Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 1161–1166
Progress in Natural Science
www.elsevier.com/locate/pnsc

Modified binary particle swarm optimization

Sangwook Lee [a], Sangmoon Soak [b], Sanghoun Oh [c], Witold Pedrycz [d], Moongu Jeon [c,*]

[a] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b] Information Systems Examination Team, Korean Intellectual Property Office (KIPO), Government Complex Daejeon Building 4, 920 Dunsandong, Seogu, Republic of Korea
[c] Department of Information and Communications, Gwangju Institute of Science and Technology, 261 Cheomdan-gwagiro, Buk-gu, Gwanju, Republic of Korea
[d] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada, T6G 2V4

Received 4 March 2008; received in revised form 28 March 2008; accepted 28 March 2008

Abstract

This paper presents a modified binary particle swarm optimization (BPSO) which adopts concepts of the genotype–phenotype representation and the mutation operator of genetic algorithms. Its main feature is that the BPSO can be treated as a continuous PSO. The proposed BPSO algorithm is tested on various benchmark functions, and its performance is compared with that of the original BPSO. Experimental results show that the modified BPSO outperforms the original BPSO algorithm.

Keywords: Binary particle swarm optimization; Genotype–phenotype; Mutation

1. Introduction

The particle swarm optimization (PSO) is an evolutionary computation technique motivated by the behavior of organisms [1]. The original aim of PSO is to adjust the weights of the artificial neural network by substituting the back-propagation. However, these days PSO is successfully applied to a wide range of continuous optimization problems. In 1997, Kennedy and Eberhart adapted the standard continuous PSO algorithm to binary spaces [2]. However, this variant of the continuous PSO, termed the binary PSO (BPSO), has not been studied much.

In the PSO algorithm, there are two update functions: the velocity update function and the position update function. Originally, the position was updated by combining its current position and velocity, but in BPSO, the position is updated by reflecting only the current velocity; generally, the sigmoid function has been used to update the position in BPSO. Due to this characteristic, it seems that the velocity domain is a search space though an actual binary search space already exists.

Based on genetics, any organism can be represented by its phenotype, which virtually determines what exactly the object is in the real world, and its genotype containing all the information about the object at the chromosome set level [3]. The role of each gene is reflected in the phenotype. In this paper, we demonstrate a modified BPSO using this genotype–phenotype paradigm of genetics. We will match the genotype and the phenotype to the velocity and the binary position, respectively, in BPSO.

The remainder of the paper is organized as follows: Section 2 provides an overview of PSO. The various core parameters of PSO are introduced and guidelines to parameter selection are given. Section 3 describes the modified BPSO which is a variant of the original BPSO by adopting the genotype–phenotype concept and the mutation operator. The experimental procedure and results are presented in Section 4. The paper concludes with Section 5, which offers a summary of the results.

* Corresponding author. Tel.: +82 62 970 2406; fax: +82 62 970 2204.
E-mail address: mgjeon@gist.ac.kr (M. Jeon).

1002-0071/$ - see front matter
doi:10.1016/j.pnsc.2008.03.018

2. Background

This section introduces the PSO algorithm and its variant, the BPSO. Performance criteria of the algorithm are discussed by considering the effect of algorithmic parameters. Particle organization and knowledge sharing through topologies are also discussed.

2.1. PSO

The PSO algorithm was first introduced by Kennedy and Eberhart as an optimization technique for continuous problems in 1995 [1]. It is a kind of evolutionary computation technique motivated by the behavior of organisms such as fish schooling and bird flocking. Because of its simple and easy implementation, PSO has been widely used in a variety of optimization problems.

The PSO algorithm is initialized with the population of individuals being randomly placed in the search space and search for an optimal solution by updating individual generations. At each iteration, the velocity and the position of each particle are updated according to its previous best position (p_{best},i,j) and the best position found by informants (g_{best},i,j). In the original continuous version [1,4], each particle's velocity and position are adjusted by the following formula:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1R_1(p_{\text{best},i,j} - x_{i,j}(t)) + c_2R_2(g_{\text{best},i,j} - x_{i,j}(t)) \quad (1)$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1), \quad (2)$$

where i is the index of a particle in the swarm ($i = 1, \ldots, n$), j is the index of position in the particle ($j = 1, \ldots, m$), t represents the iteration number, $v_{i,j}(t)$ is the velocity of the ith particle, and $x_{i,j}(t)$ is the position. Note that R_1 and R_2 are random numbers uniformly distributed between 0 and 1, c_1 and c_2 are the acceleration coefficients, and w is the positive inertia weight.

