

A Novel Probability Binary Particle Swarm Optimization Algorithm and Its Application

Ling Wang*

School of Mechatronics and Automation, Shanghai University, Shanghai, China
Email: wangling@shu.edu.cn

Xiuting Wang, Jingqi Fu and Lanlan Zhen

School of Mechatronics and Automation, Shanghai University, Shanghai, China
Email: wangxiuting521@hotmail.com

Abstract—Particle swarm optimization (PSO), an intelligent optimization algorithm inspired by the flocking behavior of birds, has been shown to perform well and widely used to solve the continuous problem. But the traditional PSO and most of its variants are developed for optimization problems in continuous space, which are not able to solve the binary combinational optimization problem. To tackle this problem, Kennedy extended the PSO and proposed a discrete binary PSO. But its performance is not ideal and just few further works were conducted based on it. In this paper, we propose a novel probability binary particle swarm optimization (PBPSO) algorithm for discrete binary optimization problems. In PBPSO, a novel updating strategy is adopted to update the swarm and search the global solution, which further simplify the computations and improve the optimization ability. To investigate the performance of the proposed PBPSO, the multidimensional knapsack problems are used as the test benchmarks. The experimental results demonstrate that PBPSO has a better performance for solving the multidimensional knapsack problem in terms of convergent speed and global search ability.

Index Terms—particle swarm optimization, probability optimization algorithm, knapsack problem

I. INTRODUCTION

Particle swarm optimization (PSO), a parallel evolutionary computation technique originally designed by Kennedy and Eberhart in 1995 [1], is an emerging population-based optimization method. PSO is a kind of random search algorithm that simulates nature evolutionary process and performs good characteristic in solving some difficulty optimization problems. The basic concept of PSO comes from a large number of birds flying randomly and looking for food together. Each bird is an individual and called a particle. As the birds looking for food, the particles fly in a multidimensional search space looking for the optimal solution. Here all the particles are composed of a family rather than the isolated individual for each other. They can remember their own

flying experience and share their companions' flying experience. Shi and Eberhart [2] called the former the cognition part and the latter the social part. According to the cognitive memory, all the particles can adjust their position moving toward their global best position or their neighbor's local best position. Therefore, PSO has much more profound intelligent background than the genetic algorithm. Also PSO could be executed more easily. On the one hand, PSO has very few parameters to adjust, so that it is convenient to make the parameters reach to the optimum values, a large amount of calculation work and much time can be saved. On the other hand, PSO can find the optimal solutions or near the optimal solutions with a fast convergent speed, because it only has two computation formulas for iteration. Based on these advantages, PSO is not only used for the science research, but also is applied in engineering field, such as neural networks [3] [4], nonlinear constrained optimization problems [5], power flow problem [6]

But the basic PSO and most of its improved versions all work in the continuous space, which can not be used to optimize the discrete combinational problems. To tackle this problem, discrete PSO is developed and applied in many fields. Ching-Jong Liao utilized a discrete version of particle swarm optimization for the flowshop scheduling problem [7]. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem is investigated by Tasgetiren et al. [8], where a new position update method is developed based on the discrete domain. Jin Y.X. et al. [9] provided a new discrete PSO methodology applied in transmission network expansion planning. Jarboui B., Damak N. and Siarry P. took up a challenge to apply a new combinatorial particle swarm optimization (CPSO) algorithm to solve a large variety of combinatorial optimization problems, namely the multi-mode resource-constrained project scheduling problem using integer values [10]. But there are just few researches on PSO for discrete binary combinational problems. Kennedy and Eberhart firstly extended the basic PSO and developed a discrete binary PSO (KBPSO) to optimize the binary optimization algorithm in 1997 [11]. And then Qi [12]

* Corresponding author

developed a modified binary PSO (MBPSO) for feature selection in MLR and PLS modeling. As the optimization ability is not perfect, just few researches [13] are conducted based on them.

To remedy this drawback, in this paper we propose a novel probability binary particle swarm optimization (PBPSO) to solve discrete binary optimization problems. The remainder of the paper is organized as follows. Section 2 firstly describes the traditional binary particle swarm optimization algorithm. Then we propose our PBPSO algorithm in detail. The overall scheme of the proposed probability binary particle swarm optimization for solving the multidimensional knapsack problem is presented in section 3. Parameter studying of PBPSO, optimization performances and experimental comparisons of PBPSO algorithm for the multidimensional knapsack problems are given in section 4. Finally, some concluding remarks are given in section 5.

