Applied Mathematical Modelling 38 (2014) 1338–1350

Contents lists available at ScienceDirect

Applied Mathematical Modelling

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

Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem

Mingchang Chih [a,*], Chin-Jung Lin [b], Maw-Sheng Chern [b], Tsung-Yin Ou [c]

[a] System Development Center, Chung-Shan Institute of Science and Technology, Lungtan Township, Taoyuan 32546, Taiwan, ROC
[b] Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30043, Taiwan, ROC
[c] Department of Industrial Engineering and Management, National Quemoy University, Jinning Township, Kinmen 89250, Taiwan, ROC

ARTICLE INFO

Article history:
Received 17 September 2012
Received in revised form 1 June 2013
Accepted 8 August 2013
Available online 21 August 2013

Keywords:
Multidimensional knapsack problem
Binary particle swarm optimization
Time-varying acceleration coefficients
OR-library

ABSTRACT

The multidimensional knapsack problem (MKP) is a difficult combinatorial optimization problem, which has been proven as NP-hard problems. Various population-based search algorithms are applied to solve these problems. The particle swarm optimization (PSO) technique is adapted in our study, which proposes two novel PSO algorithms, namely, the binary PSO with time-varying acceleration coefficients (BPSOTVAC) and the chaotic binary PSO with time-varying acceleration coefficients (CBPSOTVAC). The two proposed methods were tested using 116 benchmark problems from the OR-Library to validate and demonstrate the efficiency of these algorithms in solving multidimensional knapsack problems. The results were then compared with those in the other two existing PSO algorithms. The simulation and evaluation results showed that the proposed algorithms, BPSOTVAC and CBPSOTVAC, are superior over the other methods according to its success rate, mean absolute deviation, mean absolute percentage error, least error, and standard deviation.

1. Introduction

The multidimensional knapsack problem (MKP) is a generalization of the 0–1 knapsack problem and a special case of 0–1 integer programming [1]. MKP has been extensively discussed because of its theoretical importance and wide range of applications. Numerous real-world applications are modeled as knapsack problems, such as capital budgeting [2], project selection [3], and loading problems [4], in a variety of areas [5,6].

MKP can be mathematically formulated as follows:

$$\max z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \quad i = 1, 2, \ldots, m, \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n, \tag{3}$$

* Corresponding author. Tel.: +886 37331862.
E-mail address: d927805@oz.nthu.edu.tw (M. Chih).

0307-904X/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.apm.2013.08.009

where n is the number of items and m is the number of knapsack constraints with the capacity b_i for $i = 1,2,\ldots,m$. Each item j requires a_{ij} units of resource consumption in the ith knapsack and yields c_j units of profit upon inclusion. The goal is to find a subset of items that yields the maximum profit without exceeding the resource capacities. All entries are naturally nonnegative.

MKP is a combinatorial optimization problem that belongs to the class of NP-hard problems [7]. Similar to many other combinatorial optimization problems, MKP was intensively investigated in metaheuristics during the first decade of the 21st century. Several modern heuristic algorithms have been developed to solve MKP, namely, simulated annealing [8], tabu search [9], genetic algorithms [10], ant algorithms [11], and particle swarm optimization (PSO) [12].

The PSO technique is a promising alternative to other conventional heuristic approaches for solving problems on combinatorial optimization. This population-based optimization algorithm was designed by Kennedy and Eberhart [13] to optimize various continuous nonlinear functions. PSO can be considered a random search algorithm that simulates the natural evolutionary process and performs excellently in solving difficult and complex optimization problems. The PSO prototype is restricted to the real number space. However, numerous optimization problems are set in a space that features discrete or qualitative distinctions between variables. To address this need, Kennedy and Eberhart [14] developed a discrete version of the algorithm.

PSO has been successfully applied to a wide range of areas such as the traveling salesman problem [15], inspection policy [16], water supply system design [17], economic statistical control chart design [18], reliability networks [19], multi-modal problems [20], feature selection [21], multi-objective optimization [22], and supplier selection [23].

In this paper, two algorithms based on binary PSO (BPSO) with time-varying acceleration coefficients (TVAC) are presented. The proposed algorithms are compared with the existing classical PSO algorithms for solving MKPs using the available test data sets in the OR-Library [24].

The rest of the paper is organized as follows: Section 2 provides the basic concepts of classic and BPSO algorithms. The two proposed PSO algorithms with TVAC for solving MKP are presented in Section 3. Section 4 summarizes the simulation and evaluation results of the proposed algorithms on the test data sets. Finally, concluding remarks are made in Section 5.

2. Background

PSO is an efficient population-based optimization technique [13] that is based on the metaphors of social interaction and communication (e.g., fish schooling and bird flocking). This technique uses the collaboration among simple search agents called particles in a population to find the optima in a particular search space. The technique has been demonstrated to be effective in optimizing difficult multidimensional problems in different fields [25,26].

In PSO, the particles are potential solutions that fly through the problem space by following the current optimum particles. All of the particles have fitness values that are evaluated by the fitness function to be optimized. The particle velocities direct the flight of the particles.

PSO combines local and global searches for high search efficiency. Its algorithm is initialized in a population of random particles (with random positions and velocities inside the problem space). PSO subsequently searches for optima by updating the successive generations. Each particle is updated during each iteration according to the two "best" values. The first value, called *pbest*, is the best solution (fitness) that has been achieved by the particle. The other "best" value that is tracked by PSO is the current best value, which is obtained by any particle in the population. This best value is the global best, which is designated as *gbest*. When a particle considers parts of the population as its topological neighbors, the best value is a local best, namely, *lbest*. After the two best values are identified, the particle updates its velocity and position according to the following equations in continuous PSO:

$$v_{ij}^{t} = w \cdot v_{ij}^{t-1} + c_1 \cdot Rand() \cdot (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 \cdot Rand() \cdot (g_{ij}^{t-1} - x_{ij}^{t-1}), \tag{4}$$

$$\text{and} \quad x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}, \tag{5}$$

where v_{ij}^{t} is the velocity of ith particle at jth dimension (decision variable) in tth iteration; x_{ij}^{t-1} is the current particle (solution); p_{ij}^{t-1} and g_{ij}^{t-1} are *pbest* and *gbest*, respectively; and *Rand*() is a random number between [0,1]. c_1 and c_2 are learning factors, and the value of $(c_1 + c_2)$ is usually limited to 4 [25]. Several parameters should be determined to apply the PSO technique. These parameters include the number of population members (m), cognition learning factor (c_1), social learning factor (c_2), inertia weight (w), and the number of iterations or CPU time.

The complete computational procedure of the PSO algorithm is summarized below.

STEP 1. Initialize
Initialize the parameters and the population with random positions and velocities.
STEP 2. Evaluation
Evaluate the fitness value (the desired objective function) for each particle.
STEP 3. Find the *pbest*
If the fitness value of particle i is better than its best fitness value (*pbest*) in history, then set the current fitness value as the new *pbest* of particle i.

STEP 4. Find the *gbest*

If any *pbest* is updated and is better than the current *gbest*, then set *gbest* as the current value.

STEP 5. Update the velocity and position

Update the velocity and move to the next position according to Eqs. (4) and (5).

STEP 6. Stopping criterion

If the desired number of iterations or CPU time are achieved, then stop; otherwise go back to STEP 2.

Given that PSO was initially introduced to optimize various continuous nonlinear functions, the major obstacle of successfully applying PSO in the real world is its continuous nature. To resolve this drawback, Kennedy and Eberhart [14] developed a discrete binary version of PSO called BPSO. In BPSO, the particle is characterized by a binary solution presentation, and the velocity must be transformed towards the change in probability for each binary dimension to take a value of one.