2.2. BPSO

The BPSO algorithm was introduced by Kennedy and Eberhart to allow the PSO algorithm to operate in binary problem spaces [2]. It uses the concept of velocity as a probability that a bit (position) takes on one or zero. In the BPSO, Eq. (1) for updating the velocity remains unchanged, but Eq. (2) for updating the position is redefined by the rule [2]

$$x_{i,j}(t+1) = \begin{cases} 0 & \text{if rand}() \geqslant S(v_{i,j}(t+1)) \\ 1 & \text{if rand}() < S(v_{i,j}(t+1)) \end{cases} \quad (3)$$

where $S(\cdot)$ is the sigmoid function for transforming the velocity to the probability as the following expression:

$$S(v_{i,j}(t+1)) = \frac{1}{1 + e^{-v_{i,j}(t+1)}} \quad (4)$$

and rand() is the pseudo-random number selected from a uniform distribution over [0.0, 1.0].

It should be noted that the BPSO is susceptible to sigmoid function saturation, which occurs when velocity values are either too large or too small. In such cases, the probability of a change in bit value approaches zero, thereby limits exploration. For a velocity of 0, the sigmoid function returns a probability of 0.5, implying that there is a 50% chance for the bit to flip. However, velocity clamping will delay the occurrence of the sigmoid function saturation.

2.3. Parameters of PSO

2.3.1. Acceleration and inertia constants

In the velocity update Eq. (1), the acceleration coefficients, c_1 and c_2 determine the influence of the personal best and the neighborhood best solutions on the particle's current velocity vector. The cognitive component is defined as the influence of the personally best position vector, and the social component is defined as the stochastically weighted difference between the neighborhood best position and the current position. If the cognitive component has a relatively large value, the flight of the particle will be steered in the direction of the personal best position, limiting the influence of the social component.

The inertia parameter, w, controls the influence of the previous velocity on the current velocity [4]. It can improve the manner in which the PSO algorithm converges to a solution by smoothing particle trajectories. In [5], it was shown that a good convergence can be ensured by making these two constants (acceleration and inertia) dependent. Their relation is shown in the following equation with an intermediate parameter φ,

$$\begin{cases} w = \frac{1}{\varphi - 1 + \sqrt{\varphi^2 - 2\varphi}} \\ c_1 = c_2 = \varphi w \end{cases} \quad (5)$$

Note that for φ the above formula imposes values greater than 2, the inertia w has to be a real number. In addition, experimental results in [6] show that c_1 and c_2 must be greater than 1. These two remarks lead us to changing φ in the interval [2.01, 2.4].

2.3.2. Maximum velocity V_{max}

A restriction may be placed on the individual velocity component values to enforce the limitation that a particle does not exceed a certain acceleration. This constraint, $V_{\max}$, reduces the chance that a particle may accelerate uncontrollably and explode off the bounds of the search space. Velocity values that are not within the range $[-V_{\max}, V_{\max}]$ are clamped. Thus, the value of $V_{\max}$ must be carefully selected. If it is too small, step sizes ought to be very small, which may potentially lead the algorithm to be trapped in a local minimum, or may increase the number of iterations to reach a good solution.

2.3.3. Topology (neighborhood structure)

The first presented PSO used a global best (*GBest*) topology, implying that all particles are considered as neighbors. Because of the interconnection between particles, the *GBest* topology frequently causes the particles to converge on a local optimum. Nowadays, however, it is well-known that the local best (*Lbest*) topology, where partial particles are considered as neighbors, outperforms *GBest* topology [5]. *Lbest* topology defines neighbors based on indices in a variable space and particles are influenced by the neighboring best particle. Among various *Lbest* topologies in the literature, it has been experimentally shown that the *Von Neumann* topology is at least as effective as the *Lbest* topology [7], even outperforming it on various occasions. The *Von Neumann* topology arranges particles in a two-dimensional lattice enabling the particle to share information with more particles and in more directions.

