

# A Novel Binary Particle Swarm Optimization

Mojtaba Ahmadih Khanesar, Member, IEEE, Mohammad Teshnehlab and Mahdi Aliyari Shoorehdeli  
K. N. Toosi University of Technology  
Electrical Engineering Faculty  
Control Department  
Tehran, Iran  
Email: ahmadih@ieee.org

**Abstract**— Particle swarm optimization (PSO) as a novel computational intelligence technique, has succeeded in many continuous problems. But in discrete or binary version there are still some difficulties. In this paper a novel binary PSO is proposed. This algorithm proposes a new definition for the velocity vector of binary PSO. It will be shown that this algorithm is a better interpretation of continuous PSO into discrete PSO than the older versions. Also a number of benchmark optimization problems are solved using this concept and quite satisfactory results are obtained.

**Keywords:** Binary Particle Swarm Optimization, Discrete Optimization, Computational Intelligence

## I. INTRODUCTION

Particle swarm optimization (PSO) was originally designed and introduced by Eberhart and Kennedy [1-3] in 1995. The PSO is a population based search algorithm based on the simulation of the social behavior of birds, bees or a school of fishes. This algorithm originally intends to graphically simulate the graceful and unpredictable choreography of a bird folk. Each individual within the swarm is represented by a vector in multidimensional search space. This vector has also one assigned vector which determines the next movement of the particle and is called the velocity vector.

The PSO algorithm also determines how to update the velocity of a particle. Each particle updates its velocity based on current velocity and the best position it has explored so far; and also based on the global best position explored by swarm [5, 6 and 8].

The PSO process then is iterated a fixed number of times or until a minimum error based on desired performance index is achieved. It has been shown that this simple model can deal with difficult optimization problems efficiently.

The PSO was originally developed for continuous valued spaces but many problems are, however, defined for discrete valued spaces where the domain of the variables is finite. Classical examples of such problems are: integer programming, scheduling and routing [5]. In 1997, Kennedy and Eberhart introduced a discrete binary version of PSO for discrete optimization problems [4]. In binary PSO, each particle represents its position in binary values which are 0 or 1. Each particle's value can then be changed (or better say mutate) from one to zero or vice versa. In binary PSO the velocity of a particle defined as the probability that a particle might change its state to one. This algorithm will be discussed in more detail in next sections. Also, the difficulties of binary PSO will be shown in this paper, and then a novel binary PSO algorithm will be proposed. In novel binary PSO proposed

here, the velocity of a particle is its probability to change its state from its previous state to its complement value, rather than the probability of change to 1. In this new definition the velocity of particle and also its parameters has the same role as in continuous version of the PSO. This algorithm will be discussed. Also simulation results are presented to support the idea later in this paper.

There are also other versions of binary PSO. In [6] authors add birth and mortality to the ordinary PSO. AMPSO is a version of binary PSO, which employs a trigonometric function as a bit string generator [9]. Boolean algebra can also be used for binary PSO [10].

Binary PSO has been used in many applications like Iterated Prisoner's Dilemma [11], choosing optimum input subset for SVM [12], design of dual-band dual-polarized planar antenna [10].

This paper is organized as follows: in section II we will consider continuous PSO algorithm and also binary PSO algorithms. Then we will introduce its shortcomings. In section III we will introduce modified PSO and discuss on the logics behind it. In section IV we will examine the algorithm on a number of test problems and we will show that results obtained are quite satisfactory.

## II. THE PARTICLE SWARM OPTIMIZATION

A detailed description of PSO algorithm is presented in [1-3]. Here we will give a short description of the continuous and binary PSO proposed by Kennedy and Eberhart.

