	0.7633
9	ks_12d	12	BPSO	6989842.73	6992404	0.5875
			GPMBPSO	6988151.02	6992404	0.5875
			MBPSO	6992144.26	6992404	0.5875
10	ks_12e	12	BPSO	5316879.59	5337472	0.2186
			GPMBPSO	5301119.31	5337472	0.2186
			MBPSO	5337472	5337472	0.2186
11	ks_16a	16	BPSO	7834900.26	7850983	0.2367
			GPMBPSO	7826923.53	7850983	0.2367
			MBPSO	7843073.29	7850983	0.2367
12	ks_16b	16	BPSO	9334408.62	9352998	0.0153
			GPMBPSO	9326158.74	9352998	0.0153
			MBPSO	9350353.39	9352998	0.0153
13	ks_16c	16	BPSO	9118837.47	9151147	0.603
			GPMBPSO	9114581.85	9151147	0.6038
			MBPSO	9144118.38	9151147	0.6038
14	ks_16d	16	BPSO	9321705.87	9348889	0.1396
			GPMBPSO	9317336.67	9348889	0.1396
			MBPSO	9337915.64	9348889	0.1396
15	ks_16e	16	BPSO	7758572.21	7769117	0.2014
			GPMBPSO	7757247.79	7769117	0.2014
			MBPSO	7764131.81	7769117	0.2014
16	ks_20a	20	BPSO	10707360.91	10727049	0.0574
			GPMBPSO	10702954.99	10727049	0.0574
			MBPSO	10720314.03	10727049	0.0574
17	ks_20b	20	BPSO	9791306.65	9818261	0.2030
			GPMBPSO	9786719.85	9818261	0.2030
			MBPSO	9805480.48	9818261	0.2030
18	ks_20c	20	BPSO	10703423.34	10714023	0.1966
			GPMBPSO	10695550.75	10714023	0.1966
			MBPSO	10710947.05	10714023	0.1966
19	ks_20d	20	BPSO	8910152.57	8929156	0.0938
			GPMBPSO	8905564.36	8929156	0.0938
			MBPSO	8923712.21	8929156	0.0938
20	ks_20e	20	BPSO	9349546.98	9357969	0.1327
			GPMBPSO	9343911.1	9357969	0.1327
			MBPSO	9355930.35	9357969	0.1327

(*continued on next page*)

Table 4 (*continued*)

Example	0–1 Knapsack Problem	No. of Items	Method	AVPFT	MAXPFT	WHTGP
21	ks_24a	24	BPSO	13510432.96	13549094	0.0252
			GPMBPSO	13506115.12	13549094	0.0252
			MBPSO	13532060.07	13549094	0.0252
22	ks_24b	24	BPSO	12205346.16	12233713	0.0847
			GPMBPSO	12202425.75	12233713	0.0847
			MBPSO	12223442.61	12233713	0.0847
23	ks_24c	24	BPSO	12427880.56	12448780	0.1492
			GPMBPSO	12419101.82	12448780	0.1492
			MBPSO	12443349.03	12448780	0.1492
24	ks_24d	24	BPSO	11792064.76	11815315	0.0986
			GPMBPSO	11791581.41	11815315	0.0986
			MBPSO	11803712.38	11815315	0.0986
25	ks_24e	24	BPSO	13922797.55	13940099	0.2527
			GPMBPSO	13921046.22	13940099	0.2527
			MBPSO	13932526.16	13940099	0.2527

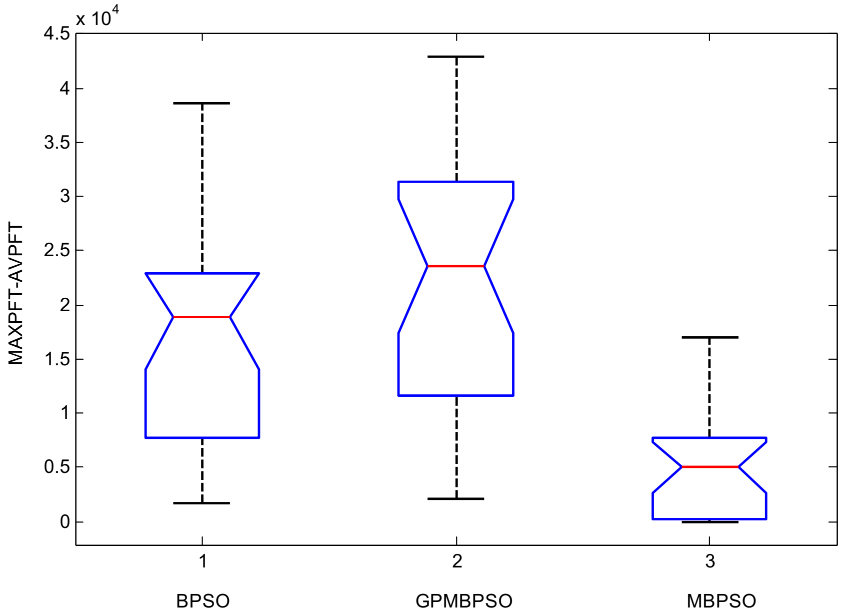

Fig. 8. Boxplot of problem set I of 0–1 Knapsack Problem.