3. Modified binary particle swarm optimization

In this section, our proposed scheme, the modified BPSO, is described using the genotype–phenotype concept and the mutation operator. The velocity and the binary position parameters of the original BPSO are corresponding to the genotype of position and phenotype of position of the modified BPSO, respectively. Furthermore, from considering a characteristic of the genotype of position (velocity in the original BPSO), we suggest a mutation operator for improving the performance of the BPSO based on our previous work [8].

3.1. Genotype–phenotype concept

The main difference between the PSO and the BPSO is in the position update function. Specifically, the position update of the BPSO does not use the information of the current position. In other words, the next position of the BPSO is not influenced by the current position but influenced by the velocity only. This implies that when updating a position in the BPSO, it is meaningless to know where a particle is currently located in the binary search space. Because of this fact, it seems that velocity is a particle, even though the binary position already exists in the BPSO. Thus, we suggest the concept that the velocity and the position of the original BPSO be taken as a particle and a solution transformed by the sigmoid function, respectively. This concept is based on the imitation of the mechanism of genotype–phenotype concept in biology.

The genotype of an individual is the genetic information carried by the individual's genes, whether or not the genes are expressed. The phenotype denotes all the observable characteristics of an individual, such as physical appearance (eye color, height, etc.) and internal physiology. It is determined partly by genes, the environment and the way of life.

In genetic algorithms, the genotype–phenotype concept has already been applied to encoding schemes. As an example, we introduce random key encoding [9]; generally, random key encoding is used for an order-based encoding scheme. If there are five genes in a chromosome, each gene is assigned a random number drawn uniformly from (0, 1). To decode the chromosome, we count the genes in ascending order with regard to their values as the following example:

Random key	0.42	0.06	0.38	0.48	0.81
Decodes as	3	1	2	4	5

where the value of random key is the genotype and the value of decode is the phenotype. Note that genes to be counted early tend to "evolve" closer to 0, and those to be counted later tend to evolve closer to 1 because of the ascending order.

The genotype–phenotype concept can be applied to the BPSO as follows: Let the velocity of the original BPSO be a continuous search space, and then a binary position can be decoded by the sigmoid function. Fig. 1 illustrates the concepts of the original and the modified BPSO. Here, let the velocity and the binary position of the original BPSO be a genotype $x_{g,i,j}$ and a phenotype $x_{p,i,j}$, respectively. Then, the update functions of (1), (3), and (4) of the original BPSO are changed as the following expressions:

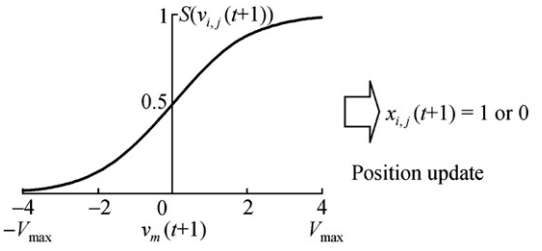
$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1R_1(p_{\text{best},i,j} - x_{p,i,j}(t)) + c_2R_2(g_{\text{best},i,j} - x_{p,i,j}(t)) \quad (6)$$

$$x_{g,i,j}(t+1) = x_{g,i,j}(t) + v_{i,j}(t+1) \quad (7)$$

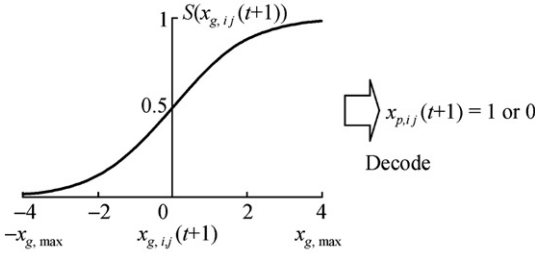
$$x_{p,i,j}(t+1) = \begin{cases} 0 & \text{if rand}() \geqslant S(x_{g,i,j}(t+1)) \\ 1 & \text{if rand}() < S(x_{g,i,j}(t+1)) \end{cases} \quad (8)$$

where

$$S(x_{g,i,j}(t+1)) = \frac{1}{1 + e^{-x_{g,i,j}(t+1)}}. \quad (9)$$



(a) Concept diagram of the original BPSO



(b) Concept diagram of the geno-phenotype BPSO

Fig. 1. Concept diagram of the BPSO.