BPSO updates the velocity based on Eq. (4), although the particles are represented by binary variables and w is not considered. Furthermore, the velocity is constrained to the interval [0,1] using the following sigmoid transformation:

$$S(v_{ij}^t) = \frac{1}{1+\exp(-v_{ij}^t)}, \tag{6}$$

where $S(v_{ij}^t)$ denotes the probability of bit v_{ij}^t taking the value of 1. To prevent $S(v_{ij}^t)$ from approaching 0 or 1, a constant $V_{\max}$ is used to limit the range of v_{ij}^t. $V_{\max}$ is often set to 4 in practice [12], such that $v_{ij}^t \in [-4,4]$ and

$$v_{ij}^t = \begin{cases} V_{\max}, & \text{if } v_{ij}^t > V_{\max}; \\ -V_{\max}, & \text{if } -v_{ij}^t < -V_{\max}; \\ v_{ij}^t & \text{otherwise.} \end{cases} \tag{7}$$

Eq. (6) implies that each particle at each time step changes its current position according to Eq. (8), instead of Eq. (5). Consequently, the following equation is derived:

$$x_{ij}^t = \begin{cases} 1, & \text{if } Rand() < S(v_{ij}^t), \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

3. Methodology

This section discusses the methodologies for modifying BPSO to solve MKP.

3.1. Penalty function

MKP is a constrained combinatorial optimization problem. The use of a penalty function is the most common PSO approach for solving constrained optimization problems [27]. The constrained problem is transformed to an unconstrained one by penalizing the constraints and building a single objective function that, in turn, is minimized using an unconstrained optimization algorithm.

Penalty functions are classified into two categories, namely, stationary and non-stationary. Stationary penalty functions use fixed penalty values throughout minimization. By contrast, the penalty values are dynamically modified in non-stationary penalty functions. In a previous study [28], the results that were obtained using non-stationary penalty functions were almost always superior to those obtained with stationary functions. Thus, the implied penalty costs in the present study can be defined as:

$$z' = \frac{\sum_{j=1}^n c_j x_j}{\sum_{j=1}^n a_{ij} x_j}, \quad \text{if} \quad \sum_{j=1}^n a_{ij} x_j > b_i, \forall i,j. \tag{9}$$

In the equation above, $z'(\cdot)$ is the penalty objective function, which directs the particles towards both the feasible and infeasible regions near the border of the feasible area. This step prevents the search from going too far into the infeasible region. Unler and Murat [29] stated that the minimization algorithms are usually trapped in the local minima if the penalty values are high. However, if the penalty values are low, the feasible optimal solutions can hardly be detected.

3.2. Time-varying acceleration coefficients (TVAC)

In population-based optimization methods, proper control of global and local exploration is crucial for the efficient identification of the optimum solution. Shi and Eberhart [30] found a significant improvement in the performance of the PSO method with a linearly varying inertia weight over generations. The mathematical representation of this concept is given by Eq. (10).

$$w = (w_{\max} - w_{\min})\frac{t_{\max} - t}{t_{\max}} + w_{\min} \tag{10}$$

Ratnweera et al. [26] further introduced the concept of TVAC in PSO with learning factors. TVAC aims to enhance the global search in the early stages of optimization and encourages the particles to converge toward the global optima at the end of the search. This concept can be mathematically represented as

$$c_1 = (c_{1f} - c_{1i})\frac{t-1}{t_{\max}} + c_{1i}, \tag{11}$$

$$c_2 = (c_{2f} - c_{2i})\frac{t-1}{t_{\max}} + c_{2i}. \tag{12}$$

In this study, the time-varying inertia weight (w) and time-varying learning factors ($c_1 c_2$) of TVAC are introduced to the proposed PSO algorithms. The velocity updating equation can be expressed as

$$\begin{aligned} v_{ij}^t = w \cdot v_{ij}^{t-1} &+ \{(c_{1f} - c_{1i})\tfrac{t-1}{t_{\max}} + c_{1i}\} \cdot Rand() \cdot (p_{ij}^{t-1} - x_{ij}^{t-1}) \\ &+ \{(c_{2f} - c_{2i})\tfrac{t-1}{t_{\max}} + c_{2i}\} \cdot Rand() \cdot (g_{ij}^{t-1} - x_{ij}^{t-1}) \end{aligned} \tag{13}$$

Shi and Eberhart [30] observed that the optimal solution can be improved by varying the value of w from 0.9 at the beginning of the search to 0.4 at the end of the search for most problems. Ratnweera et al. [26] highlighted that an improved optimum solution for most of the benchmarks was observed when changing c_1 from 2.5 to 0.5 and c_2 from 0.5 to 2.5 over the full range of the search.

3.3. Variable mapping

In PSO, the position update equation is an explicit function of the previous velocity and previous position. Each swarm is manipulated according to Eqs. (4) and (5). Instead of using the sigmoid transformation in classic BPSO, we introduced the position update equation that was previously reported by Bansal and Deep [12]. The concept is obtained from the position update equation for the direct continuous optimization. If the velocity bounds are $-V_{\max}$ and $V_{\max}$, then the term $x_{ij}^{t-1} + v_{ij}^t$ is bounded between $(0 - V_{\max} = -V_{\max})$ and $(1 + V_{\max})$ because x_{ij}^t in Eq. (5) must have a value of 0 or 1. Therefore, the position update equation can be represented as:

$$x_{ij}^t = \begin{cases} 1, \text{ if } U(-V_{\max}, 1 + V_{\max}) < x_{ij}^{t-1} + v_{ij}^t; \\ 0 \qquad \text{otherwise.} \end{cases} \tag{14}$$

$U(0,1) = \frac{U(-V_{\max}, 1+V_{\max})}{(1+2V_{\max})}$. Eq. (14) can be rewritten as:

$$x_{ij}^t = \begin{cases} 1, \text{ if } U(0,1) < \frac{x_{ij}^{t-1} + v_{ij}^t + V_{\max}}{1+2V_{\max}}, \\ 0 \qquad \text{otherwise.} \end{cases} \tag{15}$$

where $U(0,1)$ is a random number between [0,1], $U(-V_{\max}, 1 + V_{\max})$ is a random number between $[-V_{\max}, 1 + V_{\max}]$.

3.4. Chaos

Chaos is a non-linear system with deterministic dynamic behavior. Chaos has ergodicity, stochastic, and regularity properties and is sensitive to its initial conditions and parameters. This system is often embedded into the velocity updating equation of PSO for w, that is, a chaotic map is used to control the value of the parameters in the velocity equation, Eq. (4). A commonly used chaotic map is the logistic map [21,31,32].

In this study, the logistic map was applied instead of the *Rand*() function to handle the "cognition effect" and "social effect" in velocity updating. Jiang and Etorre [33] indicated that the *Rand*() function cannot completely ensure the ergodicity of optimization in the phase space because it is absolutely random in the traditional PSO. Therefore, chaotic logistic map with certainty, ergodicity, and stochastic property was introduced to improve the global convergence. The mathematical expression of this chaotic map is

$$y(t) = 4.0 \times y(t-1) \times [1 - y(t-1)], y(t) \in (0,1) \tag{16}$$

where $y(t)$ is distributed in the range (0,1), such that the initial $y(0) \in (0,\ 1)$, and $y(0) \notin (0, 0.25, 0.5, 0.75)$.

4. The proposed PSO algorithms

Two novel PSO algorithms that were based on previous methodologies are proposed in this section to solve MKP. In the proposed algorithms, the initial values for all of the particles are generated through a random method. Each particle is then evaluated to find the corresponding fitness value. If the knapsack capacity does not satisfy the knapsack constraint, the penalty function is added to the fitness function, as in Eq. (9). The *pbest* and the *gbest* of each particle are updated separately,

along with the velocity and the position of each particle. The position update equation is utilized in Eqs. (15) and (16). Finally, PSO stops if the stopping criterion is satisfied. Otherwise, the algorithm proceeds to the next generation.

4.1. Binary particle swarm optimization with time-varying acceleration coefficients (BPSOTVAC)

In this subsection, we introduce BPSO with TVAC (BPSOTVAC). The pseudo-code of the proposed PSO algorithm is likewise provided.