instances are generated randomly and the optimum solution is not known, the results of all three versions of binary PSO are compared on the basis of maximum profit over 100 runs (MAXPFT), average profit over 100 runs (AVPFT), and total weight gap, in percentage, in case of maximum profit $\left(=\left[\frac{(\text{weight limit}-\text{weight when the maximum profit is reported})}{\text{weight limit}}\right]\times 100\right)$ denoted as WHTGP. Second set of KP instances is taken from http://www.cs.colostate.edu/~cs575dl/assignments/assignment5. html and [6] with number of items between 10 and 500. The optimum solutions of these instances are known; therefore the comparison is made on the basis of success rate (= total number of runs out of 100 that produces optimum solution within the termination criterion) denoted by SR, average number of function evaluations (= average of function evaluations used in all 100 simulations) denoted by AFE, average error (= average of —optimum solution – obtained solution— over runs in which, obtained solution is feasible) denoted by AE, least error (= minimum of —optimum solution – obtained solution— over runs in which, obtained solution is feasible) denoted by LE, and the standard deviation of error denoted by SD. It should be noted that SD considered in this paper is of feasible solutions only.

From Table 4, which summarizes the results of first problem set, it is clear that maximum profit MAXPFT of 100 runs by BPSO, GPMBPSO, and MBPSO is same for all instances (i.e., performance of all three versions is same, if the best solution of 100 runs is considered). Obviously, the WHTGP will also be same for all three algorithms. Now, if we see AVPFT then a minor improvement of MBPSO over GPMBPSO and BPSO can be observed. For all the instances, AVPFT of MBPSO is slightly greater

Table 5
Comparative results of Problem Set II of 0–1 Knapsack Problem.

Example	Number of items	Optimal solution	Algorithm	SR	AFE	AE	LE	SD
1	10	295	BPSO	99	681	0.02	0	0.1989
			GPMBPSO	100	391	0	0	0
			MBPSO	100	543	0	0	0
2	20	1024	BPSO	100	1130	0	0	0
			GPMBPSO	100	1036	0	0	0
			MBPSO	100	2952	0	0	0
3	50	3112	BPSO	35	109819	2.46	0	3.2478
			GPMBPSO	46	96422	1.91	0	2.9294
			MBPSO	66	62212	0.68	0	1.4274
4	100	2683223	BPSO	20	268715	694.39	0	466.9345
			GPMBPSO	24	266190	761.96	0	556.9693
			MBPSO	50	241805	284.03	0	325.2135
5	200	5180258	BPSO	0	600000	688.55	160	276.0619
			GPMBPSO	0	600000	689.58	159	278.3393
			MBPSO	0	600000	872.74	25	432.8804
6	500	1359213	BPSO	0	1500000	1216.73	880	153.3096
			GPMBPSO	0	1500000	1251.46	793	223.4678
			MBPSO	0	1500000	1248.96	586	275.2432

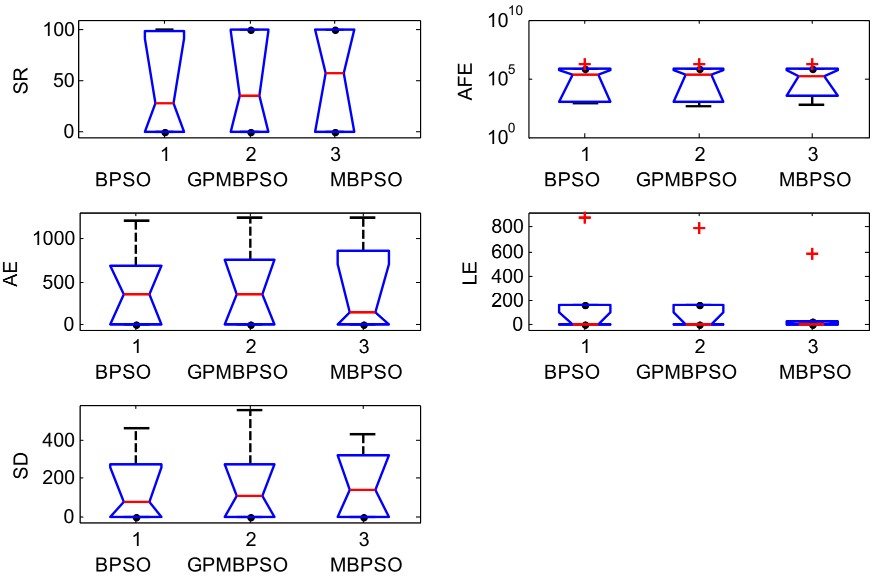

Fig. 9. Boxplots of problem set II of Knapsack Problem.