In update functions of the modified BPSO, we emphasize the following two points:

- The information concerning the positions used in the velocity update (6) is from the phenotype not from the genotype of the positions.
- The modified BPSO is identical to the original BPSO with 1.0 inertia weight if $w = 0$ in the velocity update function (6).

3.2. Mutation

When the original BPSO algorithm starts the iteration to find an optimum, the velocity tends to go into $V_{\max}$ or $-V_{\max}$ by the velocity update if the corresponding target position is one or zero. If a velocity converges to near $V_{\max}$ or $-V_{\max}$, it is hard to change the corresponding position with a small change of velocity, which makes it difficult to escape from a good local optimum in the BPSO. In order to get out of this undesired position, large movement of velocity is required, which is unconcerned with two current best positions (p_{best} and g_{best}). For accomplishing the above objective, we suggest the insertion of the following operation between velocity update (1) and position update (2) of the original BPSO process:

$$\begin{aligned}&\textbf{for}(i = 1; i < n; i = i + 1)\{\\&\textbf{if}(\text{rand}() < r_{\text{mu}})\\&\textbf{then} v_{i,j_r}(t+1) = -v_{i,j_r}(t+1)\},\end{aligned} \tag{10}$$

where r_{mu} is the probability that the proposed operation is conducted in the ith particle and j_{r} is the randomly-selected position of this particle. If this operation is executed when velocity is near $V_{\max}$ or $-V_{\max}$, the position will be changed from one to zero or from zero to one, respectively. This rule (10) similarly affects the BPSO algorithm with a bit change mutation in GAs [10]; i.e., regarding a certain probability, a bit is changed from 0 to 1 or from 1 to 0, respectively.

On the other hand, in our modified BPSO algorithm, $V_{\max}$ and v_{i,j_r} of the original BPSO are identical to $X_{g,\max}$ and x_{g,i,j_r}, respectively. For the mutation in the modified BPSO process, the following operation is inserted between (7) and (8):

$$\begin{aligned}&\textbf{for}(i = 1; i < n; i = i + 1)\{\\&\textbf{if}(\text{rand}() < r_{\text{mu}})\\&\textbf{then} x_{g,i,j_r}(t+1) = -x_{g,i,j_r}(t+1)\}\end{aligned} \tag{11}$$

Algorithms 1 and 2 are pseudo-codes for the original and modified BPSO with a mutation, respectively. When implementing the algorithm with no mutation, the mutation operation (i.e., (10) and (11)) is omitted.

Algorithm 1. An original BPSO with a mutation

Begin
  $t = 0$; {t: *generation index*}
  initialize particles $x_{i,j}(t)$;
  evaluation $x_{i,j}(t)$;
  while(*termination condition*≠*true*) ***do***
    $v_{i,j}(t) = update\ v_{i,j}(t)$; {*by Eq.* (1)}
    $v_{i,j}(t) = mutation\ v_{i,j}(t)$; {*by Eq.* (10)}
    $x_{i,j}(t) = update\ x_{i,j}(t)$; {*by Eqs.* (3) *and* (4)}
    evaluate $x_{i,j}(t)$;
    $t = t + 1$;
  end while
End