### A. Continuous particle swarm optimization

Assume that our search space is d-dimensional, and the i-th particle of the swarm can be represented by a d-dimensional position vector  $X_i = (x_{i1}, x_{i2}, \dots, x_{id})$ . The velocity of the particle is denoted by  $V_i = (v_{i1}, v_{i2}, \dots, v_{id})$ . Also consider best visited position for the particle is  $P_{ibest} = (p_{i1}, p_{i2}, \dots, p_{id})$  and also the best position explored so far is  $P_{gbest} = (p_{g1}, p_{g2}, \dots, p_{gd})$ . So the position of the particle and its velocity is being updated using following equations:

$$v_i(t+1) = w v_i(t) + c_1 \phi_1 (p_i - x_i(t)) + c_2 \phi_2 (p_g - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Where  $c_1$  and  $c_2$  are positive constants, and  $\varphi_1$  and  $\varphi_2$  are two random variables with uniform distribution between 0 and 1. In this equation,  $W$  is the inertia weight which shows the effect of previous velocity vector on the new vector. An upper bound is placed on the velocity in all dimensions  $V_{\max}$ . This limitation prevents the particle from moving too rapidly from one region in search space to another. This value is usually initialized as a function of the range of the problem. For example if the range of all  $x_{ij}$  is  $[-50,50]$  then  $V_{\max}$  is proportional to 50.

$P_{ibest}$  for each particle is updated in each iteration when a better position for the particle or for the whole swarm is obtained. The feature that drives PSO is social interaction. Individuals (particles) within the swarm learn from each other, and based on the knowledge obtained then move to become similar to their "better" previously obtained position and also to their "better" neighbors. Individual within a neighborhood communicate with one other. Based on the communication of a particle within the swarm different neighborhood topologies are defined. One of these topologies which is considered here, is the star topology. In this topology each particle can communicate with every other individual, forming a fully connected social network. In this case each particle is attracted toward the best particle (best problem solution) found by any member of the entire swarm. Each particle therefore imitates the overall best particle. So the  $P_{gbest}$  is updated when a new best position within the whole swarm is found.

The algorithm for the PSO can be summarized as follows:

1. Initialize the swarm  $X_i$ , the position of particles are randomly initialized within the hypercube of feasible space.
2. Evaluate the performance  $F$  of each particle, using its current position  $X_i(t)$ .
3. Compare the performance of each individual to its best performance so far: if  $F(X_i(t)) < F(P_{ibest})$ :  

$$F(P_{ibest}) = F(X_i(t))$$

$$P_{ibest} = X_i(t)$$
4. Compare the performance of each particle to the global best particle: if  $F(X_i(t)) < F(P_{gbest})$ :  

$$F(P_{gbest}) = F(X_i(t))$$

$$P_{gbest} = X_i(t)$$
5. Change the velocity of the particle according to (1).
6. Move each particle to a new position using equation (2).
7. Go to step 2, and repeat until convergence.

#### B. Binary particle swarm optimization

Kennedy and Eberhart proposed a discrete binary version of PSO for binary problems [4]. In their model a particle will decide on "yes" or "no", "true" or "false", "include" or "not to include" etc. also this binary values

can be a representation of a real value in binary search space.

In the binary PSO, the particle's personal best and global best is updated as in continuous version. The major difference between binary PSO with continuous version is that velocities of the particles are rather defined in terms of probabilities that a bit will change to one. Using this definition a velocity must be restricted within the range  $[0,1]$ . So a map is introduced to map all real valued numbers of velocity to the range  $[0,1]$  [4]. The normalization function used here is a sigmoid function as:

$$v'_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}} \quad (3)$$

Also the equation (1) is used to update the velocity vector of the particle. And the new position of the particle is obtained using the equation below:

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r_{ij} < sig(v_{ij}(t+1)) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Where  $r_{ij}$  is a uniform random number in the range  $[0,1]$ .

#### C. Main problems with binary PSO

Here two main problems and concerns about binary PSO is discussed the first is the parameters of binary PSO and the second is the problem with memory of binary PSO.

