An Improved Particle Swarm Optimization Algorithm Using Information of Second Best Particles

Young-Bin Shin

Acknowledgements

I am deeply indebted to my supervisor Prof. Eisuke KITA for all the help and suggestions. Without his mentorship to accomplish this research, I would not be able to finish my work. I again express my gratitude to Prof. Eisuke KITA for his efforts. I would like to express my appreciation to the examiners of this thesis, Prof. Takaya ARITA and Prof. Masahiro OHKA for their invaluable time and efforts. I would like to thank all the students of KITA Lab for their support and friendship. I would also like to thank my family for their moral support, love and encouragements. Finally, I would like to express my appreciation to Ministry of Education, Culture, Sports, Science and Technology in JAPAN for their financial support through the scholarship.

Young-Bin Shin

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Despite the rapid development of computer performance, many optimization problems, particularly combinatorial optimization problems, have approached calculation time and computer capacity limits due to increase of the problem size. Combinatorial optimization problems crop up in data analysis, integrated circuit layouts and pattern recognition in engineering and financial investment, inventory management and production schedules in business administration. In addition, they exist in a variety of fields, such as economics and biology.

Since, in the optimization problem, realizable solutions should be obtained in actual time, a heuristic approach is often used. Heuristic approaches find a practical solution for a certain level of satisfaction, rather than seeking to obtain an ideal method, since it is impossible to consider all variables and conditions. Therefore, the heuristic approach is available when there is no method to solve the problem or the solution is not yet feasible. In addition, a heuristic approach is used either when a problem has not been clearly defined or has only incomplete information.

The heuristic method obtains sufficient knowledge or efficient solutions using repetition and experience gained via trial and error. This heuristic technique has the advantage of finding a suitable solution in a limited execution time to a problem that is difficult to prove mathematically.

Optimization is a methodology for finding a solution within constraints after formalizing a decision-making problem as an optimization model to achieve a goal. Meanwhile, the heuristic method is a theory that improves the repeatability in optimization problems by specifying the computation method.

A Heuristic algorithm enables an extensive search of a solution space. Due to there being no guaranteed optimal solution, the search becomes a a stochastic optimization model in heuristic algorithms. The optimization method study is difficult to develop for

solving problems with different characteristics. Accordingly, for high-level methods that are applicable to various problems and have no constraining information about specific problems, meta-heuristics are used [55, 19]. Meta-heuristic algorithms are being studied for solving practical problems, which are acceptable for combination optimization problems by modeling natural phenomena and behavior [43, 1].

The primary meta-heuristic algorithms have common concepts and theories that are simple and superior search performance in solution space; however, they have two problems. Firstly, they are not versatile, despite the basic features of meta-heuristics. Secondly, search success rate and search time of the algorithm are relatively low. To improve these problems, the improved algorithms of the meta-heuristic one are discussed in this thesis.

Combinatorial optimization problems belong to NP-hard problems, for which it is difficult to find the optimal solution in the polynomial time. To solve NP-hard problems, in the last few decades, many studies have been published in which several practical optimization problems have been addressed with various meta-heuristic algorithms such as Genetic Algorithms (GA) [24], Simulated Annealing (SA) [53] and Particle Swarm Optimization (PSO) [26]. These meta-heuristic algorithms have different performance levels in the search rate and the efficiency of the search time by each algorithm property and problem type.

Genetic algorithms (GA) are applicable to various types of problems; however, the search time has low efficiency in large search spaces. Simulated Annealing is also a time-consuming process. Therefore, many improved meta-heuristic algorithms have been applied to given NP-hard problems to improve the search time and local optimization. Furthermore, the hybrid method has been proposed to combine two or more algorithms to improve the performance of meta-heuristic algorithms [20, 22, 35, 48]. Two algorithms can be combined in a serial process according to their characteristics. Its combination usually connects a part of the preprocessing process or data processing.

Genetic Algorithms (GA) are one of the most widely used in stochastic global search methods for efficiently solving NP-hard problems, since Holland [24] first introduced GA in 1975. Holland presented the basic concepts of GA such as selection, recombination, crossover and mutation, based on Darwin's evolution theory, to search in a solution space. Goldberg [16] presented the GA as a concise approach to solving the search optimization of various types. After his work, GA became a useful technique for optimization problems. Moreover, since Dave Davis [33, 34] also discussed the utility of GA in advanced problems in 1991, GA has been improved gradually as a proper method for solving NP-hard problems. The feature difference of GA and other algorithms is to set the representation of actual variables. In GA, each variable is encoded in a string, called a chromosome. A population, a group of strings, is used to obtain genetic recombination through crossover

and mutation.

Simulated Annealing (SA) is a generic probabilistic meta-heuristic for the global optimization problem of locating a good approximation to the global optimal solution of a given function in a large search space [11, 14]. Simulated Annealing was developed by Kirkpatrick et al. [53] in 1983; the basic idea is to imitate the annealing process, a crystal is heated and then cooled slowly until it has a regular crystal lattice configuration, in solid-state physics. They suggested an analogy between minimizing the cost function of a combinatorial optimization problem and reducing the energy of a solid via slow cooling, which is the relationship between thermodynamic processes and a heuristic method for search performance.

An algorithm for the efficient simulation of the evolution of a solid to thermal equilibrium has already been proposed Metropolis et al. [44] in 1953. The original Metropolis scheme required that the state of a substance in a thermodynamic system was chosen at an energy and temperature. When reducing the temperature in the annealing process, the resultant change in the energy is computed. The new configuration of the substance lattice is probabilistically accepted by the resulting energy, it uses a heuristic solution.

Particle Swarm Optimization (PSO), which was presented in 1995 by Kennedy and Eberhart [26], is based on a metaphor of social interaction such as birds flocking or fish schooling. PSO is a population-based optimization algorithm, which can be implemented and applied easily to solve various function optimization problems, or problems that can be transformed to a function minimization or maximization problem. In the PSO algorithm, each particle in the swarm has a position vector and a velocity vector in the search space at any one time. Each particle remembers its previous own best position vector, called the personal best particle position, and the overall best position among the swarm of particles is called global best particle position. The velocity vector and position vector of the each particle are updated according to a scheme.

Although the advantage of PSO is the simplicity of its theory, its ease of implementation and high-efficiency operation, it has the drawback of premature convergence by becoming trapped into a local optimum.

Many researchers have achieved improved performance by parameter adjusting to make up for the defects of PSO [25, 61, 2, 23]. F. Van den Bergh investigated swarm size in PSO [13]. If the swarm size is large, the initial degree of dispersion grows, and the search number is increased by every iteration in the search space. Meanwhile, the CPU time is also increased, and the search becomes a random search. The optimal swarm size depends on the problem, and it requires a larger swarm size when the search space is more complex.

In the first PSO algorithm, there is the problem that the speed of the particles gradually increases. In particular, when there is a large difference between the personal best position and the global best position, divergence frequently occurs. Eventually the par-

ticles reach a divergence phenomenon when they fly off the outside of the search space. To avoid this problem, a speed limit method was devised to limit the particles' maximum speed [63].

Inertia weight is a parameter that controls the influence of a particle's previous velocity. If the inertia weight is large, the global search capability is enhanced, since the particle's speed will be faster. On the other hand, if the inertia weight is set up as a small value, the local search capability is enhanced by reducing the velocity of the particle. P. N. Suganthan [45] and G. Venter [21] introduced linear and non-linear methods that determine the weight inertia depending on the iteration number.

In PSO, the main parameters are the acceleration coefficients which determine the participation (weighting) of the social component (global best position) and the cognitive component (personal best positions) in the velocity scheme of particles. In general, the acceleration coefficients use a fixed value (constant); some researchers have proposed variation values to improve the performance of the PSO [7, 8, 38].

Thus, the efficiency of problem solution depends on the control of parameters in PSO. However, finding the optimal parameter values via validation such as a cross-validation procedure is difficult, since the optimal parameter is varied for every problem.

In this thesis, the author proposes an improved PSO algorithm that upgrades the update scheme of the particle, rather than improve the parameters. As mentioned earlier, the particles search in a solution space by updating the velocity and position vectors. Each particle updates a new position to share information of global and personal best positions. In this regard, it should be noted that it shares information about the global and personal best positions.

In PSO, the change in the position of each particle reduces as time passes, which is called convergence. At this time, the global best position and personal best position of all particles are themselves. Therefore, when all particles converge to the local optima, they no longer share information.

In this research, to activate the particles' motion, a second best particle position is used as new information that can be shared. The objective of this study is described in the next section.

1.2 Aim of Study

In this thesis, the author will propose new update schemes of particle velocity vectors in PSO to improve their search performance. One of the problems with heuristic algorithms is the premature convergence problem. Premature convergence means the too early convergence of a population of potential solutions, resulting in not the global optimal solution, but a local (sub-) optimal solution. This study in particular forces a second

best particle position to avoid local optimization. The second best particle positions are employed to avoid the PSO algorithm's prematurity problem.

The main objective of this thesis is to achieve exceptional performance when using the proposed PSO algorithm. The effectiveness of the proposed PSO algorithm is estimated through three benchmark problems. The first benchmark problem uses test functions; researcher have commonly used test functions to evaluate their algorithms [41, 32, 31]. The proposed PSO algorithms are compared with different PSO variants involving the original PSO using 11 test functions. Average values are obtained which are objective function values, and the author observed error values between the average function value and the global minimum. Furthermore, the author investigated which the proposed PSO algorithm more rapidly converges to an optimal solution.

The second benchmark problem is a two-dimensional packing problem that maximizes the number of the packed items in a region without items overlapping. The packing problem is a typical NP-hard problem; it is extremely difficult to find the optimal solution in polynomial time. In this study, the author consider a two-dimensional packing problem in which the packing region is arbitrarily polygon-shaped. This packing region is characterized by any researcher that does not apply. Existing packing problems have used irregular, square and circular items to pack into a rectangular or circular region [58, 30, 12, 36, 49]. This study is meaningful for applying the new algorithm in the new packing problem.

Finally, the third benchmark problem is truss optimization. This study focuses on size optimization in which design variables are cross-sectional areas with allowable stress constraints. The proposed PSO is applied to a 10-bar truss structure to compare experimental results. From the results, the author examines the convergence property of the proposed PSO algorithms.

1.3 Composition of Thesis

This thesis is organized as follows. Chapter 1 is an introduction including background and previous relevant algorithms.

In chapter 2, the original Particle Swarm Optimization and different PSO variants are introduced. The particle's position vector and velocity vector, update schemes and flowchart are described.

In chapter 3, the proposed PSO algorithms using second best particle positions are described. For discussing the effect of the parameters on the proposed PSO algorithms, some numerical experiments are descried in this chapter.

In chapter 4, the proposed PSO algorithms are compared with different PSO variants on the test functions. Experimental results, function values and convergence histories are investigated to determine utility of proposed PSO algorithms.

In chapter 5, the analysis of a two-dimensional packing problem by using the proposed PSO is described. Original PSO and proposed PSO are applied for solving the packing problems which have arbitrarily polygon-shaped regions. The objective is that same items are packed in the region without their overlap.

In chapter 6, the truss structure optimization is described as a practical application. An overall weight minimization of a 10-bar truss structure is considered as an example.

Finally, in chapter 7, the conclusion is summarized again.

Chapter 2

Particle Swarm Optimization

2.1 Particle Swarm Optimization(PSO)

2.1.1 Optimization Problem

When the constraint conditions are negligible, the optimization problem can be defined as follows.

The objective function f is minimized so that

$$f(\boldsymbol{x}) \rightarrow \min_{\boldsymbol{x}}$$

The design variables are given as follows.

$$\boldsymbol{x} = \{x_1, x_2, \cdots, x_D\}^T$$

where the parameter D denotes the total number of design variables.

2.1.2 Concept of Original PSO

Particle Swarm Optimization (PSO) was presented in 1955 by Kennedy and Eberhart [26], based on the social behaviour such as bird flocking and fish schooling in nature. In fish schooling, each fish is defined as a particle in the search space, and its objective is to find the feed.

In the PSO algorithm, the particles represent potential solutions of the optimization problem and then, the particles search for the optimal solution of the problem in the feasible search space. A particle i in the swarm has a position vector $\boldsymbol{x}_i(t)$ and a velocity vector $\boldsymbol{v}_i(t)$ in the search space at time t. Each particle has memory and hence, can remember the best position in search space it ever visited. The satisfaction of the particle i for the design objective is estimated by the objective function $f(\boldsymbol{x}_i(t))$.

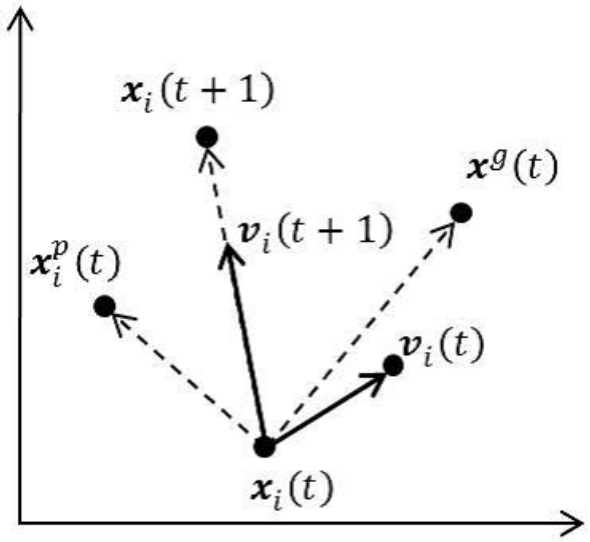


Figure 2.1: Update of position and velocity vectors of ith particle.

The position vector at which each particle takes the best fitness function is known as the personal best particle position vector $\boldsymbol{x}_i^p(t)$ and the overall best out of all particles in the swarm is as global best particle position vector $\boldsymbol{x}^g(t)$.

2.1.3 Update Rules of Position and Velocity Vectors

The position vector $\boldsymbol{x}_i(t)$ and the velocity vector $\boldsymbol{v}_i(t)$ are updated by the global best and the personal best particle position vectors as shown in Fig. 2.1.

The position and the velocity vectors of the particle i ($i = 1, \cdots, N$) are updated according to the following schemes

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2.1}$$

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) \tag{2.2}$$

where w is the inertia weight, c_1 and c_2 are acceleration coefficient, and t is the time-step, called iteration number. The variable r_1 and r_2 are random numbers in the interval $[0, 1]$. The parameter N is the swarm size or the total number of particles in the swarm.

The inertia weight w governs how much percentage of the velocity should be retained from the previous time step to the next time step. Generally the inertia weight is not fixed but varied as the algorithm progresses. The inertia weight w, in this study, is generally updated by self-adapting formula as

$$w = w_{\max} - (w_{\max} - w_{\min}) \times \frac{t}{t_{\max}} \tag{2.3}$$

where the parameter $w_{\max}$ and $w_{\min}$ denote the maximum and minimum inertia weight, respectively. The parameter t and $t_{\max}$ are the iteration step and the maximum iteration steps in the simulation, respectively.

The cognitive parameter c_1 and social parameter c_2 determine the relative pull of $\boldsymbol{x}_i^p$ and $\boldsymbol{x}^g$. According to the recent work done by Clerc [37], the parameters can be taken as $c_1 = c_2 = 1.5$.

The original PSO algorithm is summarized as follows and flowchart is given in Fig. 2.2.

1. **Initialize iteration number:** The iteration number t is initialized as $t = 0$. The maximum iteration step t_{max} and swarm size N are specified.
2. **Initialize particle position and velocity vectors:** The particle position vector $\boldsymbol{x}_i(t)$ and velocity vector $\boldsymbol{v}_i(t)$ are initialized with random numbers.
3. **Evaluate fitness function:** Fitness functions $f(\boldsymbol{x}_i(t))$ is evaluated for all particles.
4. **Check convergence criterion:** If $t = t_{max}$, the process goes to the next stop. Otherwise, the process is terminated.
5. **Update personal best particle position:** The set is defined as follows.

$$\boldsymbol{S}^p(t) = \{S_j^p(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_i^p, \boldsymbol{x}_i(t)\} & (\text{t=0}) \\ \{\boldsymbol{x}_i^p(t-1), \boldsymbol{x}_i(t)\} & (\text{o/w}) \end{array} \right\}$$

   The personal best particle position vectors are updated as follows.

$$\boldsymbol{x}_i^p(t) \leftarrow \arg \min_{S_j^p(t) \in S^p(t)} f(S_j^p(t))$$

6. **Update global best particle position:** The set is defined as follows.

$$\boldsymbol{S}^g(t) = \{S_j^g(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_1^p(t), \cdots, , \boldsymbol{x}_N^p(t)\} & (\text{t=0}) \\ \{\boldsymbol{x}^g(t-1), \boldsymbol{x}_1^p(t), \cdots, , \boldsymbol{x}_N^p(t)\} & (\text{o/w}) \end{array} \right\}$$

   The global best particle position vector is updated as follows.

$$\boldsymbol{x}^g(t) \leftarrow \arg \min_{S_j^g \in S^g} f(S_j^g)$$

7. **Set the particle number:** Particle number i is initialized as $i = 1$.
8. **Check swarm size:** If $i \leq N$, process goes to the next step. Otherwise, process goes to the Step 11.
9. **Update particle position and velocity vectors:** The position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are calculated by Eqs. (2.1) and (2.2), respectively.
10. **Update the particle:** The particle number is updated by $i = i + 1$, and process goes to Step 8.
11. **Update iteration number:** The iteration number is updated so that $t = t + 1$, and process goes to Step 3.