than GPMBPSO and BPSO. A statistical view in terms of boxplot as shown in Fig. 8 is more appropriate to see the improvement of MBPSO over GPMBPSO and BPSO. In Fig. 8, boxplots of BPSO, GPMBPSO, and MBPSO are plotted for difference (MAX-PFT – AVPFT). Clearly, the boxplot of MBPSO is close to zero and has less height than that of BPSO and GPMBPSO. This shows that the minimum value, median, maximum value, quartiles, and standard deviation of the difference discussed above are least for MBPSO as compare to other two versions and therefore, MBPSO is relatively better than BPSO and GPMBPSO.

Results of problem set II are given in Table 5. It is evident that MBPSO shows higher success rate for all problems. Thus, MBPSO is more reliable than BPSO and GPMBPSO. AFE are also least, in case of MBPSO for all instances except instance 2. This shows that MBPSO is comparatively fast. It can also be seen than MBPSO is better than BPSO and GPMBPSO from the point of view of LE, AE and SD which reflects higher accuracy of MBPSO than BPSO and GPMBPSO. A comparative analysis of BPSO, GPMBPSO, and MBPSO can be seen at a glance using boxplots. The boxplots, of these versions, for all comparison criteria are shown in Fig. 9 and establish the fact that MBPSO is more effective than BPSO and GPMBPSO in almost all criteria considered here.

Table 6
Comparative results of Problem Set I of Multidimensional Knapsack Problem.

Example	Instance	Algorithm	SR	AFE	AE	LE	SD
1	Sento1	BPSO	43	112666	19.67	0	26.1228
		GPMBPSO	41	120004	22.56	0	29.57
		MBPSO	52	111303	9.96	0	15.1195
2	Sento2	BPSO	11	162720	16.08	0	12.1579
		GPMBPSO	9	165742	18.59	0	16.397
		MBPSO	44	125271	5.4	0	6.63325
3	Weing1	BPSO	88	14197.4	63.79	0	174.729
		GPMBPSO	77	23018.8	122.71	0	227.584
		MBPSO	100	9444.4	0	0	0
4	Weing2	BPSO	90	12916.4	23.1	0	96.6405
		GPMBPSO	81	22365	89.28	0	602.013
		MBPSO	99	10502.8	1.6	0	15.9198
5	Weing3	BPSO	15	73451	787.34	0	775.814
		GPMBPSO	8	78121.4	1190.41	0	888.113
		MBPSO	37	57626.8	347.86	0	373.721
6	Weing4	BPSO	80	21802.2	401.09	0	1035.9
		GPMBPSO	77	23394	424.92	0	1044.12
		MBPSO	99	9403.8	27.15	0	270.139
7	Weing5	BPSO	59	38638.6	1274.05	0	1792.49
		GPMBPSO	52	43097.6	1745.73	0	1965.89
		MBPSO	86	20804	384.4	0	1131.66
8	Weing6	BPSO	31	59742.2	278.6	0	209.203
		GPMBPSO	37	54406.8	302.7	0	320.633
		MBPSO	74	29729	101.4	0	171.067
9	Weing7	BPSO	5	301444	321.69	0	727.039
		GPMBPSO	4	303594	502.22	0	875.657
		MBPSO	41	230180	38.33	0	33.9594
10	Weing8	BPSO	92	115354	0.08	0	0.271293
		GPMBPSO	95	98974	0.05	0	0.217945
		MBPSO	89	154350	0.11	0	0.31289