II. A PROBABILITY BINARY PARTICLE SWARM OPTIMIZATION ALGORITHM

A. The Discrete Particle Swarm Optimization Algorithm

Supposing that the real value searching space is D -dimensional and m particles form a swarm, each particle is looked as a point in the D -dimensional space, and the i -th particle represents a D -dimensional vector $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. It means that each particle is a potential global optimum of the function $f(x)$ over a given domain D . Here $f(x)$ is used for evaluating the particle, using the particle's positional coordinates as input values. The output value often is named fitness value, and according to the fitness value, the particle is updated to move towards the better area by the corresponding operators till the best point is found. In the iterative process, each particle's previous best position is remembered and denoted $p_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the globally best position in the whole swarm is recorded as $p_g = (p_{g1}, p_{g2}, \dots, p_{gD})$. The i -th particle's "flying" velocity is also a D -dimensional vector, is represented as $v_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ ($i = 1, 2, \dots, m$). At each step, the velocity of all particles is adjusted as a sum of its local best value, global best value and its present velocity, multiplied by the three constants ω , c_1 , c_2 respectively, shown in (1); the position of each particle is also modified by adding its velocity to the current position, see (2),

$$v_{ij}^{k+1} = \omega \times v_{ij}^k + c_1 \times r_1 \times (p_{ij}^k - x_{ij}^k) + c_2 \times r_2 \times (p_{gj}^k - x_{ij}^k), \quad (1)$$

$$x_{ij}^{k+1} = x_{ij}^k + v_{ij}^k. \quad (2)$$

In (1-2), k represents the iterative number, ω is the inertia weight, c_1 and c_2 are learning factors, normally set as $c_1 = c_2 = 2.0$, $r_1, r_2 \in \text{uniform} [0, 1]$. The random weighting of the control parameters in the algorithm results in a kind of explosion or a "drunkard's walk" as particles' velocities and positional coordinates careen toward infinity [14]. Generally, the particle velocity is limited by some maximum value v_{max} .

As the basic particle swarm optimization operates in continuous and real number space, it can not be used to optimize the pure discrete binary combinational problem. To tackle this problem, Kennedy and Eberhart proposed KBPSO algorithm, where the particles take the values of binary vectors of length n and the velocity defined the probability of bit x_{ij} to take the value 1. KBPSO reserved the updating formula of the velocity (see (1)) while velocity was constrained to the interval $[0.0, 1.0]$ by a limiting transformation function, that is, the particle changes its bit value by (3-4) in KBPSO

$$S(v_{ij}) = 1 / (1 + e^{-v_{ij}}), \quad (3)$$

$$x_{ij} = \begin{cases} 1 & \text{if rand() } \leq S(v_{ij}) \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where the value of $\text{rand}()$ drawn from $U(0,1)$ and the function $S(v)$ is a sigmoid limiting transformation.

As the optimization ability of KBPSO is not ideal, Qi developed a modified discrete binary PSO algorithm (MBPSO). In MBPSO, the updating formulas are defined by the following equations [13]

$$\text{If } (0 < v_{ij} \leq \alpha) \text{ then } x_{ij}(\text{new}) = x_{ij}(\text{old}), \quad (5)$$

$$\text{If } (\alpha < v_{ij} \leq \frac{1}{2}(1+\alpha)) \text{ then } x_{ij}(\text{new}) = p_{ij}, \quad (6)$$

$$\text{If } (\frac{1}{2}(1+\alpha) < v_{ij} \leq 1) \text{ then } x_{ij}(\text{new}) = g_{ij}, \quad (7)$$

where α named static probability is a random value in the range of $(0,1)$. The initial value of α is 0.5.

B. The Probability Binary Particle Swarm Optimization Algorithm

As KBPSO and MBPSO are very easy to trap in the local optimum and KBPSO updating formulas is complexity, we propose a novel algorithm named probability binary particle swarm optimization algorithm (PBPSO) to tackle these problems in this paper. Here (1) and (2) are all reserved for iterative evolution in PBPSO, and then we adopt a novel formula to determine a binary bit px_{ij} , which can be denoted as follows:

$$L(x_{ij}) = (x_{ij} - R_{\min}) / (R_{\max} - R_{\min}), \quad (8)$$

$$px_{ij} = \begin{cases} 1 & \text{if rand() } \leq L(x_{ij}) \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

where $L(x)$ is a linear function, its output value belongs to $(0,1)$; $\text{rand}()$ is a stochastic number selected from a uniform distribution in $[0.0, 1.0]$; and $[R_{\max}, R_{\min}]$ is a pre-defined range for gaining the probability value with $L(x)$ function.

Compare to KBPSO, now the vector x_i is real number vector rather than the binary vector. To obtain a probability value distributed in $[0.0, 1.0]$, we adopt the linear function $L(x)$ to calculate it, which determines px_{ij}

to be 1 or 0. For instance, if $L(x_{ij})=0.6$, $\text{rand}()=0.3$, that is, $\text{rand}() < L(x_{ij})$, then $px_{ij}=1$, ($i=1,2,\dots,m$ and $j=1,2,\dots,D$). So a binary vector $px_i=(px_{i1}, px_{i2}, \dots, px_{iD})$ can be worked out, and then we can apply this binary vector into the combinatorial optimization problem.

In summary, the flowchart of the PBPSO is given in Fig. 1, where P is on behalf of the population size, G denotes the maximum iterative times.