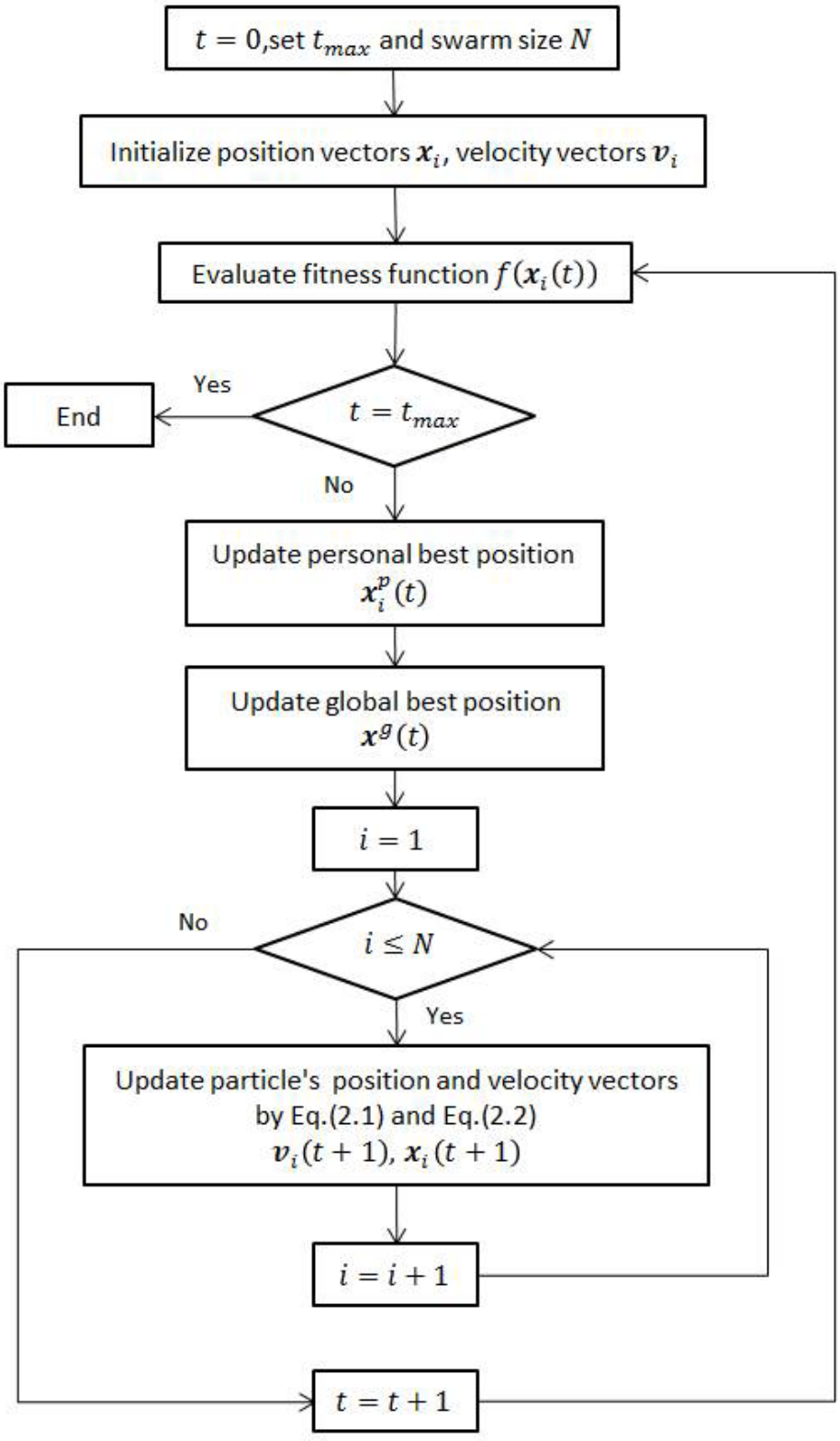


Figure 2.2: Flowchart of original PSO algorithm.

2.2 PSO Variants

Several PSO variants have been proposed by some researchers. Their update rules of position and velocity vectors can be summarized as follows.

2.2.1 Basic PSO

In the first idea, the position and the velocity vectors of the particle i $(i = 1, \cdots, N)$ are updated according to the following schemes.

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2.4}$$

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) \tag{2.5}$$

where c_1 and c_2 are acceleration coefficients, and t is the time-step, called iteration number. The variable r_1 and r_2 are random numbers in the interval $[0, 1]$. The parameter N is the swarm size or the total number of particles in the swarm.

At the right hand side of Eq. (2.5), the second and the third terms are named as the cognitive and the social components, respectively. The update rule of the velocity vector is simple because it is defined as the simple summation of the velocity term at the present time-step $\boldsymbol{v}_i(t)$, the cognitive component and social components. Therefore, I will name it as the basic PSO.

2.2.2 PSO With Inertia Weight (PSO-w)

Shi and Eberhart [62] proposed inertia weight w(PSO-w) into the original PSO. The inertia weight is used to balance the global and local search abilites.

The update rule of the position vector of the particle i $(i = 1, \cdots, N)$ is identical to that of the Basic PSO; Eq. (2.4). The velocity vector of the particle i $(i = 1, \cdots, N)$ is updated according to the following rule.

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) \tag{2.6}$$

where w is the inertia weight, c_1 and c_2 are acceleration coefficients, and t is the time-step, called iteration number. The variable r_1 and r_2 are random numbers in the interval $[0, 1]$. The parameter N is the swarm size or the total number of particles in the swarm.

2.2.3 PSO With Constriction Factor (PSO-cf)

Clerc and Kennedy [39] introduced PSO with constriction factor (PSO-cf).

The update rule of the position vector of the particle i $(i = 1, \cdots, N)$ is identical to that of the Basic PSO; Eq. (2.4). The velocity vector of the particle i $(i = 1, \cdots, N)$ is

updated according to the following rule.

$$\boldsymbol{v}_i(t+1) = K[\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t))] \tag{2.7}$$

where $K \in [0, 1]$ is constriction factor for improvement of convergence velocity.

2.2.4 Local PSO-w And Local PSO-cf

Local version of PSO with inertia weight (Local PSO-w) and constriction factor (Local PSO-cf) were introduced by Kennedy and Mendest [27]. They suggested that a small neighborhood might be more suitable to complex problems while a larger neighborhood might perform better on simple problems.

In the Local PSO-w, the update rule of the position vector of the particle i ($i = 1, \cdots, N$) is identical to that of the Basic PSO; Eq. (2.4). The velocity vector of the particle i ($i = 1, \cdots, N$) is updated according to the following rule.

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}_i^g(t) - \boldsymbol{x}_i(t)) \tag{2.8}$$

where $\boldsymbol{x}_i^g(t)$ is best particle position vector in the neighborhood.

In the Local PSO-cf, the velocity vector of the particle i ($i = 1, \cdots, N$) is updated according to the following rule.

$$\boldsymbol{v}_i(t+1) = K[\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}_i^g(t) - \boldsymbol{x}_i(t))] \tag{2.9}$$

2.2.5 Union of Global And Local PSOs (UPSO)

Parsopoulos and Vrahatis [17] focused on union of global and local PSOs, which is named as a unified particle swarm optimization(UPSO). In UPSO, the update rule of the position vector of the particle i ($i = 1, \cdots, N$) is identical to that of the Basic PSO; Eq. (2.4). The velocity vector of the particle i ($i = 1, \cdots, N$) is updated according to the following rule.

$$\boldsymbol{v}_i(t+1) = u\boldsymbol{G}_i(t+1) + (1-u)\boldsymbol{L}_i(t+1)) \tag{2.10}$$

where $\boldsymbol{G}_i(t+1)$ and $\boldsymbol{L}_i(t+1)$ are velocity vector of particle in Global PSO-cf and in Local PSO-cf, respectively. The parameter u $\in$ [0,1] is a parameter called the unification factor, which affects the global and local components.

According to the algorithms of Global PSO-cf and Local PSO-cf, the particle position vectors $\boldsymbol{G}_i(t+1)$ and $\boldsymbol{L}_i(t+1)$ are calculated from the following equations, respectively.

$$\boldsymbol{G}_i(t+1) = K[\boldsymbol{G}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t))] \tag{2.11}$$

$$\boldsymbol{L}_i(t+1) = K[\boldsymbol{L}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{x}_i^g(t) - \boldsymbol{x}_i(t))] \tag{2.12}$$

2.2.6 Comprehensive Learning PSO (CLPSO)

Liang et al. [29, 52] proposed a comprehensive learning PSO (CLPSO), which uses a novel learning strategy whereby all other particle's historical best information is used to update a particle's velocity.

In CLPSO, the update rules of the position and the velocity vectors of the particle i ($i = 1, \cdots, N$) are defined as follows.

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2.13}$$

$$\boldsymbol{v}_i(t+1) = \{v_{i1}(t+1), v_{i2}(t+1), \cdots, v_{iD}(t+1)\} \tag{2.14}$$

where $v_{id}(t+1)$ denotes the $d-$th component of the vector $\boldsymbol{v}_i(t+1)$.

The update rule of the velocity vector of the CLPSO is very different from the others because the velocity vector is updated by each component. The velocity vector component $v_{id}(t+1)$ is updated by the following rule:

$$v_{id}(t+1) = wv_{id}(t) + cr_{id}(x^p_{Id}(t) - x_{id}(t)) \tag{2.15}$$

where $d \in \{1, 2, 3, ..., D\}$ and $x^p_{Id}(t)$ denotes the personal best particle position for updating dth dimension of particle i.

The parameter $x^p_{Id}(t)$ is selected from the two personal best particles. One is the personal best particle of the particle i and the other is that of the other particle than the particle i, which is referred to as the particle a. The selection of the personal best particle depends on the learning probability P_c.

1. The learning probability P_c is specified.
2. A random number P is generated.
3. If $P > P_c$, the personal best particle of the particle i is selected; $x^p_{Id}(t) = x^p_{id}(t)$. Otherwise, another personal best particle is selected; $x^p_{Id}(t) = x^p_{ad}(t)$.

Another personal best particle is chosen according to the following process.

1. Two particles are chosen randomly out of the population, excluding the particle whose velocity is updated.
2. The fitness of the personal best particles of these two particles are compared and then, better one is selected.

Chapter 3

Proposed PSO algorithm

3.1 Introduction

The original PSO has no handling mechanism for avoiding the local optimization except for the use of personal best position. When particles are converged on the local optimization as shown in Fig. 3.1, however, particles are more difficult to move global optima since personal best position equal to local optimized global best position. Thus, each particle requires the different information of objective position to escape from local optima. In this research, proposed PSO algorithms used PSO with second global best particle position (SG-PSO) and PSO with second personal best particle position (SP-PSO) as different information of objective positions.

3.2 Proposed PSO Algorithms

3.2.1 PSO with Second Global best Particle (SG-PSO)

In PSO with Second Global best Particle (SG-PSO), each particle can remember the global best particle position vector $\boldsymbol{x}^g(t)$, the personal best particle position vector $\boldsymbol{x}^p(t)$ and the second global best particle position vector $\boldsymbol{x}^{g2}(t)$. The use of $\boldsymbol{x}^{g2}(t)$ can reduce the chance of PSO convergence to local optimal solution for diversity movement of particles. When the second global best particle position vector $\boldsymbol{x}^{g2}(t)$ is employed, the update scheme of the particle velocity vector is given as follows

$$\begin{aligned} \boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) \\ &\quad + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) + c_3 r_3(\boldsymbol{x}^{g2}(t) - \boldsymbol{x}_i(t)) \end{aligned} \tag{3.1}$$

where w is the inertia weight, c_1, c_2 and c_3 are acceleration coefficients, and t is the iteration time. Besides, r_1, r_2 and r_3 are random numbers distributed in the interval $[0, 1]$. The parameter c_1 and c_2 are taken as the same values in the original PSO; $c_1 = c_2 = 1.5$.

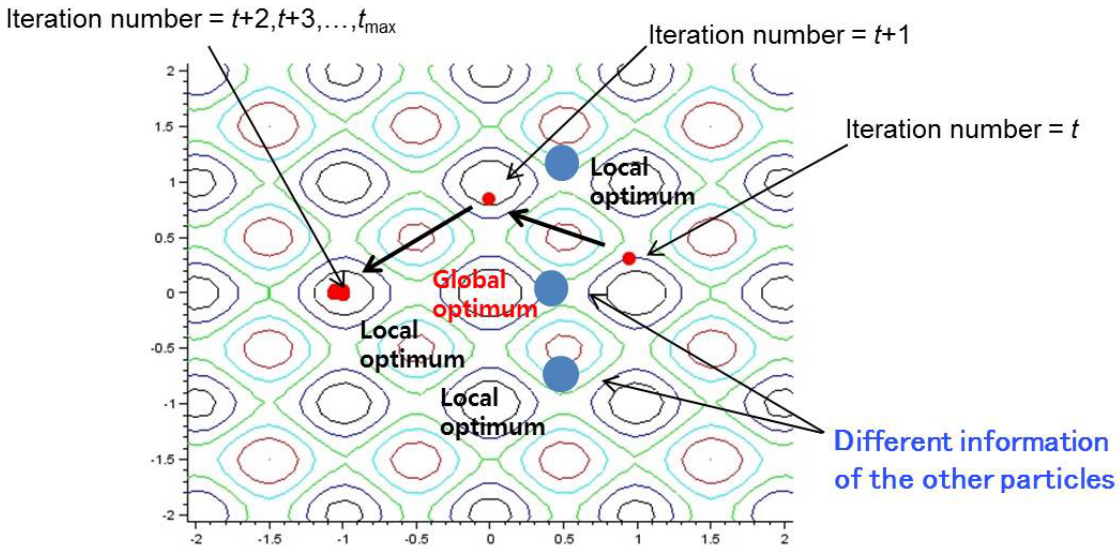


Figure 3.1: Search process to find global best position.

Effect of the parameter c_3 on the search performance is discussed in the Section 3.4. Unfortunately, the second global best position is worse than the first global best position. In other words, if only new update scheme Eq. (3.1) is used, search performance of the PSO algorithm becomes worse. Therefore, in the SG-PSO, the original and the new update schemes are randomly employed for updating particle position vectors.

The update rule Eq. (3.1) has been already presented in the reference [51]. The numerical discussions and the applications were, however, not described in the reference [51]. Therefore, in this study, it is discussed in numerical examples.

The SG-PSO algorithm is summarized as follows and flowchart is given in Fig. 3.2.

1. **Initialize iteration number:** The iteration number t is initialized as $t = 0$. The maximum iteration step t_{max} and swarm size N are specified.
2. **Initialize particle position and velocity vectors:** The particle position vector $\boldsymbol{x}_i(t)$ and velocity vector $\boldsymbol{v}_i(t)$ are initialized with random numbers.
3. **Evaluate fitness function:** Fitness functions $f(\boldsymbol{x}_i(t))$ is evaluated for all particles.
4. **Check convergence criterion:** If $t = t_{max}$, the process goes to the next stop. Otherwise, the process is terminated.
5. **Update personal best particle position:** The set is defined as follows.

$$\boldsymbol{S}^p(t) = \{S_j^p(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_i^p, \boldsymbol{x}_i(t)\} & (\text{t=0}) \\ \{\boldsymbol{x}_i^p(t-1), \boldsymbol{x}_i(t)\} & (\text{o/w}) \end{array} \right\}$$

The personal best particle position vectors are updated as follows.

$$\boldsymbol{x}_i^p(t) \leftarrow \underset{S_j^p(t) \in S^p(t)}{\operatorname{argmin}} f(S_j^p(t))$$

6. **Update global best particle position:** The set is defined as follows.

$$\boldsymbol{S}^g(t) = \{S_j^g(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_1^p(t), \cdots, , \boldsymbol{x}_N^p(t)\} & \text{(t=0)} \\ \{\boldsymbol{x}^g(t-1), \boldsymbol{x}_1^p(t), \cdots, , \boldsymbol{x}_N^p(t)\} & \text{(o/w)} \end{array} \right\}$$

The global best particle position vector is updated as follows.

$$\boldsymbol{x}^g(t) \leftarrow \underset{S_j^g \in S^g}{\operatorname{argmin}} f(S_j^g)$$

7. **Update second global best particle position:** The set is defined as follows.

$$\boldsymbol{S}^{g2}(t) = \{S_k^{g2}(t)\} = \{S_j^g(t) | S_j^g(t) \notin \boldsymbol{x}^g(t)\}$$

The second global best particle position vector is updated as follows.

$$\boldsymbol{x}^{g2}(t) \leftarrow \underset{S_k^{g2}(t) \in S^{g2}(t)}{\arg\min} f(S_k^{g2}(t))$$

8. **Set the particle number:** : The particle number i is initialized as $i = 1$.
9. **Check swarm size:** If $i \leq N$, process goes to the next step. Otherwise, process goes to the Step 13
10. **Check random number r:** A random number r is generated in the range $[0, 1]$. If $r \leq 0.5$, process goes to the step $11(a)$, Otherwise, process goes to the Step $11(b)$
11. **Update particle position and velocity vectors:**
    (a) The position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are calculated by Eqs. (2.1) and (3.1), respectively.
    (b) The position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are calculated by Eqs. (2.1) and (2.2), respectively.
12. **Update the particle:** The particle number is updated by $i = i + 1$, and process goes to Step 9.
13. **Update iteration number:** The iteration number is updated so that $t = t + 1$, and process goes to Step 3.

.

