# BPSO Algorithms for Knapsack Problem

Amira Gherboudj and Salim Chikhi

Computer Science Department, MISC Laboratory,
Mentouri University, Constantine Algeria
{gherboudj,chikhi}@ umc edu.dz

**Abstract.** Particle Swarm Optimization (PSO) is an evolutionary metaheuristic. It was created in 1995 by Kennedy and Eberhart for solving optimization problems. However, several alternatives to the original PSO algorithm have been proposed in the literature to improve its performance for solving continuous or discrete problems. We propose in this paper 4 classes of binary PSO algorithms (BPSO) for solving the NP-hard knapsack problem. In the proposed algorithms, the velocities and positions of particles are updated according to different equations. To verify the performance of the proposed algorithms, we made a comparison between algorithms of the 4 proposed classes and a comparison between the proposed algorithms with the Standard PSO2006 and the Standard BPSO. The comparison results showed that the proposed algorithms outperform the Standard PSO2006 and the Standard BPSO in terms of quality of solution found.

**Keywords:** PSO, BPSO, knapsack Problem (KP).

## 1 Introduction

The Particle Swarm Optimization (PSO) is one of population-based solution metaheuristics inspired by an analogy with the ethology. It was created in 1995 by Kennedy and Eberhart [3]. PSO mimics the collective behavior of animals living in groups such as bird flocking and fish schooling. Simplicity and performance of this method have attracted interest of several communities of researchers who have conducted studies on optimization and application of this metaheuristic for solving several optimization problems. In this paper, we propose 4 classes of Binary PSO algorithms (BPSO) for solving the knapsack problem.

The knapsack problem (KP) is a NP-hard problem [1,2]. It can be defined as follows: Assuming that we have a knapsack with maximum capacity C and a set of N objects. Each object i has a profit $p_i$ and a weight $w_i$. The problem is to select a subset of items from the set of N objects to maximize the value of all selected objects without exceeding the maximum capacity of the knapsack. KP can be formulated as:

$$\text{Maximize} \quad \sum_{i=1}^{N} p_i x_i \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{N} w_i x_i \leq C \tag{2}$$

$$x_i = \begin{cases} 1 \text{ If the object i is selected} \\ 0 \text{ Otherwise} \end{cases} \qquad i=1, \ldots\ldots, N \tag{3}$$

The remainder of this paper is organized as follows: the principle of the PSO is described in section 2. The third section concerns PSO variants. In the fourth section we describe the algorithms of each class. Comparison and experimental results are provided in section 5 and a conclusion is provided in the sixth section of this paper.

## 2 PSO Principle

The PSO method involves a set of agents for solving a given problem. This set is called swarm, each swarm is composed of a set of members, they are called particles. Each particle is characterized by position $x_{id}= (x_{i1}, x_{i2},\ldots, x_{id},\ldots, x_{iD})$ and velocity $v_{id}= (v_{i1}, v_{i2},\ldots, v_{id},\ldots, v_{iD})$ in a search space of D-dimension. During the search procedure, the particle tends to move towards the best position (solution) found. At each iteration of the search procedure, the particle moves and updates its velocity and its position in the swarm based on experience and the results found by the particle itself, its neighbors and the swarm. It therefore combines three components: its own current velocity, its best position $p_{bestid}= (p_{besti1}, p_{besti2},\ldots, p_{bestid},\ldots, p_{bestiD})$ and the best position obtained by its informants. Thus the equations for updating the velocity and position of particles are presented below:

$$v_{id}(t)= v_{id} (t-1) + c_1 r_1 (p_{bestid} (t-1) - x_{id} (t-1)) + c_2 r_2 (g_{bestd} (t-1) - x_{id} (t-1)) \tag{4}$$

