ICIC Express Letters ICIC International ©2016 ISSN 1881-803X
Volume **10**, Number **8**, August **2016** pp. 1987–1994

AN IMPROVED BINARY PARTICLE SWARM OPTIMIZATION FOR 0-1 KNAPSACK PROBLEM

Na Tian*, Meng Wang and Yan Gu

Institute of Educational Informatization
Jiangnan University
No. 1800, Lihu Road, Wuxi 214122, P. R. China
*Corresponding author: tianna@jiangnan.edu.cn; { wangmengly; zero_gy }@163.com

Received December 2015; accepted March 2016

Abstract. *In this paper, an improved binary particle swarm optimization (IBPSO) is proposed based on the hamming distance. According to the characteristic of transform function, particle with bigger absolute value of velocity is more likely to move (change its position). In other words, the absolute value of velocity should increase if a particle tends to learn from its personal or the global experience. Therefore, the velocity update equation is reformulated. 30 benchmark instances of 0-1 knapsack problem are used to test the proposed algorithm and the comparison with three latest binary algorithms is also presented. The numerical experimental results indicate the effectiveness and efficiency of IBPSO.*

Keywords: Binary particle swarm optimization, Hamming distance, 0-1 knapsack

1. **Introduction.** Many solutions of real life optimization problems can be expressed as binary strings, such as data compression [1,2], image compression [3,4], feature selection [5], and 0-1 knapsack problems [6]. Many meta-heuristic algorithms have been applied to solving the binary optimization problems, such as genetic algorithm (GA) [7,8], particle swarm optimization (PSO) [9-11], evolutionary algorithm (EA) [12], ant colony optimization (ACO) [13], and gravitational search algorithm (GSA) [14,15]. Binary PSO (BPSO), which has simple structure and is easy to implement, is widely used to address various binary optimization problems. Several improved variants of BPSO have been proposed in the literature [5,6,9-11] in terms of topology, parameter selection, transform function, and hybridization with other algorithms. In [9], a binary hybrid topology particle swarm optimization quadratic interpolation (BHTPSO-QI) was proposed to enhance the global searching capability. In [11], a modified binary PSO about steepness was investigated to generate a better transform function. Beheshti et al. introduce acceleration into binary PSO [6]. An improved binary PSO with local search was proposed and applied to feature selection problems in [5].

However, none of the above works has considered the essence of velocity update equation that particles will have more chance to change their position if they tend to learn from their own experience or the global experience. Different from continuous PSO, the sign of velocity does not mean direction. Positive and negative values of velocity have the same impact on the evolvement of positions. In other words, only the absolute value of velocity should be concerned, which should increase if the position of a particle is different from the search experience. Therefore, in this paper, a new velocity update equation is proposed based on the above motivation.

The remainder of this paper is arranged as follows. Section 2 describes the details of the proposed variant of BPSO. In Section 3, the numerical experiments and analysis are conducted. Finally, Section 4 gives the conclusions.

2. Improved Binary Particle Swarm Optimization.

2.1. Particle swarm optimization. PSO is a population-based optimization technique originally introduced by Kennedy and Eberhart in 1995 [16]. A PSO system simulates the knowledge evolvement of a social organism, in which each particle represents one candidate solution of a problem.

In the classical PSO system with M particles, each individual is treated as a volume-less particle in the D-dimensional space, with position and velocity vectors of particle i at the kth iteration represented as $x_i(k) = (x_{i,1}(k), \ldots, x_{i,D}(k))$ and $v_i(k) = (v_{i,1}(k), \ldots, v_{i,D}(k))$, in order to optimize the objective function:

$$\text{minimize } f(x), \ x \in \Omega \tag{1}$$

The particle moves according to the following equations:

$$v_{i,d}(k+1) = \omega v_{i,d}(k) + c_1 r_1 (P_{i,d}(k) - x_{i,d}(k)) + c_2 r_2 (P_{g,d}(k) - x_{i,d}(k)) \tag{2}$$