For each particle
  Initialize particle
  $x_{ij} = \begin{cases} 0, & Rand() < 0.5; \\ 1, & \text{otherwise.} \end{cases}$ where $x_{ij} \in \{0,1\}$
END
Do
  For each particle
    Calculate fitness value
    If the particle is not a feasible solution
    compute the penalty cost according to
    $z' = \frac{\sum_{j=1}^{n} c_j x_j}{\sum_{j=1}^{n} a_{ij} x_j}, if\ \sum_{j=1}^{n} a_{ij} x_j > b_i, \forall i, j.$
    If the fitness value is better than the best fitness value (*pbest*) in history
    set current value as the new *pbest*
End
    Choose the particle with the best fitness value of all the particles as the *gbest*
  For each particle
    Calculate particle velocity according to

$$v_{ij}^{t} = w \cdot v_{ij}^{t-1} + \{(c_{1f} - c_{1i}) \tfrac{t-1}{t_{\max}} + c_{1i}\} \cdot Rand() \cdot (p_{ij}^{t-1} - x_{ij}^{t-1}) + \{(c_{2f} - c_{2i}) \tfrac{t-1}{t_{\max}} + c_{2i}\} \cdot Rand() \cdot (g_{ij}^{t-1} - x_{ij}^{t-1})$$

    Update particle position
    $x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}$
    $x_{ij}^{t} = \begin{cases} 1, & \text{if } U(0,1) < \frac{x_{ij}^{t} + V_{\max}}{1 + 2V_{\max}}; \\ 0, & \text{otherwise.} \end{cases}$
  End
When maximum iterations are not attained

4.2. Chaotic binary particle swarm optimization with time-varying acceleration coefficients (CBPSOTVAC)

In this subsection, the chaos idea is introduced into BPSO with TVAC to develop a second novel PSO algorithm. The complete computational procedure of the proposed PSO algorithm is summarized below.

For each particle
  Initialize particle
  $x_{ij} = \begin{cases} 0, & Rand() < 0.5; \\ 1, & \text{otherwise.} \end{cases}$ where $x_{ij} \in \{0,1\}$
END
Do
  For each particle
    Calculate fitness value
    If the particle is not a feasible solution compute the penalty cost according to
    $z' = \frac{\sum_{j=1}^{n} c_j x_j}{\sum_{j=1}^{n} a_{ij} x_j}, if \sum_{j=1}^{n} a_{ij} x_j > b_i, \forall i, j.$
    If the fitness value is better than the best fitness value (*pbest*) in historyset current value as the new *pbest*
  End
    Choose the particle with the best fitness value of all the particles as the *gbest*

For each particle

Calculate particle velocity according to

$$v_{ij}^{t} = w \cdot v_{ij}^{t-1} + \{(c_{1f} - c_{1i})\frac{t-1}{t_{\max}} + c_{1i}\} \cdot Rand() \cdot (p_{ij}^{t-1} - x_{ij}^{t-1}) + \{(c_{2f} - c_{2i})\frac{t-1}{t_{\max}} + c_{2i}\} \cdot Rand() \cdot (g_{ij}^{t-1} - x_{ij}^{t-1})$$

Where $y_i(t-1) = 4.0 \times y_i(t-2) \times (1 - y_i(t-2)), y_i(t-1) \in (0,1)$

Update particle position

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}$$

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

End

When maximum iterations are not attained

5. Simulation and Evaluation

This section attempts to apply benchmark datasets to model evaluation. The proposed algorithms, BPSOTVAC and CBPSOTVAC, were tested using several groups of MKP benchmarks selected from OR-Library [24]. The first group contains 10 benchmark problems and corresponds to "SENTO" [34] and "WEING" [35]. The second group contains 30 benchmark problems and corresponds to "WEISH" [4]. The third group contains eight benchmark problems and corresponds to "HP" and "PB" [36]. The fourth group contains eight benchmark problems and corresponds to "cb5-10-250." The fifth group contains 30 benchmark problems and corresponds to "cb3-5-500.". The sixth group contains 30 benchmark problems and corresponds

Table 1
Simulation results of SENTO and WEING testing problems.

Problem	#Knapsacks	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Sento1	30	60	MBPSO	0.16(0.04)	44.81(4.32)	0.0058(0.0006)	229	43.23
			CBPSO1	0(0)	198.29(4.99)	0.026(0.001)	302	49.92
			BPSOTVAC	0.57(0.05)	8.74(1.15)	0.0011(0.0001)	34	11.52
			CBPSOTVAC	0.39(0.05)	136.28(35.78)	0.021(0.007)	3146	357.78
Sento2	30	60	MBPSO	0.03(0.02)	24.85(1.88)	0.0029(0.0002)	81	18.8
			CBPSO1	0(0)	103.32(2.58)	0.012(0.0003)	150	25.78
			BPSOTVAC	0.27(0.04)	9.42(0.70)	0.001(0.0001)	38	7.04
			CBPSOTVAC	0.2(0.04)	53.53(10.10)	0.0063(0.0012)	633	101.03
Weing1	2	28	MBPSO	0.82(0.04)	110.79(25.04)	0.0008(0.0002)	801	250.43
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.92(0.03)	51.25(28.20)	0.0004(0.0002)	1961	281.98
Weing2	2	28	MBPSO	0.65(0.05)	117.45(31.41)	0.0009(0.0002)	1700	314.08
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.88(0.03)	123.19(54.55)	0.0009(0.0004)	3341	545.5
Weing3	2	28	MBPSO	0.11(0.03)	1053.2(87.7)	0.0112(0.0009)	3500	876.78
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	0.92(0.03)	6.42(2.55)	0.00007(0.00003)	160	25.53
			CBPSOTVAC	0.75(0.04)	173.07(67.24)	0.0019(0.0007)	3789	672.42
Weing4	2	28	MBPSO	0.76(0.04)	570.6(127.1)	0.0049(0.0011)	4001	1270.8
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.97(0.02)	42.83(37.86)	0.0004(0.0003)	3774	378.58
Weing5	2	28	MBPSO	0.52(0.05)	1629.21(192.36)	0.017(0.002)	4778	1923.5
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.94(0.02)	85.62(57.28)	0.0009(0.0006)	4728	572.82
Weing6	2	28	MBPSO	0.36(0.05)	310.2(32.2)	0.0023(0.0002)	1340	322.4
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	0.97(0.02)	11.7(6.7)	0.00009(0.00005)	390	66.86
			CBPSOTVAC	0.87(0.03)	91.71(34.35)	0.0007(0.0003)	2460	343.45
Weing7	2	105	MBPSO	0.02(0.01)	660.86(113.06)	0.0006(0.0001)	6111	1130.6
			CBPSO1	0(0)	32690.6(500.2)	0.0308(0.0005)	42425	5002
			BPSOTVAC	0(0)	281.23(38.37)	0.00026(0.00004)	2069	383.74
			CBPSOTVAC	0(0)	11272.9(3002.1)	0.011(0.003)	154486	30020
Weing8	2	105	MBPSO	0.03(0.02)	5824.7(470.4)	0.0095(0.0008)	44731	4704.3
			CBPSO1	0(0)	118166(1599)	0.234(0.004)	160402	15988
			BPSOTVAC	0.35(0.05)	1872.44(200.09)	0.0030(0.0003)	6463	2000.9
			CBPSOTVAC	0.20(0.04)	27128.4(7516.9)	13.7(13.6)	623862	75169

Table 2
Simulation results of WEISH testing problems.