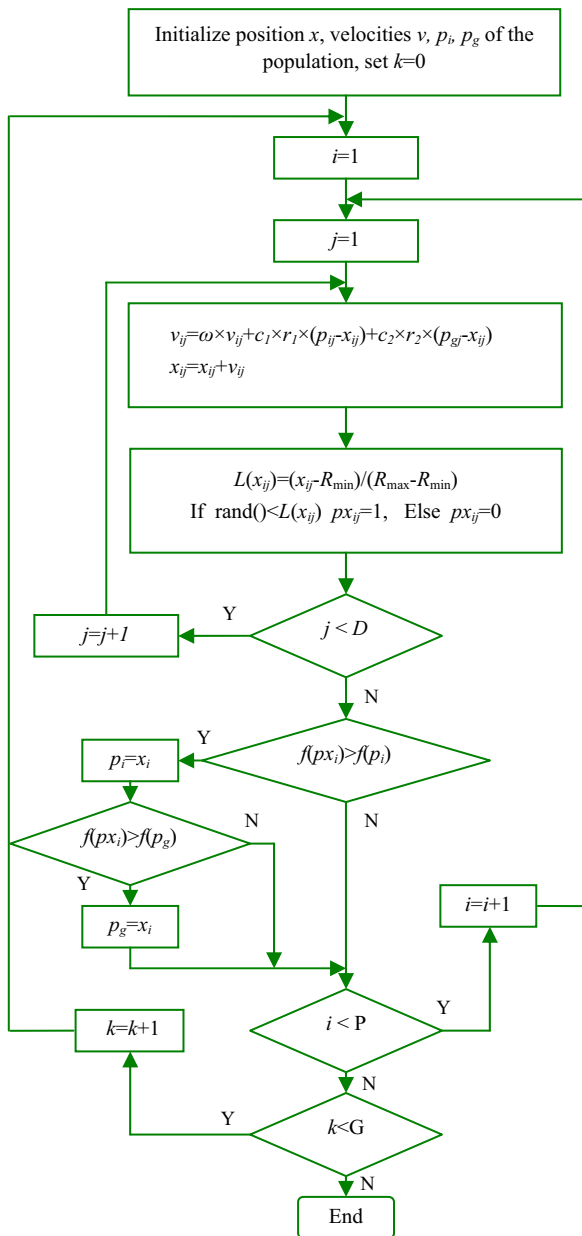


Figure 1. The flowchart of PBPSO.

III. THE PBPSO FOR THE MULTIDIMENSIONAL KNAPSACK PROBLEM

In this section, we first define the multidimensional knapsack problem (MKP) and then the PBPSO for solving MKP is presented

A. Problem Definition

The multidimensional knapsack problem (MKP) is a generalization of the 0-1 knapsack problem and a special case of general 0-1 integer programming. It also can be considered as a resource allocation problem. Generally, The MKP can be introduced as follows: suppose we have m resources and n objects, each resource $i \in I$ has a budget M_i , each object $j \in J$ has a profit p_j and consumes w_{ij} of resource i . The problem is to maximize the profit within a limited budget. The MKP can be formulated as [15]

$$\text{Maximize } f(x) = \sum_{j=1}^n p_j x_j, \quad (10)$$

$$\text{Subject to } \sum_{j=1}^n w_{ij} x_j \leq M_i \quad i=1, \dots, m, \quad (11)$$

$$x_j \in \{0,1\}, \quad j=1, \dots, n,$$

$$\text{With } p_j > 0, w_{ij} \geq 0, M_i \geq 0.$$

The constraint described in the second condition is called a knapsack constraint, so the MKP is also called the m -dimensional knapsack problem. If $m=1$, MKP is often named the one-dimensional knapsack problem (KP). Let $I = \{1, 2, \dots, m\}$ and $J = \{1, 2, \dots, n\}$, for all $j \in J$, $w_j \leq M < \sum_{j=1}^n w_j$. In the above formulas, x is a binary vector, that is, $x=(x_1, \dots, x_j, \dots, x_n)$, and each bit of x equals zero or one. $x_j=1$ represents the i -th resource is consumed w_{ij} , $x_j=0$ represents the corresponding resource is rejected. Taking a simple multidimensional knapsack instance for example, we give the process of consuming each resource with the selected objects in Fig. 2.

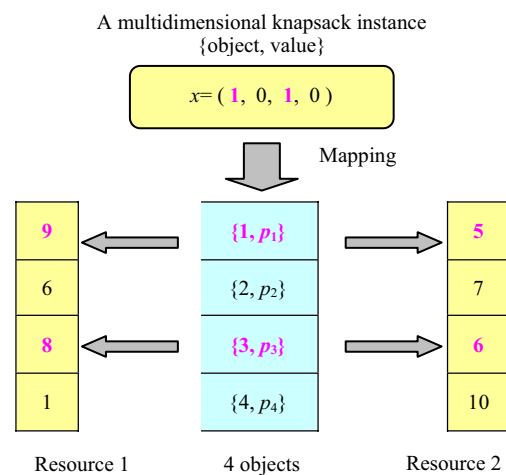


Figure 2. The process of consuming each resource with the selected objects

In the resource distributive process, the total consumption of resources chosen for the objects could exceed the maximum budget allowed. Various methods have been studied and used for dealing with constrains such as penalty function method [16], greed algorithm [17] and surrogate duality method [18]. On the one hand,

the heuristic algorithm, such as surrogate duality method, can greatly improve the quality of solution as the domain knowledge is introduced, but this process will cost much time. On the other hand, the penalty function method is easy but not helpful to find better solution additionally. To evaluate the optimization ability of PBPSO, the approach examined in this paper employs a penalty function to adjust the fitness of illegal individuals by optimizing [16]

$$F(x) = f(x) - \sum_{i=1}^m \Phi_i, \quad (12)$$

where Φ_i is the penalty function; m is the number of constrains, that is, the number of resources or the amount of knapsacks; the minus means the problem for maximization; $f(x)$ is the fitness function that gives a fitness value for each solution, and $F(x)$ is the penalized fitness value when a resource exceed its budget.