$$x_{i,d}(k+1) = x_{i,d}(k) + v_{i,d}(k+1) \tag{3}$$

where $i = 1, 2, \ldots, M$, $d = 1, 2, \ldots, D$, ω is the inertia weight, and c_1 and c_2 are acceleration coefficients. r_1 and r_2 are random numbers distributed uniformly in $(0, 1)$. Vector $P_i = (P_{i,1}, P_{i,2}, \ldots, P_{i,D})$ is the best previous position of particle i, called personal best position, and vector $P_g = (P_{g,1}, P_{g,2}, \ldots, P_{g,D})$ $(g = \arg \min_{i=1:M} f(P_i))$ is the position of the best particle in the swarm, called global best position.

The second term of Equation (2) is called cognition term and the third term is called social term. The value of ω controls the balance between exploration and exploitation. $|v_{i,d}(k)| \leq v_{\max}$ and $v_{\max}$ is set as regarding the search space bound.

2.2. Binary particle swarm optimization. A binary version of PSO (BPSO) was first proposed by Kennedy and Eberhart [17], in which the position of a particle has two possible values: '0' or '1'. The velocity is also computed as Equation (2), and then it is transformed into the interval $[0, 1]$ by sigmoid function (Equation (4)) as shown in Figure 1.

$$S\left(v_{i,d}\left(k\right)\right) = sigmoid\left(v_{i,d}\left(k\right)\right) = \frac{1}{1 + e^{-v_{i,d}(k)}} \tag{4}$$

The position of a particle is updated as follows:

$$\begin{array}{ll} \text{if} & rand() < S\left(v_{i,d}(k+1)\right) \\ \text{then} & x_{i,d}(k+1) = 1 \\ \text{else} & x_{i,d}(k+1) = 0 \end{array} \tag{5}$$

$|v_{i,d}(k)| \leq v_{\max}$ and $v_{\max}$ can be set as 6 according to Figure 1.

Although the above method has simple structure and is easy to use, it has fatal drawback. There should be no difference between positive and negative values in velocity, because the sign only means direction. However, in sigmoid function (Figure 1), a negative velocity value means smaller probability to change, while a positive velocity value means bigger probability to change. Furthermore, in classical PSO, when the velocity tends to zero, it means the particle is already in a good position and does not need to move. However, as in sigmoid function, the particle still tends to change with a probability of 0.5, which is unreasonable.

To address the above disadvantage of BPSO, an improved version is proposed in [10], in which the transform function is changed as follows (shown in Figure 2):

$$S\left(v_{i,d}(k)\right) = 2 \times \left| \frac{1}{1 + e^{-v_{i,d}(k)}} - 0.5 \right| \tag{6}$$