Problem	#Knapsacks	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish1	5	30	MBPSO	0.82(0.04)	10.9(2.6)	0.0024(0.0006)	114	26.34
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.94(0.02)	5.45(3.28)	0.0012(0.0008)	248	32.81
Weish2	5	30	MBPSO	0.55(0.05)	8.39(1.8)	0.0018(0.0004)	123	18.01
			CBPSO1	0.79(0.04)	1.05(0.2)	0.00023(0.00005)	5	2.04
			BPSOTVAC	0.64(0.05)	1.8(0.2)	0.00040(0.00005)	5	2.41
			CBPSOTVAC	0.66(0.05)	4.12(2.31)	0.0009(0.0005)	231	23.12
Weish3	5	30	MBPSO	0.63(0.05)	20.54(3.49)	0.0051(0.0009)	141	34.98
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	0.99(0.01)	0.63(0.63)	0.00015(0.00015)	63	6.3
			CBPSOTVAC	0.95(0.02)	9.21(5.27)	0.0024(0.0014)	394	52.69
Weish4	5	30	MBPSO	0.96(0.02)	1.76(0.89)	0.0004(0.0002)	56	8.99
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.99(0.01)	8.59(8.59)	0.0023(0.0023)	859	85.9
Weish5	5	30	MBPSO	0.99(0.01)	0.54(0.54)	0.00012(0.00012)	54	5.4
			CBPSO1	1(0)	0(0)	0(0)	0	0
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.98(0.01)	8.11(7.45)	0.0021(0.0019)	742	74.45
Weish6	5	40	MBPSO	0.32(0.05)	15.36(1.44)	0.0028(0.0003)	56	14.39
			CBPSO1	0.65(0.05)	5.47(0.79)	0.00098(0.00014)	34	7.92
			BPSOTVAC	0.59(0.05)	6.68(0.82)	0.00121(0.00015)	18	8.19
			CBPSOTVAC	0.53(0.05)	23.21(7.93)	0.0044(0.0016)	518	79.28
Weish7	5	40	MBPSO	0.64(0.05)	10.2(1.89)	0.0018(0.0003)	122	18.92
			CBPSO1	0.83(0.04)	3.45(0.78)	0.0006(0.0001)	25	7.79
			BPSOTVAC	0.96(0.02)	0.7(0.34)	0.00013(0.00006)	18	3.45
			CBPSOTVAC	0.78(0.04)	19.17(7.19)	0.0036(0.0014)	511	71.95
Weish8	5	40	MBPSO	0.44(0.05)	7.24(1.31)	0.0013(0.0002)	72	13.07
			CBPSO1	0.64(0.05)	0.72(0.09)	0.00013(0.00002)	2	0.96
			BPSOTVAC	0.79(0.04)	0.42(0.08)	0.00008(0.00001)	2	0.82
			CBPSOTVAC	0.68(0.05)	8.84(4.28)	0.0016(0.0008)	418	42.81
Weish9	5	40	MBPSO	0.78(0.04)	10.61(2.56)	0.0021(0.0005)	200	25.65
			CBPSO1	0.96(0.02)	1.36(0.67)	0.0003(0.0001)	34	6.69
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.85(0.04)	13.01(6.57)	0.0027(0.0014)	641	65.7
Weish10	5	50	MBPSO	0.56(0.05)	10.84(2.22)	0.0017(0.0004)	83	22.17
			CBPSO1	0.07(0.03)	42.57(3.37)	0.0068(0.0005)	141	33.65
			BPSOTVAC	0.91(0.03)	1.43(0.96)	0.0002(0.0002)	68	9.56
			CBPSOTVAC	0.67(0.05)	57.16(18.86)	0.0102(0.0037)	1394	188.63
Weish11	5	50	MBPSO	0.4(0.05)	29.48(4.39)	0.0053(0.0008)	167	43.95
			CBPSO1	0.05(0.02)	79.9(4.7)	0.0144(0.0008)	191	47.06
			BPSOTVAC	0.88(0.03)	7.42(2.57)	0.0013(0.0005)	113	25.72
			CBPSOTVAC	0.62(0.05)	110.85(40.3)	0.028(0.011)	2245	403.03
Weish12	5	50	MBPSO	0.65(0.05)	16.35(3.57)	0.0026(0.0006)	226	35.68
			CBPSO1	0.09(0.03)	57.3(4.24)	0.0092(0.0007)	191	42.35
			BPSOTVAC	0.89(0.03)	0.29(0.19)	0.00005(0.00003)	19	1.91
			CBPSOTVAC	0.71(0.04)	107.5(30.4)	0.020(0.006)	1497	304.43
Weish13	5	50	MBPSO	0.87(0.03)	8.47(2.52)	0.0014(0.0004)	155	25.19
			CBPSO1	0.15(0.04)	59.33(4.10)	0.0098(0.0007)	159	40.97
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.85(0.04)	38.62(18.00)	0.0075(0.0039)	1725	180.04
Weish14	5	60	MBPSO	0.66(0.05)	16.09(2.59)	0.0023(0.0004)	100	25.95
			CBPSO	0(0)	210.47(6.68)	0.031(0.001)	347	66.79
			BPSOTVAC	0.98(0.01)	0.62(0.44)	0.00089(0.00006)	31	4.36
			CBPSOTVAC	0.79(0.04)	116.23(36.46)	0.021(0.007)	2127	364.66
Weish15	5	60	MBPSO	0.72(0.05)	10.55(1.86)	0.0014(0.0002)	70	18.64
			CBPSO1	0(0)	193.51(5.99)	0.0266(0.0008)	359	59.99
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.8(0.04)	161.45(55.43)	0.030(0.011)	2978	554.35
Weish16	5	60	MBPSO	0.44(0.05)	7.66(1.75)	0.0011(0.0002)	87	17.49
			CBPSO1	0(0)	139.24(4.71)	0.0195(0.0007)	259	47.07
			BPSOTVAC	0.54(0.05)	1.16(0.17)	0.00016(0.00002)	8	1.71
			CBPSOTVAC	0.43(0.05)	143.29(36.73)	0.023(0.006)	1931	367.29
Weish17	5	60	MBPSO	0.56(0.05)	5.76(0.74)	0.0007(0.0001)	41	7.38
			CBPSO1	0(0)	91.8(3.28)	0.0107(0.0004)	170	32.79
			BPSOTVAC	1(0)	0(0)	0(0)	0	0
			CBPSOTVAC	0.72(0.05)	85.29(22.72)	0.011(0.003)	1035	227.16

Table 2 (*continued*)

Problem	#Knapsacks	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish18	5	70	MBPSO	0.38(0.05)	14.65(1.84)	0.0015(0.0002)	94	18.4
			CBPSO1	0(0)	259.34(5.4)	0.0278(0.0006)	367	54.01
			BPSOTVAC	0.75(0.04)	2.79(0.53)	0.00029(0.00005)	15	5.25
			CBPSOTVAC	0.53(0.05)	99.14(27.55)	0.011(0.003)	1595	275.53
Weish19	5	70	MBPSO	0.55(0.05)	20.83(3.36)	0.0027(0.0004)	149	33.67
			CBPSO1	0(0)	429.39(8.36)	0.059(0.001)	615	83.65
			BPSOTVAC	0.65(0.05)	4.9(0.71)	0.0006(0.0001)	35	7.13
			CBPSOTVAC	0.62(0.05)	169.45(48.94)	0.028(0.009)	3060	489.37
Weish20	5	70	MBPSO	0.53(0.05)	10.81(1.59)	0.0011(0.0002)	69	15.99
			CBPSO1	0(0)	336.41(8.94)	0.037(0.001)	528	89.41
			BPSOTVAC	0.78(0.04)	3.78(0.75)	0.0004(0.00008)	20	7.53
			CBPSOTVAC	0.69(0.05)	117.89(41.07)	0.015(0.005)	2482	410.74
Weish21	5	70	MBPSO	0.61(0.05)	17.85(2.49)	0.0019(0.0003)	88	24.97
			CBPSO1	0(0)	347.65(8.44)	0.039(0.001)	499	84.42
			BPSOTVAC	0.74(0.04)	6.06(1.04)	0.0007(0.0001)	24	10.41
			CBPSOTVAC	0.67(0.05)	125.78(37.84)	0.016(0.005)	2574	378.38
Weish22	5	80	MBPSO	0.33(0.05)	29.73(3.15)	0.0033(0.0004)	112	31.55
			CBPSO	0(0)	674.21(9.47)	0.082(0.001)	935	94.69
			BPSOTVAC	0.16(0.04)	15.12(0.66)	0.00169(0.00007)	18	6.63
			CBPSOTVAC	0.17(0.04)	172.8(48.67)	0.024(0.007)	3063	486.71
Weish23	5	80	MBPSO	0.24(0.04)	29.65(3.54)	0.0036(0.0004)	126	35.43
			CBPSO1	0(0)	670.43(11.96)	0.087(0.002)	902	119.64
			BPSOTVAC	0.85(0.04)	1.11(0.51)	0.00013(0.00006)	36	5.11
			CBPSOTVAC	0.58(0.05)	179(43.72)	0.026(0.007)	3114	437.23
Weish24	5	80	MBPSO	0.27(0.04)	17.48(1.81)	0.0017(0.0002)	70	18.09
			CBPSO1	0(0)	367.36(5.69)	0.0373(0.0006)	499	56.96
			BPSOTVAC	0.7(0.05)	3.04(0.64)	0.00029(0.00006)	31	6.44
			CBPSOTVAC	0.55(0.05)	113.72(29.58)	0.012(0.003)	1841	295.79
Weish25	5	80	MBPSO	0.29(0.05)	15.13(1.34)	0.0015(0.0001)	61	13.39
			CBPSO1	0(0)	449.77(8.62)	0.0475(0.0009)	648	86.21
			BPSOTVAC	0.49(0.05)	4.54(0.71)	0.00045(0.00007)	24	7.09
			CBPSOTVAC	0.32(0.05)	112.43(36.19)	0.013(0.004)	2321	361.88
Weish26	5	90	MBPSO	0.31(0.05)	27.32(2.43)	0.0028(0.0002)	114	24.27
			CBPSO1	0(0)	895.39(12.64)	0.103(0.001)	1122	126.39
			BPSOTVAC	0.36(0.05)	11.44(1.28)	0.0012(0.0001)	47	12.81
			CBPSOTVAC	0.28(0.04)	270.13(71.08)	0.04(0.01)	4084	710.77
Weish27	5	90	MBPSO	0.65(0.05)	23.7(4.86)	0.0024(0.0005)	203	48.62
			CBPSO	0(0)	967.43(12.03)	0.109(0.001)	1205	120.34
			BPSOTVAC	0.99(0.01)	0.39(0.39)	0.00004(0.00004)	39	3.9
			CBPSOTVAC	0.83(0.04)	211.46(64.04)	0.028(0.009)	3915	640.43
Weish28	5	90	MBPSO	0.64(0.05)	15.21(2.67)	0.0016(0.0003)	144	26.72
			CBPSO1	0(0)	980.45(12.23)	0.115(0.002)	1266	122.32
			BPSOTVAC	0.87(0.03)	2.99(0.78)	0.00031(0.00008)	23	7.77
			CBPSOTVAC	0.62(0.05)	368.74(88.73)	0.06(0.02)	4387	887.33
Weish29	5	90	MBPSO	0.46(0.05)	26.73(3.47)	0.0029(0.0004)	154	34.74
			CBPSO1	0(0)	981.44(10.86)	0.117(0.001)	1180	108.56
			BPSOTVAC	0.86(0.03)	3.19(1.01)	0.0003(0.0001)	79	10.09
			CBPSOTVAC	0.48(0.05)	384.5(85.5)	0.057(0.016)	4891	854.5
Weish30	5	90	MBPSO	0.38(0.05)	11.6(1.45)	0.001(0.0001)	57	14.48
			CBPSO1	0(0)	548.1(8.76)	0.0516(0.0009)	760	87.58
			BPSOTVAC	0.87(0.03)	0.52(0.14)	0.00005(0.00001)	4	1.35
			CBPSOTVAC	0.63(0.05)	203.79(49.18)	0.021(0.005)	2836	491.81

