International Journal of Innovative Computing, Information and Control
Volume **10**, Number **6**, December **2014**

ICIC International ©2014 ISSN 1349-4198
pp. 2127–2142

A MODIFIED PARTICLE SWARM OPTIMIZATION WITH MUTATION AND REPOSITION

Chiabwoot Ratanavilisagul and Boontee Kruatrachue

Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Chalongkrung Rd., Ladkrabang, Bangkok 10520, Thailand
chaibwoot@hotmail.com

Received January 2014; revised May 2014

Abstract. *The common problems of particle swarm optimization (PSO) are those of trapping in local optimum and premature convergence. This research paper aims to develop a solution to both problems by introducing mutation around particles and employing the reposition technique. The concurrent use of the introduced mutation and reposition has proved to solve both problems and enhanced the PSO performance; and thus is employed in this research. The proposed technique is termed MRPSO. MRPSO is tested on sixteen benchmark functions and the multidimensional knapsack problems (MKP). MRPSO yields the more satisfactory search results than the genetic algorithm (GA) and PSOs for the benchmark functions and the MKPs.*
Keywords: Particle swarm optimization, Binary particle swarm optimization, Mutation operator, Multidimensional knapsack problem, Genetic algorithm

1. **Introduction.** Kennedy and Eberhart [1,2] were the first to introduce the particle swarm optimization (PSO) in 1995 by observing the behaviors of animals, e.g., bird flocking and fish schooling. Their movements and communication mechanisms were thoroughly studied. In comparison with several other population-based stochastic optimization methods, such as the genetic algorithm (GA) [3,4] and the evolutionary programming (EP), PSO performs better in solving various optimization problems with fast and stable convergence rates [5-7].

The advantages of PSO [8,9] are its simplicity, rapid convergence, and few parameters to be adjusted. However, PSO has its own disadvantages of premature convergence to a local optimum and high chances of trapping in the local optimum [8,9]. To overcome both problems, researchers [9-27] increased search diversity in the population of PSO to prevent stagnation of the search in the local optimum by adding mutation operators in the PSO process.

This research has introduced mutation around individual particles (X) where the mutation values update the individual's best position (PBEST) and the best position found in the whole swarm (GBEST). In comparison with the standard PSO, this procedure increases search diversity without increasing the convergence speed; and is termed MXUPG. The experimental results of MXUPG on the benchmark test functions show MXUPG yields better solutions in the test functions than other mutation methods and the standard PSO.

The local optimum trapping is possible for both MXUPG and the standard PSO. This phenomenon leads to the stagnation of search in which the solution obtained at the

point of trapping is repeatedly produced irrespective of the length of search time. Consequently, certain research studies [28-31] added the reinitialize particles method (i.e., re-initialization) into PSO to improve the search solution. However, the reinitialize particles method requires a restart when trapping occurs. Therefore, an additional time is needed to re-converge in the new run. The particles reposition method (i.e., reposition) is proposed in this research in place of the reinitialize particles method to reduce the required re-convergence time. In addition, resetting both PBEST and GBEST yields the best results since the particles have more chances to converge in new areas after reposition. Hence, the particles reposition method with the resetting of both PBEST and GBEST (to be called RPG) is used to reduce the re-convergence time and increase the possible convergences in the new areas. In addition, the experimental results of RPG on the benchmark test functions yield better solutions in the test functions than other re-initialization methods and the standard PSO.

To improve the search performance, a novel PSO algorithm is created by combining MXUPG and RPG (to be called MRPSO). A set of benchmark test functions is used to compare the proposed MRPSO method with the standard PSO [1,2], an adaptive particle swarm optimization (APSO) [18], a hybrid particle swarm optimization which incorporates henon map mutation operation (HPSO) [19], the particle swarm optimization with reinitialize particles (PSOR) [28], and the floating point representation for genetic algorithm (FGA) [32]. The results show that the proposed MRPSO yields the best solutions in all test functions. Moreover, tests on the multidimensional knapsack problems (MKP) are performed with the proposed MRPSO, the binary representation for genetic algorithm (BGA) [32], the standard binary particle swarm optimization (BPSO) [33], the genotype-phenotype modified binary particle swarm optimization (GPMBPSO) [34] and the modified binary particle swarm optimization (MBPSO) [35]. The results are then compared. The comparison shows that the proposed MRPSO yields the best solutions in all instances of MKP.

The rest of this paper is organized as follows. Sections 2 describes the standard PSO and BPSO. The performance analysis of the standard PSO and the PSO with mutation is detailed in Section 3. The proposed MRPSO method is discussed in Section 4 while Section 5 presents the experiment setup, the experiment results, and the detailed result analysis. Section 6 discusses certain applications of the proposed method. Section 7 is the conclusion.

2. Particle Swarm Optimization and Binary Particle Swarm Optimization.

2.1. Particle Swarm Optimization.

In the standard PSO, each population member is called a "particle" with its own position and velocity. Each individual particle performs search in the search space according to its velocity, GBEST and PBEST. The standard PSO algorithm starts with randomizing particle positions and their respective velocities. The position evaluation of each particle is achieved using the objective function of the optimization problem. In a given iteration, each individual particle updates its position and velocity according to the expression below:

$$V'_{id} = \varpi V_{id} + \eta_1 \, rand()(P_{id} - X_{id}) + \eta_2 \, rand()(P_{gd} - X_{id}) \tag{1}$$

$$X'_{id} = X_{id} + V'_{id} \tag{2}$$

where X'_{id} is the current positions of i particle and d dimension, X_{id} is the previous positions of i particle and d dimension, and V_{id} is the previous velocity of i particle and d dimension. V'_{id} is the current velocity of i particle and d dimension, P_{id} is PBEST of i particle and d dimension, and P_{gd} is GBEST of d dimension. η_1 and η_2 are acceleration

constants, $0 \leq \varpi < 1$ is an inertia weight, and $rand()$ generates random number from interval $[0, 1]$. A limit velocity is represented with $V_{\max}$. Thus, if the calculated velocity of a particle exceeds $V_{\max}$, it will be replaced with $V_{\max}$.

2.2. **Binary Particle Swarm Optimization.** Kennedy and Eberhart [33] were the first to introduce the binary particle swarm optimization (BPSO) algorithm to allow the standard PSO to operate in the binary problem spaces. BPSO has since been adopted as a standard. In this algorithm, BPSO updates the velocity according to (3). The velocity is a set of real numbers and the particle position is a set of bits. Consequently, the velocity must be transformed into a set of probabilities using the sigmoid function as shown in (4). The principle of the position update is that the velocity dictates a probability that a position (bit) selects either zero or one; and the position update follows (5), where $sig(v'_{id})$ is the sigmoid function for transforming the velocity into the probability.

$$v'_{id} = v_{id} + \eta_1 \, rand()(P_{id} - X_{id}) + \eta_2 \, rand()(P_{gd} - X_{id}) \tag{3}$$

$$sig(v'_{id}) = \frac{1}{1 + e^{v'_{id}}}, \quad e \approx 2.7182818 \tag{4}$$

$$X'_{id} = \begin{cases} 1 & \text{If } rand() < sig(v'_{id}) \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

3. The Performance Analysis of Standard PSO and PSO with Mutation.

3.1. **Performance analysis of standard PSO.** The search performance of the standard PSO is subject to types of optimization problems. For instance, if it is a unimodal problem which consists of one single global optimum point [36], the standard PSO performs well although the time required to search for the global optimum point is long. On the other hand, in a multimodal problem in which there are many local optimum points and one global optimum [36], the local optimum trapping frequently occurs in the standard PSO due to the position update [10], the effect of (1) [17], and the attraction of GBEST [23].