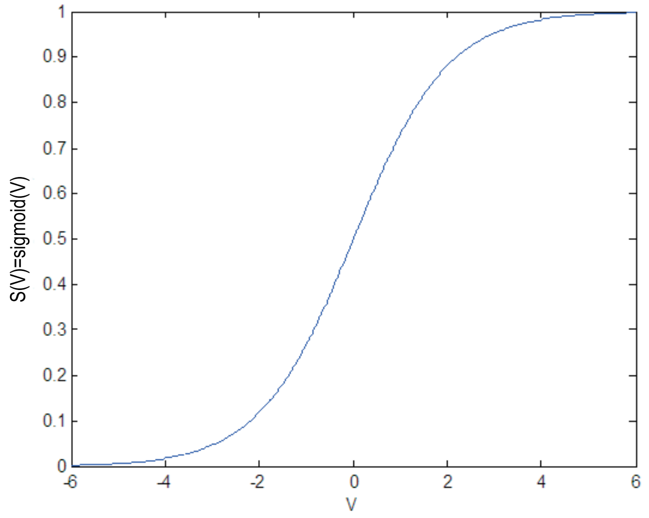

Figure 1. Sigmoid transform function

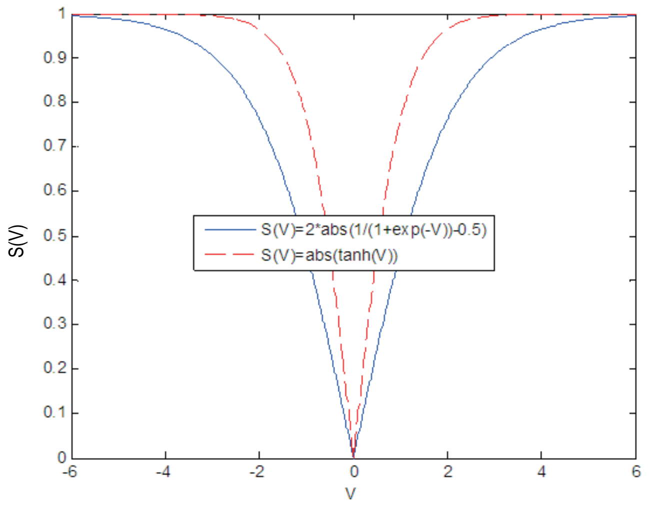

Figure 2. Two improved transform functions

A binary gravity search algorithm (BGSA) was proposed [14], in which, a new transform function is defined as follows:

$$S\left(v_{i,d}\left(k\right)\right) = \left|\tanh\left(v_{i,d}\left(k\right)\right)\right| \tag{7}$$

Note from Figure 2 that, particles with bigger absolute value of velocity are more likely to move. In addition, Equation (7) makes the particles more easily to change.

However, most researchers have been paying attention to the improvement of the transform function, but the performance of the algorithm is also affected by the velocity update equation. Different from the Euclidean distance used in the continuous PSO, the distance between two particles in BPSO is defined as the number of positions at which the corresponding bits are different (known as Hamming distance). Therefore, the operator '$-$' in the second and third term of Equation (2) means whether $P_{i,d}$ and $x_{i,d}$, $P_{g,d}$ and $x_{i,d}$ are the same or different. It will be '1' if they are the same, '0' if they are different, and '-1' is meaningless.

Therefore, only the absolute values of $P_{i,d}(k)-x_{i,d}(k)$ and $P_{g,d}(k)-x_{i,d}(k)$ are considered here. If they are equal to '1' ($x_{i,d}$ is different from $P_{i,d}$ or $P_{g,d}$), $x_{i,d}$ requires more chance (larger probability) to change. So, a new velocity update equation is proposed as follows:

$$v_{i,d}(k+1) = \pm\left(\omega\left|v_{i,d}(k)\right| + c_1 r_1 \left|P_{i,d}(k) - x_{i,d}(k)\right| + c_2 r_2 \left|P_{g,d}(k) - x_{i,d}(k)\right|\right) \quad (8)$$

which ensures the absolute value of $v_{i,d}(k+1)$ to increase when particle i tends to learn from its own experience or the best particle, and the sign of $v_{i,d}(k+1)$ can be '+' or '−' because positive and negative values have the same probability according to Figure 2.

At the early search stage (exploration), large ω and frequently learning from previous experience make particles have more opportunity to evolve. While at the later search stage, particles may have stopped evolving and get trapped into a local optimum. At that time, the second term and the third term of Equation (8) are usually equal to 0, and then we get $v_{i,d}(k+1) = \pm\left(\omega\left|v_{i,d}(k)\right|\right)$ with a relatively small ω. Note from Figure 2 that the particle still has nearly 50% chance to change even with a small value of velocity (e.g., $v_{i,d}(k+1) = 1$). Therefore, the swarm has a chance to jump from the local optimum and continue to exploit a new area. When particles stagnate, Equation (7) may generate more "energy" than Equation (6) for particles to move. Therefore, these two transform functions are both used in the proposed binary PSO.

3. IBPSO for 0-1 Multi-Dimensional Knapsack Problems (0-1 MKP).

3.1. Mathematical model of 0-1 MKP.

In order to evaluate the performance of IBPSO, the 0-1 MKP, which is NP-complete, is adopted in this section. Many real problems are formulated as the 0-1 MKP, such as cargo loading [18], resource allocating [19], capital budgeting [8], and pollution prevention and control [11].