to "cb6-10-500.". The first three groups are low-dimensional knapsack problems, wherein the number of decision variables (# items) ranges from 20 to 105. The other groups are high-dimensional knapsack problems, wherein the number of decision variables (# items) ranges from 250 to 500. These benchmark instances were likewise solved using two existing PSO algorithms: the modified binary PSO (MBPSO) [12] and the CBPSO1 [21].

Previous methods have considered MKP benchmark instances where the optimal solution is known. Thus, a comparison among MBPSO, CBPSO1, BPSOTVAC, and CBPSOTVAC was performed based on five performance measures: success rate (SR), mean absolute deviation (MAD), mean absolute percentage error (MAPE), least error (LE), and standard deviation (SD). SR refers to the number of runs out of all of the executions that produce the optimum solution within the termination criterion. MAD is the average of the absolute difference between the simulation data and the given optimal solution. MAPE is obtained by dividing MAD by the corresponding optimal solution and usually expresses the accuracy as a percentage. LE is the least error obtained by the minimum of absolute difference between the optimum solution and final solution. SD is the standard deviation of final solutions over runs.

Table 3
Simulation results of HP and PB testing problems.

Problem	#Knapsacks	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Hp1	4	28	MBPSO	0.1(0.03)	37.52(2.55)	0.0112(0.0008)	109	25.52
			CBPSO1	0.64(0.05)	5.5(0.78)	0.0016(0.0002)	30	7.84
			BPSOTVAC	0.38(0.05)	11.44(1.07)	0.0034(0.0003)	33	10.69
			CBPSOTVAC	0.29(0.04)	14.1(1.37)	0.0042(0.0004)	72	13.69
Hp2	4	35	MBPSO	0.11(0.03)	46.22(3.92)	0.015(0.001)	136	39.15
			CBPSO1	0.73(0.04)	4.58(0.76)	0.0014(0.0002)	19	7.65
			BPSOTVAC	0.67(0.05)	6.51(1.39)	0.0021(0.0004)	116	13.95
			CBPSOTVAC	0.59(0.05)	12.39(2.13)	0.0039(0.0007)	114	21.35
Pb1	4	27	MBPSO	0.11(0.03)	32.62(2.43)	0.0107(0.0008)	104	24.32
			CBPSO1	0.61(0.05)	5.93(0.79)	0.0019(0.0003)	30	7.92
			BPSOTVAC	0.46(0.05)	9(0.94)	0.0029(0.0003)	30	9.44
			CBPSOTVAC	0.4(0.05)	10.26(1.05)	0.0033(0.0003)	61	10.52
Pb2	4	34	MBPSO	0.16(0.04)	44.69(3.93)	0.014(0.001)	154	39.31
			CBPSO1	0.8(0.04)	4.28(1.15)	0.00135(0.00037)	95	11.49
			BPSOTVAC	0.73(0.04)	4.5(0.77)	0.00142(0.00024)	31	7.68
			CBPSOTVAC	0.51(0.05)	14.45(1.87)	0.0046(0.0006)	87	18.73
Pb4	2	29	MBPSO	0.27(0.04)	2639.8(180.39)	0.029(0.002)	4751	1803
			CBPSO1	0.99(0.01)	2.03(2.03)	0.00002(0.00002)	203	20.3
			BPSOTVAC	0.91(0.03)	228.1(79.71)	0.0025(0.0009)	3233	797.1
			CBPSOTVAC	0.84(0.04)	304.33(87.51)	0.0033(0.0009)	3498	875.1
Pb5	10	20	MBPSO	0.08(0.03)	49.42(2.44)	0.024(0.001)	117	24.36
			CBPSO	0.99(0.01)	0.17(0.17)	0.00008(0.00008)	17	1.7
			BPSOTVAC	0.84(0.04)	2.72(0.63)	0.0013(0.0003)	17	6.26
			CBPSOTVAC	0.8(0.04)	3.4(0.68)	0.0016(0.0003)	17	6.83
Pb6	30	40	MBPSO	0.28(0.04)	27.36(2.91)	0.038(0.004)	147	29.12
			CBPSO1	0.55(0.05)	8.47(1.09)	0.011(0.001)	34	10.99
			BPSOTVAC	0.5(0.05)	8.7(0.99)	0.012(0.001)	31	9.99
			CBPSOTVAC	0.54(0.05)	17.74(4.02)	0.028(0.008)	351	40.17
Pb7	30	37	MBPSO	0.05(0.02)	19.89(1.63)	0.019(0.002)	82	16.29
			CBPSO1	0.41(0.05)	5.64(0.59)	0.0055(0.0006)	22	5.88
			BPSOTVAC	0.47(0.05)	5.43(0.57)	0.0053(0.0005)	20	5.71
			CBPSOTVAC	0.4(0.05)	13.05(2.42)	0.013(0.003)	126	24.25

Table 4
Simulation results of cb5-10-250 testing problems (10 knapsacks and 250 items).