3.2. **Performance analysis of PSO with mutation.** This subsection discusses the performances after adding mutation operators in the PSO process; and the advantages and disadvantages.

The major advantage of mutation implementation is the increased population variability. The mutation could reduce the trapping of the standard PSO and yield the better search results. On the other hand, the disadvantage is the difficulty of adjusting the mutation parameters to any specific problems. In other words, if the parameters were under-mutated, the trapping could occur, making the mutation useless.

In the case of a non-compared mutation (NCM) operator [9-18] in which a mutation value from any given point is applied without comparing against the original value prior to mutation, if the parameters were over-mutated, the convergence of particles would be less likely. This induces a random search and the search result is poorer than that of the standard PSO.

In the case of a compared mutation (CM) operator [19-27] in which a mutation value from any given point is applied only if the mutation value is better than the original value prior to mutation, over-mutation of the parameters in a multimodal problem leads to a faster convergence speed than the standard PSO. This could result in premature convergence and a poorer search result than the standard PSO. In addition, CM is sometimes called the local search [23,24].

4. A Modified Particle Swarm Optimization with Mutation and Reposition.

4.1. The mutation around a point.

To improve the performance of CM in multimodal problems, this research has proposed the novel algorithm that maintains the variability in the population without significant increase in the convergence speed when compared with the PSO. In applying the novel algorithm, each iteration mutation is introduced around particles. PBEST and GBEST are compared and updated with the resulting mutation values, rather than directly updating the particle positions with the mutation values. The update criteria are that if the resulting mutation values are better than the existing PBEST and GBEST, the PBEST and GBEST are replaced with the mutation values; and that if the resulting mutation values are poorer than the current PBEST and GBEST, both PBEST and GBEST are retained. The goal of this indirect update is to maintain the convergence speed of the standard PSO. The proposed novel algorithm is called the "mutation around particle position and update of PBEST and GBEST (MXUPG)". The MXUPG method is less likely to disrupt the convergence of particles since the update of particle positions is regulated by the standard PSO.

4.2. The particles reposition.

As previously mentioned, the major shortcoming of the standard PSO is the frequent local optimum trapping. Fortunately, the trapping is readily detectable since GBEST is not updated for an indefinite period of time, as shown in Figure 3.

In the multimodal problem, when trapping is detected, the re-initialization method restarts another round of search until a maximum iteration is reached, rather than leaving the trapping problem unsolved. This allows the reinitialization particles to search for solutions in other areas and returns the better search solutions than the standard PSO.

However, the re-initialization process requires the initializing of all particles. In addition, a significant amount of time is needed for the particles to converge in each subsequent restart. The amount of time required for re-convergence in each restart reduces the number of possible reruns and subsequently the opportunity to search for better solutions in other areas. To quicken the convergence in each new run, this research has introduced the particles reposition method into the standard PSO.

The reposition is the use of NCM to allow for a vast distribution of particles prior to the solution search by the standard PSO. The reposition is different from the re-initialization in that it applies mutation to certain dimensions of the particles instead of re-initializing all values.

However, the inappropriate application of the re-initialization and reposition methods should be avoided since overuse could disturb the convergence of particles and results in a random search. On the other hand, the methods should be applied when local optimum trapping is detected and the trapping is long enough to guarantee that the trapping problem does occur.

When the reposition begins, particles change their positions while PBEST and GBEST could be either altered or unaltered. In the case of unaltered GBEST or unaltered PBEST or both unaltered before reset, the particles return from their new positions to the previous GBEST and PBEST. In the return to the previous GBEST, if the particles locate a better position than the previous GBEST, the better position is then occupied as a new GBEST. Typically, the positions along the return path are worse than the previous GBEST. Therefore, the swarm often returns to the previous GBEST and is re-trapped in the former area. This restricts the search area to the area around the previous GBEST.

To avoid the search area restriction in the reposition, both GBEST and PBEST should be reset. The reset allows particles to converge in new areas. The new areas could be either better or worse than the areas prior to reset since the process is random, as shown

by the restart from point A in Figure 3. However, resetting both GBEST and PBEST increases the chances of searching in better areas although there is a possibility that the swarm returns to the pre-reset area even with the resetting of both GBEST and PBEST, as shown by the restart from point B in Figure 3. The reposition method used together with the resetting of both PBEST and GBEST is called RPG.

The RPG method can solve the local optimum trapping when it occurs and reduce the required re-convergence time. Moreover, the method increases the chances of convergence in new areas after reposition.

4.3. **A modified particle swarm optimization with mutation and reposition.** MXUPG and RPG can be concurrently applied to improve the PSO performance. MXUPG increases the variability in the population to avoid the local optimum trapping and RPG solves the trapping problem when it occurs. Therefore, this research proposes integrating the two methods to improve the PSO performance. The use of the integrated methods (i.e., MXUPG and RPG) yields better results than the use of either one of the methods, as shown in Tables 2 and 3. The proposed integrated method is called MRPSO.

MRPSO begins searching using MXUPG and the search stops when an optimal solution is found. On the other hand, if a trapping occurs, RPG is activated to solve the trapping. When the trapping is solved, MXUPG resumes the search. RPG is reactivated if another trapping is detected until another optimal solution is found. The pseudo code of MRPSO is shown below:

```
Initial particles of each particle                                          1
While (termination condition ≠ true) do                                     2
    Evaluate the fitness of each particle                                   3
    If fitness of each particle is better than PBEST, update PBEST          4
    If fitness of each particle is better than GBEST, update GBEST          5
    Update each particle position according to (1) and (2)                  6
    For i = 1 to N                                  (Mutation step)         7
        For j = 1 to RM                                                     8
            For d = 1 to D                                                  9
                Tx_d = x_id                                                 10
                If PM > rand() then                                         11
                    Apply (6)                                               12
                End If                                                      13
            Next d                                                          14
            If fitness of Tx is better than PBEST, update PBEST = Tx        15
            If fitness of Tx is better than GBEST, update GBEST = Tx        16
        Next j                                                              17
    Next i                                  (End mutation step)             18
    If (times of GBEST consecutive unchanged) ≥ TR   (Reposition step)      19
        Reset GBEST, PBEST                                                  20
        For i = 1 to N                                                      21
            For d = 1 to D                                                  22
                If PR > rand() then                                         23
                    Apply (7)                                               24
                End If                                                      25
            Next d                                                          26
        Next i                                                              27
    End If                                  (End reposition step)           28
End while                                                                   29
```

Equations (6) and (7) are mutation equations used in MRPSO. A positive operator is selected if the random number generated uniformly in the range $[0, 1]$ is less than 0.5 and

a negative operator otherwise.

$$Tx_d = Tx_d \pm (Tx_d \times rand()) \tag{6}$$

$$x_{id} = x_{id} \pm (x_{id} \times rand()) \tag{7}$$

where RM is the rounds of mutation, PM is the probability of mutation, PR is the probability of reposition, TR is the threshold of reposition, and Tx is the temporary particle. Tx_d is the temporary particle of d dimension, x_{id} is the positions of i particle of d dimension, N is the size of the population, and D is the dimension of the solution space.