Algorithm 2. A modified BPSO with a mutation

Begin
  $t = 0$; {t: *generation index*}
  initialize particles $x_{p,i,j}(t)$;
  evaluation $x_{p,i,j}(t)$;
  while (*termination condition*≠*true*) ***do***
    $v_{i,j}(t) = update\ v_{i,j}(t)$; {*by Eq.* (6)}
    $x_{g,i,j}(t) = update\ x_{g,i,j}(t)$; {*by Eq.* (7)}
    $x_{g,i,j}(t) = mutation\ x_{g,i,j}(t)$; {*by Eq.* (11)}
    $x_{p,i,j}(t) = decode\ x_{g,i,j}(t)$; {*by Eqs.* (8) *and* (9)}
    evaluate $x_{p,i,j}(t)$;
    $t = t + 1$;
  end while
End

4. Experiments

In experiments, we will apply the modified BPSO and the original BPSO to ten benchmark functions, and then compare their performance. We used a Pentium IV with a 2.4-GHz CPU and 1 GByte of memory, and Visual C++ as the programming language.

4.1. Benchmark functions

Ten benchmark functions were used for the performance tests. The first functions are De Jong's suite, taken from [11], and the last five functions are multi-modal benchmark functions which were taken from [12]. Since several functions have many local optima, it is very hard to find a global optimum point with conventional numerical optimization methods.

The details of functions are as follows:

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

$$f_2(x) = 100(x_1^2 - x_2^2)^2 + (1 - x_1)^2$$

$$f_3(x) = 6 \cdot \sum_{i=0}^{5} int(x_i)$$

$$f_4(x) = \sum_{i=1}^{n} i \cdot x_i^4 + N(0,1)$$

$$\frac{1}{f_5(x)} = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}$$

$$f_6(x) = 0.5 + \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$$

$$f_7 = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$$

$$f_8 = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$$

$$f_9 = \sum_{i=1}^{n} 100((x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

$$f_{10}(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10),$$

where int(x_i) is the floor of the number, $N(0,1)$ is Gaussian noise, and

$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots \end{pmatrix}$$

All functions are for minimization problems. Table 1 summarizes characteristics of the functions. In the table, 'F.' indicates a function, 'Dim' indicates a dimension of function, 'B. dim' indicates a binary dimension of function, and 'Opt.' indicates the global optimum of function.

4.2. Parameters selection

Parameter selection is one of the most important issues in the PSO algorithm. Here, we attempt to determine an optimal parameter selection based on previous research in the literature. At first, regarding random number generation, it is known that a rand() function, provided by the computer language C, cannot provide a high quality of pseudo-randomness [6]. For this reason, we used KISS, which is excellent for generating a pseudo-random number. Regarding the coefficients of the update functions (i.e. w, c_1, and c_2), Clerc [6] suggested 2.07 for the intermediate parameter φ to obtain good results. As results from (5), we used $w = 0.689343$ and $c_1 = c_2 = 1.42694$. In the modified BPSO, the maximum search space of *genotype* ($X_{g,\max}$) is identical to the maximum velocity $V_{\max}$ of the original BPSO. In practice, $V_{\max}$ of the original BPSO is often set to be 4, so that there is always at least a chance of $S(V_{\max}) \cong 0.0180$ that a bit will change state [12]. Thus, we used $V_{\max} = 4$ and $X_{g,\max} = 4$ in the original BPSO and the modified BPSO, respectively. Finally, regarding topology, we used the *Von Neumann* topology to obtain good results based on [7].

Table 1
Details of benchmark functions

F	Common name	Dim	Search space	B. dim	Opt.
f_1	Spherical	3	(−5.12, 5.12)	30	0
f_2	2D Rosenbrock	2	(−2.048, 2.048)	24	0
f_3	Step Function	5	(−5.12, 5.12)	50	0
f_4	Quartic	30	(−1.28, 1.28)	240	0
f_5	Foxholes	2	(−65536.0, 65536.0)	34	≈1
f_6	Schaffer's F6	2	(−100.0, 100.0)	22	0
f_7	Griewank	30	(−300.0, 300.0)	390	0
f_8	Ackley	30	(−30.0, 30.0)	300	0
f_9	Rosenbrock	30	(−2.048, 2.048)	360	0
f_{10}	Rastrigin	30	(−5.12, 5.12)	300	0

4.3. Results

Tables 2 and 3 show the comparison between the modified BPSO and the original BPSO without mutation and with mutation, respectively. The selected numbers for particles and generations in the experiments are 40 and 1000, respectively. Each experiment was conducted 30 times in order to report statistically relevant results. In Table 3, the r_{mu} indicates a mutation rate which provides the best performance among the nine tests from 0.1 to 0.9; the r_{mu} value of 'all' and '0' indicates that all mutation tests provide an identical result and that a test with no mutation provides the best result, respectively. In Table 2, the results of the continuous PSO are also used to investigate a performance difference between the BPSO and the PSO.