##### 1) Parameters of the binary PSO

It is not just the interpretation of the velocity and particle trajectories that changes for the binary PSO. The meaning and behavior of the velocity clamping and the inertia weight differ substantially from the real-valued PSO. In fact, the effects of these parameters are the opposite of those for the real valued PSO. In fact, the effects of these parameters are the opposite of those for the real-valued PSO [5].

In continuous version of PSO large numbers for maximum velocity of the particle encourage exploration. But in binary PSO small numbers for  $V_{\max}$  promotes exploration, even if a good solution is found. And if  $V_{\max} = 0$ , then the search changes into a pure random search. Large values for  $V_{\max}$  limit exploration. For example if  $V_{\max} = 4$ , then  $sig(V_{\max}) = 0.982$  is the probability of  $x_{ij}$  to change to bit 1.

There is also some difficulties with choosing proper value for inertia weight  $w$ . For binary PSO, values of  $w < 1$  prevents convergence. For values of  $-1 < w < 1$ ,  $V_{ij}$  becomes 0 over time. For which  $sig(0) = 0.5$  so for  $w < 1$  we have  $\lim_{t \rightarrow \infty} sig(v_{ij}(t)) = 0.5$ . If  $w > 1$  velocity increases over time and  $\lim_{t \rightarrow \infty} sig(v_{ij}(t)) = 1$  so all bits change to 1. If  $w < -1$  then  $\lim_{t \rightarrow \infty} sig(v_{ij}(t)) = 0$  so the probability that bits change to bit 0 increases.

As discussed in [5] the inertia weight and its effect is a problem. Also two approaches are suggested there:

First is to remove the momentum term. According to [5], as the change in particle's position is randomly influenced by  $r_{ij}$ , so the momentum term might not be needed. This approach is unexplored approach although it is used in [7], but no comparisons are provided there.

The second approach is to use a random number for  $w$  in the range:  $(-1, 1)$ .

In fact inertia weight has some valuable information about previously found directions found. Removing this term can't give any improvement to the binary PSO and the previous direction will be lost in this manner. Also using a random number for  $w$  in the range  $(-1, 1)$  or any range like this can't be a good solution. It is desired that the algorithm is quite insensible to the values selected for  $w$ . Also using negative values for  $w$  makes no sense because this term provides the effect of previous directions in the next direction of the particle. Using a negative value for this parameter is not logical.

## 2) Memory of the binary PSO

Considering equation (4) the next value for the bit is quite independent of the current value of that bit and the value is solely updated using the velocity vector. In continuous version of PSO the update rule uses current position of the swarm and the velocity vector just determines the movement of the particle in the space.

### III. THE NOVEL BINARY PARTICLE SWARM OPTIMIZATION

Here, the  $P_{ibest}$  and  $P_{gbest}$  of the swarm is updated as in continuous or binary version. The major difference between this algorithm and other version of binary PSO is the interpretation of velocity. Here, as in continuous version of PSO, velocity of a particle is the rate at which the particle changes its bit's value. Two vectors for each particle are introduced as  $\vec{V}_i^0$  and  $\vec{V}_i^1$ .  $\vec{V}_i^0$  is the probability of the bits of the particle to change to zero while  $\vec{V}_i^1$  is the probability that bits of particle change to one. Since in update equation of these velocities, which will be introduced later, the inertia term is used, these velocities are not complement. So the probability of change in j-th bit of i-th particle is simply defined as follows:

$$V_{ij}^c = \begin{cases} V_{ij}^1 & ,if\ x_{ij} = 0 \\ V_{ij}^0 & ,if\ x_{ij} = 1 \end{cases} \quad (5)$$