3.2.2 PSO with Second Personal best Particle (SP-PSO)

The SG-PSO uses the second global best particle position vector $\boldsymbol{x}^{g2}$ for avoiding the local optimization. On the other hand, PSO with second personal best particle (SP-PSO) uses

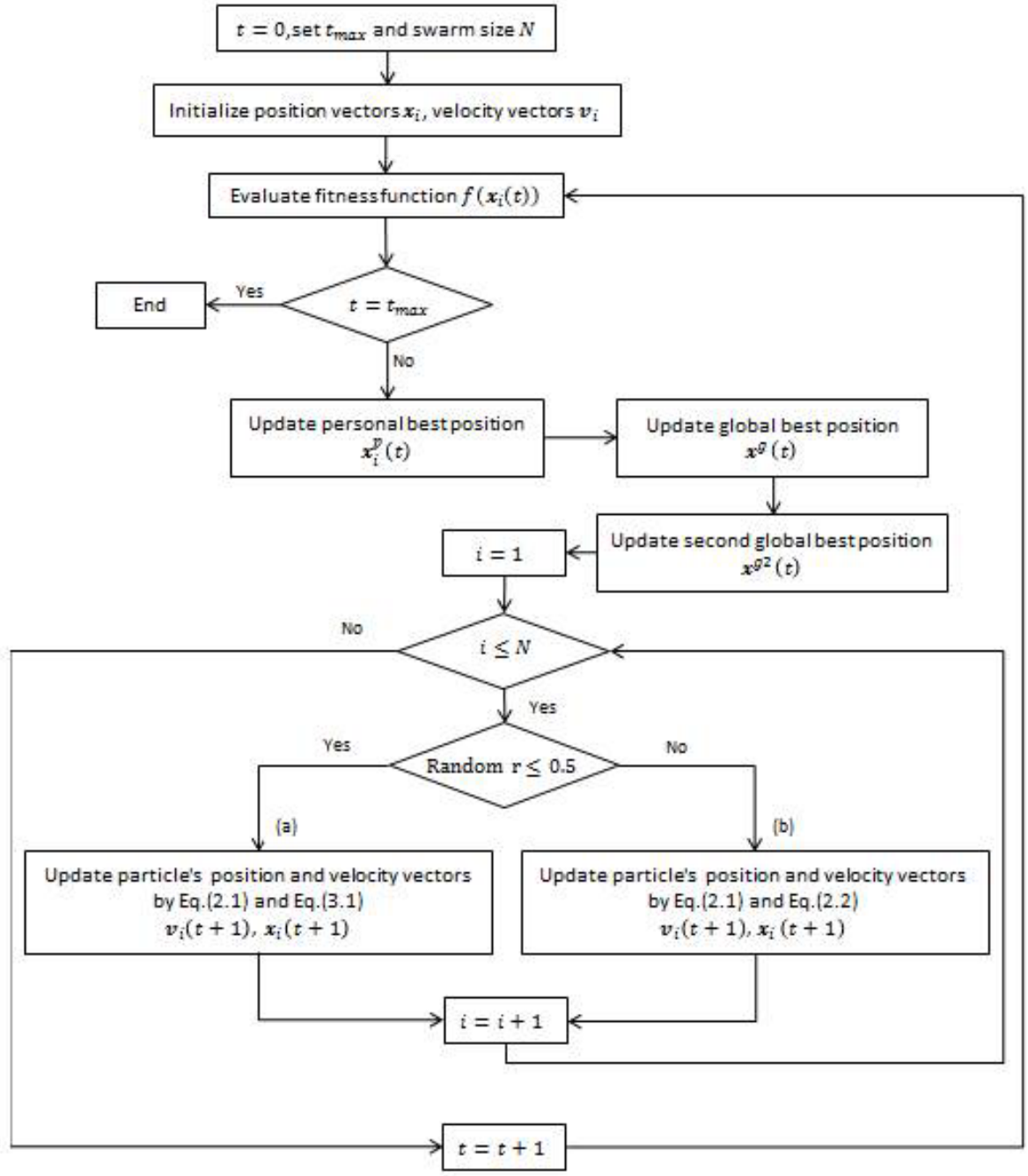


Figure 3.2: Flowchart of SG-PSO algorithm.

the second personal best particle position vector $\boldsymbol{x}_i^{p2}$ instead of the second global best particle position vector $\boldsymbol{x}^{g2}$.

In PSO with Second Personal best Particle (SP-PSO), the second global best particle position vector $\boldsymbol{x}^{g2}$ gives the similar effect on all particles because the second global best particle is only one in the swarm. In the SP-PSO, the second personal best particle position vector $\boldsymbol{x}_i^{p2}$ gives the different effect on each particle because the second personal best particle is generally different for each particle. Therefore, particles in the SP-PSO tend to search wider region than those in the SP-PSO.

In the SP-PSO, each particle can remember the global best particle position vector $\boldsymbol{x}^g$, the personal best particle position vector $\boldsymbol{x}_i^p$ and the second personal best particle position vector $\boldsymbol{x}_i^{p2}$. When the second personal best particle position vector $\boldsymbol{x}_i^{p2}(t)$ is employed, the update scheme of the particle velocity vector is given as follows and flowchart is given in Fig. 3.3.

$$\begin{aligned} \boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) \\ &\quad + c_2 r_2(\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) + c_4 r_4(\boldsymbol{x}^{p2}(t) - \boldsymbol{x}_i(t)) \end{aligned} \tag{3.2}$$

where w is the inertia weight, c_1, c_2 and c_4 are acceleration coefficients, and t is the iteration time. Besides, r_1, r_2 and r_4 are random numbers distributed in the interval $[0, 1]$.

1. **Initialize iteration number:** The iteration number t is initialized as $t = 0$. The maximum iteration step t_{max} and swarm size N are specified.
2. **Initialize particle position and velocity vectors:** The particle position vector $\boldsymbol{x}_i(t)$ and velocity vector $\boldsymbol{v}_i(t)$ are initialized with random numbers.
3. **Evaluate fitness function:** Fitness functions $f(\boldsymbol{x}_i(t))$ is evaluated for all particles.
4. **Check convergence criterion:** If $t = t_{max}$, the process goes to the next stop. Otherwise, the process is terminated.
5. **Update personal best particle position:** The set is defined as follows.

$$\boldsymbol{S}^p(t) = \{S_j^p(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_i^p, \boldsymbol{x}_i(t)\} & \text{(t=0)} \\ \{\boldsymbol{x}_i^p(t-1), \boldsymbol{x}_i(t)\} & \text{(o/w)} \end{array} \right\}$$

   The personal best particle position vectors are updated as follows.

$$\boldsymbol{x}_i^p(t) \leftarrow \underset{S_j^p(t) \in S^p(t)}{\operatorname{argmin}} f(S_j^p(t))$$

6. **Update global best particle position:** The set is defined as follows.

$$\boldsymbol{S}^g(t) = \{S_j^g(t)\} = \left\{ \begin{array}{ll} \{\boldsymbol{x}_1^p(t), \cdots, , \boldsymbol{x}_N^p(t)\} & \text{(t=0)} \\ \{\boldsymbol{x}^g(t-1), \boldsymbol{x}_1^p(t), \cdots, \boldsymbol{x}_N^p(t)\} & \text{(o/w)} \end{array} \right\}$$

The global best particle position vector is updated as follows.

$$\boldsymbol{x}^g(t) \leftarrow \underset{S_j^g \in S^g}{\operatorname{argmin}} f(S_j^g)$$

7. **Update second personal best particle position:** The set is defined as follows.

$$\boldsymbol{S}^{p2}(t) = \{S_k^{p2}(t)\} = \{S_j^g(t) | S_j^g(t) \notin \boldsymbol{x}_i^p(t)\}$$

The second global best particle position vector is updated as follows.

$$\boldsymbol{x}_i^{p2}(t) \leftarrow \underset{S_k^{p2}(t) \in S^{p2}(t)}{\arg\min} f(S_k^{p2}(t))$$

8. **Set the particle number:** The particle number i is initialized as $i = 1$.
9. **Check swarm size:** If $i \leq N$, process goes to the next step. Otherwise, process goes to the Step 13
10. **Check random number r:** A random number r is generated in the range $[0, 1]$. If $r \leq 0.5$, process goes to the step $11(a)$, Otherwise, process goes to the Step $11(b)$
11. **Update particle position and velocity vectors:**
    (a) The position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are calculated by Eqs. (2.1) and (3.2), respectively.
    (b) The position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are calculated by Eqs. (2.1) and (2.2), respectively.
12. **Update the particle:** The particle number is updated by $i = i + 1$, and process goes to Step 9.
13. **Update iteration number:** The iteration number is updated so that $t = t + 1$, and process goes to Step 3.

3.2.3 Trajectory of Particles

In order to compare the search process of the SG-PSO and SP-PSO, the trajectory of the global best particle positions is illustrated. Rastrigin function with $n = 2$ is taken as an example and the swarm size is 10. The trajectory of the global best particles in the original PSO, SG-PSO and SP-PSO are illustrated in Figs. 3.4, 3.5 and 3.6, respectively. The red points denote the global best particle position. In Fig. 3.5, the trajectory of the second global best particle is also shown with blue points. Fig. 3.4 shows that, in the original PSO, the global best particle is attracted to the local optimum at 4th iteration and therefore, the global optimum cannot be found. Fig. 3.5 shows that the global best particle can avoid local optimum at 1st, 2nd, 5th and 6th iterations and therefore, the global optimum can be found rapidly. Fig. 3.6 shows that the SP-PSO can also avoid to converge to the local optimum. The global best particle, however, moves to find global optima extensively. Consequently, it is considered that SP-PSO is suitable to local search than SG-PSO.

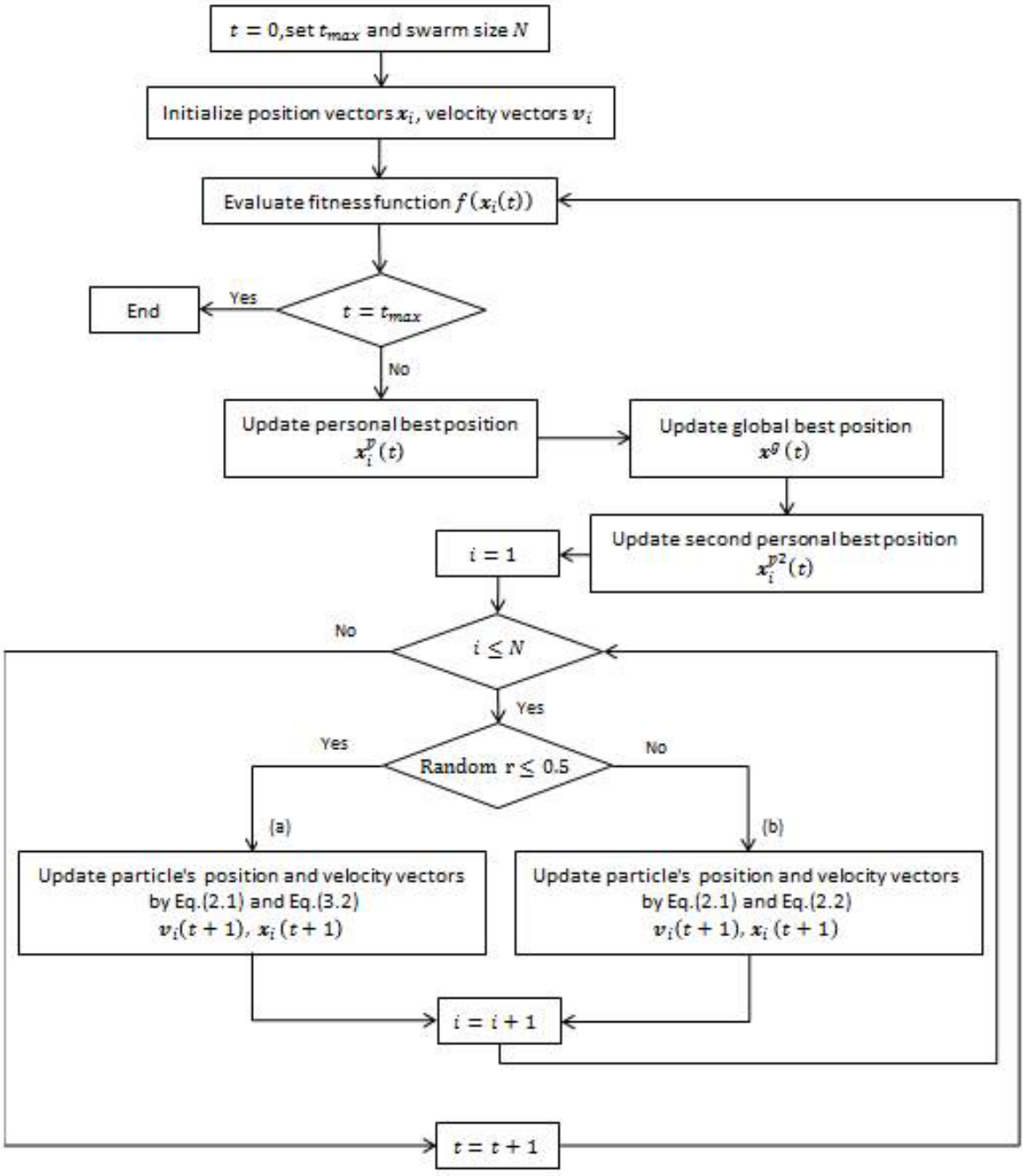
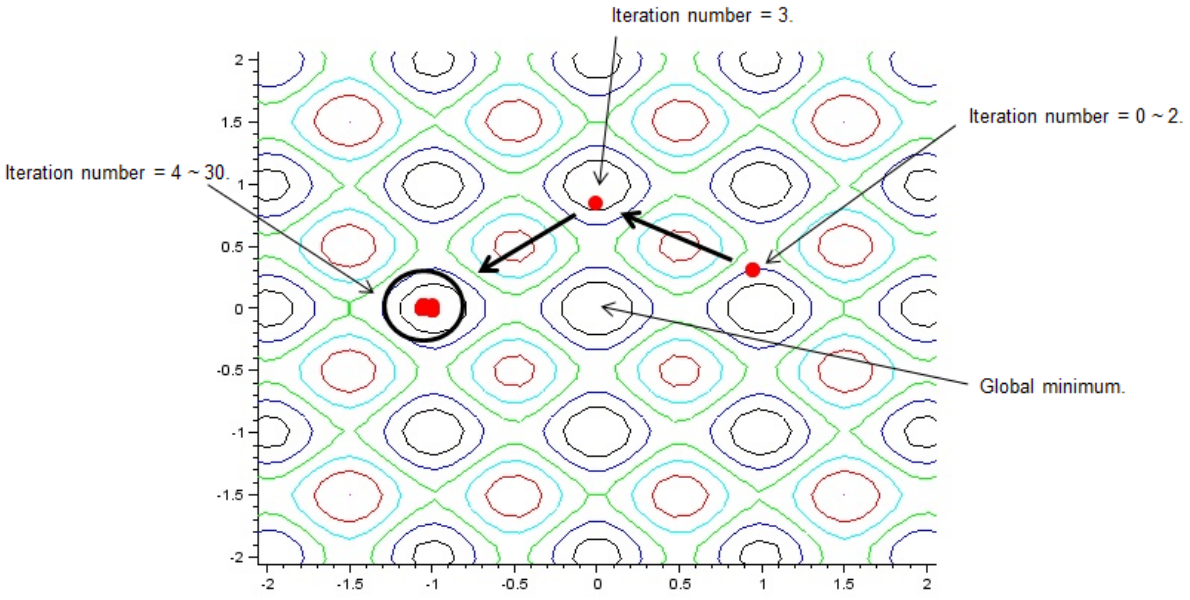
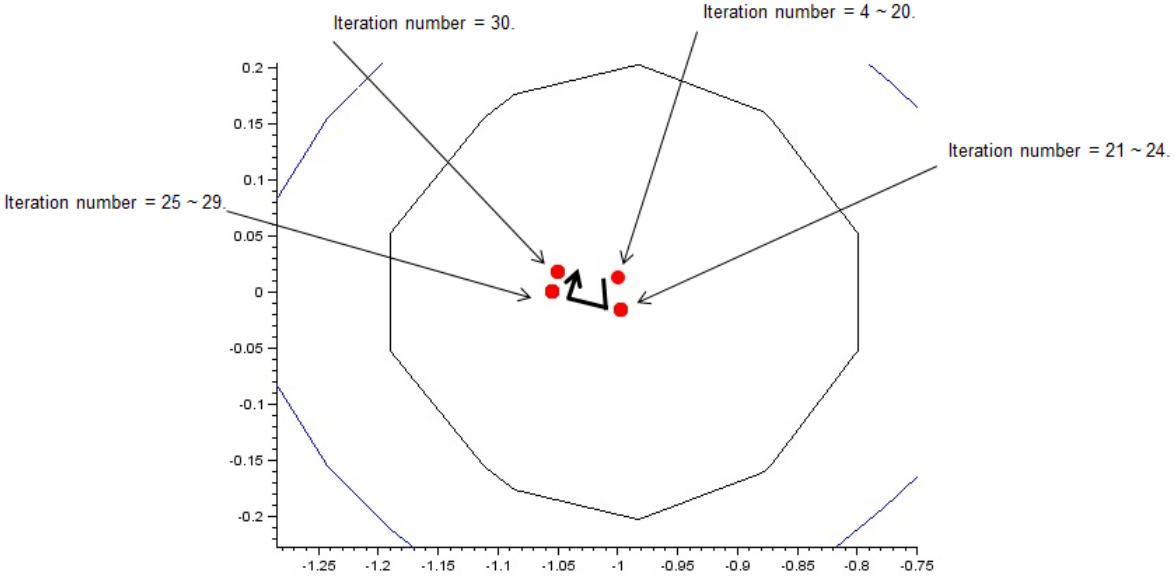


Figure 3.3: Flowchart of SP-PSO algorithm.



(a) Whole view



(b) Expanded view from 4 - 30 iterations.