$$x_{id} (t)= x_{id} (t-1) + v_{id} (t) \tag{5}$$

$(x_{id} (t), x_{id} (t-1))$, $(v_{id} (t), v_{id} (t-1))$: Position and Velocity of particle i in dimension d at times t and t-1, respectively. $p_{bestid} (t-1)$, $g_{bestd}(t-1)$ : the best position obtained by the particle i and the best position obtained by the swarm in dimension d at time t-1, respectively. $c_1$, $c_2$: two constants representing the acceleration coefficients. $r_1$, $r_2$: random numbers drawn from the interval [0,1[. $v_{id} (t-1)$, $c_1 r_1 (p_{bestid} (t-1) - x_{id} (t-1))$, $c_2 r_2 (g_{bestd}(t-1) - x_{id} (t-1))$: the three components mentioned above, respectively.

The position of particle i represents a solution of the addressed problem. The value of the objective function (or fitness) of the particle i is denoted by f $(x_{id})$. To estimate the quality of particle i, it is necessary to calculate its fitness. This one is calculated using a special function for the addressed problem. In the knapsack problem, the fitness is calculated according to equation (1).

The PSO algorithm begins by initializing the size of the swarm and the various parameters. Assign randomly to each particle an initial position and velocity. Initialize $p_{bestid}$, then calculate the fitness of particles in order to calculate the best position found by the swarm ($g_{bestd}$). At each iteration, particles are moved using equations (4) and (5). Their objective functions are calculated and $p_{bestid}$, $g_{bestd}$ are updated. The process is repeated until the satisfaction of stopping criterion. A pseudo PSO algorithm is presented below:

```
                    Particle Swarm Optimization Algorithm
1. Initialization :
     • Parameters and size of the swarm (S);
     • Randomly initialize particles positions and velocities;
     • For each particle, $p_{bestid} = x_{id}$;
     • Calculate f ($x_{id}$) of each particle;
     • Calculate $g_{bestd}$; // the best $p_{bestid}$
2. While (termination criterion is not met) {
     For (i = 1 to S) {
     • Calculate the new velocity using equation (4);
     • Calculate the new position using equation (5);
     • Calculate f ($x_{id}$) of each particle;
     • If (f ($x_{id}$) >f ($p_{bestid}$)) $p_{bestid} = x_{id}$;  // Maximization case
     • If (f ($p_{bestid}$) >f ($g_{bestd}$)) $g_{bestd} = p_{bestid}$;
       }
   }
3. Show the best solution found $g_{bestd}$;
```

## 3 PSO Variants

The idea of the pioneers of PSO algorithm: Kennedy and Eberhart [3] has sought the attention of several researchers who have conducted studies in the aim of improving the performance of the proposed method (PSO) which is not a global convergence-guaranteed optimization algorithm [5].

In 1996, Eberhart and al [15] proposed to limit the velocity of the particles in $[-V_{max}, V_{max}]$ to avoid the problem of deviation of the search space during the movement of particles. The role of the new parameter $V_{max}$ is to control the movement of particles.

In 1998, Shi and Eberhart [4] proposed to apply the inertia coefficient ω, to control the particles velocities as follows:

$$v_{id}(t)= \omega\, v_{id} (t-1) + c_1 r_1 (p_{bestid} (t-1) - x_{id} (t-1)) + c_2 r_2 (g_{bestd} (t-1) - x_{id} (t-1)) \quad (6)$$

ω is an inertia coefficient. It is used to control the influence of particle velocity on his next move to keep a balance between exploitation and exploration of the search space.

On the other hand, Clerc and Kennedy [9] proposed an alternative of equation (4). Their solution is to add a constriction coefficient K in the aim of controlling the speed of the particles to escape the divergence problem of the swarm that causes premature convergence of the algorithm. The proposed equation is:

$$v_{id} (t)= K [v_{id} (t-1) + c_1 r_1 (p_{bestid} (t-1) - x_{id} (t-1)) + c_2 r_2 (g_{bestd} (t-1) - x_{id} (t-1))] \quad (7)$$

Where $K= \frac{2}{\left|2-\varphi-\sqrt{\varphi(\varphi-4)}\right|}$ ;With $\varphi= c_1+ c_2$ and $\varphi > 4$ ; $c_1 = c_2 = 2.05$, $K=0.729844$.