In this way the velocity of particle is simply calculated. Also the update algorithm for  $\vec{V}_i^1$  and  $\vec{V}_i^0$  is as follows: consider the best position visited so far for a particle is  $P_{ibest}$  and the global best position for the particle is  $P_{gbest}$ . Also consider that the j-th bit of i-th best particle is one. So to guide the bit j-th of i-th particle to its best position, the velocity of change to one ( $\vec{V}_i^1$ ) for that particle increases and the velocity of change to zero ( $\vec{V}_i^0$ )

is decreases. Using this concept following rules can be extracted:

$$If\ P_{ibest}^j = 1\ Then\ d_{ij,1}^1 = c_1 r_1\ and\ d_{ij,1}^0 = -c_1 r_1$$

$$If\ P_{ibest}^j = 0\ Then\ d_{ij,1}^0 = c_1 r_1\ and\ d_{ij,1}^1 = -c_1 r_1$$

$$If\ P_{gbest}^j = 1\ Then\ d_{ij,2}^1 = c_2 r_2\ and\ d_{ij,2}^0 = -c_2 r_2$$

$$If\ P_{gbest}^j = 0\ Then\ d_{ij,2}^0 = c_2 r_2\ and\ d_{ij,2}^1 = -c_2 r_2$$

Where  $d_{ij}^1, d_{ij}^0$  are two temporary values.  $r_1$  and  $r_2$  are two random variable in the range of  $(0,1)$  which are updated each iteration. Also  $c_1, c_2$  are two fixed variables which are determined by user. then:

$$V_{ij}^1 = w V_{ij}^1 + d_{ij,1}^1 + d_{ij,2}^1 \quad (6)$$

$$V_{ij}^0 = w V_{ij}^0 + d_{ij,1}^0 + d_{ij,2}^0 \quad (7)$$

Where  $w$  is the inertia term. In fact in this algorithm if the j-th bit in the global best variable is zero or if the j-th bit in the corresponding personal best variable is zero the velocity ( $V_{ij}^0$ ) is increased. And the probability of changing to one is also decreases with the same rate. In addition, if the j-th bit in the global best variable is one  $V_{ij}^1$  is increased and  $V_{ij}^0$  decreases. In this approach previously found direction of change to one or change to zero for a bit is maintained and used so particles make use of previously found direction. After updating velocity of particles,  $\vec{V}_i^0$  and  $\vec{V}_i^1$ , the velocity of change is obtained as in (5).

A normalization process is also done. Using sigmoid function as introduced in (3). And then the next particles state is computed as follows:

$$x_{ij}(t+1) = \begin{cases} \bar{x}_{ij}(t) & ,if\ r_{ij} < V_{ij}^1 \\ x_{ij}(t) & ,if\ r_{ij} > V_{ij}^1 \end{cases} \quad (8)$$

Where  $\bar{x}_{ij}$  is the 2's complement of  $x_{ij}$ . That is, if  $x_{ij} = 0$  then  $\bar{x}_{ij} = 1$  and if  $x_{ij} = 1$  then  $\bar{x}_{ij} = 0$ . And  $r_{ij}$  is a uniform random number between 0 and 1.

The meaning of the parameters used in velocity equation, are exactly like those for the continuous PSO. The inertia weight used here maintains the previous direction of bits of particle to the personal best bit or

global best bit whether it is 1 or 0. Also the meaning of velocity is the same as meaning of the velocity in continuous version of PSO which is the rate of change in particle's position. Also as in continuous PSO if the maximum velocity value considered is large, random search will happen. Small values for maximum velocity cause the particle to move less. Here also the previous states of the bits of the particles are taken into account. Using the equation (7) the previous value of the particle is taken into account, while in binary PSO just velocity determined the next value of particle. So, better performance and better learning from experiments in this algorithm is achieved. Experimental results in the next section support these complain.