The 0-1 MKP consists of D items and n knapsacks with limited capacities. Each item has a profit and weight. The objective is to select a subset of items having maximum total profit without exceeding the capacity constraints. Therefore, the 0-1 MKP can be formulated as follows:

$$\begin{array}{c} \text{maximize} \sum_{j=1}^{D} p_j x_j \\ \text{subject to } \sum_{j=1}^{D} w_{ij} x_j \le c_i,\ i = 1, 2, \ldots, n,\ x_j \in \{0, 1\} \end{array} \quad (9)$$

where $p_j \ge 0$ is the profit of item j, $w_{ij} \ge 0$ is the weight of item j in knapsack i, and c_i is the capacity of knapsack i. $x_j = 1$ means that item i is selected.

When IBPSO is applied to solving 0-1 MKP, the position of a particle represents a candidate solution, in which the dimension equals the number of items. However, in the swarm, some solutions are infeasible because the constraints in Equation (9) are not satisfied (the total weights exceed the capacities of some knapsacks). Many methods have been used to deal with these infeasible solutions [6,20]. One type of method is to repair the solution to a feasible solution, and another type is to decrease the probability to select infeasible solutions by using penalty function. In this paper, a repair method using greedy algorithm is adopted, in which the item with smallest profit weight ratio will be removed.

The test instances are selected from OR-Library [21].

3.2. Results and analysis.

The latest research about 0-1 MKP is given in [9], where BHTPSO and BHTPSO-QI have been empirically proved to outperform other algorithms. Therefore, in this paper, BHTPSO, BHTPSO-QI and BGSA [14] are taken to do comparison with the proposed IBPSO. All the algorithms are independently run 30 times under the same circumstances. The population size is set to 100 ($M = 100$). The maximum number of iterations is set to 3000. The inertia weight ω in IBPSO is set to linearly decrease from 0.9 to 0.4. The acceleration coefficients $c_1 = c_2 = 2.0$.

The best, mean and worst maximum profits obtained by the algorithms are listed in Table 1 (instances 1-15) and Table 2 (instances 16-30), in which, IBPSO-E and IBPSO-T represent improved binary particle swarm optimization with Equations (6) and (7) as their transform functions, respectively.