5. **Results and Discussions.** To prove the efficiency of the proposed MRPSO, it is tested on sixteen benchmark functions [37,38], as listed in Table 1. Functions 1-9 are multimodal problems while the remaining Functions 10-16 are unimodal problems. The MRPSO results on the benchmark functions are compared with those of PSO, FGA, APSO, HPSO, PSOR, MXUPG, and RPG. The benchmark functions are representative of the floating-point representation. To prove the MRPSO ability in solving other representations, MKP has been added as a binary representation. This research selects the following MKPs from OR-Library [39], i.e., SENTO [40], WEING [41], WEISH [42], PB [43] and HP [43]. The MRPSO results on MKPs are compared with those of BPSO, BGA, GPMBPSO, MBPSO, MXUPG, and RPG.

TABLE 1. Details of benchmark test functions

Problem no.	Function name	Expression	Search space $[X_{\max}, X_{\min}]$	Objective function value	Dim	Attribute	$V_{\max}$
1	Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$x \in [-32.768, 32.768]^n$	0	50	Multimodal	32.768
2	Griewank	$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \Pi_{i=1}^{n} \cos\left(x_i/\sqrt{i}\right) + 1$	$x \in [-300, 300]^n$	0	50	Multimodal	300
3	Rastrigin	$f(x) = 10n + \sum_{i=1}^{n}(x^2 - 10\cos(2\pi x_i))$	$x \in [-5.12, 5.12]^n$	0	50	Multimodal	5.12
4	RosenBrock	$f(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$x \in [-2.048, 2.048]^n$	0	50	Multimodal	2.048
5	Schwefel	$f(x) = 418.9829 \times n + \sum_{i=1}^{n}\left(x_i \times \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$x \in [-500, 500]^n$	0	50	Multimodal	500
6	Schaffer's F6	$f(x) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1.0 + 0.001 \times (x^2+y^2))^2}$	$x \in [-100, 100]^n$	0	2	Multimodal	100
7	Step	$f(x) = \sum_{i=0}^{n-1} (\lfloor x_i \rfloor + 0.5)^2$	$x \in [-5.12, 5.12]^n$	0	50	Multimodal	5.12
8	Cosine Mixture	$f(x) = -0.1 \times \sum_{i=0}^{n}\cos(5\pi x_i) + \sum_{i=0}^{n} x^2 + 0.1n$	$x \in [-1, 1]^n$	0	50	Multimodal	1
9	Exponential	$f(x) = -\exp(-0.5\sum_{i=1}^{n} x_i^2) + 1$	$x \in [-1, 1]^n$	0	50	Multimodal	1
10	Spherical	$f(x) = \sum_{i=1}^{n} x_i^2$	$x \in [-5.12, 5.12]^n$	0	50	Unimodal	5.12
11	Parallel Ellipsoid	$f(x) = \sum_{i=1}^{n} (i \times x_i^2)$	$x \in [-5.12, 5.12]^n$	0	50	Unimodal	5.12
12	Multimod	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert \times \prod_{i=1}^{n} \lvert x_i\rvert$	$x \in [-10, 10]^n$	0	50	Unimodal	10
13	Rotated Ellipsoid	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j^2)$	$x \in [-65.536, 65.536]^n$	0	50	Unimodal	65.536
14	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left[\sum_{i=1}^{n} \frac{i}{2} x_i\right]^2 + \left[\sum_{i=1}^{n} \frac{i}{2} x_i\right]^4$	$x \in [-5.12, 5.12]^n$	0	50	Unimodal	5.12
15	Cigar	$f(x) = x_i^2 + 100000\sum_{i=2}^{n} x_i^2$	$x \in [-10, 10]^n$	0	50	Unimodal	10
16	Brown 3	$f(x) = \sum_{i=0}^{n-1}\left[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}\right]$	$x \in [-1, 1]^n$	0	50	Unimodal	1

This research uses a personal computer with Intel Core i7 3770 with a 2.4-GHz CPU and 8 GB RAM, and Visual C++ as the programming language. To guarantee fairness, the positions and velocity of particles are set identical in all the experiments. The measures of algorithm performance in the experiments are as follows.

The mean best fitness value (MBF): MBF is the mean of best fitness in the final iteration from all running and indicates the search efficiency of an algorithm. In the case of the experiments with the benchmark functions, all functions have a zero as the minimum point. In Table 1, an entry less than 10^{-324} is given a value of zero. The closer the MBF is to the zero point, the better the method is. In the case of MKP, where the best known values are given, the closer the MBF is to the best known value, the better the method is.

The success round (SR): SR is the number of running rounds that yields the optimum solution and shows the reliability of an algorithm. A good algorithm can be identified with a high SR. The higher the SR is, the better the algorithm is.

The logarithm mean fitness values of all particles (LMFP): LMFP is the logarithm mean of fitness of all particles in each iteration. For the benchmark functions, any algorithm that produces LMFP closer to the zero point is a good algorithm. The faster the LMFP is to entering the stabilizing stage, the faster the convergence speed is.

The logarithm of population average distance among points (LD(t)): $D(t)$ [44] isthe distribution discrete degree between the particles in the population. $D(t)$ indicates the diversity of the population. In PSO, $D(t)$ can be calculated from (8). In MXUPG, $D(t)$ is summation of the results of (8) and (9).

$$D(t) = \frac{1}{M \times L} \sum_{i=1}^{M} \sqrt{\sum_{d=1}^{D} (x_{id}^t - \bar{p}_{id})^2} \tag{8}$$

$$D(t) = \frac{1}{M \times L \times RM} \sum_{r=1}^{RM} \sum_{i=1}^{M} \sqrt{\sum_{d=1}^{D} (mx_{idr}^t - \overline{mp}_{idr})^2} \tag{9}$$

where L is the diagonal length of search space, M is the size of the population, D is the dimension of the solution space, and x_{id}^t is the d dimension coordinate values of the i particle. $\overline{p}_{id}$ is the average value of the d dimension coordinate values of all particles, mx_{idr}^t is the d dimension coordinate mutation values from the i particle in the j round of mutation, and $\overline{mp}_{idr}$ is the average value of the d dimension coordinate values of mutation position from all particles in all rounds of mutation.

5.1. **Experiment of the mutation around a point.** This subsection discusses the property of the proposed MXUPG algorithm in relation to the standard PSO. The parameters of the benchmark test functions are presented in Table 1. The PSO parameters are as follows. η_1 and η_2 are 1.496180 and ϖ is 0.729844, as suggested by Bergh [30]. $V_{\max}$ is shown in Table 1. PM and RM are 0.10 and 5 rounds, respectively. To guarantee fairness, the number of evaluations is set identical in all methods, resulting in 1200 particles in PSO and 200 particles in MXUPG. In Figures 1 and 2, each function has 50 runs. The MXUPG pseudo code is similar to the MRPSO pseudo code except that no reposition step is present in MXUPG.