The algorithm proposed here for the binary PSO can be summarized as follows:

1. Initialize the swarm  $X_i$ , the position of particles are randomly initialized within the hypercube. Elements of  $X_i$  are randomly selected from binary values 0 and 1.
2. Evaluate the performance  $F$  of each particle, using its current position  $X_i(t)$ .
3. Compare the performance of each individual to its best performance so far: if  $F(X_i(t)) < F(P_{ibest})$ :

$$F(P_{ibest}) = F(X_i(t))$$

$$P_{ibest} = X_i(t)$$

4. Compare the performance of each particle to the global best particle: if  $F(X_i(t)) < F(P_{gbest})$ :

$$F(P_{gbest}) = F(X_i(t))$$

$$P_{gbest} = X_i(t)$$

5. Change the velocity of the particle,  $\vec{V}_i^0$  and  $\vec{V}_i^1$  according to (6,7).
6. Calculate the velocity of change of the bits,  $\vec{V}_i^c$  as in (5).
7. Generate the random variable  $r_{ij}$  in the range: (0,1). Move each particle to a new position using equation (8).
8. Go to step 2, and repeat until convergence.

#### IV. EXPERIMENTAL RESULTS

In this section we will compare the performance of proposed binary PSO and the binary PSO proposed by Kennedy and Eberhart in [4] and the binary PSO used in [7]. In our experiments we investigated methods on the minimization of test functions set which is proposed in [4]. The functions used here are: Sphere, Rosenbrock, Griewangk and Rastrigin which are represented in equations (9-12) respectively. The global minimum of all of these functions is zero. The expression of these test functions are as follows:

$$f_1(x) = \sum_{i=1}^N x_i^2 \quad (9)$$

$$f_2(x) = \sum_{i=1}^{N-1} \left( 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \quad (10)$$

$$f_3(x) = \frac{1}{4000} \sum_{i=1}^N x_i^2 - \prod_{i=1}^N \cos \frac{x_i}{\sqrt{i}} + 1 \quad (11)$$

$$f_4(x) = \sum_{i=1}^N \left( x_i^2 - 10 \cos(2\pi x_i) + 10 \right) \quad (12)$$

These functions have been used by many researchers as benchmarks for evaluating and comparing different optimization algorithms. In all of these functions N is the dimension of our search space. In our experiments the range of the particles were set to  $[-50, 50]$  and 20 bits are used to represent binary values for the real numbers. Also population size is 100 and the number of iteration assumed to be 1000. The different values assumed in tests for N are 3,5,10, where N is the dimension of solution space.

As it is shown in Table (1-8), the results are quite satisfactory and much better than the algorithms proposed in [4] and [7]. As it was mentioned earlier, the method proposed here uses the previous direction found effectively and velocity has the same interpretation as the continuous PSO, which is the rate of changes. The method of selecting inertia weight in binary PSO proposed in [4] is still a problem [5]. But removing the inertia weight is also undesirable because the previous direction is completely losses. In fact the previous velocities of a particle contain some information about the direction to previous personal best and global bests of the particle and surely have some valuable information which can help us faster and better find the solution. But in the proposed algorithm the effect of previous direction and also the effect of previous state of the system is completely taken into account. The results obtained here quite support the idea.

#### V. CONCLUSION

In this study a new interpretation for the velocity of binary PSO was proposed, which is the rate of change in bits of particles. Also the main difficulty of older version of binary PSO which is choosing proper value for  $w$  is solved. The previous direction and previous state of each particle is also taken into account and helped finding good solutions for problems. This approach tested and returned quite satisfactory results in number of test problems.

The binary PSO can be used in variety of applications, especially when the values of the search space are discrete like decision making, solving lot sizing problem, the traveling salesman problem, scheduling and routing.

#### REFERENCES

- [1] R. Eberhart, and J. Kennedy, A New Optimizer Using Particles Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, pp. 39-43, 1995.
- [2] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization", IEEE International Conference on Neural Networks (Perth,