Figure 3.4: Trajectory of global best particles in original PSO.

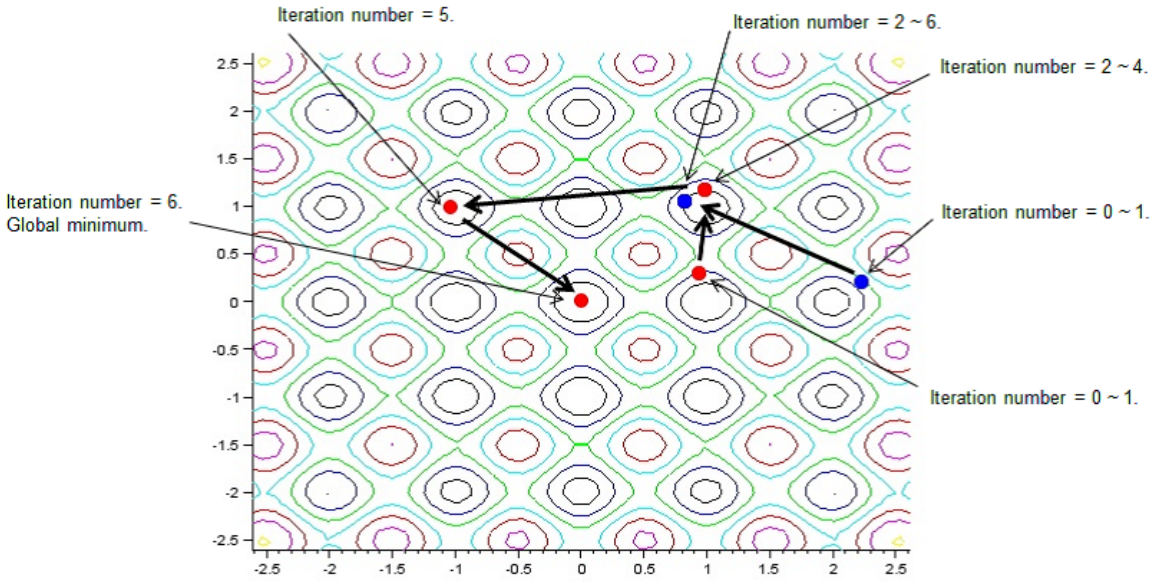


Figure 3.5: Trajectory of first and second global best particles in SG-PSO.

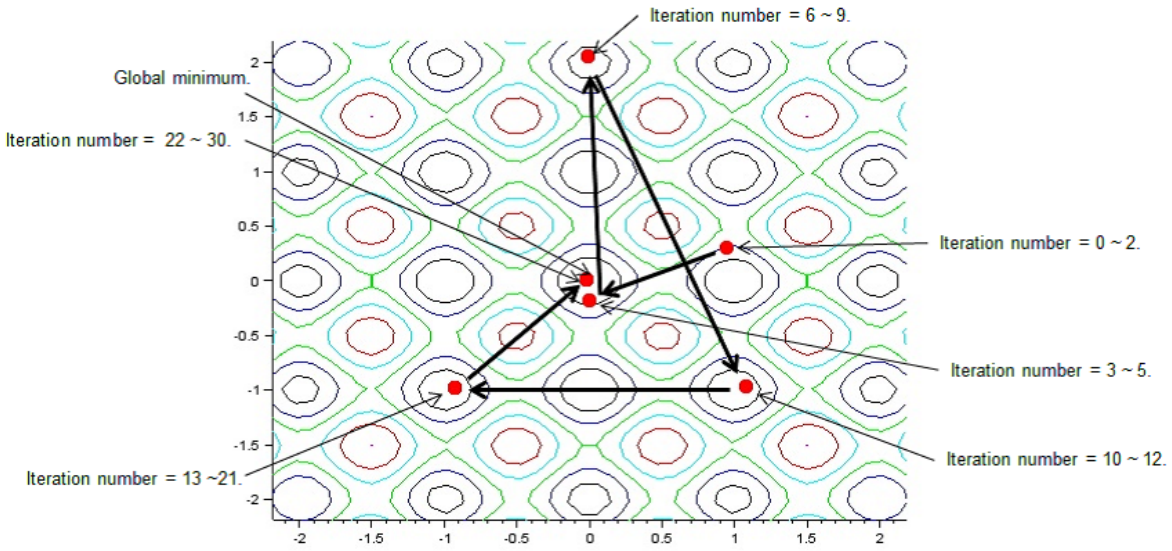


Figure 3.6: Trajectory of global best particles in SP-PSO.

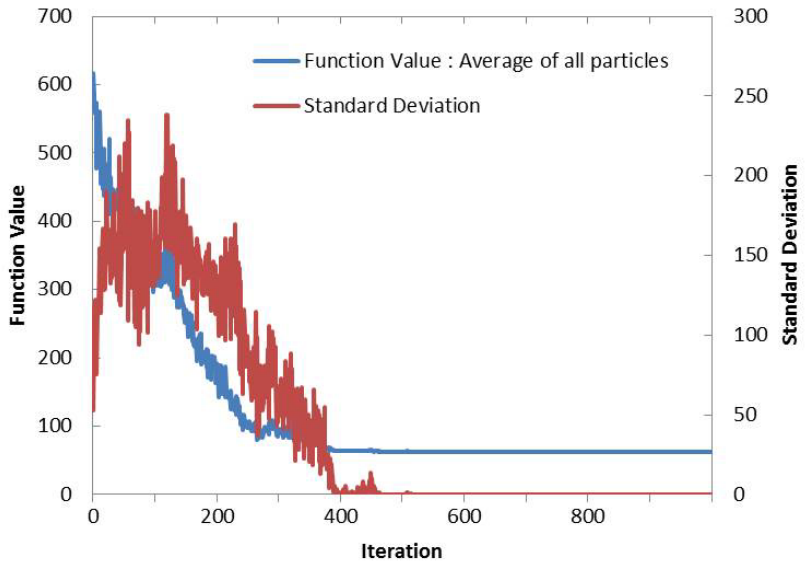


Figure 3.7: Convergence history of all particles (Original PSO).

3.2.4 Convergence History of All Particles

The two-dimensional Rastrigin function is taken as the test function and then the convergence histories of the average function value of all particles and its standard deviation are compared. At time t, the average function value of all particles as below

$$\text{Average function value of all particles} \quad = \quad \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x_i}(t)) \tag{3.3}$$

The convergence histories of the SG-PSO, SP-PSO and original PSO, are shown in Figs. 3.7, 3.8 and 3.9, respectively. The figures are plotted with the iteration as the horizontal axis, and the function value and the standard deviation of the function value as the vertical axes, respectively.

Fig. 3.7 shows that, in the original PSO, standard deviation converges to zero. Figs. 3.8 and 3.9 show that, in the SG-PSO and SP-PSO, the standard deviation does not converge to zero during the simulation process, and the average function value does not converge unlike the original PSO.

These results show that, in the SG-PSO and SP-PSO, the use of the second global or the second personal position can avoid the rapid gathering of all particles to local optimum which has been found ever. Therefore, the SG-PSO and SP-PSO can find better solution than the original PSO.

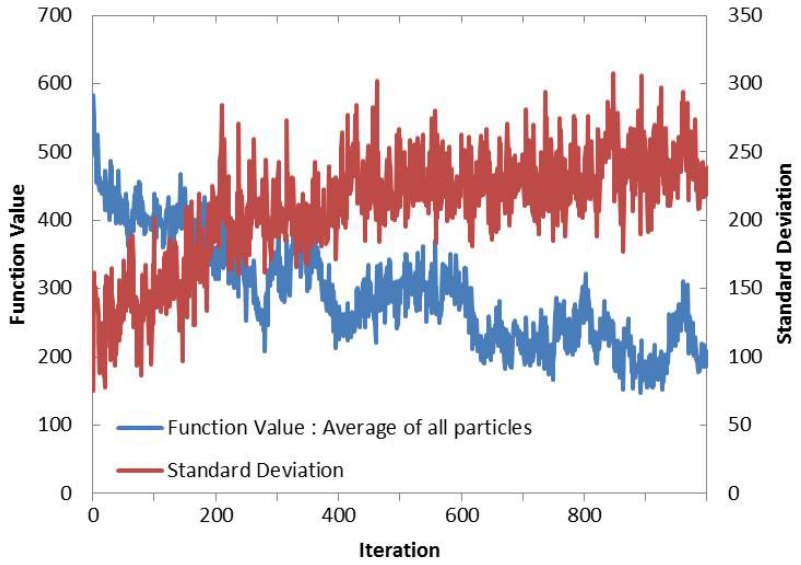


Figure 3.8: Convergence history of all particles (SG-PSO).

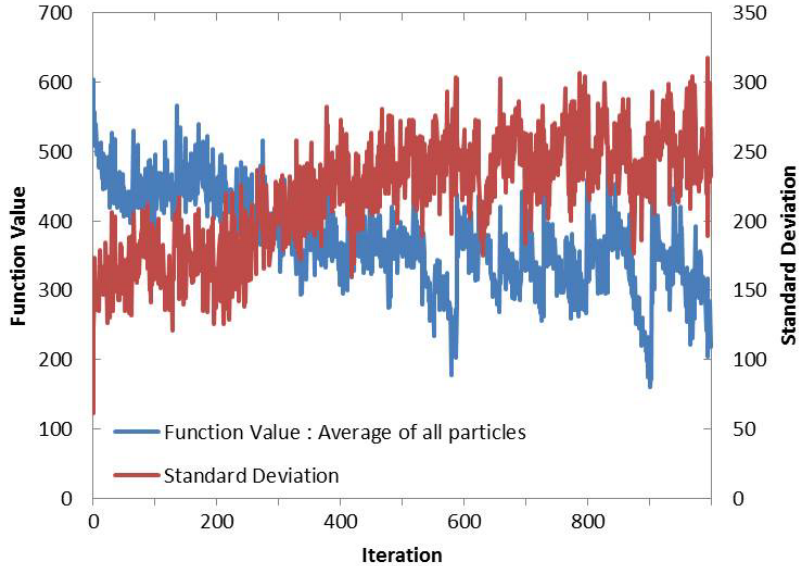


Figure 3.9: Convergence history of all particles (SP-PSO).

Table 3.1: Benchmark functions for c_3 and c_4: Trelea(2003).

Name	Function	Dimension
Sphere	$f_{Sp}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	30
Rosenbrock	$f_{Ro}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	30
Rastrigin	$f_{Ra}(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30
Griewank	$f_{Gr}(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30
Schaffers f6	$f_{Sc}(\boldsymbol{x}) = 0.5 + \frac{\left(\sin\sqrt{(x_1^2 + x_2^2)}\right)^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	2

Name	Search range$[x_{min}, x_{max}]$	Threshold value
Sphere	$[-100, 100]^n$	0.01
Rosenbrock	$[-30, 30]^n$	100
Rastrigin	$[-5.12, 5.12]^n$	100
Griewank	$[-600, 600]^n$	0.1
Schaffers f6	$[-100, 100]^n$	0.00001

3.3 Effect of Parameters on SG-PSO and SP-PSO

3.3.1 Benchmark Functions

For discussing the effect of the parameters c_3 and c_4 on the SG-PSO and SP-PSO, some numerical experiments are described here. Trelea [9, 50] discussed convergence speed of the deterministic PSO algorithm using the five benchmark functions. In this section, Trelea's functions are used to compare the effects of c_3 and c_4. Trelea's functions are summarized in Table 3.1. Five test functions in two design variables are shown in Figs. 3.10 to 3.14.

The function dimension is $n = 2$ for Schaffer's f6 function or $n = 30$ for the other functions. The global minimum of all functions is 0. Swarm size and maximum iteration number are 30 and 10000, respectively. According to the work done by Clerc [37], the acceleration coefficients c_1 and c_2 are specified as $c_1 = c_2 = 1.5$. The parameters of the inertia weight are $w_{\max} = 0.9$ and $w_{\min} = 0.4$. Simulations are performed 20 times from different initial populations. The search performance is compared in "Estimation" value,

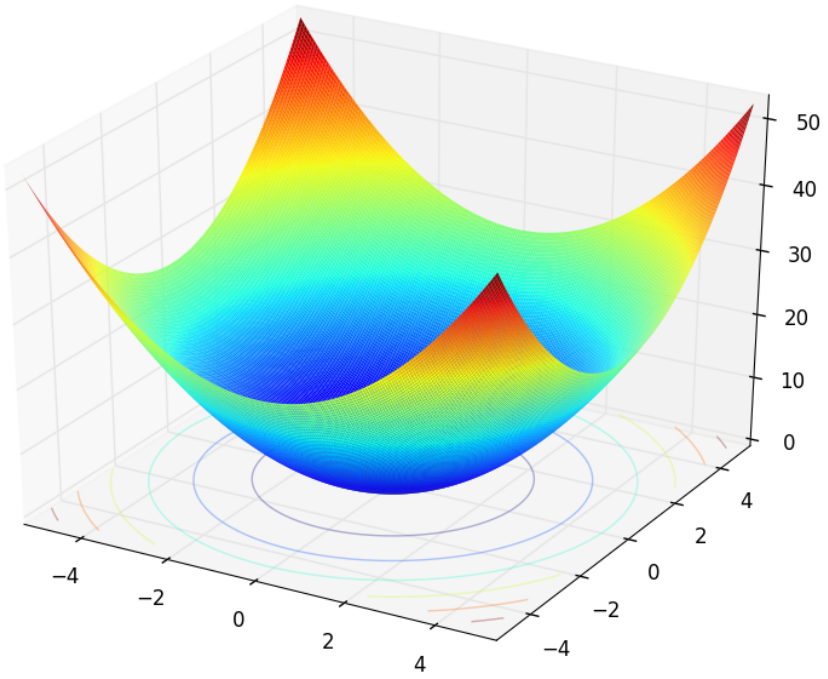


Figure 3.10: 3D graph of two-dimensional Sphere function.

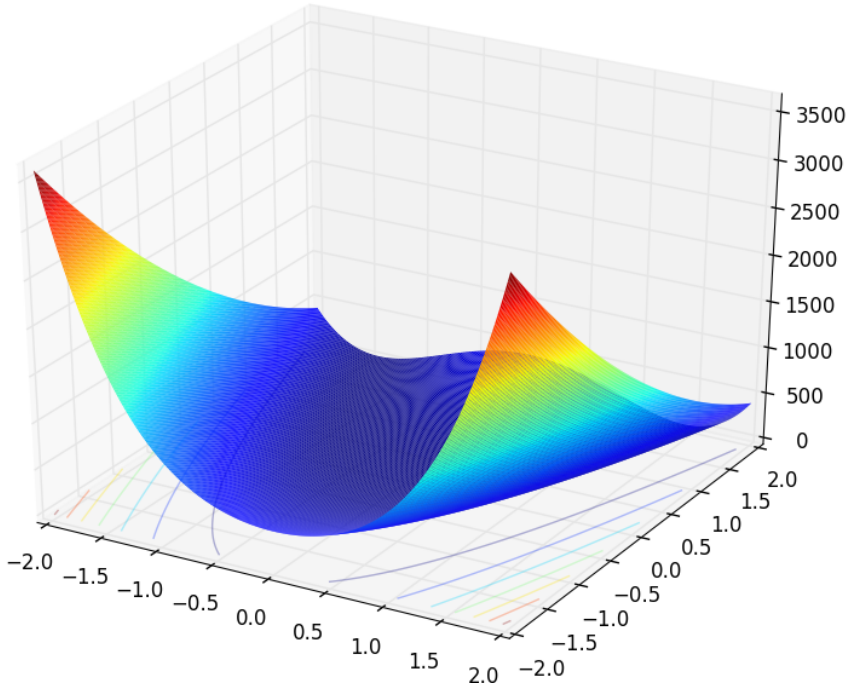


Figure 3.11: 3D graph of two-dimensional Rosenbrock function.

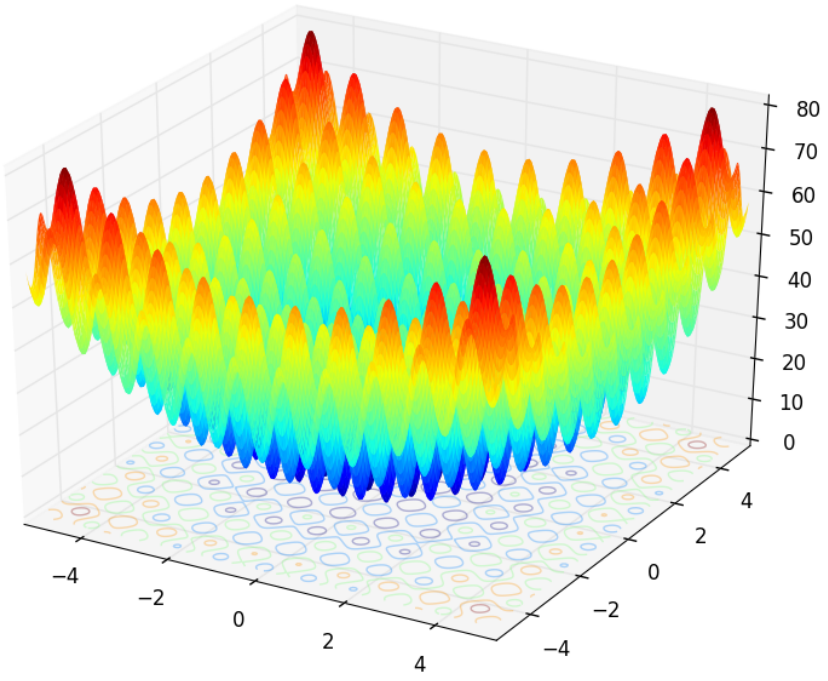


Figure 3.12: 3D graph of two-dimensional Rastrigin function.