As seen in Figures 1 and 2, LD(t) and LMFP are high during the initial search stage. They decrease and then stabilize toward the end. As shown in Figure 1, at the stabilizing stage, LD(t) of PSO is much less than LD(t) of MXUPG. This phenomenon indicates that the population variability of MXUPG is more than that of PSO. Figure 2 shows that LMFPs of PSO and MXUPG stabilize at approximately 1200 iterations. Thus, the convergence speeds of MXUPG and PSO are almost identical. It is possible to conclude that MXUPG is less likely to induce to premature convergence.

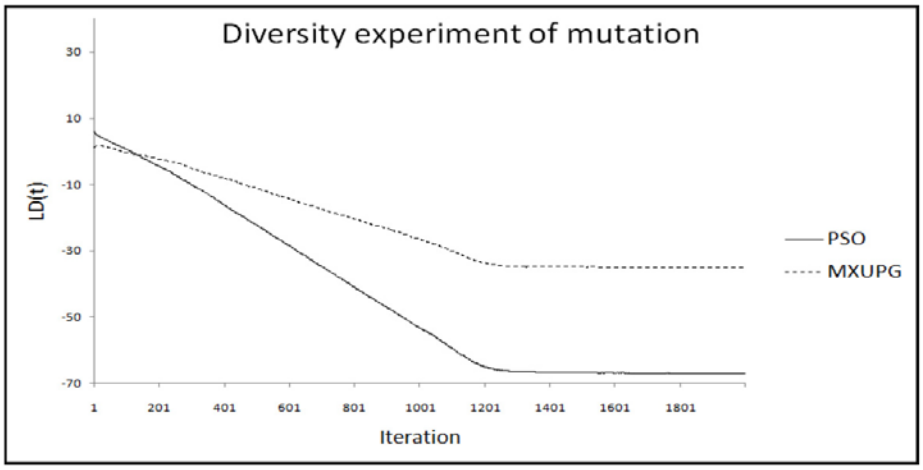

FIGURE 1. LD(t) of PSO and MXUPG of Ackley function

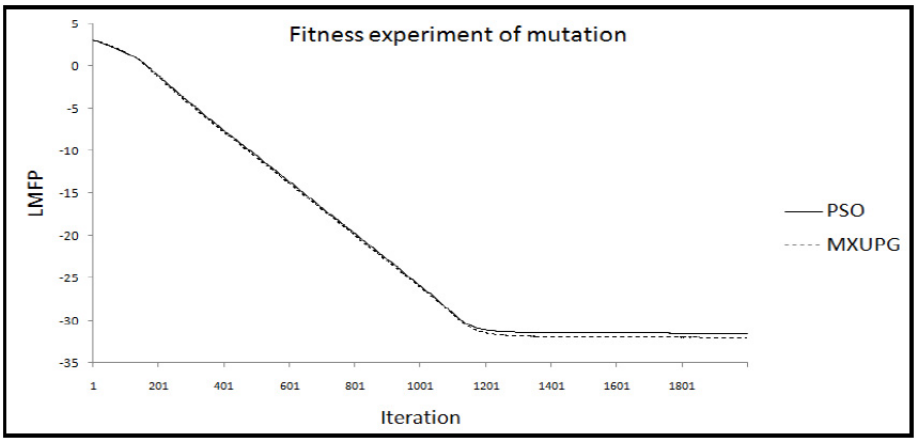

FIGURE 2. LMFPs of PSO and MXUPG of Ackley function

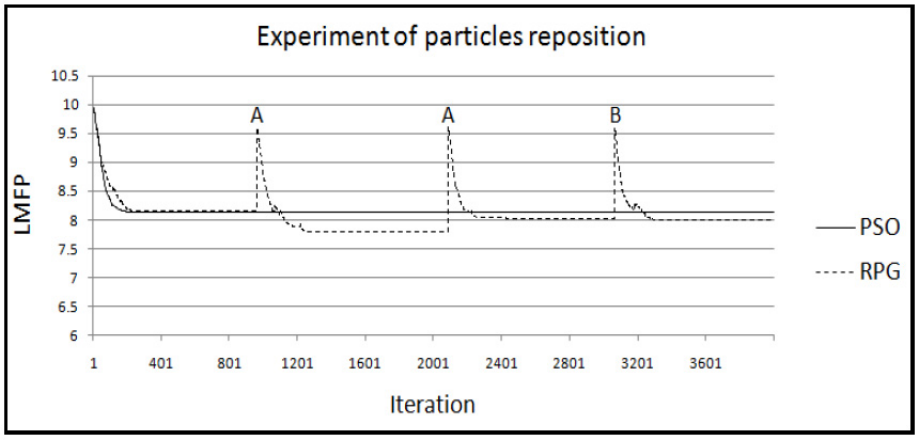

FIGURE 3. LMFPs of PSO and RPG of Schwefel function

5.2. **Experiment of particles reposition.** This subsection examines the property of the proposed RPG in relation to the standard PSO. In the reposition experiments, the parameters are set similarly to Subsection 5.1 except that the number of particles is 200 in all methods. In addition, TR and PR are set at 100 and 0.70, respectively. In Figure 3, this function runs only once since an average of the resetting results is difficult to detect.

The RPG pseudo code is similar to the MRPSO pseudo code except that the mutation step is not present in the RPG.

Figure 3 illustrates the local optimum trapping of PSO where LMFP initially decreases and then stabilizes. In the case of RPG, LMFP initially decreases and stabilizes until the local optimum trapping is detected. When the trapping occurs, the reposition is executed and rapidly increases LMFP. Afterward, the standard PSO starts searching for solutions and thus causes a decrease in LMFP.

5.3. **Experiment of proposed algorithm on benchmark functions.** This subsection investigates the performance of the proposed MRPSO on the benchmark functions listed in Table 1. The search results of MRPSO are compared with those of PSO, FGA, APSO, HPSO, RPG, PSOR and MXUPG. The PSO parameters, which are the underlying parameters of all algorithms, are identical to those in Subsection 5.1 except for the number of particles. The particles of PSO, FGA, APSO, HPSO, RPG, and PSOR are each 1200 particles while those of MRPSO and MXUPG are each 200 particles.

The non-PSO parameters are as follows. For APSO and HPSO, all parameters are identical to those in the original papers of APSO [18], HPSO [19] and PSOR [28], respectively. For FGA, the crossover probability and the mutation probability were set to 0.8 and 0.1, respectively, according to [45,46]. For RPG, MXUPG and MRPSO, the parameters are similar to those in Subsections 5.1 and 5.2. To guarantee fairness of the performance measurement, the mutation Equations (6) and (7) are applied to all methods. In Table 2, each function has 100 runs with a maximum iteration of 40000 iterations.