Table 1. Experimental results on the benchmarks 1-15 from OR-Library

MKP benchmark	Profit	BHTPSO	BHTPSO-QI	BGSA	IBPSO-T	IBPSO-E
mknapcb1-5.100-00	Best	24,169	24,301	24,152	**24,326**	24,302
	Mean	23822.8	23821.7	23835.7	24,161	**24,167**
	Worst	23,415	23,287	23,175	23,998	**24,017**
mknapcb1-5.100-01	Best	24,109	23,944	23,986	**24,274**	**24,274**
	Mean	23657.2	23688.7	23563.3	24,124	**24,160**
	Worst	22,953	23,375	23,177	23,864	**23,982**
mknapcb1-5.100-02	Best	23,435	23,418	23,386	23,494	**23,523**
	Mean	23072.7	23073.1	23041.5	23,438	**23,469**
	Worst	22,678	22,621	22,543	23,308	**23,308**
mknapcb1-5.100-03	Best	23,253	23,192	23,172	23,468	**23,486**
	Mean	22,928	22923.1	22,863	23,303	**23,322**
	Worst	22,507	22,234	22,468	23,142	**23,235**
mknapcb1-5.100-04	Best	23,815	23,774	23,755	**23,959**	**23,959**
	Mean	23473.6	23527.9	23459.2	23,905	**23,932**
	Worst	23,155	23,053	23,106	23,742	**23,821**
mknapcb2-5.250-00	Best	57,814	57,800	57,565	58,900	**58,957**
	Mean	56874.3	56685.2	56554.7	58,685	**58,777**
	Worst	54,935	55,255	55,191	58,327	**58,477**
mknapcb2-5.250-01	Best	59,982	59,767	60,057	61,206	**61,360**
	Mean	58588.8	58680.6	58613.9	61,036	**61,115**
	Worst	56,807	56,821	57,707	60,816	**60,848**
mknapcb2-5.250-02	Best	60,630	60,524	59,936	**61,786**	61,734
	Mean	59234.1	59186.3	58975.3	61,468	**61,523**
	Worst	57,435	57,278	57,723	61,070	**61,297**
mknapcb2-5.250-03	Best	57,736	57,884	57,970	59,055	**59,139**
	Mean	56,773	56,584	56744.4	58,859	**58,962**
	Worst	55,589	55,164	55,371	58,507	**58,613**
mknapcb2-5.250-04	Best	57,378	57,550	56,959	58,680	**58,688**
	Mean	56129.2	56361.1	55961.3	58,485	**58,550**
	Worst	54,364	53,929	54,637	58,297	**58,298**
mknapcb3-5.500-00	Best	114,493	114,438	111,206	119,528	**119,729**
	Mean	111,017	111,469	108,930	119,240	**119,340**
	Worst	106,454	107,005	106,951	118,862	**118,966**
mknapcb3-5.500-01	Best	112,821	112,147	108,522	117,128	**117,322**
	Mean	109,276	109,247	106,631	116,780	**117,000**
	Worst	100,118	104,696	104,519	116,368	**116,544**
mknapcb3-5.500-02	Best	114,774	116,099	111,271	120,557	**120,807**
	Mean	112,035	112,001	109,430	120,180	**120,390**
	Worst	106,406	104,627	107,683	119,716	**119,975**
mknapcb3-5.500-03	Best	115,828	114,327	111,283	119,719	**120,102**
	Mean	112,200	111,671	109,062	119,390	**119,700**
	Worst	106,222	107,578	107,061	118,778	**119,386**
mknapcb3-5.500-04	Best	115,889	117,242	112,391	121,691	**121,785**
	Mean	112,253	113,364	110,564	121,270	**121,470**
	Worst	102,820	103,910	108,670	120,987	**121,125**

All the 15 instances in Table 1 have 5 knapsacks (constraints). It is obvious to see that the results obtained by IBPSO (both IBPSO-T and IBPSO-E) are much better than those obtained by other algorithms (the best values of each instance are marked in bold). In addition, IBPSO with simple update equation is easy to implement and runs faster than BHTPSO and BGSA under the same circumstances. BGSA requires complex computation of masses, forces, distances and accelerations of the agent, and BHTPSO has