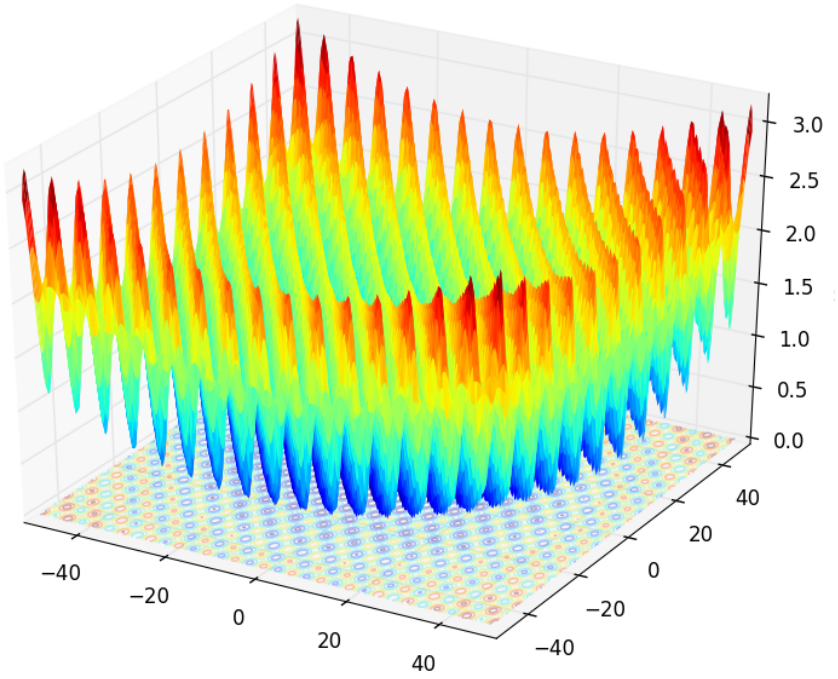


Figure 3.13: 3D graph of two-dimensional Griewank function.

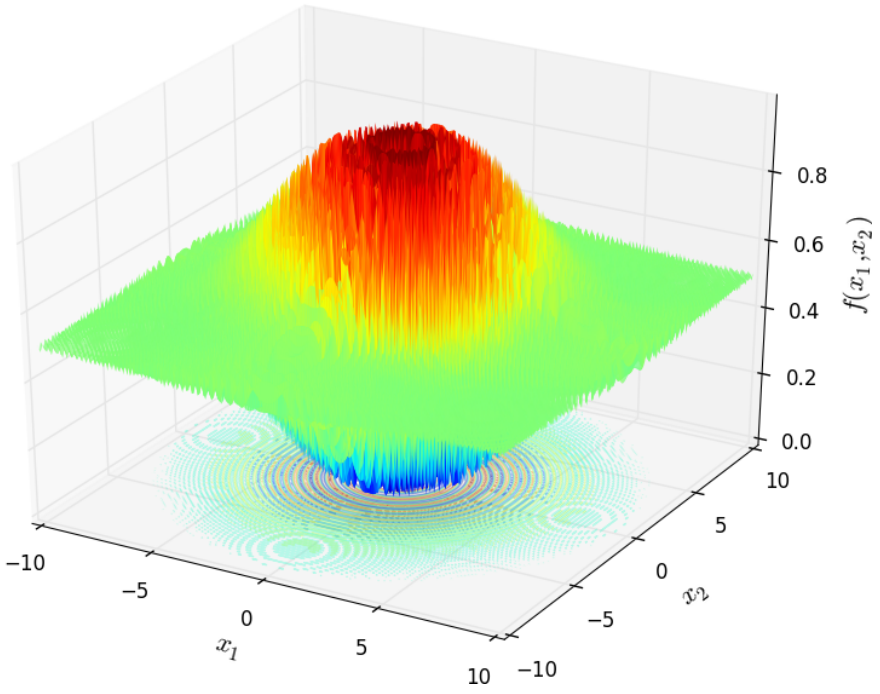


Figure 3.14: 3D graph of two-dimensional Schaffer's f6 function function.

which is defined as the quotient of the average search time and the success rate as follows.

$$\text{Estimation} = \frac{\text{Average search time}}{\text{Success rate}}$$

The value of "Success rate" in the above equation denotes the percentage of the simulations at which the global optimal solution could be found. It is concluded that the global minimum of the function is found when the function value is smaller than the threshold value. The value of "Average search time" in the above denotes the average iteration number in case where the global optimal solution is found. The search performance is better as the value of "Estimation value" in the above is smaller.

3.3.2 Comparison of Experimental Results

For different c_3 values, results are shown in Table 3.2. The smallest estimation value of each function is underlined. The smallest estimation values are observed at $c_3 = 5$ for Sphere, Rosenbrock and Schaffer's f6 functions, $c_3 = 2.5$ for Rastrigin function and $c_3 = 5.5$ for Griewank function. The second smallest values for Rastrigin and Griewank functions are observed at $c_3 = 5$. The smallest and the second smallest estimation values, for Rastrigin function, are 109 at $c_3 = 2.5$ and 112 at $c_3 = 5.0$, respectively. In case of Griwank fuction, the smallest and the second smallest estimation values are 209 at $c_3 = 5.5$ and 223 at $c_3 = 5.0$. Since their difference is small, it is concluded that $c_3 = 5.0$

is suitable for all functions.

For the different values of c_4, the results are shown in Table 3.3. The smallest estimation value for each function is underlined in the table. The smallest estimation values are observed at $c_4 = 5.5$ for Sphere, Rosenbrock, Rastrigin, and Griewank functions and at $c_4 = 5$ for Schaffer's f6 function. The second smallest estimation value for Schaffer's f6 function is observed at $c_4 = 5.5$. The smallest and the second smallest estimation values, for Schaffer's f6 function, are 2.8 at $c_3 = 5.0$ and 3.1 at $c_3 = 5.5$, respectively. Since there are not much difference in values, it is concluded that $c_4 = 5.5$ is suitable for all functions.

3.4 Conclusion

In this chapter, the SG-PSO and SP-PSO algorithms were proposed for improving the search performance. In the original PSO, the particle positions are updated from the personal best and the global best positions which particles have ever found. The proposed algorithms focus on the use of the second global best and the second personal best particle positions in order to avoid to converge to the local optimum. In the SG-PSO and SP-PSO, the second global best and the second personal best positions are randomly used with the update rules of the original PSO for updating the particle positions.

The effect of the parameters c_3 and c_4 on the SG-PSO and SP-PSO was discussed using five test functions such as Sphere, Rosenbrock, Rastrigin, Griewank and Schaffer's f6 functions. The numerical simulation results revealed that $c_3 = 5.0$ and $c_4 = 5.5$ are suitable for all functions. The trajectory of the global best particles was illustrated. While, in the original PSO, the global best particle was attracted to the local optimum quickly, SG-PSO and SP-PSO could avoid the local optimum and finally find the global optimum.

The applicability of SG-PSO and SP-PSO to various numerical examples will be discussed in the following chapters.

Table 3.2: Estimation values on SG-SPO.

(a) Sphere function

c_3	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	0.8	0.55	0.85	0.85	0.9	0.8	0.9	0.95	0.95	0.7
Average search time	276	338	457	560	386	239	316	292	277	291
Estimation	345	614	538	659	429	299	351	308	<u>292</u>	416

(b) Rosenbrock function

c_3	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	0.5	0.9	0.95	0.8	0.95	0.9	0.9	0.9	1	0.85
Average search time	716	843	516	382	422	258	254	317	240	207
Estimation	1431	936	543	477	444	287	282	352	<u>240</u>	244

(c) Rastrigin function

c_3	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	0.9	0.95	0.8	1	0.95	1	0.9	1	0.9	0.85
Average search time	149	215	336	109	226	155	107	202	101	663
Estimation	165	226	419	<u>109</u>	238	155	119	202	112	780

(d) Griewank function

c_3	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	0.45	0.8	0.65	0.95	1	0.85	0.8	0.95	0.9	1
Average search time	327	250	365	462	373	302	216	312	201	209
Estimation	728	313	561	486	373	355	270	328	223	<u>209</u>

(e) Schaffer's f6 function

c_3	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	1	1	1	1	1	1	1	1	1	1
Average search time	11.8	8.9	12.1	11.4	8.6	9.0	10.6	11	6.8	12
Estimation	11.8	8.9	12.1	11.4	8.6	9.0	10.6	11	<u>6.8</u>	12

Table 3.3: Estimation values on SP-PSO.

(a) Sphere function

c_4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	1	1	1	1	1	1	1	1	1	1
Average search time	1024	1205	1178	922	489	421	685	851	387	291
Estimation	1024	1205	1178	922	489	421	685	851	387	291

(b) Rosenbrock function

c_4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	1	0.95	1	1	0.95	1	1	0.95	0.95	1
Average search time	618	849	1415	1035	613	729	370	557	378	273
Estimation	618	894	1415	1035	645	729	370	586	398	273

(c) Rastrigin function

c_4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	1	0.95	1	1	1	1	1	1	1	1
Average search time	131	141	98	115	81	62	84	86	68	58
Estimation	131	148	98	115	81	62	84	86	68	58

(d) Griewank function

c_4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	0.9	0.9	1	0.95	1	1	1	1	1	1
Average search time	1192	507	1136	561	675	512	315	352	402	309
Estimation	1325	563	1136	590	675	512	315	352	402	309

(e) Schaffer's f6 function

c_4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
Success rate	1	1	1	1	1	1	1	1	1	1
Average search time	14.9	9.5	5.6	10.2	5.4	3.8	6.0	10.3	2.8	3.1
Estimation	14.9	9.5	5.6	10.2	5.4	3.8	6.0	10.3	2.8	3.1

Chapter 4

Search Performance Evaluation of Proposed Algorithms

4.1 Introduction

In order to estimate the premature convergence problem by any algorithm, test functions are frequently used. The premature convergence means too early convergence of a population of potential solutions, resulting in not the global optimal solution but a local (sub-) optimal solution. The aim of this chapter is to discuss that SG-PSO and SP-PSO are compared with different PSO variants using test functions. For this research, simple unimodal, multimodal function and some functions of CEC2005 are applied: Sphere, Rosenbrock, Schwefel, Rastrigin, Weierstrass, Shifted Sphere, Shifted Schwefel's Problem 1.2, Shifted Rosenbrock, Shifted Rastrigin, Shifted Rotated Rastrigin and Shifted Rotated Weierstrass. The Global PSO-w, Global PSO-cf, Local PSO-w, Local PSO-cf, UPSO and CLSPSO are compared with SG-PSO and SP-PSO as different PSO variants. Then, the convergence property of the SG-PSO and SP-PSO are discussed in the convergence of the average function values.

4.2 Test Functions

The all of functions are on 30 dimensions to evaluate the performance of SG-PSO and SP-PSO. The 11 test functions and 3D maps of 2 dimensions are summarized as follows.

Sphere function

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 \tag{4.1}$$

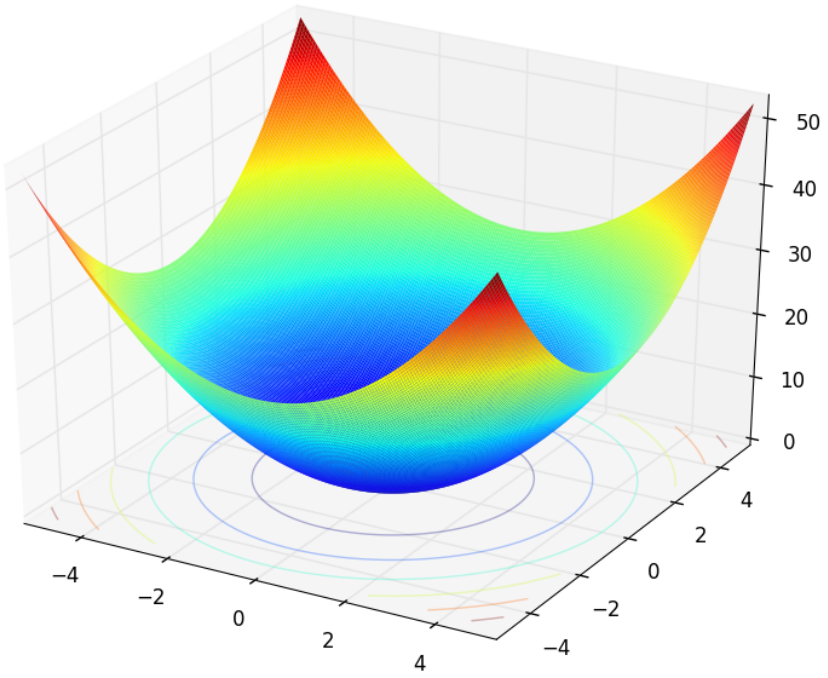


Figure 4.1: 3D map of two-dimensional Sphere function.

Rosenbrock function

$$f_2(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right) \tag{4.2}$$

Schwefel function

$$f_3(\boldsymbol{x}) = 418.9829 \times n \sum_{i=1}^{n} (x_i \sin(|x_i|^{\frac{1}{2}})) \tag{4.3}$$

Rastrigin function

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{4.4}$$

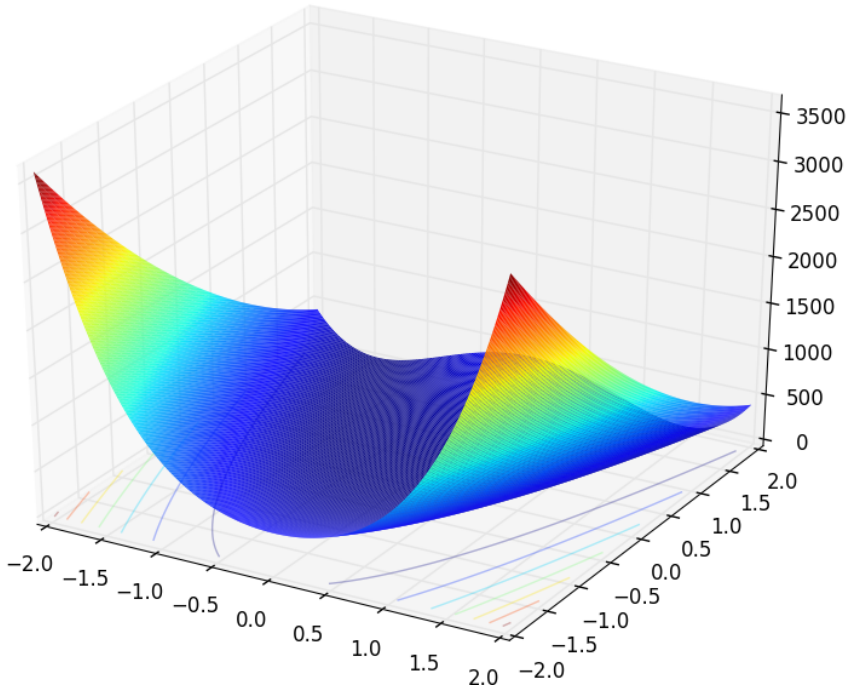


Figure 4.2: 3D map of two-dimensional Rosenbrock function.

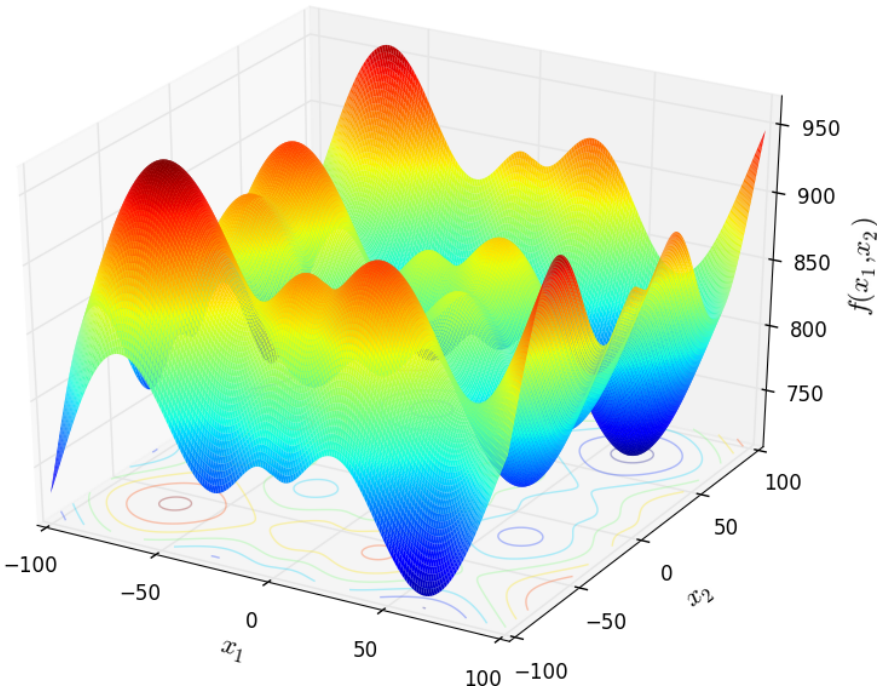


Figure 4.3: 3D map of two-dimensional Schwefel function.