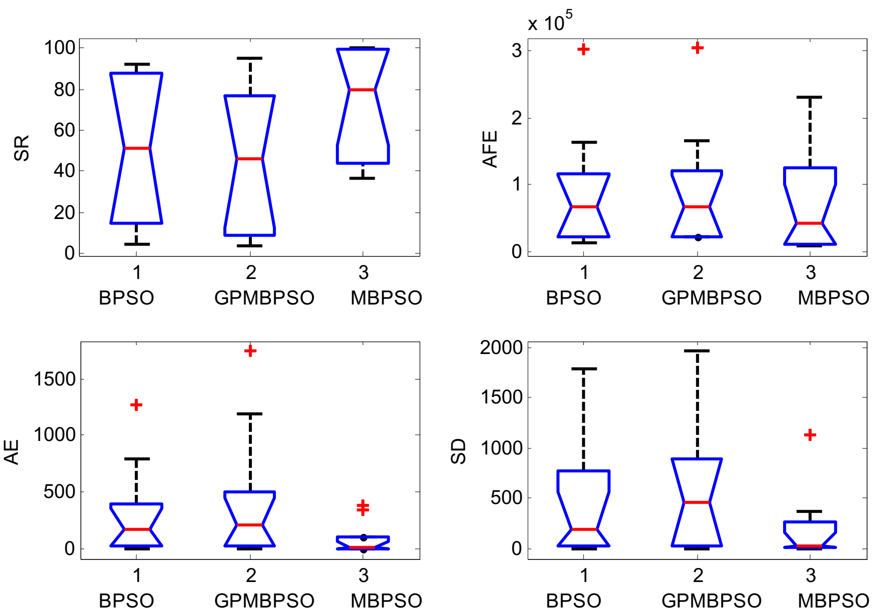

Fig. 10. Boxplots of problem set I of Multidimensional Knapsack Problem.

Table 7
Comparative results of Problem Set II of Multidimensional Knapsack Problem.

Example	Algorithm	SR	AFE	AE	LE	SD	No. of Ifs
1	BPSO	92	11697	5.53	0	19.2408	0
	GPMBPSO	89	13677	6.46	0	19.833	0
	MBPSO	100	10548	0	0	0	0
2	BPSO	68	33414	2.89	0	6.77332	0
	GPMBPSO	60	41091	7.06	0	14.3205	0
	MBPSO	80	26451	1	0	2	0
3	BPSO	85	18598.5	10.5	0	26.2661	0
	GPMBPSO	81	21778.5	13.09	0	30.4963	0
	MBPSO	98	12973.5	0.72	0	6.3231	0
4	BPSO	100	3492	0	0	0	0
	GPMBPSO	99	5832	0.3	0	2.98496	0
	MBPSO	100	9447	0	0	0	0
5	BPSO	100	3640.5	0	0	0	0
	GPMBPSO	100	3487.5	0	0	0	0
	MBPSO	100	9688.5	0	0	0	0
6	BPSO	52	63414	8.45	0	9.68646	0
	GPMBPSO	34	82518	12.66	0	11.961	0
	MBPSO	80	41088	3.25	0	6.58692	0
7	BPSO	74	37090	5.38	0	9.24097	0
	GPMBPSO	61	52018	9.49	0	16.6418	0
	MBPSO	99	21236	0.18	0	1.79098	0
8	BPSO	52	61776	3	0	6.44515	0
	GPMBPSO	44	70108	6.86	0	15.7022	0
	MBPSO	95	25154	0.1	0	0.43589	0
9	BPSO	95	13262	1.82	0	8.00547	0
	GPMBPSO	83	28034	8.71	0	24.8311	0
	MBPSO	100	17920	0	0	0	0
10	BPSO	61	73976	9.25	0	16.9313	0
	GPMBPSO	62	71134	11.37	0	21.4363	0
	MBPSO	98	26960	0.81	0	5.9828	0
11	BPSO	24	121124	18.7895	0	36.9759	62
	GPMBPSO	13	134650	4.20E + 19	0	395.779	42
	MBPSO	41	102264	41.3371	0	200.864	11
12	BPSO	79	42046	5.45	0	17.0613	0
	GPMBPSO	74	52362	10.15	0	24.0746	0
	MBPSO	99	26214	0.01	0	0.099499	0
13	BPSO	89	33798	9.6	0	30.2949	0
	GPMBPSO	82	43350	2.00E + 18	0	30.3244	2
	MBPSO	95	32464	0.791667	0	7.71621	4
14	BPSO	70	66608	11.65	0	19.4779	0
	GPMBPSO	54	93602	2.00E + 18	0	34.3155	2
	MBPSO	88	49094	2.28421	0	8.09894	5
15	BPSO	91	45150	0.774194	0	5.35238	7
	GPMBPSO	79	62866	3.00E + 18	0	19.3973	3
	MBPSO	97	26418	1.29	0	7.81447	0
16	BPSO	49	100968	5.13	0	13.3227	0
	GPMBPSO	26	139708	11.75	0	22.4082	0
	MBPSO	91	44278	0.9	0	7.36682	0
17	BPSO	75	60796	2.94	0	5.54945	0
	GPMBPSO	60	84712	4.76	0	6.24999	0
	MBPSO	100	26606	0	0	0	0
18	BPSO	36	142116	11.25	0	13.1129	0
	GPMBPSO	34	146216	13.05	0	14.6447	0
	MBPSO	85	60386	1.78	0	5.28504	0
19	BPSO	29	161562	255	0	248.444	66
	GPMBPSO	32	163814	2.80E + 19	0	68.837	28
	MBPSO	51	125192	13.5684	0	22.9474	5
20	BPSO	79	68938	4.24	0	9.23485	0
	GPMBPSO	69	85398	6.97	0	12.4398	0
	MBPSO	96	42082	0.86	0	5.28398	0