All benchmark functions are defined to have a global minimum value of 0.0, except for f_5 with a minimum of about 1. From the comparison of the results, it can be seen that the modified BPSO provides better or similar performance compared with that of the original BPSO. Specifically, in f_4, f_7, f_8, and f_{10} tests, it is shown that the modified BPSO is significantly superior to the original

Table 2
Comparison of the results (without mutation)

F	Modified BPSO	Original BPSO	PSO ($V_{\max} = 4.0$)
f_1	0.00015 ± 0.00001	0.00095 ± 0.00067	0.00317 ± 0.00058
f_2	0.00041 ± 0.00007	0.00009 ± 0.00002	0.00246 ± 0.00065
f_3	0.0 ± 0.0	0.0 ± 0.0	0.21928 ± 0.01074
f_4	2.66643 ± 0.36769	31.91915 ± 4.58588	0.24186 ± 0.09542
f_5	1.02889 ± 0.07352	0.998 ± 0.0	1.13258 ± 0.16271
f_6	0.0073 ± 0.00042	0.00923 ± 0.00212	0.00219 ± 0.0004
f_7	8.50211 ± 4.80884	121.806 ± 10.3229	2.79257 ± 0.16396
f_8	3.63762 ± 0.62202	20.935 ± 0.05193	1.59434 ± 0.0864
f_9	26769.2 ± 0.0	26772.3 ± 1.25772	130.486 ± 12.5939
f_{10}	49.82 ± 7.35563	309.605 ± 9.19442	88.275 ± 5.73963

Table 3
Comparison of the results (with mutation)

F	r_{mu}	Modified BPSO	r_{mu}	Original BPSO
f_1	0.3	0.00013 ± 0.000014	0	0.00095 ± 0.00067
f_2	0.7	0.000099 ± 0.000002	0.2	0.000035 ± 0.000111
f_3	all	0.0 ± 0.0	all	0.0 ± 0.0
f_4	0.7	2.27598 ± 0.529546	0.4	27.655961 ± 5.796165
f_5	0.7	0.998004 ± 0.0	all	0.998004 ± 0.004209
f_6	0	0.007303 ± 0.000422	0.5	0.007291 ± 0.004209
f_7	0.4	6.128562 ± 3.650294	0.4	117.545935 ± 9.677798
f_8	0	3.637617 ± 0.622022	0.4	20.875734 ± 0.08873
f_9	0	26769.17 ± 0.0	0.4	26771.578 ± 0.8541
f_{10}	0	49.82 ± 7.355633	0.4	307.561 ± 13.2992