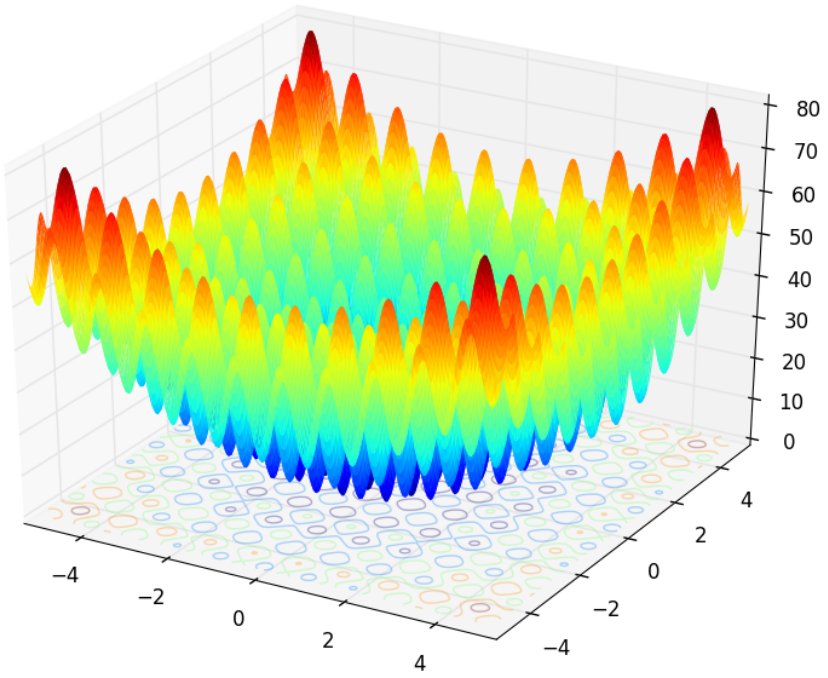


Figure 4.4: 3D map of two-dimensional Rastrigin function.

Weierstrass function

$$
\begin{aligned}
f_5(\boldsymbol{x}) &= \sum_{i=1}^{n}\left(\sum_{k=0}^{k_{\max}}[a^k \cos\left(2\pi b^k(x_i+0.5)\right)]\right) - n\sum_{k=0}^{k_{\max}}[a^k \cos(2\pi b^k \cdot 0.5)] \qquad (4.5)\\
a &= 0.5, b=3, k_{\max}=20
\end{aligned}
$$

Shifted Sphere function

$$
\begin{aligned}
f_6(\boldsymbol{x}) &= \sum_{i=1}^{n} z_i^2 - 450, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} \qquad (4.6)\\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum}
\end{aligned}
$$

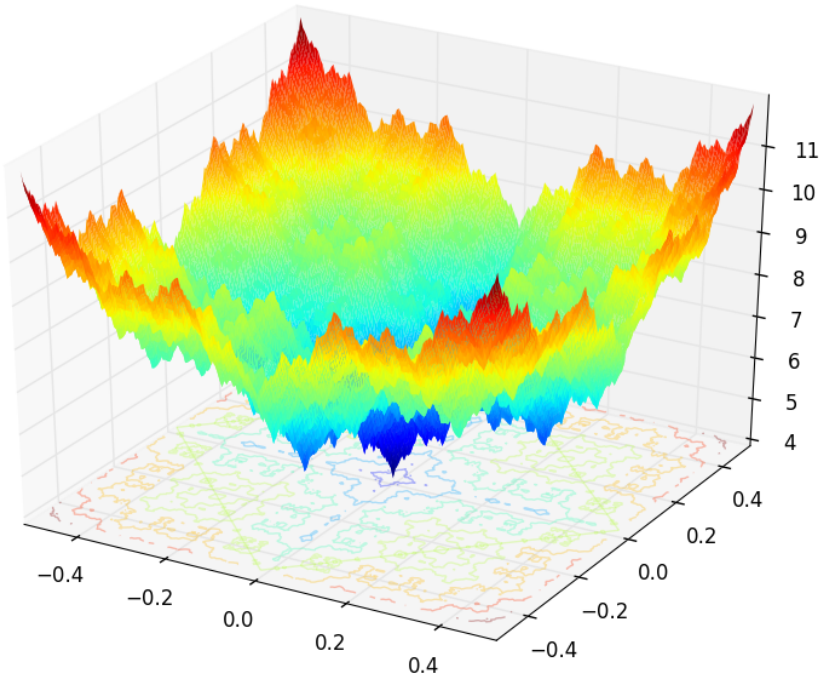


Figure 4.5: 3D map of two-dimensional Weierstrass function.

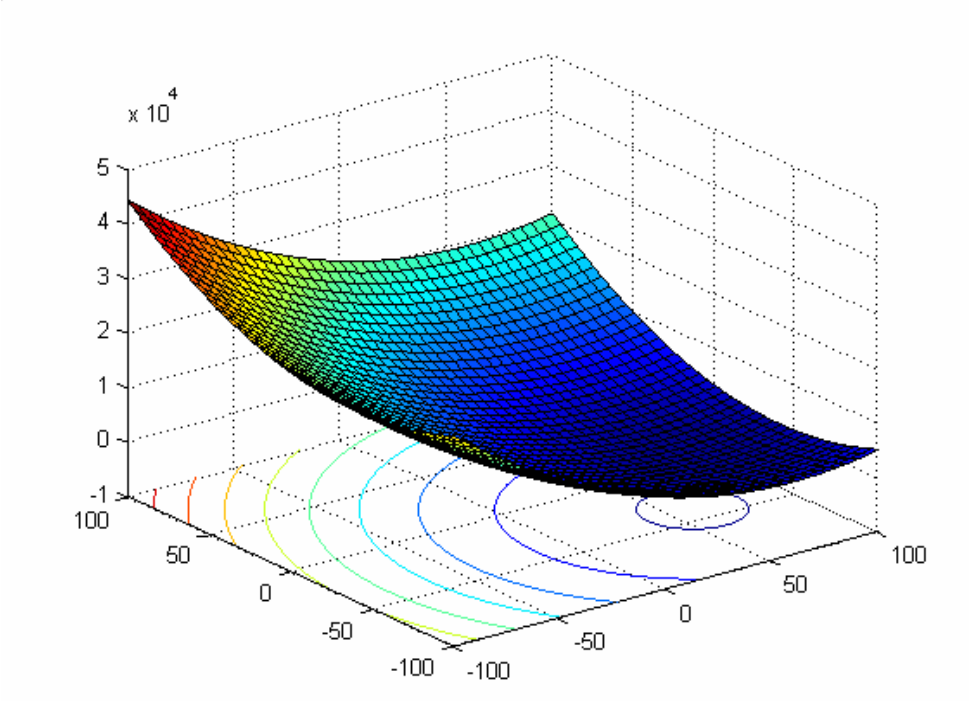


Figure 4.6: 3D map of two-dimensional Shifted Sphere function.

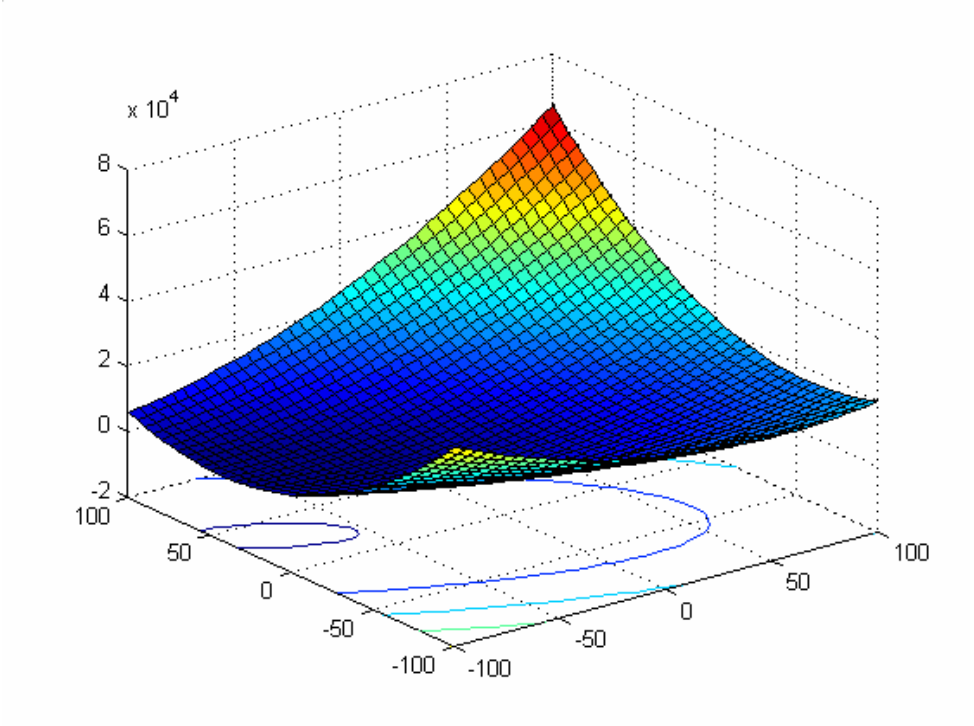


Figure 4.7: 3D map of two-dimensional Shifted Schwefel's Problem 1.2 function.

Shifted Schwefel's Problem 1.2 function

$$
\begin{aligned}
f_7(\boldsymbol{x}) &= \sum_{i=1}^{n}(\sum_{j=1}^{i} z_i)^2 - 450, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} \\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum}
\end{aligned} \tag{4.7}
$$

Shifted Rosenbrock function

$$
\begin{aligned}
f_8(\boldsymbol{x}) &= \sum_{i=1}^{n-1} \left(100\left(x_{z+1} - z_i^2\right)^2 + \left(z_i - 1\right)^2\right) + 390, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} \\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum}
\end{aligned} \tag{4.8}
$$

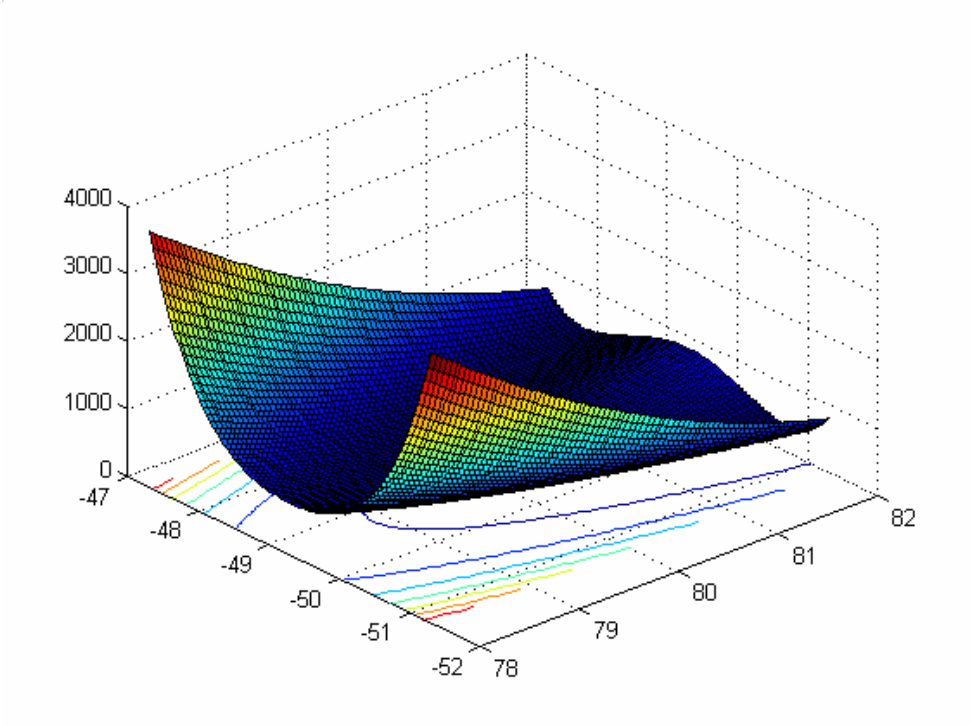


Figure 4.8: 3D map of two-dimensional Shifted Rosenbrock function.

Shifted Rastrigin function

$$
\begin{aligned}
f_9(\boldsymbol{x}) &= \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi z_i) + 10) - 330, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} \\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum}
\end{aligned} \tag{4.9}
$$

Shifted Rotated Rastrigin function

$$
\begin{aligned}
f_{10}(\boldsymbol{x}) &= \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi z_i) + 10) - 330, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} * \boldsymbol{M} \\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum} \\
\boldsymbol{M} &: N \times N \text{ orthogonal matrix}
\end{aligned} \tag{4.10}
$$

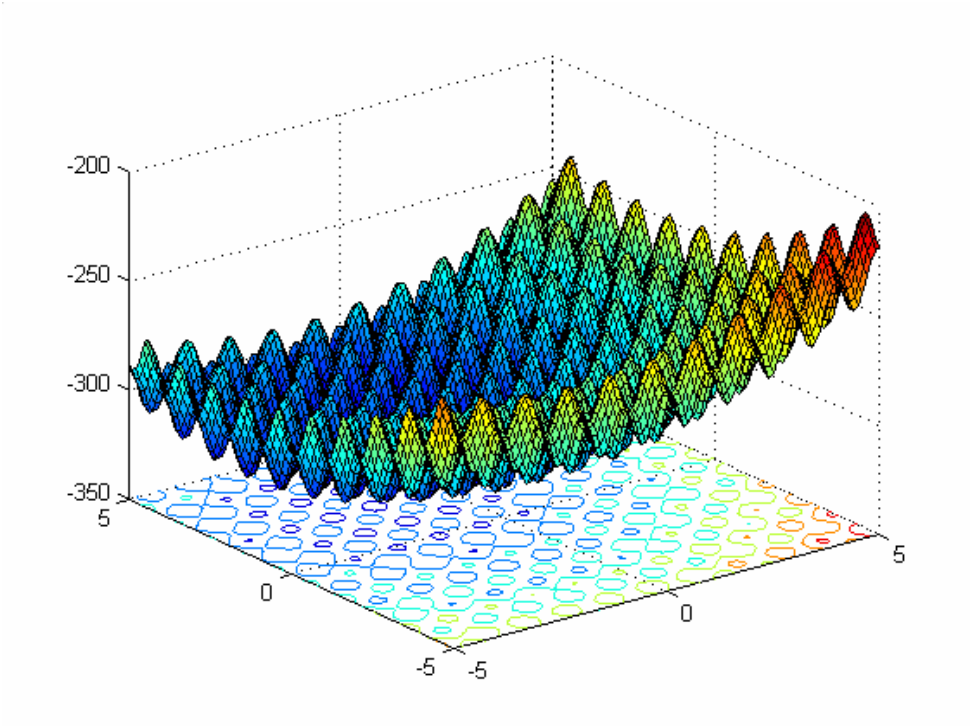


Figure 4.9: 3D map of two-dimensional Shifted Rastrigin function.

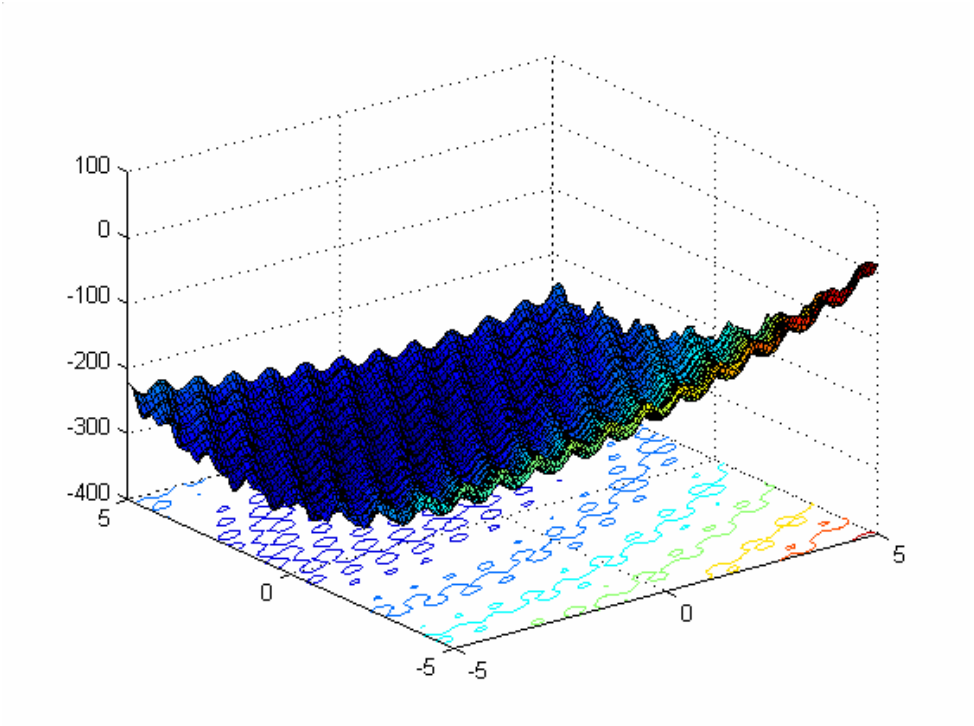


Figure 4.10: 3D map of two-dimensional Shifted Rotated Rastrigin function

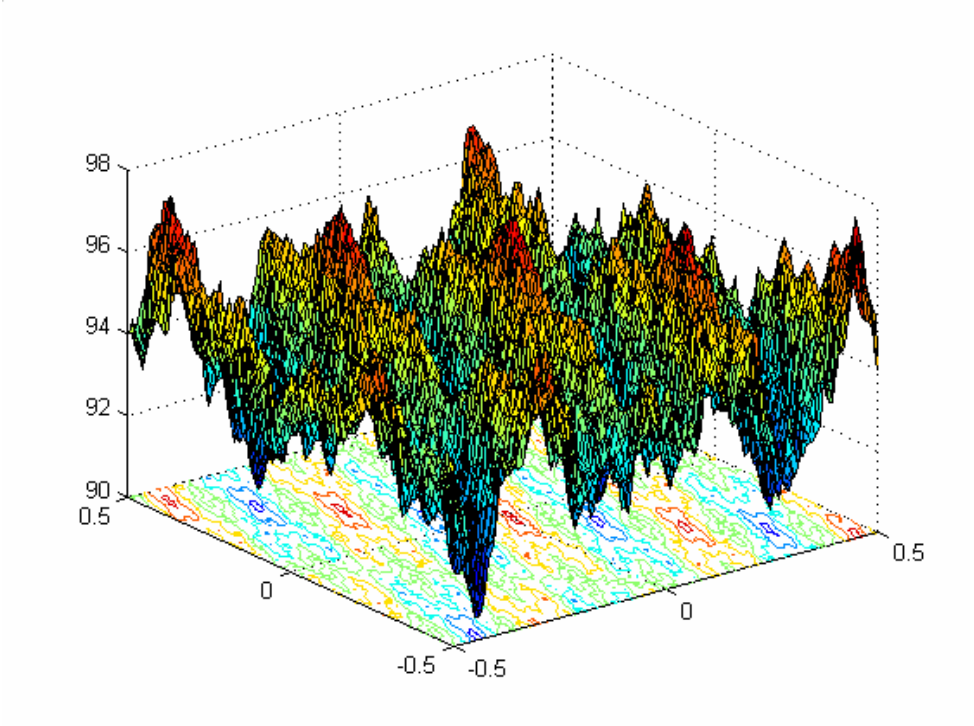


Figure 4.11: 3D map of two-dimensional Shifted Rotated Weierstrass function.