TABLE 2. Comparative results of PSO, FGA, APSO, HPSO, PSOR, RPG, MXUPG and MRPSO on the benchmark test functions

Techniques	PSO		FGA		APSO		HPSO		PSOR		MXUPG		RPG		MRPSO	
Problem no.	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR
1	1.21E-14	0	7.10e-015	0	0.109254	0	6.04E-15	0	2.39E-14	0	7.11E-15	0	0	100	0	100
2	0.00689239	54	0	100	0.00447973	67	0	100	0.00334614	80	0	100	0	100	0	100
3	91.138	0	0	100	147.532	0	92.1927	0	60.9643	0	0	100	0	100	0	100
4	0	100	13.521	0	0	100	0	100	0	100	0	100	0	100	0	100
5	2395.47	0	2221.31	0	2545.11	0	2283.74	0	2530.3	0	0	100	1968.11	0	0	100
6	0	100	0	100	0	100	0	100	0	100	0	100	0	100	0	100
7	0	100	0	100	0.22	86	0	100	0.05	95	0	100	0	100	0	100
8	0.0709364	63	0	100	0.791897	0	0	100	0.0635472	18	0	100	0	100	0	100
9	1.24E-16	0	1.11e-016	0	2.02E-16	0	7.11E-17	36	2.69E-16	0	1.09E-16	2	0	100	0	100
10	0	100	0	100	0	100	0	100	6.43E-317	0	0	100	0	100	0	100
11	0	100	0	100	0	100	0	100	2.23E-279	0	0	100	0	100	0	100
12	0	100	0	100	2.11E-159	64	0	100	0	100	0	100	0	100	0	100
13	0	100	0	100	0	100	0	100	1.90E-287	0	0	100	0	100	0	100
14	1.10E-148	0	1.3256e-154	0	4.94066e-324	51	1.26E-160	0	0.353901	0	0	100	5.27E-147	0	0	100
15	0	100	0	100	0	100	0	100	2.53E-192	0	0	100	0	100	0	100
16	0	100	0	100	0	100	0	100	9.57E-306	0	0	100	0	100	0	100

As seen in Table 2, for the multimodal functions, RPG and MXUPG generally outperform APSO, FGA, HPSO, PSOR, and PSO. Hence, RPG and MXUPG can improve the PSO performance. RPG can solve the local optimum trapping when it occurs. In addition, RPG can increase the possibility of convergence in new areas after reposition and reduce the required re-convergence time. Hence, RPG produces better results than PSO in the multimodal problems. MXUPG increases variability without increase in the convergence speed when compared with the standard PSO. The search results of MXUPG are better than those of the standard PSO. For the unimodal functions, the search results of RPG are similar to those of PSO since the reposition is not executed. In some multimodal functions, MXUPG is unable to locate the optimum points due to local optimum trapping. The MBF and SR results indicate that MRPSO can locate the optimum points

of all functions in Table 1. MRPSO outperforms PSO, FGA, APSO, HPSO, PSOR, RPG and MXUPG in the floating-point representation problems. MRPSO is obtained from the combination of RPG and MXUPG. Thus, the application of MXUPG with RPG produces better search results than the use of either MXUPG or RPG.

PSO yields the better search results than FGA in some functions, such as the Rosenbrock function. On the other hand, the research results of FGA are better than those of PSO in some other functions, e.g., the Rastrigin function. This indicates that PSO and FGA are appropriate for solving different optimization problems. Table 2 shows that MRPSO could overcome the drawbacks of the standard PSO.

5.4. **Experiment of proposed algorithm on multidimensional knapsack problem.** The multidimensional knapsack problem (MKP) is a widely researched discrete programming problem [47] and belongs to the group of NP-hard combinatorial optimization problems [47-49]. The MKP formula can be applied to various business requirements, such as the project selection problem, the capital budgeting problem, the cutting stock problem, the cargo loading problem, the production planning problem, the scheduling problem, the distributed processor problem, and the database allocation problem [49,50]. Hence, an algorithm that can solve an MKP would be able to solve every optimization problem with the same MKP formula and has many uses [50].

MKP consists of m knapsacks with maximum capacity C_j $(j = 1, \ldots, m)$ for each knapsack and a set of n objects, where each object i $(i = 1, \ldots, n)$ has a price (P_i) and a weight (w_{ij}). The goal of MKP is to select a subset of objects that maximizes the total price without exceeding the total capacity of all knapsacks. MKP can be expressed as

$$Maximize \sum_{i=1}^{n} p_i x_i \quad x_i \in \{0, 1\} \tag{10}$$

$$Object \sum_{i=1}^{n} w_{ij} x_i \leq C_j \quad x_i \in \{0, 1\} \tag{11}$$

This subsection investigates the performance of the proposed MRPSO on MKP. The search results of MRPSO are compared with those of BPSO, GPMBPSO, MBPSO, MXUPG, RPG and BGA. The application of MRPSO pseudo code to MKP requires changes to some parts of the MRPSO pseudo code. The changes and reasons are detailed as the following: a dimension (d) or bit, which represents an object, selects 1 if the object is selected and 0 otherwise. The standard PSO is replaced with BPSO, and if a bit changes from 0 to 1, BPSO checks the capacity of all knapsacks. If the capacity of all knapsacks can handle the weight of this object, the bit is allowed to change from 0 to 1; otherwise, the bit remains unchanged. The capacity check is to guarantee that the search does not exceed the solution space boundaries. The mutation step of the MRPSO pseudo code is to replace Line 12 (6) with $T_{xd} = 1$. The reposition step is edited by replacing Line 24 (7) with a flipped bit of x_{id}. In addition, a velocity reset is required since the particle velocity influences the particle bit change [51]. Unless the velocity is reset, the particle bits would return to the pre-reposition state and thus make the reposition useless.

The BPSO parameters are as follows: η_1 and η_2 are both set at 2, and $V_{\max}$ is set at 4, as recommended by Deep [35]. On the other hand, the non-BPSO parameters are as follows: the parameters of GPMBPSO and MBPSO are identical to those in the original papers of MBPSO [35]. In RPG, MXUPG and MRPSO, PM is 0.05, RM is 1 round, TR is 30 iterations, and PR is 0.3. For BGA, the crossover probability and the mutation probability were set at 0.8 and 0.1, respectively, according to [45,46]. To guarantee fairness, the number of evaluations is set identical in all methods. This results

in 1000 particles each in BPSO, BGA, GPMBPSO, MBPSO and RPG; and 500 particles each in MRPSO and MXUPG.

In Table 3, each instance has 100 runs with a maximum iteration of 5000 iterations. In the same table, the results of MBF and sums of SR show that MRPSO can locate the optimum points of all instances. MRPSO outperforms BPSO, BGA, MBPSO, GPMBPSO,

TABLE 3. Comparative results of BPSO, BGA, GPMBPSO, MBPSO, MXUPG, RPG and MRPSO on MKP