(*continued on next page*)

Table 7 (*continued*)

Example	Algorithm	SR	AFE	AE	LE	SD	No. of Ifs
21	BPSO	73	74202	11.73	0	20.3955	0
	GPMBPSO	42	143970	6.00E + 18	0	23.7576	6
	MBPSO	77	76974	8.08511	0	17.6838	6
22	BPSO	37	163616	22.81	0	25.9806	0
	GPMBPSO	26	193898	1.10E + 19	0	31.2043	11
	MBPSO	45	150994	12.0706	0	17.1277	15
23	BPSO	1	238056	14.5	0	19.1563	86
	GPMBPSO	7	230698	6.40E + 19	0	32.6514	64
	MBPSO	10	225526	25.0517	0	42.3526	42
24	BPSO	52	136256	7.74	0	13.0043	0
	GPMBPSO	48	156092	11.5	0	17.5137	0
	MBPSO	90	58560	0.5	0	1.5	0
25	BPSO	25	191000	13.59	0	9.84083	0
	GPMBPSO	25	194452	14.68	0	11.5394	0
	MBPSO	52	136124	7.84	0	8.28941	0
26	BPSO	0	240000	899	550	833.4	49
	GPMBPSO	0	240000	5.30E + 19	550	28284.9	53
	MBPSO	0	240000	587.485	550	27.5674	3
27	BPSO	19	228766	98	0	95.5186	80
	GPMBPSO	27	223438	5.80E + 19	0	47.2344	58
	MBPSO	77	122464	20.3371	0	90.701	11
28	BPSO	35	205826	40.7368	0	215.87	43
	GPMBPSO	13	247132	7.30E + 19	0	534.32	73
	MBPSO	10	247890	149	0	140	26
29	BPSO	0	270000	678	459	702.51	95
	GPMBPSO	0	270000	9.90E + 19	486	0	99
	MBPSO	0	270000	586	586	0	99
30	BPSO	47	159064	7.72	0	11.0309	0
	GPMBPSO	37	191942	8.95	0	11.6562	0
	MBPSO	72	106162	1.73	0	4.7241	0
31	BPSO	21	65487.2	27.35	0	22.4229	0
	GPMBPSO	10	73510.2	35.28	0	26.3492	0
	MBPSO	45	50067.4	10.85	0	12.0982	0
32	BPSO	19	85207.4	42.07	0	41.9157	0
	GPMBPSO	17	86609.9	39.62	0	36.224	0
	MBPSO	65	48954.9	7.27	0	11.7217	0
33	BPSO	31	61568.4	200.86	0	169.25	0
	GPMBPSO	21	70313.4	2935.64	0	1732.02	0
	MBPSO	40	58014.5	102.86	0	108.55	0
34	BPSO	12	53750	44.62	0	23.9778	0
	GPMBPSO	7	57180	48.79	0	21.5078	0
	MBPSO	36	42081	22	0	22.1418	0
35	BPSO	38	78800	17.64	0	20.7627	0
	GPMBPSO	54	62148	14.94	0	27.0059	0
	MBPSO	59	61812	8.95	0	14.0224	0
36	BPSO	8	103467	13.56	0	8.91327	0
	GPMBPSO	20	91850.6	12.65	0	10.2834	0
	MBPSO	44	72677.3	5.19	0	5.89694	0
37	BPSO	29	61070.8	21.81	0	19.7386	0
	GPMBPSO	11	76214.6	32.5	0	25.345	0
	MBPSO	48	50002.4	10.96	0	13.5033	0
38	BPSO	21	85300.3	41.29	0	36.0559	0
	GPMBPSO	22	83774.3	39.42	0	42.0179	0
	MBPSO	58	56075.3	10.51	0	16.9555	0

5.2. Multidimensional Knapsack Problem

MBPSO has also been tested on two groups of benchmarks of MKP selected from OR-Library [48], the first group corresponds to series "*sento*" [49] and "*weing*" [50] which contains 10 instances and the number of items ranging between 28 to 105. The second group corresponds to "*weish*"[51], which contains 38 instances and the number of items ranging between 20 and 90. These instances have also been solved by BPSO in [37]. Since we have considered MKP instances, whose optimal