To ensure the diversity of the swarm, Hi et al [7] proposed to update the particle velocity according to equation (8):

$$v_{id}(t) = \omega\, v_{id}(t-1) + c_1 r_1 (p_{bestid}(t-1) - x_{id}(t-1)) + c_2 r_2 (g_{bestd}(t-1) - x_{id}(t-1)) + c_3 r_3 (P^r_{id}(t-1) - x_{id}(t-1)) \quad (8)$$

$P^r_{id}$ is the position of a particle i of swarm in the dimension d of the search space, this particle is selected randomly at time (t-1). The role of the component ($P^r_{id}$ (t-1) - $x_{id}$ (t-1)) is to ensure the diversity of the swarm based on the value of the coefficient $c_3$.

## 4 BPSO Algorithm

The first version of BPSO algorithm (The Standard BPSO algorithm) was proposed in 1997 by Kennedy and Eberhart [11]. In the BPSO algorithm, the position of particle i is represented by a set of bit. The velocity $v_{id}$ of the particle i is calculated from equation (4). $v_{id}$ is a set of real numbers that must be transformed into a set of probabilities, using the sigmoid function as follows:

$$S(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \quad (9)$$

Where S ($v_{id}$) represents the probability of bit $x_{id}$ takes the value 1.

To avoid the problem of the divergence of the swarm, the velocity $v_{id}$ is generally limited by a maximum value $V_{max}$ and a minimum value $-V_{max}$, i.e. $v_{id} \in [-V_{max}, V_{max}]$. The position $x_{id}$ of the particle i is updated as follows:

$$x_{id} = \begin{cases} 1 \text{ if } r < S(v_{id}) \\ 0 \text{ Otherwise} \end{cases} \qquad r \in [0, 1[ \quad (10)$$

In addition to the version of the Standard BPSO algorithm they exist other versions of BPSO algorithm, such as those proposed in [8, 12, 13, 14].

### 4.1 Representation

To represent the positions and velocities of the particles we used binary vectors of size D. The representation of position of particle i is as follows:

$$x_{id} = [x_{i1}, x_{i2}, \ldots, x_{id}, \ldots, x_{iD}]$$

$$x_{id} = \begin{cases} 1 \text{ If the object is selected} \\ 0 \text{ Otherwise} \end{cases}$$

### 4.2 Velocity and Position Update

To represent the PSO principle, we need a number of operations and operators which are defined in [6].

### 4.3 Proposed Classes

In the aim of solving the KP, we have proposed four classes of BPSO algorithm. In each class, we have proposed four algorithms with different equations and parameters.

**4.3.1 The First Class.** In the first class we adapt and use the PSO version with inertia coefficient $\omega$, proposed in 1998 by Shi and Eberhart [4]. In the algorithms of this class, the position of particles is updated according to equation (5).

**1) BPSO6:** It is an adaptation of the Standard PSO2006. In BPSO6, the velocity of particles is updated using the following equation:

$$v_{id}(t) = \omega \times v_{id}(t-1) + r_1 c_1 \times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2 c_2 \times (l_{bestd}(t-1) - x_{id}(t-1)) \qquad (11)$$

$l_{bestd}$ (t-1) is the best position found by the particles in dimension d of a given neighborhood. $c_1$ and $c_2$ are chosen randomly at each iteration. But in contrast to the standard PSO2006, The size of the swarm is equal to the dimension of the problem.