Techniques		BPSO		BGA		GPMBPSO		MBPSO		MXUPG		RPG		MRPSO	
Problem	Best known	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR	MBF	SR
Sento1	7772	7771.3	95	7772	100	7772	100	7770.88	98	7771.72	98	7771.58	97	7772	100
Sento2	8722	8718.91	67	8722	100	8722	100	8719.97	74	8719.64	76	8720.61	79	8722	100
Weing1	141278	141278	100	141278	100	141278	100	141278	100	141278	100	141278	100	141278	100
Weing2	130883	130883	100	130883	100	130883	100	130883	100	130883	100	130883	100	130883	100
Weing3	95677	95660.6	96	95677	100	95676	98	95661.6	98	95676.5	99	95677	100	95677	100
Weing4	119337	119337	100	119337	100	119337	100	119337	100	119337	100	119337	100	119337	100
Weing5	98796	98796	100	98796	100	98796	100	98796	100	98796	100	98796	100	98796	100
Weing6	130623	130623	100	130623	100	130623	100	130623	100	130623	100	130623	100	130623	100
Weing7	1095450	1095340	6	1094150	0	1095380	0	1095370	27	1095300	8	1095440	90	1095450	100
Weing8	624319	620857	14	624319	100	620594	0	621366	14	624319	100	622038	32	624319	100
Weish01	4554	4554	100	4554	100	4554	100	4554	100	4554	100	4554	100	4554	100
Weish02	4536	4535.9	98	4536	100	4536	100	4536	100	4536	100	4535.9	98	4536	100
Weish03	4115	4115	100	4115	100	4115	100	4115	100	4115	100	4115	100	4115	100
Weish04	4561	4561	100	4561	100	4561	100	4561	100	4561	100	4561	100	4561	100
Weish05	4514	4514	100	4514	100	4514	100	4514	100	4514	100	4514	100	4514	100
Weish06	5557	5556.02	94	5557	100	5549.72	44	5556.46	97	5556.87	99	5556.87	99	5557	100
Weish07	5567	5567	100	5567	100	5567	100	5567	100	5567	100	5567	100	5567	100
Weish08	5605	5605	100	5605	100	5605	100	5605	100	5605	100	5605	100	5605	100
Weish09	5246	5246	100	5246	100	5246	100	5246	100	5246	100	5246	100	5246	100
Weish10	6339	6339	100	6339	100	6339	100	6339	100	6339	100	6339	100	6339	100
Weish11	5643	5643	100	5643	100	5643	100	5643	100	5643	100	5643	100	5643	100
Weish12	6339	6339	100	6339	100	6339	100	6339	100	6339	100	6339	100	6339	100
Weish13	6159	6159	100	6159	100	6159	100	6159	100	6159	100	6159	100	6159	100
Weish14	6954	6954	100	6944.08	68	6954	100	6954	100	6954	100	6954	100	6954	100
Weish15	7486	7486	100	7486	100	7486	100	7486	100	7486	100	7486	100	7486	100
Weish16	7289	7288.94	97	7289	100	7289	100	7288.78	97	7288.96	98	7289	100	7289	100
Weish17	8633	8633	100	8633	100	8633	100	8633	100	8633	100	8633	100	8633	100
Weish18	9580	9579.26	94	9578.16	76	9580	100	9579.85	99	9580	100	9580	100	9580	100
Weish19	7698	7697.74	98	7664.08	8	7695.14	78	7697.61	97	7698	100	7698	100	7698	100
Weish20	9450	9450	100	9449.4	88	9450	100	9450	100	9450	100	9450	100	9450	100
Weish21	9074	9074	100	9072.95	85	9074	100	9073.76	99	9074	100	9074	100	9074	100
Weish22	8947	8940.88	66	8922.2	5	8930.44	8	8940.7	65	8943.58	81	8945.38	91	8947	100
Weish23	8344	8343.79	93	8306.55	0	8343.94	98	8343.76	96	8343.94	98	8344	100	8344	100
Weish24	10220	10219.8	99	10215.6	80	10220	100	10220	100	10220	100	10220	100	10220	100
Weish25	9939	9935.32	78	9918.95	20	9939	100	9937.56	91	9938.01	93	9938.84	99	9939	100
Weish26	9584	9575.93	69	9550.9	21	9584	100	9576.96	78	9578.24	82	9582.72	96	9584	100
Weish27	9819	9819	100	9798.15	33	9819	100	9819	100	9819	100	9819	100	9819	100
Weish28	9492	9492	100	9459	20	9492	100	9491.77	99	9492	100	9492	100	9492	100
Weish29	9410	9410	100	9381.85	39	9410	100	9409.52	98	9410	100	9410	100	9410	100
Weish30	11191	11190.6	90	11131.5	0	11191	100	11190.8	94	11190.8	94	11191	99	11191	100
PB1	3090	3090	100	3090	100	3090	100	3090	100	3090	100	3090	100	3090	100
PB2	3186	3185.28	96	3113.92	14	3186	100	3184.2	93	3185.82	99	3185.82	99	3186	100
PB4	95168	95168	100	95168	100	95168	100	95168	100	95168	100	95168	100	95168	100
PB5	2139	2137.98	94	2129.36	50	2137.64	96	2139	100	2139	100	2138.83	99	2139	100
PB6	776	776	100	776	100	776	100	776	100	776	100	776	100	776	100
PB7	1035	1034.96	96	1035	100	1035	100	1034.8	98	1034.91	99	1035	100	1035	100
HP1	3418	3416.18	92	3418	100	3404	0	3416.61	95	3418	100	3417.58	97	3418	100
HP2	3186	3186	100	3106.48	4	3186	100	3185.82	99	3186	100	3185.82	99	3186	100
Sum of SRs			4432		3611		4322		4506		4624		4674		4800

RPG and MXUPG in terms of reliability and solution quality of the binary representation problems. This is shown by MBFs and the sums of SR, both of which are higher than other algorithms. In addition, Table 3 reaffirms the fact that the use of MXUPG with RPG produces better search results than the use of either MXUPG or RPG.

As shown in Table 3, the BPSO search results are better than those of BGA in some instances, such as Weish23 and Weish30. On the other hand, BGA gives better search results than BPSO in some other instances, e.g., Sento2 and Weing8. The phenomena confirm that both algorithms are capable of solving different optimization problems. In addition, the table shows that MRPSO could overcome the PSO drawbacks.

6. **Applications of the Proposed Algorithm.** The PSO and GA algorithms belong to the evolutionary algorithm (EA) group [52] and have several similar useful properties to EA [52]. This research demonstrates the practical use of the proposed MRPSO algorithm by applying it to the optimization problems and presents the benefits of EA.

The optimization problems are found in various business problems [53], such as the project selection problem, the capital budgeting problem, the cutting stock problem, the cargo loading problem, the production planning problem, the scheduling problem, the distributed processor problem, the database allocation problem, and the knapsack problem (KP). In addition, they are found in different logistic operations, e.g., the logistics and transportation problem, the shortest path problem, the maximum flow problem, the generalized assignment problem, the traveling salesman problem (TSP), and the vehicle routing problem. The optimization problems are also discovered in several other problem types, e.g., the blending problem and the crop planning problem.

In the case of optimization problems with small solution space, e.g., KP which consists of 5 items, the brute force method can locate the optimum point because the search is carried out with all feasible solutions (i.e., 32 solutions). On the other hand, in the case of optimization problems with very large solution space, such as KP with 500 items, the total number of feasible solutions is 2^{500}. Therefore, the brute force method is unsuitable since it requires a very long time for the method to locate the optimum point. In this case, other methods (e.g., PSO and GA) should be adopted to solve the optimization problems with very large solution space.

One major benefit of PSO is its applicability to the optimization problems with very large solution space. The basic principle of PSO is that it would not search for all feasible solutions but would focus on the areas that produce good results. This narrows down the scope of solution search. Therefore, PSO can search for the solutions very efficiently since it is typical to locate better solutions around the areas that previously produced good solutions.