Problem	Algorithm	SR	MAD	MAPE	LE	SD
cb5-10-250-1	MBPSO	0(0)	2860(121)	0.051(0.002)	4256	663.16
	CBPSO1	0(0)	12766(91)	0.275(0.002)	13691	499.31
	BPSOTVAC	0(0)	626(23)	0.0107(0.0004)	921	127.73
	CBPSOTVAC	0(0)	529(25)	0.0091(0.0004)	849	139.42
cb5-10-250-2	MBPSO	0(0)	3239(120)	0.058(0.002)	5266	658.97
	CBPSO1	0(0)	12177(153)	0.262(0.004)	13926	840.44
	BPSOTVAC	0(0)	607(27)	0.0104(0.0005)	978	147.88
	CBPSOTVAC	0(0)	631(29)	0.0109(0.0005)	1108	158.11
cb5-10-250-3	MBPSO	0(0)	3193(99)	0.058(0.002)	3942	541.58
	CBPSO1	0(0)	12641(141)	0.278(0.004)	14281	772.68
	BPSOTVAC	0(0)	563(27)	0.0098(0.0005)	812	150.47
	CBPSOTVAC	0(0)	542(29)	0.0094(0.0005)	973	157.59
cb5-10-250-4	MBPSO	0(0)	2860(115)	0.049(0.002)	4273	628.55
	CBPSO1	0(0)	13184(173)	0.276(0.005)	15175	949.44
	BPSOTVAC	0(0)	661(29)	0.0109(0.0005)	1008	159.44
	CBPSOTVAC	0(0)	582(23)	0.0096(0.0004)	841	128.81
cb5-10-250-5	MBPSO	0(0)	2727(101)	0.049(0.002)	3780	555.86
	CBPSO1	0(0)	12072(128)	0.263(0.003)	13210	701.79
	BPSOTVAC	0(0)	666(31)	0.0116(0.0005)	1098	170.07
	CBPSOTVAC	0(0)	632(30)	0.0111(0.0005)	974	164.61
cb5-10-250-6	MBPSO	0(0)	2872(100)	0.051(0.002)	4027	547.94
	CBPSO	0(0)	12251(180)	0.263(0.005)	15030	987.17
	BPSOTVAC	0(0)	626(30)	0.0108(0.0005)	934	165.59
	CBPSOTVAC	0(0)	559(19)	0.0096(0.0003)	706	105.04
cb5-10-250-7	MBPSO	0(0)	2883(100)	0.052(0.002)	4011	550.55
	CBPSO1	0(0)	12123(110)	0.261(0.003)	13425	604.85
	BPSOTVAC	0(0)	740(30)	0.0128(0.0005)	1106	164.62
	CBPSOTVAC	0(0)	705(28)	0.0122(0.0005)	960	152.84
cb5-10-250-8	MBPSO	0(0)	2676(102)	0.048(0.002)	4490	558.11
	CBPSO1	0(0)	12159(148)	0.260(0.004)	13792	812.65
	BPSOTVAC	0(0)	654(33)	0.0112(0.0006)	1088	178.76
	CBPSOTVAC	0(0)	679(31)	0.0117(0.0005)	1036	168.47

Table 5
Simulation results of cb3-5-500 testing problems (5 knapsacks and 500 items).

Problem	Algorithm	SR	MAD	MAPE	LE	SD
cb3-5-500-1	BPSOTVAC	0(0)	1414(58)	0.0119(0.0005)	2160	315.67
	CBPSOTVAC	0(0)	1213(56)	0.0102(0.0005)	1906	311.97
cb3-5-500-2	BPSOTVAC	0(0)	1304(43)	0.0112(0.0004)	1800	236.66
	CBPSOTVAC	0(0)	1220(34)	0.0105(0.0003)	1693	185.68
cb3-5-500-3	BPSOTVAC	0(0)	1332(47)	0.0111(0.0004)	2126	262.20
	CBPSOTVAC	0(0)	1225(50)	0.0102(0.0004)	1928	273.89
cb3-5-500-4	BPSOTVAC	0(0)	1418(44)	0.0119(0.0004)	1982	245.50
	CBPSOTVAC	0(0)	1292(38)	0.0108(0.0003)	1730	213.22
cb3-5-500-5	BPSOTVAC	0(0)	1400(59)	0.0116(0.0005)	2080	323.44
	CBPSOTVAC	0(0)	1211(58)	0.0100(0.0005)	1928	320.16
cb3-5-500-6	BPSOTVAC	0(0)	1206(42)	0.0099(0.0004)	1852	232.95
	CBPSOTVAC	0(0)	1065(45)	0.0088(0.0004)	1725	250.14
cb3-5-500-7	BPSOTVAC	0(0)	1358(43)	0.0115(0.0004)	1799	239.13
	CBPSOTVAC	0(0)	1243(51)	0.0105(0.0004)	1998	281.14
cb3-5-500-8	BPSOTVAC	0(0)	1164(43)	0.0097(0.0004)	1717	234.55
	CBPSOTVAC	0(0)	1124(45)	0.0094(0.0004)	1944	245.54
cb3-5-500-9	BPSOTVAC	0(0)	1358(46)	0.0113(0.0004)	1775	252.88
	CBPSOTVAC	0(0)	1268(45)	0.0105(0.0004)	1976	248.89
cb3-5-500-10	BPSOTVAC	0(0)	1266(44)	0.0106(0.0004)	1630	241.13
	CBPSOTVAC	0(0)	1222(52)	0.0102(0.0004)	1824	283.69
cb3-5-500-11	BPSOTVAC	0(0)	1401(42)	0.0065(0.0002)	1890	232.46
	CBPSOTVAC	0(0)	1284(51)	0.0059(0.0002)	1917	277.75
cb3-5-500-12	BPSOTVAC	0(0)	1441(49)	0.0066(0.0002)	2029	266.88
	CBPSOTVAC	0(0)	1287(48)	0.0059(0.0002)	1881	265.14
cb3-5-500-13	BPSOTVAC	0(0)	1281(51)	0.0059(0.0002)	2007	278.75
	CBPSOTVAC	0(0)	1202(43)	0.0056(0.0002)	1745	236.39
cb3-5-500-14	BPSOTVAC	0(0)	1235(44)	0.0056(0.0002)	1771	241.56
	CBPSOTVAC	0(0)	1194(47)	0.0054(0.0002)	1834	254.92
cb3-5-500-15	BPSOTVAC	0(0)	1183(49)	0.0060(0.0002)	1633	233.89
	CBPSOTVAC	0(0)	1021(43)	0.0053(0.0002)	1729	257.11
cb3-5-500-16	BPSOTVAC	0(0)	1349(50)	0.0062(0.0002)	2052	275.63
	CBPSOTVAC	0(0)	1142(45)	0.0052(0.0002)	1744	246.99
cb3-5-500-17	BPSOTVAC	0(0)	1378(53)	0.0063(0.0002)	2004	290.73
	CBPSOTVAC	0(0)	1305(48)	0.0060(0.0002)	1757	261.91
cb3-5-500-18	BPSOTVAC	0(0)	1474(41)	0.0068(0.0002)	1914	224.13
	CBPSOTVAC	0(0)	1339(61)	0.0062(0.0003)	2063	333.52
cb3-5-500-19	BPSOTVAC	0(0)	1534(56)	0.0071(0.0003)	2326	306.98
	CBPSOTVAC	0(0)	1252(47)	0.0058(0.0002)	1976	257.16
cb3-5-500-20	BPSOTVAC	0(0)	1398(45)	0.0064(0.0002)	1836	249.07
	CBPSOTVAC	0(0)	1181(36)	0.0054(0.0002)	1595	198.39
cb3-5-500-21	BPSOTVAC	0(0)	938(38)	0.0032(0.0001)	1355	210.01
	CBPSOTVAC	0(0)	880(28)	0.0030(0.0001)	1183	153.43
cb3-5-500-22	BPSOTVAC	0(0)	928(42)	0.0030(0.0001)	1637	231.84
	CBPSOTVAC	0(0)	763(33)	0.0025(0.0001)	1176	179.72
cb3-5-500-23	BPSOTVAC	0(0)	858(43)	0.0029(0.0001)	1388	233.54
	CBPSOTVAC	0(0)	799(32)	0.0027(0.0001)	1211	177.71
cb3-5-500-24	BPSOTVAC	0(0)	892(40)	0.0029(0.0001)	1343	221.39
	CBPSOTVAC	0(0)	863(41)	0.0028(0.0001)	1279	225.92
cb3-5-500-25	BPSOTVAC	0(0)	904(45)	0.0030(0.0001)	1518	248.76
	CBPSOTVAC	0(0)	739(30)	0.0025(0.0001)	1069	162.85
cb3-5-500-26	BPSOTVAC	0(0)	910(38)	0.0030(0.0001)	1330	211.09
	CBPSOTVAC	0(0)	772(24)	0.0026(0.0001)	980	132.98
cb3-5-500-27	BPSOTVAC	0(0)	856(40)	0.0029(0.0001)	1395	221.13
	CBPSOTVAC	0(0)	797(35)	0.0027(0.0001)	1300	194.33
cb3-5-500-28	BPSOTVAC	0(0)	939(38)	0.0031(0.0001)	1369	210.70
	CBPSOTVAC	0(0)	881(36)	0.0029(0.0001)	1519	199.88
cb3-5-500-29	BPSOTVAC	0(0)	824(31)	0.0027(0.0001)	1338	173.94
	CBPSOTVAC	0(0)	821(36)	0.0027(0.0001)	1416	197.96
cb3-5-500-30	BPSOTVAC	0(0)	960(99)	0.0029(0.0001)	1241	185.59
	CBPSOTVAC	0(0)	878(91)	0.0026(0.0001)	1269	189.78

For BPSOTVAC and CBPSOTVAC, the control parameters were selected as follows [12,26,30]:

$c_{1i} = c_{2f} = 2.5, \quad c_{2i} = c_{1f} = 0.5, \quad w_{\max} = 1.5,$

$w_{\min} = 0.5, \quad V_{\max} = 4, \quad y_1(0) = y_2(0) = Rand()$ and $y(0) \notin (0, 0.25, 0.5, 0.75).$

Table 6
Simulation results of cb6-10-500 testing problems (10 knapsacks and 500 items).