**2) BP3:** In BP3, the velocity is updated using Equation (11). $c_1$ and $c_2$ are constants.

**3) BP2:** In BP2, the velocity is updated according to equation (12) defined below:

$$v_{id}(t) = \omega \times v_{id}(t-1) + r_1 c_1 \times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2 c_2 \times (l_{bestd}(t-1) - x_{id}(t-1)) + r_3 c_3 \times (g_{bestd}(t-1) - x_{id}(t-1)) \qquad (12)$$

$c_1$, $c_2$ and $c_3$ are constants.

**4) BP1:** To provide greater diversification within the swarm, we were inspired by the PSOPC algorithm [7] and we proposed to update the velocity of particles in BP1 algorithm using the following equation:

$$v_{id}(t) = \omega \times v_{id}(t-1) + r_1 c_1 \times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2 c_2 \times (l_{bestd}(t-1) - x_{id}(t-1)) + r_3 c_3 \times (g_{bestd}(t-1) - x_{id}(t-1)) + r_4 c_4 \times (P^r_{id}(t-1) - x_{id}(t-1)) \qquad (13)$$

Where $c_1$, $c_2$, $c_3$ and $c_4$ are constants. $P^r_{id}$ is the position of a particle i of swarm in the dimension d of the search space, this particle is selected randomly at time t-1.

**4.3.2 The Second Class.** In the second class we drew mutation factor used in the C3DPSO algorithm proposed by Zhong and Zhang [8] and we proposed a new acceleration coefficient F that we used to update particle position.

**1) BFP6:** In the BFP6 algorithm, the position of particles is updated according to equation (14) defined below:

$$x_{id}(t) = rF \times x_{id}(t-1) + v_{id}(t) \qquad (14)$$

The velocity of particles is updated according to equation (11).

**2) BFP3:** In BFP3, position and velocity of particles are updated according to equation (14) and (11), respectively. But $c_1$ and $c_2$ are constants.

**3) BFP2:** In BFP2, the position of particles is updated according to equation (14) and the velocity is updated according to the equation (12).

**4) BFP1:** In BFP1, the position of particles is updated according to equation (14). The velocity is updated according to equation (13).

**4.3.3 The Third Class.** In the third class, we adapted and used the PSO version proposed in [9], because we noticed that the PSO algorithm with constriction coefficient K is not widely used in the literature. To our knowledge, there is no paper that addresses the KP using PSO algorithm with constriction coefficient. In the algorithms of this class, the position of particles is updated according to equation (5).

**1) BCP6:** In the BCP6 algorithm, the velocity of particles is updated using the following equation:

$$v_{id}(t)= K\times[v_{id}(t-1) + r_1c_1\times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2\, c_2\times (l_{bestd}(t-1) - x_{id}(t-1))] \quad (15)$$

$c_1$ and $c_2$ are chosen randomly at each iteration.

**2) BCP3:** In the BCP3 algorithm, the velocity of particles is updated using the equation (16), but $c_1$ and $c_2$ are constants.

**3) BCP2:** In BCP2 algorithm, the velocity of particles is updated using the following equation:

$$v_{id}(t)= K\times[v_{id}(t-1) + r_1c_1\times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2\, c_2\times (l_{bestd}(t-1) - x_{id}(t-1)) + r_3\, c_3\times (g_{bestd}(t-1) - x_{id}(t-1))] \quad (16)$$

$c_1$, $c_2$ and $c_3$ are constants.

**4) BCP1:** In BCP1 algorithm, we proposed to update the velocity of particles using the following equation:

$$v_{id}(t)= K\times [v_{id}(t-1) + r_1c_1\times (p_{bestid}(t-1) - x_{id}(t-1)) + r_2\, c_2\times (l_{bestd}(t-1) - x_{id}(t-1)) + r_3\, c_3\times (g_{bestd}(t-1) - x_{id}(t-1)) + r_4\, c_4\times (P^r_{id}(t-1) - x_{id}(t-1))] \quad (17)$$

Where $c_1$, $c_2$, $c_3$ and $c_4$ are constants; K=0.7.