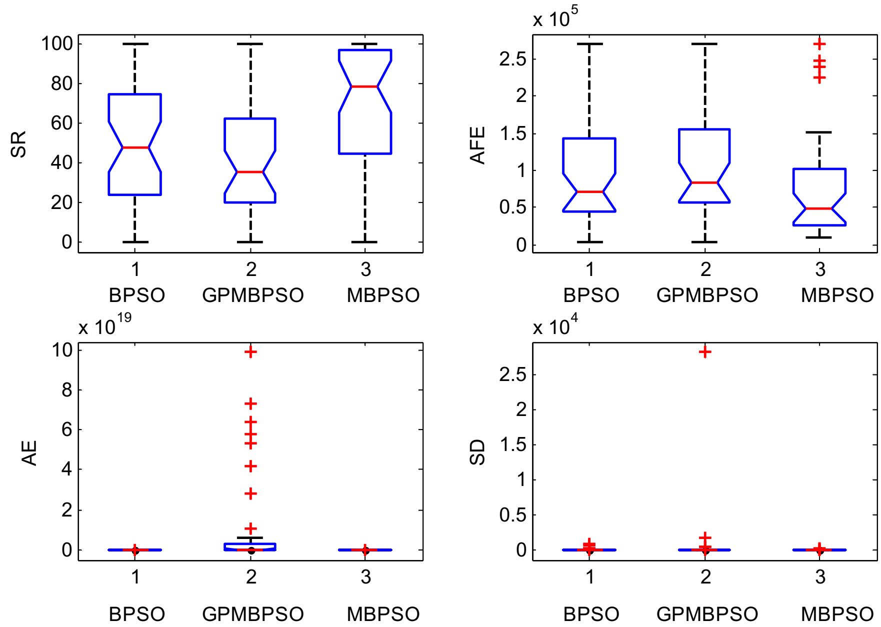

Fig. 11. Boxplots of problem set II of Multidimensional Knapsack Problem.

solution is known, therefore the comparison between BPSO, GPMBPSO, and MBPSO is carried out, on the basis of SR, AFE, AE, LE and SD as in Section 5.1.

The experimental results of instances of first group obtained by BPSO, GPMBPSO, and MBPSO are shown in Table 6. Here, second column contains the name of the instance. From Table 6, it is clear that MBPSO is more reliable than BPSO and GPMBPSO as success rate (SR) for MBPSO is higher than that of other two for 9 instances out of 10. Average number of function evaluations, AFE for MBPSO are also less than that of BPSO and GPMBPSO for 9 instances indicating capability of MBPSO of giving solution faster. Also MBPSO is able to provide better quality solution for MKP as AE, LE and SD is least for MBPSO for most of the instances. An overall strength of MBPSO with respect to BPSO and GPMBPSO can be seen from boxplots, shown in Fig. 10. In Fig. 10, the boxplots of BPSO, GPMBPSO and MBPSO for SR, AFE, AE and SD are shown.

Table 7 shows the results of problem set II of MKP. Since all solutions are not feasible so for this problem set, number of infeasible solutions (No. of IFs) are shown in the last column in addition to the other information. Note that the AE, LE and SD are computed only for feasible solutions. Fig. 11 shows the boxplot for problem set II of MKP. From Table 7 and Fig. 11, it is obvious that MBPSO is better in reliability, accuracy, and computational cost than BPSO and GPMBPSO. Number of infeasible solutions obtained by MBPSO is higher in 5 problems, lower in 7 problems and same in other problems. This fact also verifies a better reliability of MBPSO as compare to BPSO and GPMBPSO.

6. Conclusion

In this paper, a new Binary Particle Swarm Optimization technique, namely, Modified Binary Particle Swarm Optimization (MBPSO) algorithm for Knapsack Problems (KPs) is proposed. The proposed algorithm is first tested on 10 benchmark problems and the obtained results are compared with that of BPSO as well as a modified version of BPSO found in literature, namely, Genotype–Phenotype Modified Binary Particle Swarm Optimization (GPMBPSO). The proposed algorithm is then applied to 0–1 Knapsack Problem (KP) and Multidimensional Problem (MKP). 31 (25 + 6) instances of KP and 48 (10 + 38) instances of MKP are considered to verify the performance of proposed MBPSO. Obtained results prove that MBPSO outperforms BPSO and GPMBPSO in terms of reliability, cost and quality of solution.

The method can also be extended to other combinatorial optimization problems. However, our aim is specially to design MBPSO only for KPs, but for other real world binary optimization problems, MBPSO may be examined. MBPSO, with different parameter settings may also be tested as in this paper, no computations are carried out to optimize the MBPSO parameters except Vmax, what could provide better results.

Acknowledgements

Authors gratefully acknowledge the discussion with Prof. Andries Engelbrecht, University of Pretoria, and Dr. Adnan Acan, Eastern Mediterranean University, CYPRUS, in preparation of this research article. The author expresses his gratitude to the anonymous reviewers whose suggestions have resulted in an improved presentation of this paper.