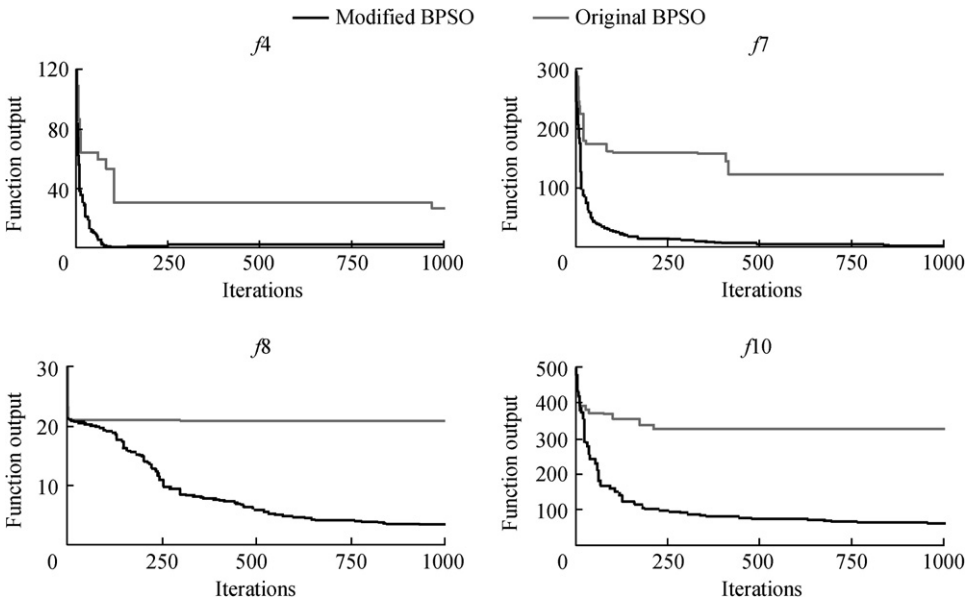


Fig. 2. Modified BPSO vs. original BPSO.

BPSO. Fig. 2 further illustrates the comparison of the performances according to iterations for the above four functions (i.e., f_4, f_7, f_8, and f_{10}). In the graph, regarding f_8 and f_{10}, it was predicted that the modified BPSO would not converge even in 1000 iterations. Thus, we continued the iterations to 10,000, and obtained the results of 0.49683 and 20.53986 for f_8 and f_{10}, respectively. Moreover, the encouraging fact is that the modified BPSO provides superior performance to the continuous PSO for the f_{10} test. On the other hand, from the mutation tests (Table 3), it was also demonstrated that a mutation operation has the potential to enhance the performance of the BPSO. However, the impact of *genotype–phenotype* adoption is more powerful than that of a mutation, in the BPSO.

5. Conclusion

This paper proposed a modified BPSO to optimize binary problems using concepts of the genotype–phenotype and the mutation of genetic algorithm. The main characteristic of the modified BPSO is that it can be considered as a continuous PSO. The proposed method was compared with the original BPSO and the standard continuous PSO on ten benchmark functions. Experimental results demonstrated that our proposed BPSO has a high possibility of providing an excellent performance for binary space problems.

Acknowledgements

This work was supported by Plant Technology Advancement Program funded by Ministry of Construction & Transportation (Grant No. C106A1520001-06A085600210) and the UCN Project of the MIC 21C Frontier R&D Program of Korean government.

References

[1] Kennedy J, Eberhart RC. Particle swarm optimization. Proceedings of IEEE international conference on neural networks, vol. IV. NJ: Piscataway; 1995. p. 1942–8.
[2] Eberhart RC. A discrete binary version of the particle swarm algorithm. In: Proceedings of 1997 conference systems man cybernetics, NJ: Piscataway; 1997. p. 4104–8.
[3] Alberts B, Bray D, Lewis J, et al. Molecular biology of the cell. 3rd ed. New York: Garland Publishing; 1994.
[4] Shi Y, Eberhart RC. A modified particle swarm optimizer. In: Proceedings of IEEE international conference on evolutionary computation, Alaska, USA: Anchorage; 1998. p. 69–72.
[5] Clerc M, Kenney J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 2002;6:58–73.
[6] Clerc M. Particle swarm optimization. ISTE; 2006.
[7] Kennedy J, Mendes R. Population structure and particle swarm performance. In: Proc 2002 Congress Evol Comput 2002; 2: p. 1671–6.
[8] Lee S, Park H, Jeon M. Binary particle swarm optimization with bit change mutation. IEICE Trans Fundam Electron Commun Comput Sci 2007;E-90A(10):2253–6.
[9] Bean JC. Genetics and random keys for sequencing and optimization. ORSA J Comput 1994;6:154–60.
[10] Holland JH. Adaptation in natural and artificial system. USA: University of Michigan Press; 1975.
[11] Available from: http://www.denizyuret.com/pub/aitr1569/node19.html.
[12] Pampara G, Franken N, Engelbrecht AP. Combining particle swarm optimization with angle modulation to solve binary problems. Evolut Comput IEEE Congress 2005;1:89–96.