Shifted Rotated Weierstrass function.

$$
\begin{aligned}
f_{11}(\boldsymbol{x}) &= \sum_{i=1}^{n}\left(\sum_{k=0}^{k_{\max}}[a^k \cos\left(2\pi b^k(z_i+0.5)\right)]\right) \\
&- n\sum_{k=0}^{k_{\max}}[a^k \cos(2\pi b^k \cdot 0.5)] + 90 \qquad (4.11)\\
\boldsymbol{z} &= \boldsymbol{x} - \boldsymbol{o} * \boldsymbol{M}, a = 0.5, b = 3, k_{\max} = 20 \\
\boldsymbol{o} &= [o_1, o_2, ..., o_n] : \text{the shifted global optimum} \\
\boldsymbol{M} &: N \times N \text{ orthogonal matrix}
\end{aligned}
$$

The Sphere function and Rosenbrock function are well-known simple unimodal functions. The Sphere fuction is no trouble to find the global minimum. The Rosenbrock has a narrow valley shape the perceived local optima to the global optimum. The Schwefel function is a multimodal function which has a deep local optima far from the global minimum, and the Rastigin function is a typical multimodal function for mathematical optimization. The Weierstrass function is continuous everywhere, but differentiable nowhere. The functions, shifted the global optimum, are the Shifted Sphere fuction, Shifted Schwefel's

Table 4.1: Global minimum and search range.

f_{num}	$f(x^*)$	x^*	Search Range
f_1	0	$[0,0]^n$	$[-100,100]$
f_2	0	$[1,1]^n$	$[-2.048,2.048]$
f_3	0	$[420.968,420.968]^n$	$[-500,500]$
f_4	0	$[0,0]^n$	$[-5.12,5.12]$
f_5	0	$[0,0]^n$	$[-0.5,0.5]$
f_6	-450	$[o,o]^n$	$[-100,100]$
f_7	-450	$[o,o]^n$	$[-100,100]$
f_8	390	$[o,o]^n$	$[-100,100]$
f_9	-330	$[o,o]^n$	$[-5,5]$
f_{10}	-330	$[o,o]^n$	$[-5,5]$
f_{11}	90	$[o,o]^n$	$[-0.5,0.5]$

Problem 1.2 function, Shifted Rosenbrock function and the Shifted Rastrigin function. The functions, shifted the global optimum and rotated by orthogonal matrix, are the Shifted Rotated Rastrigin function and the Shifted Rotated Werierstrass function [46].

The global minimums and search ranges of each function are given in Table 4.1.

4.3 Numerical Results

The search performance of eight PSO algorithms are compared on the 11 benchmark functions shown in Eqs.(4.1) to (4.12). The dimension of the benchmark function is specified to be $n = 30$. 30 simulations are performed at each function and then, the error value is estimated by the following equation.

$$\text{Error value} = |f_i(x) - f_i(x^*)| \quad (i = 1, 2, ..., n)$$

Since the variable x^* denotes the optimal solution, the expression $f_i(x^*)$ denotes the minimum value of the function $f_i(x)$. The average values are shown in Table 6.2. The smallest value for each benchmark function is underlined in the table. Except for f_1, f_{10} and f_{11}, the smallest values are found by SP-PSO. In the function f_1, f_{10} and f_{11}, the smallest values found by CLPSO and/or SG-PSO. The SP-PSO could find the second smallest values at the functions. Since the differences between the values by SP-PSO and the values by CLPSO or SG-PSO are small, it is concluded that SP-PSO can find the better solutions for all functions.

The convergence histories of the average errors are shown in Figs. 4.12 to 4.22. These figures are plotted with the iteration as the horizontal axis and the average error as the vertical axes, respectively. The result shows that, except for Shifted Rotated Weierstrass function f_{11}, SP-PSO converges earlier and closed to zero than the other PSO algorithms. Especially, SP-PSO shows much faster convergence property in cases of Schwefel function f_3 and Shifted Rosenbrock function f_8. In case of Shifted Rotated Weierstrass function f_{11}, the first and the second best performances are observed at the SG-PSO and SP-PSO, respectively.

4.4 Conclusion

In the original Particle Swarm Optimization (PSO), the particle positions are updated from the positions of the personal best and the global best particles which have ever been found. This research focuses on the use of SG-PSO and SP-PSO for improving the search performance of the original PSO.

Then, the SG-PSO and SP-PSO were compared with the other PSO algorithms in 11 benchmark functions. SP-PSO algorithms showed the best search performance in 10 benchmark functions except for Shifted Rotated Weierstrass function. In case of Shifted Rotated Weierstrass function, SG-PSO and SP-PSO algorithms were the first and the second best ones, respectively.

In the SG-PSO and SP-PSO, the update rules by the original PSO are selected at equal probability. Their selection probability, however, may affect the search performance of

Table 4.2: Results of SG-PSO, SP-PSO and other PSOs for 11 test functions.

f_{num}	Global PSO-w	Global PSO-cf	Local PSO-w	Local PSO-cf
f_1	1.00e+03	1.82e+03	1.79e+02	1.34e+04
f_2	1.07e+02	4.60e+02	1.27e+02	1.72e+03
f_3	4.46e+03	5.84e+03	6.41e+03	7.36e+03
f_4	4.11e+01	1.38e+02	4.67e+01	2.30e+02
f_5	1.83e+00	2.70e+01	4.37e+00	2.70e+01
f_6	9.87e+03	1.43e+04	9.23e+03	4.92e+04
f_7	6.00e+06	5.88e+06	5.89e+06	1.01e+07
f_8	2.48e+09	2.43e+09	9.29e+08	1.86e+10
f_9	1.38e+02	2.41e+02	2.93e+02	4.01e+02
f_{10}	2.28e+02	3.96e+02	3.83e+02	5.72e+02
f_{11}	2.99e+01	3.53e+01	3.37e+01	3.65e+01
f_{num}	UPSO	CLPSO	SG-PSO	SP-PSO
f_1	9.07e+03	0.00e+00	0.00e+00	1.48e-06
f_2	1.03e+03	4.81e+01	2.08e+02	2.20e+01
f_3	7.98e+03	5.46e+03	4.20e+03	4.03e+03
f_4	2.00e+02	5.68e+00	2.86e+01	3.28e+00
f_5	2.64e+01	2.42e-01	1.60e+00	0.00e+00
f_6	3.19e+04	1.36e+04	1.24e+04	1.63e+01
f_7	7.85e+06	5.49e+06	5.70e+06	1.73e+05
f_8	1.47e+10	1.11e+10	2.99e+09	1.45e+05
f_9	2.48e+02	2.16e+02	1.57e+02	1.29e+02
f_{10}	3.90e+02	1.70e+02	2.31e+02	1.72e+02
f_{11}	3.98e+01	3.27e+01	2.77e+01	2.89e+01

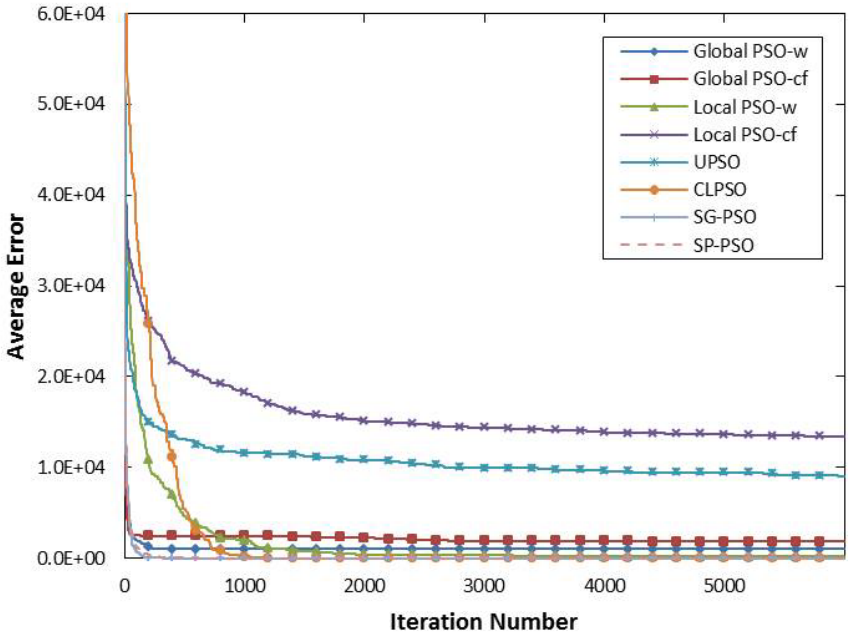


Figure 4.12: Convergence history in Sphere function.

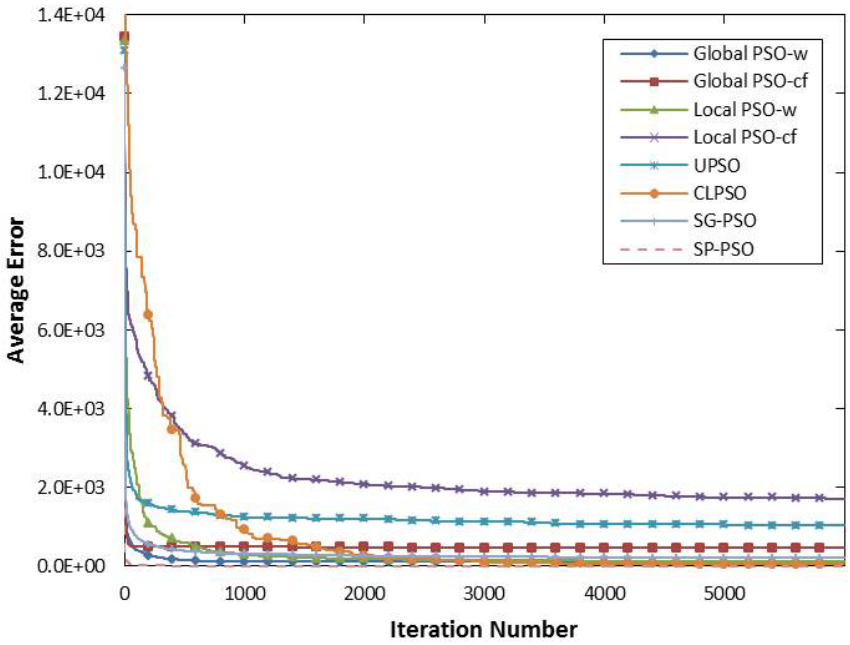


Figure 4.13: Convergence history in Rosenbrock function.

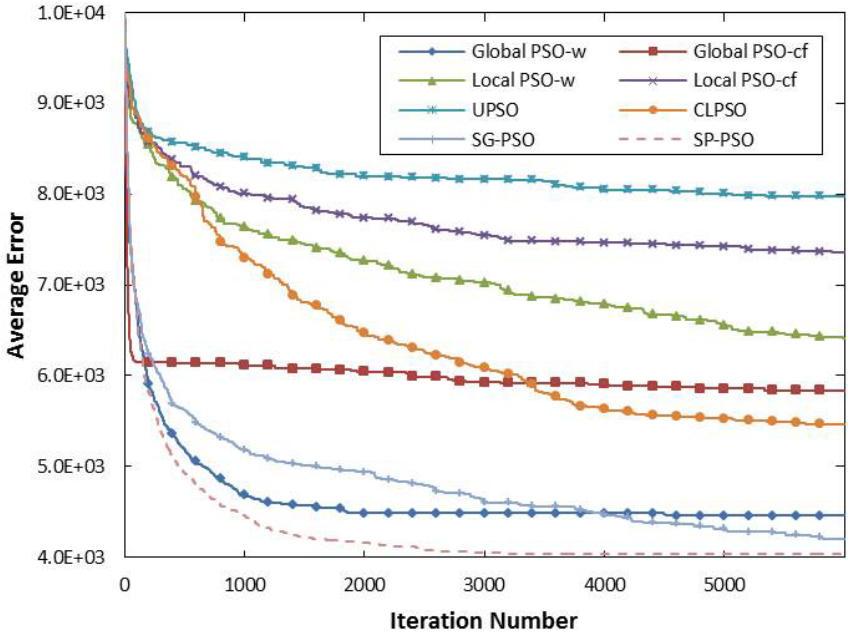


Figure 4.14: Convergence history in Schwefel function.

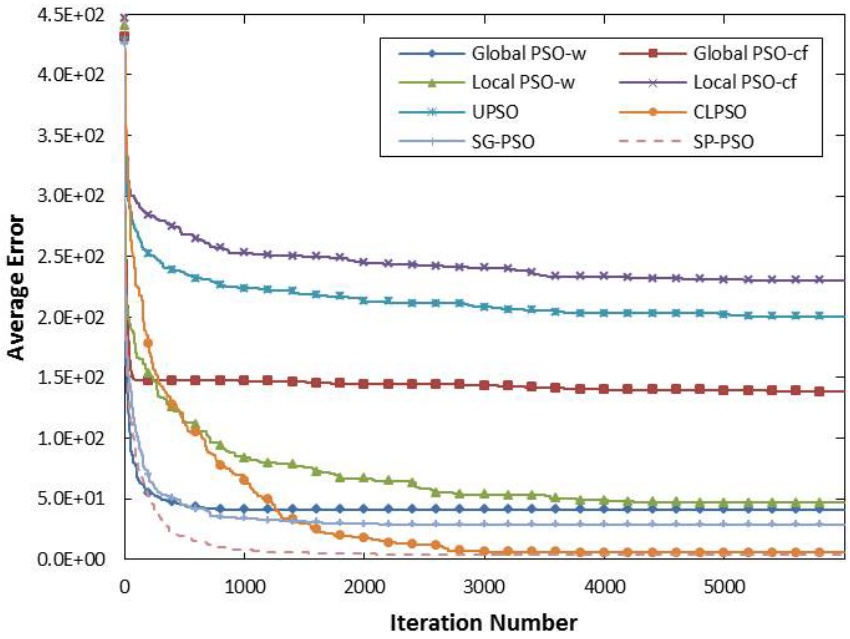


Figure 4.15: Convergence history in Rastrigin function.

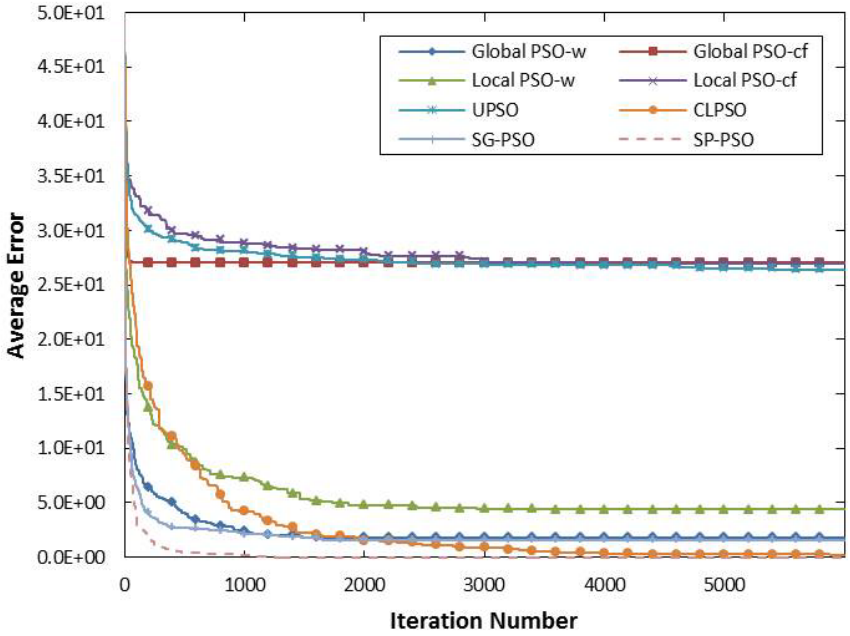


Figure 4.16: Convergence history in Weierstrass function.