References

[1] G.B. Dantzig, Discrete variable extremum problems, Operations Research 5 (1957) 266–277.
[2] A. Liu, J. Wang, G. Han, S. Wang, J. Wen, Improved simulated annealing algorithm solving for 0/1 knapsack problem, in: Proceedings of the Sixth international Conference on Intelligent Systems Design and Applications, ISDA, vol. 02, IEEE Computer Society, Washington, DC, 2006, pp. 1159–1164.
[3] J. Thiel, S. Voss, Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms, INFOR, Canada, vol. 32, 1994, pp. 226–242.
[4] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of Heuristics 4 (1998) 63–86.
[5] Hongwei Huo, Jin Xu, Zheng Bao, Solving 0/1 knapsack problem using genetic algorithm, Journal of Xidian University 26 (4) (1999) 493–497.
[6] P. Zhao, P. Zhao, X. Zhang, A new ant colony optimization for the knapsack problem, in: Proceedings of Seventh International Conference on Computer – Aided Industrial Design and Conceptual Design, November 17–19, 2006, pp. 1–3.
[7] H. Shi, Solution to 0/1 knapsack problem based on improved ant colony algorithm, international conference on information acquisition, in: IEEE International Conference on Information Acquisition, 2006, pp. 1062–1066.
[8] C. Peng, Z. Jian Li, Liu, Solving 0–1 knapsack problems by a discrete binary version of differential evolution, in: Second International Symposium on Intelligent Information Technology Application, IITA '08, vol. 2, 2008, pp. 513–516.
[9] W. Lei, P. Jin, J. Licheng, Immune algorithm, Acta Electronica Sinia 28 (7) (2000) 74–78.
[10] B. Ye, J. Sun, Wen-Bo Xu, Solving the hard knapsack problems with a binary particle swarm approach, ICIC 2006, LNBI 4115, 2006, pp. 155–163.
[11] X. Shen, Y. Li, Wei Wang, A dynamic adaptive particle swarm optimization optimization for knapsack problem, in: Proceedings of the Sixth World Congress on Intelligent Control and Automation, June 21–23, Dalian, China, 2006, pp. 3183–3187.
[12] X. Shen, W. Wang, P. Zheng, Modified particle swarm optimization for 0–1 knapsack problem, Computer Engineering 32 (18) (2006) 23–24. 38.
[13] Yi-Chao He, L. Zhou, Chun-Pu Shen, A greedy particle swarm optimization for solving knapsack problem, in: International Conference on Machine Learning and Cybernetics, vol. 2, 2007, pp. 995–998.
[14] Guo-Li Zhang, Yi Wei, An improved particle swarm optimization algorithm for solving 0–1 knapsack problem, in: Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12–15 July 2008, pp. 915–918 .
[15] J. Puchinger, G. Raidl, U. Pferschy, The multidimensional knapsack problem, Structure and algorithms, Technical Report 006149, National ICT Australia, Melbourne, Australia, 2007.
[16] F. Dammeyer, S. Voss, Dynamic tabu list management using reverse elimination method, Annals of Operations Research 41 (1993) 31–46.
[17] R. Aboudi, K. Jornsten, Tabu search for general zero-one integer programs using the pivot and complement heuristic, ORSA Journal on Computing 6 (1994) 82–93.
[18] R. Battiti, G. Tecchiolli, Local search with memory: benchmarking RTS, OR Spektrum 17 (1995) 67–86.
[19] F. Glover, G.A. Kochenberger, Critical event tabu search for multidimensional knapsack problem, in: I.H. Osman, J.P. Kelly (Eds.), Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, 1996, pp. 407–427.
[20] M. Vasquez, Jin-Kao Hao, A hybrid approach for the 0–1 multidimensional knapsack problem, in: Proceedings of IJCAI-01, Seatle, Washington, 2001, pp. 328-333.
[21] V.C. Li, Tight oscillations tabu search for multidimensional knapsack problems with generalized upper bound constraints, Computers and Operations Research 32 (11) (2005) 2843–2852.
[22] V.C. Li, G.L. Curry, Solving multidimensional knapsack problems with generalized upper bound constraints using critical event tabu search, Computers and Operations Research 32 (4) (2005) 825–848.
[23] S. Khuri, T. Back, J. Heitkotter, The zero/one multiple knapsack problem and genetic algorithm, Proceedings of the 1994 ACM Symposium on Applied Computing (SAC'94), ACM Press, 1994, pp. 188–193.
[24] G. Rudolph, J.A. Sprave, Cellular genetic algorithm with self-adjusting acceptance threshold, in: Proceeding of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, IEE, London, 1995, pp. 365–372.
[25] C. Cotta, J.Ma. Troya, A Hybrid genetic Algorithm for the 0–1 multiple knapsack problem, in: Proceedings of the International Conference on Artificial Networks and Genetic Algorithm, Springer-Verlag, Berlin, 1997, pp. 250–254.
[26] K. Kato, M. Sakawa, Genetic algorithms with decomposition procedures for multidimensional 0–1 knapsack problems with block angular structures, IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics 33 (3) (2003) 410–419.
[27] Hong Li, Yong-Chang Jiao, Li Zhang, Ze-Wei Gu, Genetic algorithm based on the orthogonal design for multidimensional knapsack problems, ICNC 2006, Part I, LNCS 4221, 2006, pp. 696–705.
[28] F. Djannaty, S. Doostdar, A hybrid genetic algorithm for the multidimensional knapsack problem, International Journal of Contemporary Mathematics Sciences 3 (9) (2008) 443–456.
[29] M. Kong, P. Tian, Y. Kao, A new ant colony optimization algorithm for the multidimensional Knapsack problem, Computers and Operations Research 35 (8) (2008) 2672–2683.
[30] Ji Junzhong, Huang Zhen, Chunnian Liu, An ant colony optimization algorithm for solving the multidimensional knapsack problems, in: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2007, pp. 10–16.
[31] I. Alaya, C. Solnon, K. Ghdira, Ant algorithm for the multi-dimensional knapsack problem, in: International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004), 2004, pp. 63–72.
[32] S. Uyar, G. Eryigit, Improvements to penalty-based evolutionary algorithms for the multi-dimensional knapsack problem using a gene-based adaptive mutation approach, GECCO (2005) 1257–1264.
[33] A. Drexl, A simulated annealing approach to the multiconstraint zero-one knapsack problem, Computing 40 (1) (1988) 1–8.
[34] M. Gong, L. Jiao, Ma Wenping, Gou Shuiping, Solving multidimensional knapsack problems by an immune-inspired algorithm, in: IEEE Congress on Evolutionary Computation, 2007, pp. 3385–3391.
[35] M. Kong, P. Tian, Apply the particle swarm optimization to the multidimensional knapsack problem, ICAISC 2006, vol. 4029, Springer, Berlin, Heidelberg, 2006, pp. 1140–1149.
[36] F. Hembecker, Heitor S. Lopes, Godoy Walter Jr., Particle swarm optimization for the multidimensional knapsack problem, Adaptive and Natural Computing Algorithms, vol. 4431, Springer, Berlin, Heidelberg, 2007, pp. 358–365.
[37] K. Deep, J.C. Bansal, A socio-cognitive particle swarm optimization for multi-dimensional knapsack problem, in: First International Conference on Emerging Trends in Engineering and Technology ICETET, India, 2008, pp. 355–360.
[38] K. Fleszar, K.S. Hindi, Fast, effective heuristics for the 0–1 multi-dimensional knapsack problem, Computers and Operations Research 36 (5) (2009) 1602–1607.
[39] J. Gottlieb, Permutation-based evolutionary algorithms for multidimensional knapsack problems, in: J. Carroll, E. Damiani, H. Haddad, D. Oppenheim (Eds.), Proceedings of the 2000 ACM Symposium on Applied Computing, SAC '00, vol. 1, ACM, New York, NY, 2000, pp. 408–414. Como, Italy.
[40] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings IEEE International Conference Neural Networks, vol. 4, 1942–1948, 1995.
[41] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of Sixth International Symposium Micro Machine Human Science, 1995, pp. 39–45.
[42] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: Background and development, Natural Computing: An International Journal 6 (4) (2007) 467–484.
[43] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization and indicative applications, Natural Computing 7 (1) (2008) 109–124.
[44] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: an overview, Swarm Intelligence 1 (2007) 33–57.