- Australia), IEEE Service Center, Piscataway, NJ, IV, pp. 1942-1948, 1995.
- [3] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2001.
- [4] Kennedy, J.; Eberhart, R.C. "A discrete binary version of the particle swarm algorithm", IEEE International Conference on Systems, Man, and Cybernetics, 1997.
- [5] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005
- [6] J. Sadri, and Ching Y. Suen, "A Genetic Binary Particle Swarm Optimization Model", IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 2006
- [7] M. Fatih Tasgetiren. & Yun-Chia Liang, "A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem" ,*Journal of Economic and Social Research* 5 (2), pp. 1-20
- [8] A. P. Engelbrecht, "computational Intelligence", John Wiley and Sons, 2002
- [9] Pampara, G. ,Franken, N. ,Engelbrecht, A.P. "Combining particle swarm optimisation with angle modulation to solve binary problems", IEEE Congress on Evolutionary Computation, 2005 pp 89-96
- [10] Marandi, A., Afshinmanesh, F., Shahabadi, M., Bahrami, F., "Boolean Particle Swarm Optimization and Its Application to the Design of a Dual-Band Dual-Polarized Planar Antenna", CEC 2006, pp. 3212-3218
- [11] Franken, N., Engelbrecht, A.P., "Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma", IEEE Transactions on Evolutionary Computation, 2005 pp.562 - 579
- [12] Chunkai Zhang; Hong Hu, "Using PSO algorithm to evolve an optimum input subset for a SVM in time series forecasting", IEEE International Conference on Systems, Man and Cybernetics, 2005 pp. 3793-3796

TABLE I. THE RESULTS OF BEST GLOBAL BEST OF MINIMIZATION OF SPHERE FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$6.8212 \times 10^{-9}$	0.0561	0.1545
N = 5	$1.9213 \times 10^{-6}$	7.9578	22.8995
N = 10	0.1121	216.6069	394.7066

TABLE II. THE RESULTS OF BEST MEAN OF PERSONAL BESTS FOR MINIMIZATION OF SPHERE FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$2.5739 \times 10^{-8}$	9.2145	0.1542
N = 5	$5.2909 \times 10^{-4}$	171.5407	224.4042
N = 10	1.9819	1532.9	1718.3

TABLE III. THE RESULTS OF BEST GLOBAL BEST OF MINIMIZATION OF ROSENBRCK FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	0.0934	0.9384	0.8645
N = 5	2.2470	1406	3746.5
N = 10	32.8310	$1.3094 \times 10^6$	$1.52321 \times 10^6$

TABLE IV. THE RESULTS OF BEST MEAN OF PERSONAL BESTS FOR MINIMIZATION OF ROSENBRCK FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	0.5164	837.6181	2945.8
N = 5	2.5162	304210	600530
N = 10	367.8391	$3.6247 \times 10^7$	$5.0179 \times 10^7$

TABLE V. THE RESULTS OF BEST GLOBAL BEST OF MINIMIZATION OF GRIENWANGK FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$2.0860 \times 10^{-9}$	0.003	0.0277
N = 5	$7.4 \times 10^{-3}$	0.2113	0.1503
N = 10	0.0579	0.8282	1.0254

TABLE VI. THE RESULTS OF BEST MEAN OF PERSONAL BESTS FOR MINIMIZATION OF GRIEWANGK FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$3.7825 \times 10^{-8}$	0.1716	0.2025
N = 5	0.0125	0.5824	0.6574
N = 10	0.3009	1.3864	1.4333

TABLE VII. THE RESULTS OF BEST GLOBAL BEST OF MINIMIZATION OF RASTRIGRIN FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$1.3533 \times 10^{-6}$	2.6693	3.7127
N = 5	0.0034	25.8756	51.3154
N = 10	10.3925	490.8208	539.3371

TABLE VIII. THE RESULTS OF BEST MEAN OF PERSONAL BESTS FOR MINIMIZATION OF RASTRIGRIN FUNCTION IN 10 TIMES OF THE RUN OF ALGORITHMS

Dimension of the input space	The Novel PSO algorithm	Binary PSO as in [4]	Binary PSO as in [7]
N = 3	$6.5138 \times 10^{-6}$	32.0306	46.7851
N = 5	0.3799	215.5889	268.3968
N = 10	39.1396	1664.3	1820.2