A problem-specific algorithm produces a better optimization performance than an EA (e.g., PSO and GA), but the problem-specific algorithm is only applicable to the type of the optimization problem for which it was designed [36]; for example, the Lin-Kernighan heuristics can solve TSP efficiently but cannot be applied to MKP. On the other hand, the repair operator algorithm [54] can solve MKP efficiently but cannot be applied to TSP. In addition, implementing the problem-specific algorithm requires the knowledge of the method to solve that specific optimization problem. Hence, the implementation of the problem-specific algorithm is complicated and is limited to only one specific problem. In contrast, implementing PSO is not subject to the optimization problems. In other words, implementing PSO requires adjusting certain parameters to fit the problems [36], such as amount of generation, amount of population, and constraint of the optimization problem.

Another main benefit of PSO is its ease of implementation. In addition, after the implementation, PSO can be reused with other optimization problems.

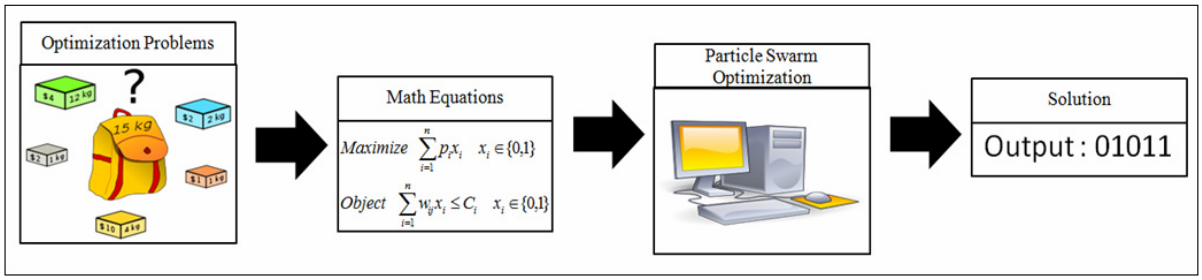

FIGURE 4. A practical application of PSO

Figure 4 illustrates a practical application of PSO. The optimization problems are converted into mathematical equations. If the equations can be solved, the optimization problems would be solved at the same time. Then, PSO searches for the solutions of the problems.

Many optimization problems contain local optimum in solution space [36], so if an algorithm were poorly designed, the local optimum trapping incidence would occur. If the trapping happened, the best solution would not be found. However, if there were no trapping, the best solution would be located quickly.

One main drawback of the standard PSO is a tendency of the local optimum trapping to occur. Therefore, the aim of this research is to minimize the local optimum trapping of PSO. The improved performance of PSO is achieved by combining the mutation technique with the reposition technique.

The experimental outcomes in Subsections 5.3 and 5.4 confirm that the proposed MRPSO method can improve the efficiency and effectiveness of PSO and that the best results are achieved with the proposed MRPSO when compared with the other algorithms. The proposed MRPSO has the highest efficiency because it has the same number of evaluations as the other algorithms but produces the best solutions. In terms of effectiveness, the proposed MRPSO achieves the highest effectiveness since it minimizes the local optimum trapping of PSO. Thus, the MRPSO algorithm can locate every global optimum point in the experiments, whereas PSO fails to locate the global optimum points in several functions and instances.

In practice, the proposed MRPSO is applicable to solving the optimization problems. In addition, it has been proved that the proposed MRPSO solves the optimization problems better than the standard PSO. Many previous research papers studied hybrid algorithms, such as that which combines PSO with the repair operator algorithm to solve MKP [54], and that which combines the Lin-Kernighan heuristics with PSO to solve TSP [55]. The hybrid algorithms improve the solution search performance. In the hybrid algorithms, if PSO were replaced with the proposed MRPSO, the MRPSO hybrid algorithm would produce a better result than the PSO hybrid algorithm.

7. **Conclusion.** This research paper has proposed MRPSO derived from the concurrent application of MXUPG and RPG to solving the optimization problems. MXUPG increases the population diversity without increasing the convergence speed relative to the standard PSO. This reduces the occurrence of local optimal trapping. However, since some trappings are inevitable with the application of only MXUPG, RPG is therefore introduced and used in combination. The reason is that RPG increases the chances of finding better solutions when the trappings occur.

Thus, the concurrent application of MXUPG and RPG is proposed to improve the PSO performance. On the benchmark functions, the MRPSO, PSO, FGA, APSO, PSOR and

HPSO are tested and the results are compared. In addition, the MRPSO, BGA, BPSO, GPMBPSO and MBPSO are tested on MKP and the results are compared. The results indicate that the proposed MRPSO outperforms PSO, FGA, APSO, HPSO, PSOR, BGA, BPSO, MBPSO, and GPMBPSO with regard to the reliability and quality of solutions in all the experiments. The proposed MRPSO is able to find the optimal points.

REFERENCES

[1] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of IEEE International Conference on Neural Networks*, Perth, Western Australia, vol.4, pp.1942-1947, 1995.

[2] J. Kennedy and R. Eberhart, A new optimizer using particle swarm theory, *The 6th International Symposium on Micro Machine and Human Science*, pp.39-43, 1995.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1979.

[4] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[5] K. E Parsopoulos, V. P. Plagianakos, G. D. Magoulas and M. N. Vrahatis, Objective function stretching to alleviate convergence to local minima, *Nonlinear Analysis, Theory, Methods and Applications*, vol.47, pp.3419-3424, 2003.

[6] R. Eberhart and Y. Shi, Comparison between genetic algorithms and particle swarm optimization, *Proc. of the 7th Annual Conference on Evolutionary Programming*, pp.611-619, 1998.

[7] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Mateo, 2001.

[8] B. Qinghai, Analysis of particle swarm optimization algorithm, *Computer and Information Science*, vol.3, no.1, pp.180-184, 2010.

[9] R. Chiabwoot and K. Boontee, Mutation period calculation for particle swarm optimization, *The 1st International Symposium on Technology for Sustainability*, pp.213-216, 2011.

[10] M. Lin and Z. Hua, Improved PSO algorithm with adaptive inertia weight and mutation, *World Congress on Computer Science and Information Engineering*, pp.627-625, 2009.

[11] N. Hiqushi and H. Iba, Particle swarm optimization with Gaussian mutation, *IEEE Conf. Swam Intelligence Symposium (SIS)*, pp.72-79, 2003.

[12] R. Thangaraj, M. Pant and A. Abraham, A new diversity guided particle swarm optimization with mutation, *World Congress on Nature and Biologically Inspired Computing*, pp.294-299, 2009.

[13] P. S. Andrews, An investigation into mutation operators for particle swarm optimization, *IEEE Congress on Evolutionary Computation*, pp.1044-1051, 2006.

[14] A. Stacey, M. Jancic and I. Grundy, Particle swarm optimization with mutation, *Proc. of IEEE International Conference on Evolutionary Computation*, vol.2, pp.1425-1430, 2003.

[15] J. Wei, L. Guangbin and L. Dong, Elite particle swarm optimization with mutation, *Asia Simulation Conference 7th Intl. Conf. on Sys. Simulation and Scientific Computing*, pp.800-803, 2008.