[45] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm optimization, in: Proceedings of the world Multiconference on Systemics, Cybernetics, Informatics, 1997, pp. 4104–4109.
[46] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley and Sons, Ltd., 2005. p. 330.
[47] Lee Chou-Yuan, Lee Zne-Jung, Su Shun-Feng, A new approach for solving 0/1 knapsack problem, IEEE International Conference on Systems, Man, and Cybernetics October 8–11, Taipei, Taiwan, 2006, pp. 3138–3143.
[48] J.E. Beasley, ORLib – Operations Research Library. [http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html], (2005).
[49] S. Senyu, Y. Toyoda, An approach to linear programming with 0–1 variables, Management Science 15 (1967) B196–B207.
[50] H.M. Weingartner, D.N. Ness, Methods for the solution of the multidimensional 0/1 knapsack problem, Operations Research 15 (1967) 83–103.
[51] W. Shih, A branch and bound method for the multiconstraint zero-one knapsack problem, Journal of Operations Research Society 30 (1979) 369–378.
[52] Sangwook Lee, Sangmoon Soak, Sanghoun Oh, Witold Pedrycz, Moongu Jeon, Modified binary particle swarm optimization, Progress in Natural Science 18 (9) (2008) 1161–1166.
[53] K. Deep, J.C. Bansal, Mean particle swarm optimisation for function optimisation, International Journal of Computational Intelligence Studies 1 (1) (2009) 72–92.