B. PBPSO for the MKP

When we use PBPSO algorithm to optimize the knapsack problem, firstly, all the particles are initialized with the random numbers, then take the formulas for evolution until the terminal rule coming.

The whole procedure of PBPSO optimizing the multidimensional knapsack problem can be described as follows:

- Step 1: Initialize the swarm and parameters of PBPSO; the pop_size is the population size to be kept constant through the generations. The inertia weight is set as $\omega=0.8$, and the learning factors are set as $c_1=c_2=2.0$.
- Step 2: Evaluate the fitness values according the fitness function; in this paper, the fitness value is defined by the total profit and calculated by (10) and (12). According the fitness value, we will decide whether the solution is good or not.
- Step 3: Update the local optima and the global optimum;
- Step 4: Update particles according the updating rules of PBPSO;
- Step 5: If the terminal rule is satisfied, stop the iteration and output the results, otherwise go to step2.

IV EXPERIMENT RESULTS AND DISCUSSION

A. Parameter Studying

Obviously, the parameters of PBPSO will seriously affect the real optimization performance. To know the PBPSO well, we study and test the parameters of PBPSO. On the one hand, as the previous works have proven that the traditional values of parameters in PSO can keep algorithm work well, the same values of c_1 , c_2 and w in KBPSO are used in PBPSO. On the other hand, the bound $[R_{max}, R_{min}]$ is a new parameter in PBPSO. To determine the value settings of R_{max} and R_{min} , the MKP is adopted as the test benchmark for parameter studying. Here we adopted two type MKP instances with different dimension as the argument for analyzing. One instance is weing1.dat and the other one is sent1.dat.

To the first benchmark test, we set pop_size=30, iterations=3000; for the second benchmark test, the pop_size is 60, iterations=4000. Other parameters are all set as the default values of KBPSO and the different range $[R_{min}, R_{max}]$ are adopted for parameter studying. These two benchmarks both run 20 times. The testing results are shown in Table I and Table II. The terms that emerge in the tables are introduced in the following:

- Worst Fitness--the worst fitness value in the 20 times running.
- Best Fitness --the best fitness value in the 20 times running.
- Success Rate--the percent of finding the optimal solution in the 20 times running.
- Average Fitness--the average fitness value of all the 20 times running.
- Min iteration--the minimum step of finding the optimal solution in the 20 times running.
- Average iteration--the average step of finding the optimal solution of all the 20 times running.

From the Table I-II, we can find PBPSO is not sensitive to the parameter $[R_{min}, R_{max}]$ as its optimization performances are similar with different values. But it is obvious that too small value of R_{max} and R_{min} is harmful to PBPSO as algorithm can not perform meticulous search well, and the optimization results are both poorest in two cases. The simulation results show that the $R_{min}=50$ and $R_{max}=50$ may be encouraged as PBPSO both achieve the best optimization results in two testing benchmark. With the further increasing of R_{max} and R_{min} , the optimization performances of PBPSO decrease gradually.

TABLE I.
THE TEST RESULTS OF WEING1.DAT

$[R_{min}, R_{max}]$	Worst Fitness	Best Fitness	Success Rate	Average Fitness	Average iteration	Min iteration
[-1, 1]	141258	141278	95%	141241.25	492.68	72
[-5, 5]	141278	141278	100%	141278	374.45	66
[-10, 10]	141278	141278	100%	141278	536.35	72
[-20, 20]	141278	141278	100%	141278	566.12	89
[-50, 50]	141278	141278	100%	141278	632.1	124
[-100, 100]	141278	141278	100%	141278	390.3	59
[-200, 200]	141278	141278	100%	141278	749.55	151
[-500, 500]	141278	141278	100%	141278	445.2	99
[-1000, 1000]	141278	141278	100%	141278	540.05	149
[-10000, 10000]	141278	141278	100%	141278	596.5	92

TABLE II.
THE TEST RESULTS OF SENT1.DAT

$[R_{min}, R_{max}]$	Worst Fitness	Best Fitness	Success Rate	Average Fitness	Average iteration	Min iteration
[-1, 1]	7479	7761	0	7636.2	920	920
[-5, 5]	7518	7772	5%	7665.6	2828	2828
[-10, 10]	7552	7772	5%	7694.85	2692	2692
[-20, 20]	7549	7772	5%	7692.7	2968	2968
[-50, 50]	7585	7772	5%	7695.9	3158	3158
[-100, 100]	7460	7739	0	7662.5	953	953
[-200, 200]	7473	7749	0	7654.1	2383	2383
[-500, 500]	7569	7762	0	7678.1	3025	3025
[-1000, 1000]	7542	7762	0	7682.9	1787	1787
[-10000, 10000]	7550	7761	0	7671.2	2497	2497