Table 2. Experimental results on the benchmarks 16-30 from OR-Library

MKP benchmark	Profit	BHTPSO	BHTPSO-QI	BGSA	IBPSO-T	IBPSO-E
mknapcb4-10.100-00	Best	22,905	22,876	22,836	**23,055**	**23,055**
	Mean	22425.8	22449.6	22334.3	22,924	**22,946**
	Worst	21,980	21,999	21,975	22,670	**22,700**
mknapcb4-10.100-01	Best	22,573	22,408	22,441	22,694	**22,763**
	Mean	22047.8	22017.3	21991.8	**22,523**	22,330
	Worst	21,322	21,454	21,435	**22,440**	22,413
mknapcb4-10.100-02	Best	21,797	21,949	21,849	**22,751**	**22,751**
	Mean	21342.3	21461.3	21313.5	22,455	**22,545**
	Worst	20,958	20,886	20,957	22,207	**22,383**
mknapcb4-10.100-03	Best	22,418	22,376	22,325	**22,594**	22,548
	Mean	22037.8	22029.1	21961.9	**22,483**	22,479
	Worst	21,228	21,533	21,488	**22,371**	22,367
mknapcb4-10.100-04	Best	22,215	22,254	22,168	21,725	**21,755**
	Mean	21822.8	21903.3	21840.8	21,645	**21,653**
	Worst	21,362	21,339	21,271	**21,559**	21,435
mknapcb5-10.250-00	Best	57,530	57,036	56,928	58,779	**58,840**
	Mean	55854.1	55960.7	55759.4	58,550	**58,650**
	Worst	53,570	53,381	54,217	58,086	**58,359**
mknapcb5-10.250-01	Best	56,568	56,490	56,337	**58,548**	58,368
	Mean	55443.9	55708.1	55455.9	58,076	**58,156**
	Worst	53,274	52,907	53,739	57,730	**57,865**
mknapcb5-10.250-02	Best	56,426	55,982	55,573	57,670	**57,778**
	Mean	54793.2	54727.8	54638.3	57,393	**57,517**
	Worst	52,871	52,714	53,516	57,113	**57,227**
mknapcb5-10.250-03	Best	59,030	59,077	58,595	**60,604**	60,583
	Mean	58057.8	57721.9	57766.2	60,336	**60,384**
	Worst	56,254	53,774	56,701	59,978	**60,117**
mknapcb5-10.250-04	Best	56,217	56,204	56,186	**57,743**	57,715
	Mean	54941.1	54872.6	54,850	57,444	**57,485**
	Worst	51,850	50,832	53,612	57,077	**57,232**
mknapcb6-10.500-00	Best	110,996	111,669	108,487	116,745	**117,112**
	Mean	107,698	108,367	105,760	116,340	**116,680**
	Worst	104,239	103,802	102,725	115,909	**116,324**
mknapcb6-10.500-01	Best	114,262	113,001	109,569	118,212	**118,464**
	Mean	108,648	109,197	106,775	117,820	**118,150**
	Worst	100,740	100,764	103,478	117,406	**117,814**
mknapcb6-10.500-02	Best	113,987	112,419	109,705	117,854	**118,018**
	Mean	108,576	109,004	106,853	117,380	**117,690**
	Worst	102,439	103,703	104,565	116,599	**116,945**
mknapcb6-10.500-03	Best	112,476	112,198	108,628	115,386	**115,740**
	Mean	107,692	107,796	105,679	115,060	**115,440**
	Worst	101,860	99,470	102,679	114,497	**115,151**
mknapcb6-10.500-04	Best	109,567	109,287	106,972	118,501	**118,664**
	Mean	106,217	106,212	104,509	118,120	**118,350**
	Worst	100,836	100,509	102,665	117,504	**117,924**

several more parameters to control (NF, T' and three iteration-dependent acceleration coefficients).

In comparison with the two transform functions (Equations (6) and (7)), the mean and worst values of maximum profits obtained by IBPSO-E are better than those got by IBPSO-T. This phenomenon indicates that IBPSO-E has stable performance, while

IBPSO-T has potential to obtain excellent solution. However, for high dimensional instances (e.g., $D = 500$), IBPSO-E outperforms its counterpart.

When the number of knapsacks increases to 10 (more constraints), IBPSO still has amazing performance over the other three algorithms. IBPSO-T and IBPSO-E have comparative performance. In some instances, IBPSO-T is able to obtain excellent best solutions, but the average performance is not so good as IBPSO-E. Similar to Table 1, IBPSO-E shows absolutely best performance over other algorithms including IBPSO-T, which indicates the scalability of IBPSO-E to large scale binary optimization problems.

4. **Conclusions.** In the binary space, distance between two particles is defined as the number of positions in which the value is different. This is known as hamming distance, based on which, an improved binary particle swarm optimization was proposed. The absolute value of velocity will increase if the particle wants to learn from other particles. 30 benchmark instances of 0-1 knapsack problem were used to test the proposed algorithm and the comparison with other three latest binary algorithms was also presented. The numerical experimental results indicate the effectiveness and efficiency of the improved algorithm.

The future work will try to apply the proposed algorithm to the multi-objective 0-1 knapsack problems.

Acknowledgment. This work is supported by Jiangsu Postdoctoral Funding (1401004B), National Natural Science Foundation of China (61300152), and National Hi-tech Research and Development Program of China (2014AA041505).