**4.3.4 The Fourth Class.** This class includes algorithms defined in the third class with application of the new acceleration coefficient F. The position of particles is updated according to equation (14).

**1) BFCP6:** In the BFCP6 algorithm, the velocity of particles is updated according to equation (15).

**2) BFCP3:** In the BFCP3 algorithm, the velocity of particles is updated according to equation (15), but $c_1$ and $c_2$ are constants.

**3) BFCP2:** In the BFCP2 algorithm, velocity of particles is updated according to equation (16).

**4) BFCP1:** In the BFCP1 algorithm, velocity of particles is updated according to equation (17).

## 5 Comparison and Experimental Results

To verify and compare the performance of the algorithms of the 4 proposed classes, 7 instances with different numbers of items were generated. In the first instance the number N of objects is equal to 120, in the second instance N = 200, in the third one

N = 500 then N = 700, 900, 1000 and 2000 in the fourth, fifth, sixth and seventh instances respectively.

Initially, we conducted a comparative study between the proposed algorithms of the four classes. Then we compared the proposed algorithms with the Standard PSO2006 [10] that we have adapted to the binary representation used. We also compared the proposed algorithms with the Standard BPSO [11].

The algorithms are coded in Java. Each algorithm is executed 125 times.

The capacity of the knapsack is calculated using the following formula:

$$C = \frac{3}{4} \sum_{i=1}^{N} w_i$$

The weights $w_i$ and profits $p_i$ of objects were selected randomly. For the algorithms of each class, the size of the swarm is equal to the number of items. In the 1st and 2nd classes, $\omega = 0.7$. In the 3rd and 4th Classes, K was not calculated from the formula defined by Clerc and Kennedy i.e. $K = \frac{2}{|2-\varphi-\sqrt{\varphi(\varphi-4)}|}$, but it was set at 0.7.

The values of the parameters $c_1$, $c_2$, $c_3$, $c_4$ and F are equal to 0.3, 0.4, 0.6, 0.1, and 0.9 respectively. Exceptionally in BPSO6, BFP6, BCP6 and BFCP6, the parameters $c_1$ and $c_2$ are drawn randomly from [0, 1[.

The positions of particles were randomly initialized for each execution. The velocities were initialized with the value 0. The number of iterations in each run is chosen equal to 15 and is used as stopping criteria for each run.

Concern the parameters of the standard PSO2006, we kept the same parameters defined in [10], but with binary representation of positions and velocities of particles.

About the Standard BPSO, we followed the equations, representation and parameters defined in [11], except that the values of $c_1$ and $c_2$ are equal to those used for testing the proposed algorithms, i.e. $c_1, c_2 = 0.3, 0.4$, respectively.

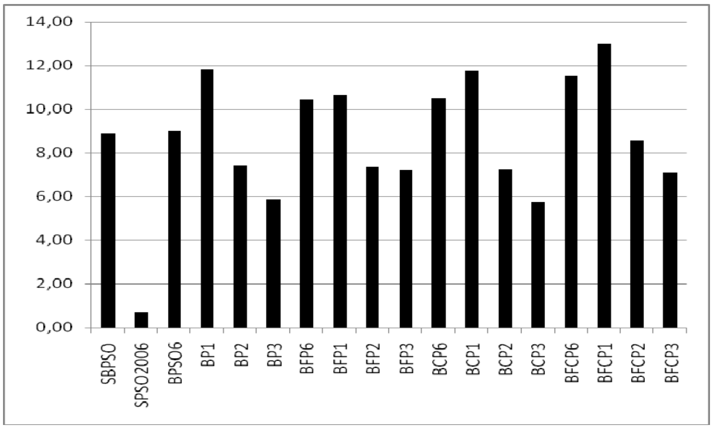

**Fig. 1.** Comparison of average computation time of the proposed algorithms with the Standard PSO2006 and the Standard BPSO

Fig. 1 shows a comparison of average computation time with 1000 objects, estimated by seconds for the proposed algorithms, the Standard PSO2006 (SPSO2006 in the figure) and the Standard BPSO (SBPSO in the figure).