B. Results and Discussion

To evaluate the effectiveness and viability of PBPSO algorithm, we take more MKP benchmarks to test its performance, and its simulation results are compared with other discrete binary PSO versions that include KBPSO and MBPSO. All the benchmarks are selected from *MP-Test-data SAC-94 Suite*. According the complexity of the problems, the benchmarks are divided into two groups. We take Pet1.dat, Pet2.dat, Pet3.dat, Pb4.dat, Pb5.dat and Pb6.dat as the simple group. The population size of algorithms is equal to 30, the maximum iteration is 3000. The Sent1.dat, Sent2.dat, Weish12.dat and Weish20.dat form the complicated testing group, and the size of the swarm is set as 60 and the maximum generation is 4000. To make a fair comparison, we set the same parameter values for both KBPSO and PBPSO, that is, $c_1=c_2=2.0$ and $\omega=0.8$; and α is set as 0.5 in MBPSO. According to the results of parameter studying presented above, $[R_{min}, R_{max}] = [-50, 50]$ are adopted in PBPSO. PBPSO, KBPSO and MBPSO all executed 20 times on each case, respectively. The experimental results of three algorithms on each

benchmark problem are shown in Table III-IV where n and m mean objects and resource constrains.

For all testing benchmarks, Pet1, Pet2 and Pet3 are the easiest instances, and their optimization results of three algorithms are extremely similar, so the average fitness values generated by KBPSO, MBPSO and PBPSO are not depicted here. The average fitness values of other benchmarks are depicted in Fig.3-9.

Table III shows that all three algorithms are valid and can obtain the global best solution, but the PBPSO is prior to MBPSO and KBPSO as it has the best average fitness values. The same conclusion can be made from Table IV. In most cases, MBPSO performs better than KBPSO as its updating strategy is more effectively. But it is obvious that MBPSO is more likely to stick in the local optimal as its value of bit will be not changed if it has the same value of its two optimal solutions. So MBPSO sometimes will be trapped in a poor local optimum which makes its average fitness value worse than KBPSO. To KBPSO algorithm, as its updating strategy spoils its velocity updating, its optimization ability is unsatisfied, especially for the complicated cases. These conclusions can be validated from Fig. 3-9.

TABLE III.
OPTIMIZATION RESULTS OF KBPSO, MBPSO AND PBPSO ON THE SIMPLE GROUP

Benchmark	Best Known	Algorithm	Optimization Code of particle's position	Best Fitness	Success Rate	Average Fitness	Average iteration	Min iterations
Pet1 m=10 n=6	3800	KBPSO	011001	3800	100%	3800	2.3	1
		MBPSO	011001	3800	100%	3800	1.2	1
		PBPSO	011001	3800	100%	3800	1	1
Pet2 m=10 n=10	8706.1	KBPSO	0101100101	8706.1	100%	8706.1	25	4
		MBPSO	0101100101	8706.1	100%	8706.1	53	2
		PBPSO	0101100101	8706.1	100%	8706.1	25	8
Pet3 m=10 n=15	4015	KBPSO	110101101100011	4015	100%	4015	61	11
		MBPSO	110101101100011	4015	100%	4015	268	5
		PBPSO	110101101100011	4015	100%	4015	92	10
Pb4 m=2 n=29	95168	KBPSO	1110111101110011010100000000	95168	20%	91879.15	1304	438
		MBPSO	1110111101110011010100000000	95168	15%	92419	1744	540
		PBPSO	1110111101110011010100000000	95168	40%	93114.1	729	181
Pb5 m=10 n=20	2139	KBPSO	010101010101010101	2139	65%	2131.1	1078	97
		MBPSO	010101010101010101	2139	5%	2110.9	90	90
		PBPSO	010101010101010101	2139	75%	2134.45	1012	129
Pb6 m=30 n=40	776	KBPSO	0110000000011000010110000010000000000001	776	10%	746.95	2640	2495
		MBPSO	0110000000011000010110000010000000000001	776	10%	708.60	642	143
		PBPSO	0110000000011000010110000010000000000001	776	15%	752.85	937	312