REFERENCES

[1] J. Fu, D. Miao, W. Yu, S. Wang, Y. Lu and S. Li, Kinect-like depth data compression, *IEEE Trans. Multimedia*, vol.15, no.6, pp.1340-1352, 2013.
[2] L. R. Iyer and S.-B. Ho, A connectionist model of data compression in memory, *Biol. Inspir. Cogn. Archit.*, vol.6, pp.58-66, 2013.
[3] L. Cinque, S. D. Agostino and L. Lombardi, Binary image compression via monochromatic pattern substitution: Sequential and parallel implementations, *Math. Comput. Sci.*, vol.7, no.2, pp.155-166, 2013.
[4] X. Lv and Z. J. Wang, Compressed binary image hashes based on semi-supervised spectral embedding, *IEEE Trans. Inform. Forensic Secur.*, vol.8, no.11, pp.1838-1849, 2013.
[5] S. M. Vieira, L. F. Mendon, G. J. Farinha and J. M. C. Sousa, Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients, *Appl. Soft. Comput.*, vol.13, no.8, pp.3494-3504, 2013.
[6] Z. Beheshti, S. M. Shamsuddin and S. S. Yuhaniz, Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems, *J. Glob. Optimiz.*, vol.57, no.2, pp.549-573, 2013.
[7] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – A comparative experiment, *IEEE Trans. Evol. Comput.*, vol.6, no.4, pp.402-412, 2002.
[8] P. C. Chu and J. E. Beasley, A genetic algorithm for the multidimensional knapsack problem, *J. Heuristics*, vol.4, pp.63-86, 1979.
[9] Z. Beheshti, A. M. Shamsuddin and S. Hasan, Memetic binary particle swarm optimization for discrete optimization problems, *Inform. Sci.*, vol.299, no.1, pp.58-84, 2015.
[10] H. Nezamabadi-pour, M. Rostami-Shahrbabaki and M. Maghfoori-Farsangi, Binary particle swarm optimization: Challenges and new solutions, *CSI J. Comput. Sci. Eng.*, vol.6, no.1, pp.21-32, 2008.
[11] J. C. Bansal and K. Deep, A modified binary particle swarm optimization for knapsack problems, *Appl. Math. Comput.*, vol.218, no.22, pp.11042-11061, 2012.
[12] Q. F. Zhang and H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Comput.*, vol.11, no.6, pp.712-731, 2007.
[13] L. Ke, Q. Zhang and R. Battiti, MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony, *IEEE Trans. Syst. Man Cybernet. Part B: Cybernet.*, vol.43, no.6, pp.1845-1859, 2012.
[14] E. Rashedi and H. Nezamabadi-Pour, Feature subset selection using improved binary gravitational search algorithm, *J. Intell. Fuzzy Syst.*, vol.26, pp.1211-1221, 2014.

[15] E. Rashedi, S. Nezamabadi and S. Saryazdi, BGSA: Binary gravitational search algorithm, *Nat. Comput.*, vol.9, no.3, pp.727-745, 2010.
[16] J. Kennedy and R. C. Eberhart, Particle swarm optimization, *IEEE Int. Conf. Neural Networks*, Perth, Australia, 1995.
[17] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, *Proc. of IEEE International Conference on Computational Cybernetics and Simulation*, vol.5, pp.4104-4108, 1997.
[18] W. Shih, A branch and bound method for the multi-constraint zero-one knapsack problem, *J. Oper. Res. Soc.*, vol.30, pp.369-378, 1979.
[19] B. Gavish and H. Pirkul, Allocation of databases and processors in a distributed computing system, *Management of Distributed Data Processing*, North Holland, Amsterdam, Netherlands, pp.215-231, 1982.
[20] L. Wang, S. Wang and Y. Xu, An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem, *Expert Syst. Appl.*, vol.39, no.5, pp.5593-5599, 2012.
[21] J. E. Beasley, *OR-Library*, http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/mknapinfo.html.