Problem	Algorithm	SR	MAD	MAPE	LE	SD
cb6-10-500-1	BPSOTVAC	0(0)	2297(59)	0.0199(0.0005)	2938	326.18
	CBPSOTVAC	0(0)	2129(65)	0.0184(0.0006)	2754	357.39
cb6-10-500-2	BPSOTVAC	0(0)	2156(49)	0.0184(0.0004)	2653	271.48
	CBPSOTVAC	0(0)	2066(63)	0.0176(0.0006)	2649	347.08
cb6-10-500-3	BPSOTVAC	0(0)	2014(69)	0.0172(0.0006)	2884	382.66
	CBPSOTVAC	0(0)	1963(56)	0.0167(0.0005)	2449	305.01
cb6-10-500-4	BPSOTVAC	0(0)	2183(73)	0.0187(0.0006)	3048	398.84
	CBPSOTVAC	0(0)	1956(72)	0.0167(0.0006)	2717	395.62
cb6-10-500-5	BPSOTVAC	0(0)	2046(55)	0.0179(0.0005)	2784	301.20
	CBPSOTVAC	0(0)	2002(53)	0.0175(0.0005)	2490	288.93
cb6-10-500-6	BPSOTVAC	0(0)	1815(44)	0.0154(0.0004)	2322	240.05
	CBPSOTVAC	0(0)	1873(72)	0.0159(0.0006)	2789	394.38
cb6-10-500-7	BPSOTVAC	0(0)	2091(66)	0.0178(0.0006)	2825	364.03
	CBPSOTVAC	0(0)	1889(62)	0.0160(0.0005)	2516	339.91
cb6-10-500-8	BPSOTVAC	0(0)	2181(75)	0.0188(0.0007)	2935	410.72
	CBPSOTVAC	0(0)	1969(68)	0.0169(0.0006)	2588	371.94
cb6-10-500-9	BPSOTVAC	0(0)	2084(58)	0.0180(0.0005)	2641	319.64
	CBPSOTVAC	0(0)	2053(65)	0.0177(0.0006)	2890	356.94
cb6-10-500-10	BPSOTVAC	0(0)	2050(70)	0.0175(0.0006)	3012	382.07
	CBPSOTVAC	0(0)	1973(54)	0.0168(0.0005)	2754	296.14
cb6-10-500-11	BPSOTVAC	0(0)	2179(77)	0.0101(0.0004)	2940	421.99
	CBPSOTVAC	0(0)	1984(88)	0.0092(0.0004)	3092	481.92
cb6-10-500-12	BPSOTVAC	0(0)	1941(49)	0.0089(0.0002)	2410	269.25
	CBPSOTVAC	0(0)	1833(66)	0.0084(0.0003)	2869	363.35
cb6-10-500-13	BPSOTVAC	0(0)	2008(79)	0.0093(0.0004)	2784	432.21
	CBPSOTVAC	0(0)	1985(60)	0.0092(0.0003)	2682	328.20
cb6-10-500-14	BPSOTVAC	0(0)	1960(76)	0.0091(0.0004)	2699	416.84
	CBPSOTVAC	0(0)	1816(54)	0.0084(0.0003)	2241	296.17
cb6-10-500-15	BPSOTVAC	0(0)	2122(60)	0.0100(0.0003)	2782	328.21
	CBPSOTVAC	0(0)	1952(57)	0.0092(0.0003)	2565	313.63
cb6-10-500-16	BPSOTVAC	0(0)	1922(60)	0.0090(0.0003)	2719	330.89
	CBPSOTVAC	0(0)	1831(56)	0.0086(0.0003)	2293	305.58
cb6-10-500-17	BPSOTVAC	0(0)	1930(60)	0.0089(0.0003)	2570	329.07
	CBPSOTVAC	0(0)	1734(61)	0.0080(0.0003)	2635	333.71
cb6-10-500-18	BPSOTVAC	0(0)	2046(79)	0.0094(0.0004)	3329	432.57
	CBPSOTVAC	0(0)	1974(59)	0.0091(0.0003)	2649	323.21
cb6-10-500-19	BPSOTVAC	0(0)	2032(75)	0.0096(0.0004)	3064	412.26
	CBPSOTVAC	0(0)	1864(71)	0.0088(0.0003)	2577	390.38
cb6-10-500-20	BPSOTVAC	0(0)	2242(83)	0.0103(0.0004)	3288	454.79
	CBPSOTVAC	0(0)	2181(61)	0.0100(0.0003)	2640	332.92
cb6-10-500-21	BPSOTVAC	0(0)	1280(42)	0.0042(0.0001)	1906	232.59
	CBPSOTVAC	0(0)	1232(36)	0.0041(0.0001)	1735	200.77
cb6-10-500-22	BPSOTVAC	0(0)	1437(49)	0.0048(0.0002)	2043	271.02
	CBPSOTVAC	0(0)	1233(46)	0.0041(0.0002)	1712	250.74
cb6-10-500-23	BPSOTVAC	0(0)	1337(42)	0.0044(0.0001)	1915	229.62
	CBPSOTVAC	0(0)	1242(36)	0.0041(0.0001)	1665	196.80
cb6-10-500-24	BPSOTVAC	0(0)	1342(62)	0.0045(0.0002)	2318	340.77
	CBPSOTVAC	0(0)	1181(43)	0.0039(0.0001)	1632	234.86
cb6-10-500-25	BPSOTVAC	0(0)	1224(58)	0.0040(0.0002)	2248	317.77
	CBPSOTVAC	0(0)	1091(34)	0.0036(0.0001)	1518	185.82
cb6-10-500-26	BPSOTVAC	0(0)	1486(55)	0.0050(0.0002)	2186	300.02
	CBPSOTVAC	0(0)	1379(44)	0.0046(0.0001)	1813	240.06
cb6-10-500-27	BPSOTVAC	0(0)	1331(52)	0.0044(0.0002)	2033	284.89
	CBPSOTVAC	0(0)	1209(54)	0.0040(0.0002)	1832	293.38
cb6-10-500-28	BPSOTVAC	0(0)	1393(53)	0.0047(0.0002)	2016	289.12
	CBPSOTVAC	0(0)	1287(37)	0.0044(0.0001)	1832	204.77
cb6-10-500-29	BPSOTVAC	0(0)	1314(54)	0.0044(0.0002)	2000	299.47
	CBPSOTVAC	0(0)	1237(43)	0.0041(0.0001)	1846	233.81
cb6-10-500-30	BPSOTVAC	0(0)	1378(47)	0.0045(0.0002)	1949	258.98
	CBPSOTVAC	0(0)	1268(47)	0.0041(0.0002)	1742	258.33

For MBPSO and CBPSO1, the control parameters were based on the publications of Bansal and Deep [12] and Chuang et al. [21], respectively. For a fair comparison, similar values of common parameters were used in these four algorithms.

The simulation experiments were conducted using an Intel Core 2 processor with 2.83 GHz CPU and 2G RAM. The swarm size was set to be five times the number of decision variables. The maximum iterations (t_{max}) was set to 20000. The simulation experiments were based on a total of 100 runs.