[16] J. Liu, J. Sun and W. B. Xu, Quantum-behaved particle swarm optimization with mutation operator, *Proc. of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pp.237-240, 2005.

[17] H. Choi, O. Shunichi, Y. Kazuho and O. Hiroaki, Improvement of particle swarm optimization application of the mutation concept for the escape from local minima, *The 8th International Conference on Supply Chain Management and Information Systems*, pp.1-5, 2010.

[18] A. ALFI, PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems, *Acta Automatica Sinica*, vol.37, no.5, pp.541-549, 2011.

[19] H. Gao and W. Xu, Particle swarm algorithm with hybrid mutation strategy, *Applied Soft Computing*, pp.5129-5142, 2011.

[20] J. Tang and X. Zhao, Particle swarm optimization with adaptive mutation, *WASE International Conference on Information Engineering*, pp.234-237, 2009.

[21] X. Wu and M. Zhong, Particle swarm optimization based on power mutation, *ISECS International Colloquium on Computing, Communication, Control, and Management*, pp.464-467, 2009.

[22] C. Li, S. Yang and I. A. Korejo, An adaptive mutation operator for particle swarm, *Proc. of the UK Workshop on Computational Intelligence*, pp.165-170, 2008.

[23] J. Tang and X. Zhao, A hybrid particle swarm optimization with adaptive local search, *Journal of Networks*, pp.411-418, 2010.

[24] J. Tang and X. Zhao, Particle swarm optimization using adaptive local search, *International Conference on Future BioMedical Information Engineering*, pp.300-303, 2009.

[25] M. Pant, R. Thangaraj and A. Abraham, Particle swarm optimization using adaptive mutation, *Proc. of the 19th International Conference on Database and Expert Systems Application*, pp.519-523, 2008.
[26] L. Chen, Particle swarm optimization with a novel mutation operator, *International Conference on Mechatronic Science, Electric Engineering and Computer*, pp.970-973, 2011.
[27] M. S. Darweesh, M. M. Ghoneimy and H. A. Kamal, A new fixed channel assignment algorithm using adaptive mutation particle swarm optimization, *Innovations on Communication Theory*, pp.1-5, 2012.
[28] X. Liu, Q. Wang, H. Liu and L. Li, Particle swarm optimization with dynamic inertia weight and mutation, *The 3rd International Conference on Genetic and Evolutionary Computing*, pp.620-623, 2009.
[29] Y. Gao and Y. Duan, A new particle swarm optimization algorithm with adaptive mutation operator, *The 2nd International Conference on Information and Computing Science*, pp.58-61, 2009.
[30] F. van den Bergh, *An Analysis of Particle Swarm Optimizers*, Ph. D. Thesis, University of Pretoria, 2001.
[31] L. Ning, S. Debao, C. Yigang and Z. Tong, Particle swarm optimization with mutation operator, *Computer Engineering and Applications*, vol.l7, pp.12-14, 2004.
[32] C. Z. Janikow and Z. Michalewicz, An experimental comparison of binary and floating point representation in genetic algorithm, *Proc. of the 4th International Conference on Genetic Algorithm*, pp.31-36, 1991.
[33] J. Kennedy and R. Eberhart, A discrete binary version of the particle swarm algorithm, *Proc. of the World Multiconference on Systemics, Cybernetics, Informatics*, pp.4104-4109, 1997.
[34] S. Lee, S. Soak, S. Oh, W. Pedrycz and M. Jeon, Modified binary particle swarm optimization, *Progress in Natural Science*, pp.1161-1166, 2008.
[35] J. C. Bansal and K. Deep, A modified binary particle swarm optimization for knapsack problems, *Applied Mathematics and Computation*, pp.11042-11061, 2012.
[36] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, 2003.
[37] M. Marcin and S. Czeslw, *Test Functions for Optimization Needs*, 2005.
[38] P. A. Ernesto, *Multivariate Test Functions Library in C for Unconstrained Global Optimization*, Department of Mathematics U.P. Diliman, 2005.
[39] J. E. Beasley, *ORLib – Operations Research Library*, http://brunel.ac.uk/∼mastjjb/jeb/orlib/files/mknap2.txt.
[40] S. Senyu and Y. Toyoda, An approach to linear programming with 0-1 variables, *Management Science*, vol.15, pp.196-207, 1967.
[41] H. M. Weingartner and D. N. Ness, Methods for the solution of the multidimensional 0/1 knapsack problem, *Operations Research*, vol.15, pp.83-103, 1967.
[42] W. Shih, A branch and bound method for the multiconstraint zero-one knapsack problem, *Journal of Operations Research Society*, vol.30, pp.369-378, 1979.
[43] A. Freville and G. Plateau, Hard 0-1 multiknapsack test problems for size reduction methods, *Investigation Operativa*, vol.1, pp.251-270, 1990.
[44] T. Krink, J. S. Vesterstrom and J. Riget, Particle swarm optimization with spatial particle extension, *Proc. of IEEE Congress on Evolutionary Computation*, Honolulu, HI, USA, pp.1474-1479, 2002.
[45] J. J. Grefenstette, Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, pp.122-128, 1986.
[46] J. D. Schaffer, R. A. Caruana, L. J. Eshelman and R. Das, Study of control parameters affecting online performance of genetic algorithms for function optimization, *Proc. of the 3rd International Conference on Genetic Algorithms*, pp.51-60, 1989.
[47] S. Martello and P. Toth, Bin-packing problem in knapsack problems, *Algorithms and Computer Implementations*, pp.221-245, 1990.
[48] Y. Zhou, Z. Kuang and J. Wang, A chaotic neural network combined heuristic strategy for multidimensional knapsack problem, *ISICA*, pp.715-722, 2008.
[49] P. C. Chu and J. E. Beasley, A genetic algorithm for the multidimensional knapsack problem, *Journal of Heuristics*, vol.4, pp.63-86, 1998.
[50] J. Deane and A. Agarwal, Neural, genetic, and neurogenetic approaches for solving the 0-1 multidimensional knapsack problem, *International Journal of Management and Information Systems*, vol.17, pp.43-54, 2013.
[51] M. Khanesar, M. Teshnehlab and M. Shoorehdeli, A novel binary particle swarm optimization, *Proc. of the 15th Mediterranean Conference on Control and Automation*, Greece, Athens, pp.1-6, 2007.

[52] E. Elbeltagia, T. Hegazy and D. Griersonb, Comparison among five evolutionary-based optimization algorithms, *Advanced Engineering Informatics*, pp.43-53, 2005.

[53] R. A. Sarker and C. S. Newton, *Optimization Modelling: A Practical Approach*, CRC Press, Taylor and Francis Group, 2008.

[54] M. Kong and P. Tian, Apply the particle swarm optimization to the multidimensional knapsack problem, *Artificial Intelligence and Soft Computing*, vol.4029, pp.1140-1149, 2006.

[55] W. Pang, K. Wang, C. Zhou, L. Dong, M. Liu, H. Zhang and J. Wang, Modified particle swarm optimization based on space transformation for solving traveling salesman problem, *Proc. of the 3rd International Conference on Machine Learning and Cybernetics*, pp.2342-2346, 2004.