In terms of computing time, Fig.1 shows that:

- The BCP3 algorithm is the best one and the BFCP1 algorithm is the worst one among the proposed algorithms.
- The B*P2 and B*P3 algorithms (i.e. BP2, BFP2, BCP2, BFCP2, BP3, BFP3, BCP3 and BFCP3) converge faster than the Standard BPSO algorithm.
- The Standard PSO2006 converges faster than the proposed algorithms.

Tables 1 and 2 show the experimental results of algorithms of each class, classes 1 and 2 in Table 1 and Class 3 and 4 in Table 2. First column of each table represents the instance i.e. the number of items. The second and third column (Class 1 and Class 2 in the first table and Class 3 and Class 4 in the second table) represent the best solutions and averages found for each instance by the algorithms of the relevant class.

Table 3 completes the tables 1 and 2. It represents the experimental results of the proposed algorithms, the Standard PSO2006 and the Standard BPSO for each instance during 125 executions. The first column represents the instance. The second column represents the best values of best and averages obtained by the proposed algorithms of the 4 classes. The third and fourth columns represent the bests and Averages obtained by the Standard BPSO and the Standard PSO2006 respectively. For each instance in tables 1, 2 and 3, the first row represents the best solution and the second row represents the average.

**Table 1.** Comparison results of the proposed algorithms of Class 1 and Class 2

| Instance | Class 1 | | | | Class 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | BPSO6 | BP1 | BP2 | BP3 | BFP6 | BFP1 | BFP2 | BFP3 |
| 120 | 4439 | 4469 | 4552 | 4457 | 4463 | 4489 | 4497 | 4564 |
| | 4180,8 | 4130,4 | 4136 | 4236,4 | 4203,2 | 4140,4 | 4137,4 | 4236,4 |
| 200 | 7559 | 7522 | 7642 | 7490 | 7491 | 7339 | 7648 | 7624 |
| | 7104,4 | 6989,9 | 6938 | 7169,8 | 7108,6 | 6972,2 | 6979,8 | 7168,4 |
| 500 | 17776 | 17647 | 17682 | 18058 | 17810 | 17642 | 17647 | 17974 |
| | 16949 | 16848,2 | 16598,4 | 17224 | 17094,2 | 16864,8 | 16564,2 | 17175,8 |
| 700 | 24678 | 24407 | 24101 | 24431 | 24809 | 24407 | 24469 | 24563 |
| | 23252,6 | 23335,6 | 23019,4 | 23867,6 | 23767,6 | 23368,4 | 23027,8 | 30507,4 |
| 900 | 31428 | 30898 | 31215 | 31686 | 31192 | 31018 | 31525 | 31276 |
| | 29659,4 | 29833,8 | 29369 | 30509,8 | 30391,2 | 29841,6 | 29330 | 30507,4 |
| 1000 | 34654 | 34319 | 34213 | 35019 | 34847 | 34273 | 33999 | 34596 |
| | 32912,2 | 33065 | 32558,6 | 33810,6 | 33637 | 33023 | 32543,2 | 33882,4 |
| 2000 | 68857 | 66605 | 66546 | 68548 | 67914 | 67110 | 67507 | 67829 |
| | 63232,75 | 65030,6 | 63754,6 | 66478 | 66185,2 | 65037 | 63725 | 66716,6 |