TABLE IV.
OPTIMIZATION RESULTS OF KBPSO, MBPSO AND PBPSO ON THE COMPLEX GROUP

Benchmark	Best Known	Algorithm	Optimization Code of particle's position	Best Fitness	Success Rate	Average Fitness	Average iteration	Min iteration
Sent1 m=30 n=60	7772	KBPSO	010010011000110101100000101001 000000100001000110000010010010	7676	0%	7562.4	1723	1723
		MBPSO	010010010000110101100010100101 000001101000100110000010010010	7762	0%	7683.55	1447	1447
		PBPSO	010010011010110101101000101001 000000100000000110000010010010	7772	5%	7695.9	3158	3158
Sent2 m=30 n=60	8722	KBPSO	111100110101101111010101010101 000101101001010111000010011010	8655	0%	8603.5	3355	3355
		MBPSO	011010011010110111110010101111 000101101010111110000010011010	8711	0%	8651	1740	1740
		PBPSO	011110011010111111101000101101 000101101001011111000010011010	8722	5%	8671.1	2221	2221
Weish12 m=5 n=50	6339	KBPSO	1001101000111100010011000 1000110101000001010000111	6339	15%	6295.1	2600	2183
		MBPSO	1001101000111100010011000 1000110101000001010000111	6339	35%	6317.05	2162	890
		PBPSO	1001101000111100010011000 1000110101000001010000111	6339	45%	6331.75	1436	112
Weish20 m=5 n=70	9450	KBPSO	1000101010111011010100011 00100011010000011011000 011110000101111011010011	9146	0%	9092.05	2296	2296
		MBPSO	10001010111110010110011 00100011000100010011000 011111000101011110011011	9445	0%	9352.95	1367	1367
		PBPSO	10001010111110010110011 0010001100100010011000 011110000101011110011011	9450	5%	9362.05	3652	3652

Simulation results display that KBPSO and MBPSO have difficulty dealing with the large size MKP problems. From Table IV, we can see that KBPSO and MBPSO cannot find the global optima in Sent1, Sent2 and Weish20 while PBPSO is still able to find the global best

solutions. To improve its results, it is essential to introduce some heuristic local search algorithms [18]. As the main focus of this paper is PBPSO, here we just adopted the penalty function strategy.