Initially, the leading investigations of the low-dimensional problems were verified. The simulation results of the first group of benchmarks are shown in Table 1. The first three columns include the name of the instance, number of knapsacks, and number of items of the instance. The value enclosed in parentheses is the quality measure (standard error) of the corresponding estimate. Based on the results shown in Table 1, CBPSO1 and BPSOTVAC are more reliable than the other two PSO algorithms. Furthermore, BPSOTVAC outperformed CBPSO1 in terms of the three criteria.

Table 2 summarizes the experimental results of the second group of benchmarks. The results show that BPSOTVAC is superior to the other methods. CBPSO1 performed well for instances with small items (<50 items). However, the algorithm is insufficiently robust for instances with larger decision variables (larger items). The simulation results of the third group of benchmarks revealed a pattern similar to that in Table 3. CBPSO1 performed adequately for the instances with small items. However, BPSOTVAC and CBPSO1 performed similarly for the instances with a smaller number of decision variables (small items). One important observation in the investigations of the low-dimensional problems is that BPSOTVAC outperformed all others.

To ensure that functions with more dimensions are tested, we continued to solve some difficult problems to show the robustness of the proposed algorithm. To reduce the computation efforts, some parameter settings were relaxed. The simulation experiments were based on a total of 30 runs in high-dimensional testing problems. Second, swarm size was set as the number of decision variables for groups "cb3-5-500" and "cb6-10-500."

Table 4 indicates the experimental results of the fourth group of benchmarks. The results showed that BPSOTVAC and CBPSOTVAC are superior to the other two existing methods in high-dimensional testing problems. Based on this observation, algorithms MBPSO and CBPSO1 were phased out in the remaining high-dimension problem simulation. Tables 5 and 6 summarize the simulation results of the benchmarks "cb3-5-500" and "cb6-10-500," respectively. The simulation results of high-dimensional problems suggest that CBPSOTVAC is more robust for difficult high-dimension problems. Generally, the low dimensional simulation results verified that BPSOTVAC is more reliable than the other three PSO algorithms. However, the high-dimensional simulation results verified that CBPSOTVAC is the more reliable algorithm.

6. Conclusions and Future Studies

In our study, two novel PSO algorithms, namely, BPSOTVAC and CBPSOTVAC, were proposed to solve the MKPs. A total of 116 benchmark problems (including low- and high-dimensional knapsack problems) from OR-Library were tested, and the results were compared with those of the other two existing PSO algorithms. The simulation and evaluation results showed that BPSOTVAC is superior to the other methods for low-dimensional knapsack benchmark problems. However, CBPSOTVAC outperformed others in high-dimensional knapsack testing problems. Thus, we can conclude that BPSOTVAC and CBPSOTVAC possess competitive strength in solving multi-dimensional knapsack problems.

To establish the proposed algorithms for general use, further research is necessary. First, the algorithms must be tested and evaluated over other binary test problems to show their robustness in handling other problem types. Moreover, statistical tests such as "Friedman test" must be employed to confirm the performance of the proposed methods statistically. Finally, testing against existing non-PSO state-of-the-art methods is helpful in establishing the algorithms for general purposes.

References

[1] A. Fréville, The multidimensional 0–1 knapsack problem: an overview, Eur. J. Oper. Res. 155 (2004) 1–21.
[2] H. Meier, N. Christofides, G. Salkin, Capital budgeting under uncertainty – an integrated approach using contingent claims analysis and integer programming, Operations Research 49 (2001) 196–206.
[3] G.J. Beaujon, S.P. Martin, C.C. McDonald, Balancing and optimizing a portfolio of R&D projects, Naval Research Logistics 48 (2001) 18–40.
[4] W. Shih, A branch and bound method for the multiconstraint zero-one knapsack problems, Journal of the Operations Research Society 30 (1979) 369–378.
[5] D. Bertsimas, R. Demir, An approximate dynamic programming approach to multidimensional knapsack problems, Manage. Sci. 48 (2002) 550–565.
[6] M. Vasquez, J.K. Hao, A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite, Computational Optimization and Applications 20 (2001) 137–157.
[7] M.D. Garey, D.S. Johnson, Computers and intractability: a guide to the theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[8] A. Drexel, A simulated annealing approach to the multiconstraint zero-one knapsack problem, Computing 40 (1988) 1–8.
[9] M. Vasquez, Y. Vimont, Improved results on the 0–1 multidimensional knapsack problem, Eur. J. Oper. Res. 165 (2005) 70–81.
[10] G.R. Raidl, J. Gottlieb, Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithm: a case study for the multidimensional knapsack problem, Evolutionary Computation 13 (2005) 441–475.
[11] L. Ke, Z. Feng, Z. Ren, X. Wei, An ant colony optimization approach for the multidimensional knapsack problem, Journal of Heuristics 16 (2010) 65–83.
[12] J.C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Appl. Math. Comput. 218 (2012) 11042–11061.
[13] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks 4 (1995) 1942–1948.
[14] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, Proceedings of IEEE international conference on systems, man, and cybernetics 5 (1997) 4104–4109.
[15] Y.-F. Liao, D.-H. Yau, C.-L. Chen, Evolutionary algorithm to traveling salesman problems, Comput. Math. Appl. 64 (2012) 788–797.
[16] A. Azadeh, M.S. Sangari, A.S. Amiri, A particle swarm algorithm for inspection optimization in serial multi-stage processes, Applied Mathematical Modeling 36 (2012) 1455–1464.
[17] I. Montalvo, J. Izquierdo, R. Pérez, M. Tung, Particle swarm optimization applied to the design of water supply systems, Comput. Math. Appl. 56 (2008) 769–776.
[18] M. Chih, L.-L. Yeh, F.-C. Li, Particle swarm optimization for the economic and economic statistical designs of the $\bar{X}$ control chart, Applied Soft Computing 11 (2011) 5053–5067.

[19] W.C. Yeh, Y.C. Lin, Y.Y. Chung, M. Chih, A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem, IEEE Trans. Reliab. 59 (2010) 212–221.
[20] H. Cho, D. Kim, F. Olivera, S.D. Guikema, Enhanced speciation in particle swarm optimization for multi-modal problems, Eur. J. Oper. Res. 213 (2011) 15–23.
[21] L.-Y. Chuang, C.-H. Yang, J.-C. Li, Chaotic maps based on binary particle swarm optimization for feature selection, Applied Soft Computing 11 (2011) 239–248.
[22] Y. Wang, Y. Yang, Particle swarm with equilibrium strategy of selection for multi-objective optimization, Eur. J. Oper. Res. 200 (2010) 187–197.
[23] R.J. Kuo, S.Y. Hong, Y.C. Huang, Integration of particle swarm optimization-based fuzzy neural network and artificial neural network for supplier selection, Applied Mathematical Modeling 34 (2010) 3976–3990.
[24] J.E. Beasley, ORLib – Operations Research Library. [http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/mknapinfo.html], (2005).
[25] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.
[26] A. Ratnweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (2004) 240–255.
[27] J.M. Yang, Y.P. Cheng, J.T. Horng, C.Y. Kao, Applying family competition to evolution strategies for constrained optimization, Lect. Notes Comput. Sci. 1213 (1997) 201–211.
[28] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method for constrained optimization problems, Proceedings of the Euro-International Symposium on Computational Intelligence (E-ISCI), 2002.
[29] A. Unler, A. Murat, A discrete particle swarm optimization method for feature selection in binary classification problems, Eur. J. Oper. Res. 206 (2010) 528–539.
[30] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, Proceedings of the congress on evolutionary computation 3 (1999) 101–106.
[31] J. Chuanwen, E. Bompard, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization, Mathematics and Computers in Simulation 68 (2005) 57–65.
[32] T. Xiang, X. Liao, K.-w. Wong, An improved particle swarm optimization algorithm combined with piecewise linear chaotic map, Appl. Math. Comput. 190 (2007) 1637–1645.
[33] C. Jiang, B. Etorre, A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment, Energy Convers. Manage. 46 (2005) 2689–2696.
[34] S. Senyu, Y. Toyada, An approach to linear programming with 0–1 variables, Manage. Sci. 15 (1967) B196–B207.
[35] H.M. Weingartner, D.N. Ness, Methods for the solution of the multi-dimensional 0/1 knapsack problem, Operations Research 15 (1967) 83–103.
[36] A. Freville, G. Plateau, Hard 0–1 multiknapsack test problems for size reduction methods, Investigation Operativa 1 (1990) 251–270.