**Table 2.** Comparison results of the proposed algorithms of Class 3 and Class 4

| Instance | Class 3 | | | | Class 4 | | | |
|---|---|---|---|---|---|---|---|---|
| | BCP6 | BCP1 | BCP2 | BCP3 | BFCP6 | BFCP1 | BFCP2 | BFCP3 |
| 120 | 4543 | 4497 | 4545 | 4533 | 4552 | 4538 | 4512 | 4487 |
| | 4226,2 | 4216,2 | 4204 | 4242,4 | 4229,8 | 4230,4 | 4207 | 4257,4 |
| 200 | 7506 | 7614 | 7474 | 7556 | 7531 | 7646 | 7681 | 7555 |
| | 7139,4 | 7126,2 | 7110,4 | 7183 | 7151 | 7133,6 | 7097 | 7203,6 |
| 500 | 17922 | 17983 | 17810 | 17906 | 17642 | 17838 | 17731 | 18332 |
| | 17069,2 | 17093,8 | 17159,6 | 17265,4 | 17175,8 | 17192,4 | 17129,4 | 17268 |
| 700 | 24651 | 24603 | 24558 | 24729 | 24540 | 24634 | 24326 | 24951 |
| | 23589,6 | 23637,4 | 23787,2 | 23926,8 | 23876 | 23835 | 23772,6 | 23934,6 |
| 900 | 31488 | 31295 | 31105 | 31240 | 31324 | 31448 | 31082 | 31478 |
| | 30111 | 30063 | 30439,2 | 30537,6 | 30448 | 30504,6 | 30455,2 | 30495,2 |
| 1000 | 34680 | 34606 | 34863 | 35897 | 34934 | 34745 | 34510 | 34717 |
| | 33291,6 | 33384,8 | 33819,6 | 33910,8 | 33879,4 | 33882,4 | 33776 | 33863 |
| 2000 | 66703 | 68266 | 67856 | 67995 | 68784 | 68378 | 67645 | 67689 |
| | 65275,2 | 66656,8 | 66615 | 66779,75 | 66322,8 | 66740 | 66682,4 | 66630,8 |

**Table 3.** Comparison of best values obtained by the proposed algorithms, the Standard PSO2006 and the Standard BPSO

| Instance | Best Known | Standard BPSO | Standard PSO2006 |
|---|---|---|---|
| 120 | 4564 | 4296 | 4331 |
| | 4257,4 | 3840,8 | 4027 |
| 200 | 7681 | 7456 | 7391 |
| | 7203,6 | 5703 | 6819,4 |
| 500 | 18332 | 13116 | 17618 |
| | 17268 | 12471,2 | 16244,4 |
| 700 | 24951 | 18276 | 23893 |
| | 30537,6 | 17097,4 | 22400,2 |
| 900 | 31686 | 22857 | 30770 |
| | 30537,6 | 21736,6 | 28574,2 |
| 1000 | 35897 | 24933 | 34025 |
| | 33910,8 | 24050 | 31682,2 |
| 2000 | 68857 | 47674 | 67006 |
| | 66779,75 | 46538,8 | 63265,8 |

Tables 1, 2 and 3 show that:

- The use of PSO algorithm version with constriction coefficient gives good averages compared to the PSO version with inertia coefficient.

- In most cases, the application of the acceleration coefficient F on the algorithms of the first class (which gave birth to algorithms of the second class) has improved their results in terms of averages.

- The use of the acceleration coefficient F in the algorithms of the third class improves their results.

- In most cases, the application of acceleration coefficient F on the version of PSO algorithm with constriction coefficient gives good averages compared with its application to the version of PSO algorithm with inertia coefficient ω.

- Best averages are obtained by B*P3 i.e. BP3, BFP3, BCP3 and BFCP3.

- In most cases, best values are obtained by B*P3 i.e. BP3, BFP3, BCP3 and BFCP3.

- The performance of the 16 proposed algorithms exceed those of the Standard PSO2006 and the Standard BPSO in terms of best solution found and average.