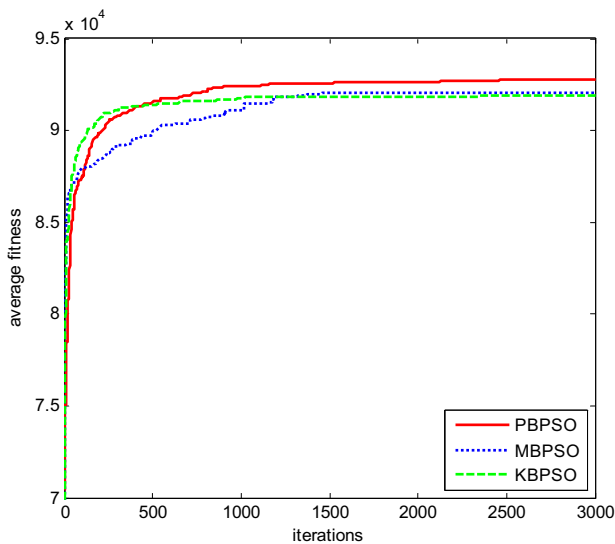


Figure 3. The average fitness value of KBPSO, MBPSO and PBPSO on Pb4.dat

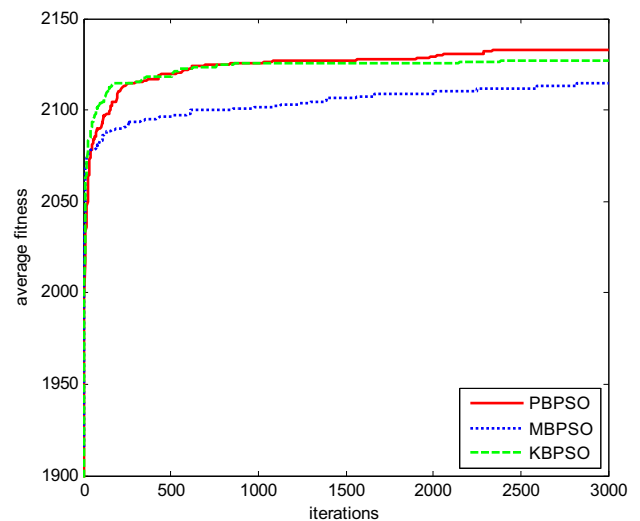


Figure 4. The average fitness value of KBPSO, MBPSO and PBPSO on Pb5.dat

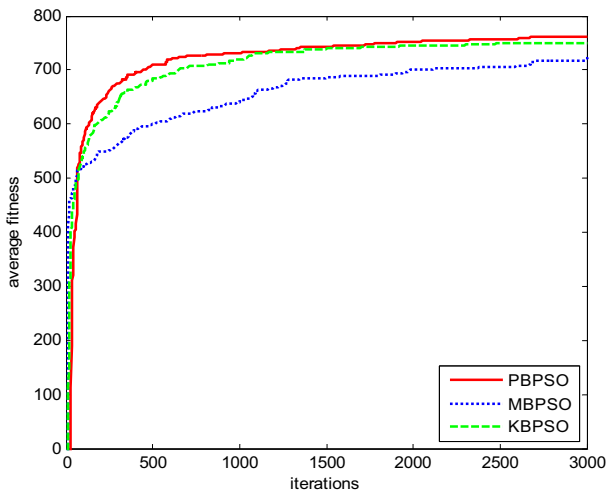


Figure 5. The average fitness value of KBPSO, MBPSO and PBPSO on Pb6.dat

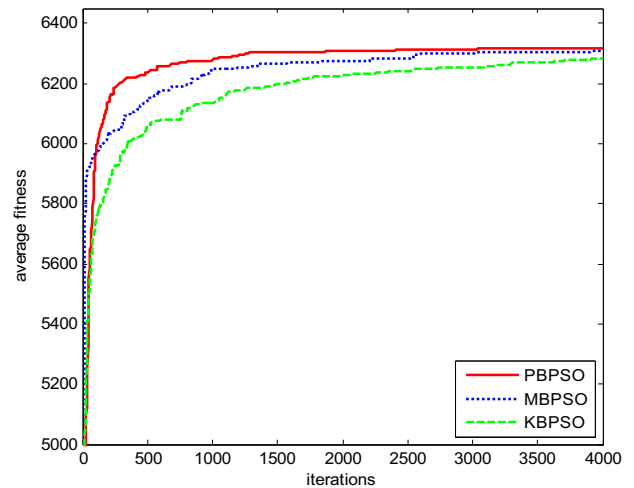


Figure 8. The average fitness value of KBPSO, MBPSO and PBPSO on Weish12.dat

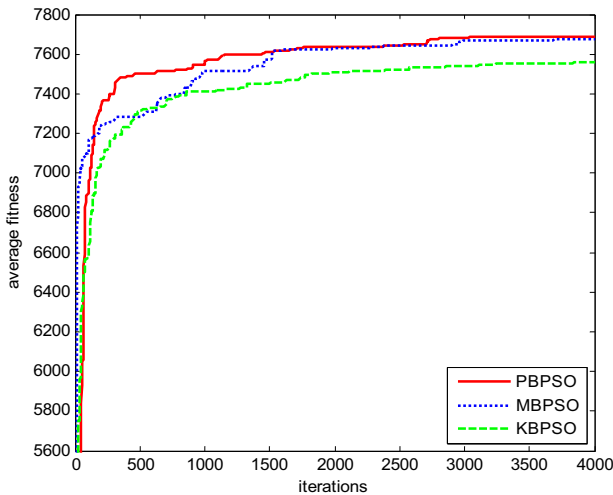


Figure 6. The average fitness value of KBPSO, MBPSO and PBPSO on Sent1.dat

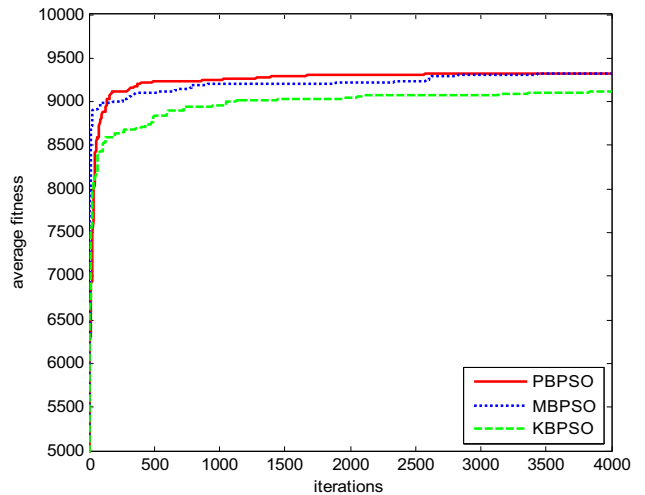


Figure 9. The average fitness value of KBPSO, MBPSO and PBPSO on Weish20.dat

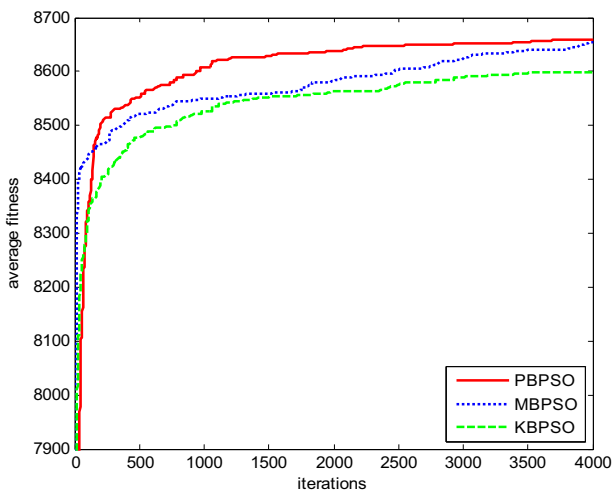


Figure 7. The average fitness value of KBPSO, MBPSO and PBPSO on Sent2.dat

V. CONCLUSION

This paper proposes a novel probability binary particle swarm optimization algorithm. The performance of PBPSO algorithm is evaluated and compared with the well-known KBPSO and MBPSO algorithm on a number of the benchmark multidimensional knapsack problem instances. The experimental results support the claim that the proposed PBPSO algorithm exhibits better optimization performance in terms of convergent speed and global search ability. Specially, PBPSO further simplify the updating computation.

ACKNOWLEDGMENT

This work is supported by ChenGuang Plan (2008CG48), Innovative Foundation of Shanghai University, Shanghai Colleges Excellent Youth Teacher Promotion Science and Research Foundation (shu07021 & sdl06082), Shanghai Municipal Natural Science

Foundation (06ZR14133), National Natural Science Foundation of China (Grant No. 60804052) and the Funds for International Science and Technology Cooperation of Shanghai (08160705900).

REFERENCES

- [1] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," *In: Proceeding of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.
- [2] Y. Shi, R. C. Eberhart, "A Modified Particle Swarm Optimizer," *In: Proceedings of the IEEE congress on evolutionary computation*. NJ: Piscataway, pp. 69-173, 1998.
- [3] R. C. Eberhart, Y. Shi, "Evolving artificial neural networks," *In: Proceeding of the International Conference on Neural Networks and Brain*, pp. 5-13, 1998.
- [4] M. Meissner, M. Schmuker, G. Schneider, "Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training," *Bmc Bioinformatics*, England, vol. 7, pp. 125, March 2006.
- [5] AI. EI-Galland, ME. EI-Hawary, AA. Sllam, "Swarming of intelligent particles for solving the nonlinear constrained optimization problem," *Engineering Intelligent System for Electrical Engineering and Communications*, England, vol. 9, pp. 155-163, September 2001.
- [6] M.A. Abido,, "Optimal power flow using particle swarm optimization," *International Journal of Electrical Power and Energy Systems*, England, vol. 24, pp. 563-571, September 2002.
- [7] CJ. Liao, CT. Tseng, P. Luarn, "A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems," *Computers & Operrations Research*, England, vol. 34, pp. 3099-3111, October 2007.
- [8] QK. Pan, MF. Tasgetiren, YC. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers& Operations Research*, England, pp. 2807-2839, September 2008.
- [9] YX. Jin, HZ. Cheng, HY. Yan, et al, "New discrete method for particle swarm optimization and its application in transmission network expansion planning", *Electric Power Systems Research*, Lausanne, Switzerland, pp. 227-233, March 2007.
- [10] B. Jarboui, N. Damak, P. Siarry, "A Combinatorial Particle Swarm Optimization for Solving Multi-Mode Resource-Constrained Project Scheduling Problems," *Applied Mathematics and Computation*, New York, USA, vol. 195, pp. 299-308, January 2008.
- [11] J. Kennedy, R. C. Eberhart,, "A Discrete Binary Version of the Particle Swarm Algorithm, in: 1997 IEEE International Conference, Systems, Man, and Cybernetics," *Computational Cybernetics and Simulation*, pp. 4104-4108, 1997.
- [12] Q. Shen, J. H. Jiang, "Modified Particle Swarm Optimization Algorithm for Variable Selection in MLR and PLS Modeling: QSAR Studies of Antagonism of Angiotensin II Antagonists," *European Journal of Pharmaceutical Sciences*, Amsterdam, Netherlands, vol. 22, pp. 145-152, June 2004.
- [13] PY. Yin, "A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves," *Journal of Visual Communication and Image Representation*, San Diego, USA, pp. 241-260, 2004.
- [14] M. Clerc, J. Kennedy, "The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, New York, USA, vol. 6, pp. 58-73, February 2002.
- [15] M. Kong, P. Tian, YC. Kao, "A New Ant Colony Optimization Algorithm for the Multidimensional Knapsack Problem," *Computers & Operations Research*, England, vol. 35, pp. 2672-2683, August 2008.
- [16] Anne L. Oken, "Penalty Functions and the Knapsack Problem," *Evolutionary Computation*, 1994. *IEEE World Congress on Computational Intelligence, Proceedings of the First IEEE Conference on, Orlando, FL, USA*, pp. 554-558, June 1994.
- [17] G. N. Diubin, A. A. Korbut, "Greedy Algorithms for the Minimization Knapsack Problem: Average Behavior," *Journal of Computer and Systems Sciences International*, New York, USA, vol. 47, pp.14-24, February 2008.
- [18] Hasan Pirkul, "A Heuristic Solution Procedure for the Multiconstraint Zero-One Knapsack Problem," *Naval Research Logistics*, pp.161-172, 1987.

Ling Wang was born in Shanghai, China, in 1979. He received B.S. degree in automation from East China University of Science and Technology, Shanghai, 2002 and Ph.D. degree in control theory and control engineering from East China University of Science and Technology, Shanghai, 2007. His research interests include intelligent optimization algorithm and fault diagnosis.

Currently, he is a lecturer, faculty of Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronics and Automation, Shanghai University.

Xiuting Wang was born in Shandong province, China, in 1983. He received B.S. degree from Shandong University of Technology, Shandong, China, in 2007. His major research topics include optimization algorithm and fault diagnoses.

Currently, he is enrolled in a master program in Shanghai University, Shanghai, China.

Jingqi Fu was born in Heilongjiang province, China, in 1962. He received Ph.D. degree from Nanjing University of Science and Technology, Nanjing, China, 1995. His research interests include: intelligent computing, intelligent sensor and wireless sensor.

Currently, he is a professor, faculty of School of Mechatronics and Automation, Shanghai University.

Lanlan Zhen was born in Hebei province, China, in 1972. She received B.S. degree from Jilin University of Technology, 1996 and M.S. degree from University of Shanghai for Science and Technology, 2002. Her major research topics include modeling and optimal control.

Currently, she is enrolled in a Ph.D. program in Shanghai University, and she is also a lecturer of Shanghai University of Electric Power.