## 6 Conclusion

PSO is a recent metaheuristic. It has sought the attention of several research communities. PSO has proved its simplicity of implementation and effectiveness. Several variants to the original PSO algorithm have been proposed in the literature to improve its performance. In this contribution, we drew some works and applications of the PSO algorithm presented in the literature, and we proposed 4 classes of BPSO algorithms with different equations for updating velocities and positions of particles. We have grouped the proposed algorithms into four classes: in the first class, we adapted and used the PSO version with inertia coefficient [4]. The new acceleration coefficient F is used in the second class for updating the particles positions. F was applied on the algorithms proposed in the first class which has given birth to the second class of algorithms. In the third class we adapted and used the PSO version with constriction coefficient [9] because we noticed that few studies use this version. In the fourth class, we used the acceleration coefficient F for the update of particles positions and the constriction coefficient for the update of particles velocities. We applied the proposed algorithms for solving the NP-hard knapsack problem using multiple instances (120, 200, 500, 700, 900, 1000 and 2000 objects).

To verify the performance of the proposed algorithms, we conducted a comparative study between the proposed algorithms of the four classes and a comparison of the proposed algorithms with the Standard PSO2006 [10] and the Standard BPSO [11]. Comparative studies of the proposed algorithms show performance improvements with the use of the new acceleration coefficient F for the updating of position and the application of the constriction coefficient K for the updating of velocity. In terms of average and best solutions, experimental results show that the proposed algorithms outperform the Standard PSO2006 and the Standard BPSO. In terms of average, best solutions and computation time, experimental results show that the B*P2 and B*P3 algorithms (i.e. BP2, BFP2, BCP2, BFCP2, BP3, BFP3, BCP3 and BFCP3) outperform the Standard BPSO.

## References

1. Xie, X., Liu, J.: A Mini-Swarm for the quadratic Knapsack Problem. In: IEEE Swarm Intelligence Symposium (SIS), Honolulu, HI, USA, pp. 190–197 (2007)
2. Pisinger, D.: Where are the hard knapsack problems? Computers and Operations Research 32(9), 2271–2284 (2005)
3. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proc. IEEE Int. Conf. On Neural Networks, WA, Australia, pp. 1942–1948 (1995)
4. Shi, Y., Eberhart, R.: Parameter Selection in Particle Swarm Optimisation. In: Proceedings of the 7th Annual Conference on Evolutionary Programming. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
5. Wang, J., Zhou, Y.: Quantm-behaved Particle Swarm Optimization with Generalized Local Search Operator for Global Optimization. In: Advanced Intelligent Computing Theories and Applications With Aspects of Artificial Intelligence, pp. 851–860. Springer, Heidelberg (2007)
6. Gherboudj, A., Chikhi, S.: Algorithme d'OEPB pour Résoudre le Problème du Sac à Dos. In: Laouar, M.R. (ed.) Proceedings of the 1st International Conference on Information Systems and Technologies, ICIST 2011, Tebessa, Algeria, pp. 460–466 (2011) ISBN: 978-9931-9004-0-5
7. He, S., Wu, Q.H., Wen, J.Y., Saunders, J.R., Paton, R.: A Particle Swarm Optimizer with Passive Congregation. Biosystems, 135–147 (2004)
8. Zhong, W., Zhang, J., Chen, W.: A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 3283–3287 (2007)
9. Clerc, M., Kennedy, J.: The Particle Swarm: Explosion, Stability, and Convergence in Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
10. Standard PSO2006, `http://www.particleswarm.info/Programs.html`
11. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Piscatawary, NJ, pp. 4104–4109 (1997)
12. Afshinmanesh, F., Marandi, A., Rahimi-Kian, A.: A novel binary particle swarm optimization method using artificial immune system. In: Proccedings of IEEE international conference on computer as a tool, pp. 217–220 (2005)
13. Liao, C., Tseng, C., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research 34(10), 3099–3111 (2007)
14. Zhan, Z.-h., Zhang, J.: Discrete particle swarm optimization for multiple destination routing problems. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 117–122. Springer, Heidelberg (2009)
15. Eberhart, R.C., Simpson, P., Dobbins, R.: Computational PC Tools, ch. 6, pp. 212-22, AP Professional (1996)