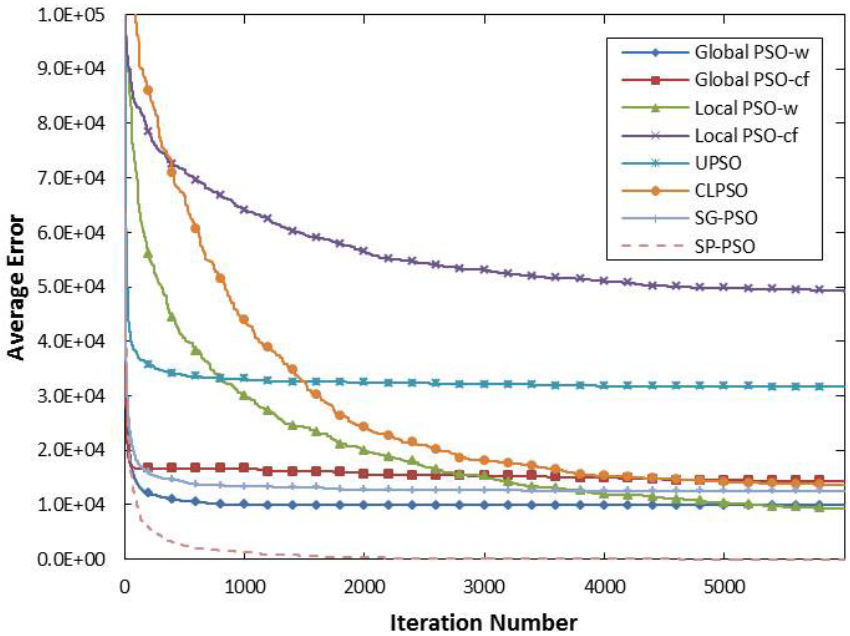


Figure 4.17: Convergence history in Shifted Sphere function.

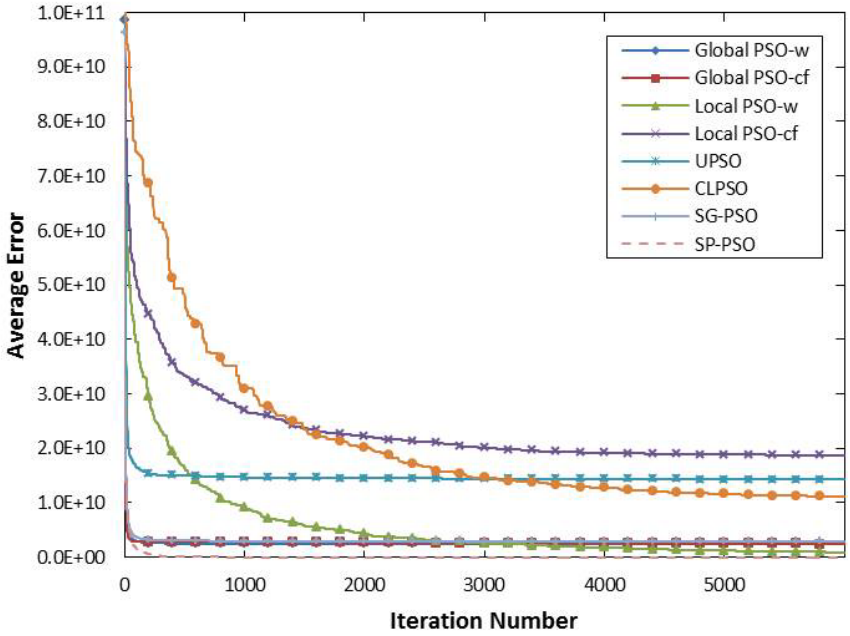


Figure 4.18: Convergence history in Shifted Schwefel's problem 1.2 function.

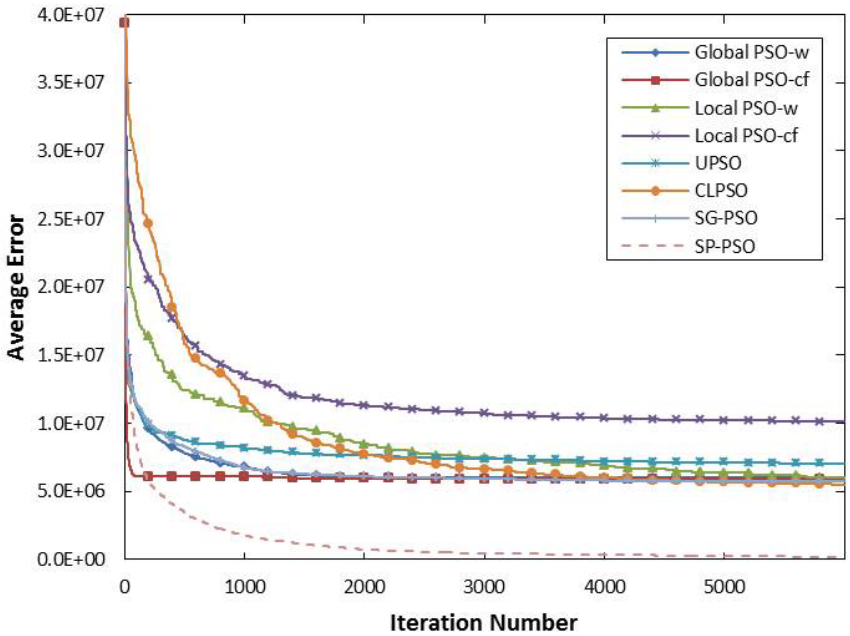


Figure 4.19: Convergence history in Shifted Rosenbrock function.

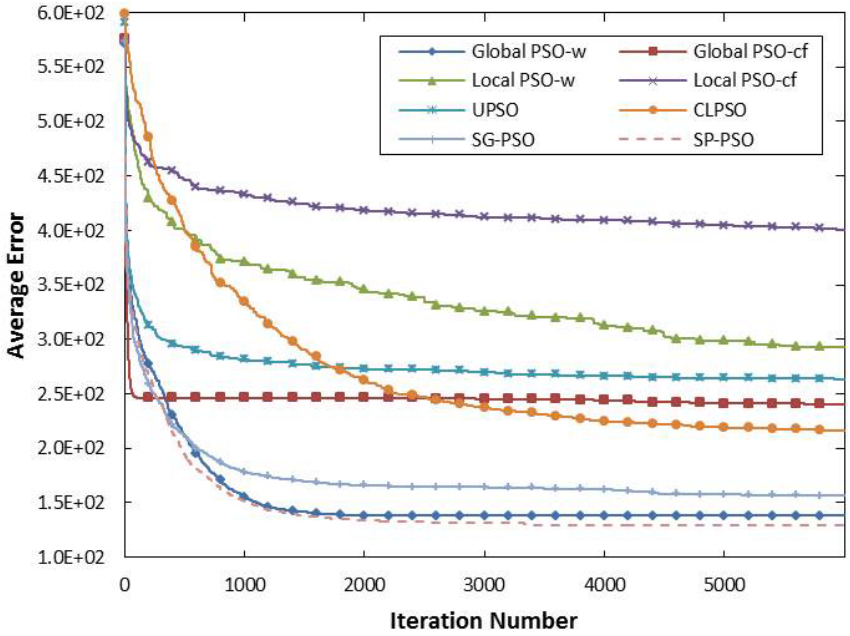


Figure 4.20: Convergence history in Shifted Rastrigin function.

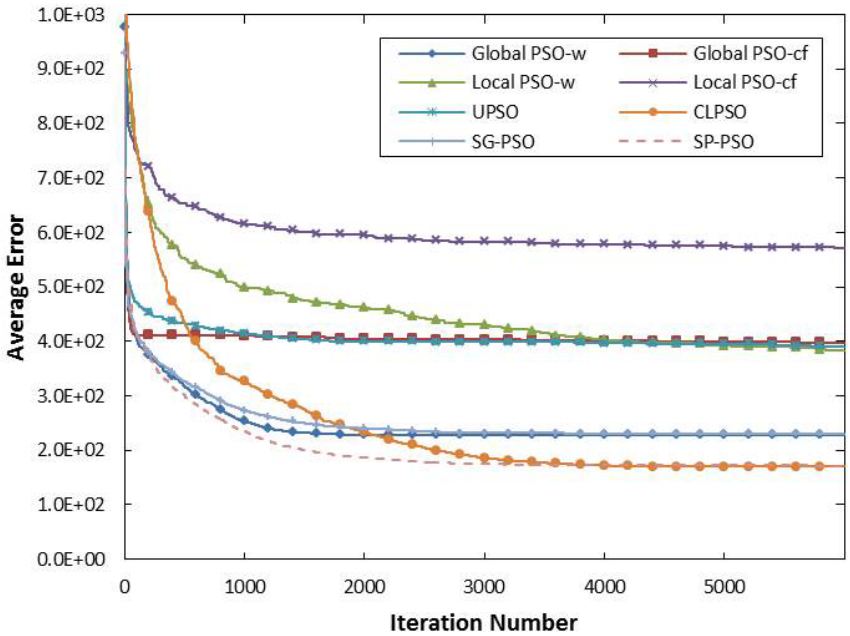


Figure 4.21: Convergence history in Shifted Rotated function.

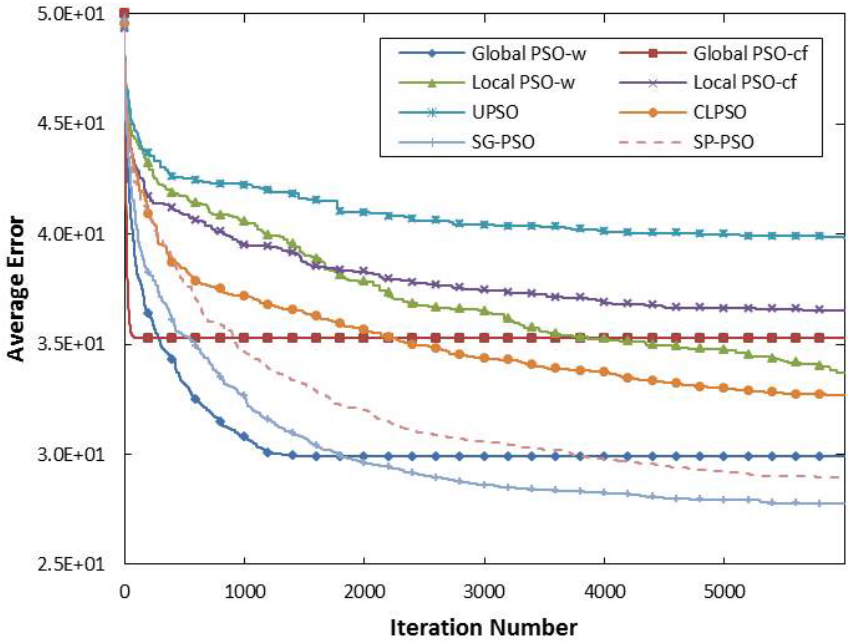


Figure 4.22: Convergence history in Shifted Rotated Weierstrass function.

the algorithms. This point has to be discussed. In near future, the applicability of the SG-PSO and SP-PSO will be studied to actual engineering applications.

Chapter 5

Application to Packing Problem

5.1 Introduction

Packing problems are a class of optimization problems in mathematics which involve attempting to pack objects together (often inside a container), as densely as possible. There are many variations of this problem, such as two-dimensional packing, linear packing, packing by weight and packing by cost. They have many applications, such as filling up containers, loading trucks with weight capacity, creating file backup in removable media and technology mapping in Field-programmable gate array semiconductor chip design.

The author focuses on the two-dimensional packing problems. Popular problems in two-dimensional packing are to pack circles or squares in a larger circle or a square. The problems are studied analytically and the maximum numbers of items are determined [56, 3, 15]. In this chapter, the author considers that the packing regions have the arbitrary polygon-shaped packing region and then, same items are packed in the region without their overlap. The typical example in the steel industry is to stamp same polygonal figures from a rectangular board. The aim of this job is to minimize the remainder region on board. Since the packing problem is one of typical NP-hard problems, it is quite difficult to find optimal solution in the polynomial time.

For solving NP-hard problems, many researchers have applied evolutionary computations such as Genetic Algorithm (GA) [24], Simulated Annealing (SA) [53] and Particle Swarm Optimization (PSO) [26]. In this study, PSO is applied for solving two-dimensional packing problems.

The application of PSO for solving packing problem has been presented by some researchers [54, 10, 5, 47]. Liu et al. [54] presented evolutionary PSO for solving bin packing problem. Zhao et al. [10, 5] applied the discrete PSO for solving rectangular packing problem. Thapatsuwan et al. [47] compared GA and PSO for solving multiple container packing problems. They focus on the packing problem of container in the storage or the ship cabin. Since the storage and the ship cabin are designed so that their

sizes are equal to the integral multiple of the container sizes, it is assumed that the items are placed every certain interval. On the other hand, the author will consider that the packing region is arbitrarily polygon-shaped. Since, in this case, the packing region sizes do not depend on the item sizes, the problems to be solved are much more difficult than the previous studies.

In this study, PSO is applied for solving the packing problems which have arbitrarily polygon-shaped regions. The design objective is to maximize the total number of the items packed in the region without the item overlap. The total number of items and the position vectors of the item centers are taken as the design variables. The problem is solved by the original PSO and SG-PSO. In the PSO, the candidate solutions of the optimization problem to be solved are defined as the particle position vectors. Then, the particle positions are updated by PSO update rules. In the original PSO, the particle position vector is updated by the global best position and the personal best position in previous positions of each particle. The SG-PSO utilizes, in addition to them, second global best position used in probability P_s.

The remaining part of this chapter is organized as follows. The optimization problem is explained in 5.2. In section 5.3, the packing problem in two-dimensional regions is solved. Finally, the conclusions are summarized in section 5.4.

5.2 Packing Problem

5.2.1 Optimization Problem

The packing problem can be formulated to maximize the number of items z included into a two-dimensional polygonal region P.

The objective function is the number of items z included into a two-dimensional polygonal region P.

$$\max z \tag{5.1}$$

When the vector $\{p_x^i, p_y^i\}$ denotes the center position vector of the item i, the design variable vector is defined as follows.

$$\boldsymbol{x} = \{p_x^1, p_y^1, \cdots, p_x^i, p_y^i, \cdots, p_x^z, p_y^z\}^T \tag{5.2}$$

The side constraint conditions for the design variables are given as follows.

$$0.5w \leq p_x^i \leq W - 0.5w \tag{5.3}$$

$$0.5h \leq p_y^i \leq H - 0.5h \tag{5.4}$$

where w and h denote item sizes, and W and H feasible space sizes, respectively.

All items should be included in the region P without their overlapping. The constraint conditions for such situation are defined by two constraint conditions.

$$\begin{aligned} g_1(i,P) &= 0 && (5.5)\\ g_2(i,j) &= 0 && (5.6)\\ & i=1,2,\ldots,z; j=1,2,\ldots,z \end{aligned}$$

The function $g_1(i,P)$ estimates the inclusion of the item i in the region P, which is defined as follows:

$$g_1(i,P) = \begin{cases} 0 & \text{The item } i \text{ is included in the region } P. \\ 1 & \text{The item } i \text{ is not included in the region } P. \end{cases} \quad (5.7)$$

The function $g_2(i,j)$ estimates the overlap between the item i and the item j, which is defined as follows:

$$g_2(i,j) = \begin{cases} 0 & \text{The item } i \text{ and } j \text{ are not overlapped.} \\ 1 & \text{The item } i \text{ and } j \text{ are overlapped.} \end{cases} \quad (5.8)$$

5.2.2 PSO Implementation

The optimization problem was defined in the previous section. It it very difficult to solve the optimization problem directly because the constraint conditions are fragile. Therefore, the optimization problem is solved according to the following steps.

1. The number of the items z is initialized as $z = 0$.
2. The number of the items z is updated as $z = z + 1$.
3. z items are arranged in the region P so that

$$G(\boldsymbol{x}) = \sum_{i=1}^{z} \left\{ g_1(i,P) + \sum_{j=1}^{z, i\neq j} g_2(i,j) \right\} \rightarrow \min . \quad (5.9)$$

4. If $G(\boldsymbol{x}) = 0$, the process goes to step 2.
5. Otherwise, $z = z - 1$ because z items could not be arranged in the region P.

PSO is employed for solving the step 3 in the previous algorithm.

When the number of the items is given, PSO is applied for solving the item packing problem within the packing region without violating the constraint conditions. The optimization problem is defined as follows.

- Fitness function

$$f(\boldsymbol{x}_i) = \frac{1}{1 + \sum_{i=1}^{z} \left\{ g_1(i,P) + \sum_{j=1}^{z, i\neq j} g_2(i,j) \right\}} \quad (5.10)$$

- Design variable vector (Particle position vector)

$$\boldsymbol{x}_i = \{p_x^1, p_y^1, \cdots, p_x^i, p_y^i, \cdots, p_x^z, p_y^z\}^T \tag{5.11}$$

- Side constraint conditions for design variables

$$0.5w \leq p_x^i \leq W - 0.5w \quad (i = 1, 2, \cdots, z) \tag{5.12}$$

$$0.5h \leq p_y^i \leq H - 0.5h \quad (i = 1, 2, \cdots, z) \tag{5.13}$$

5.2.3 Optimization Process

The process of the packing problem optimization by using SG-PSO can be summarized as follows (Fig. 5.1).

1. The maximum iteration step t_{max}, swarm size N and P_s are specified.
2. The item number z is initialized as $z = 0$.
3. The item number is updated by $z = z + 1$.
4. PSO algorithm is performed for minimizing the function (5.10).

(a) The iteration number is initialized as $t = 0$.
(b) The particle position vector $\boldsymbol{x}_i(t)$ and velocity vector $\boldsymbol{v}_i(t)$ are initialized with random numbers.
(c) The fitness function for each particle $f(\boldsymbol{x}_i(t))$ is evaluated.
(d) If $t \leq t_{max}$, the process goes to the next step. Otherwise, the process goes to the step (5).
(e) Personal best particle position $\boldsymbol{x}^g(t)$ is updated.
(f) Global best particle position $\boldsymbol{x}_i^p(t)$ is updated.
(g) Second global best particle position $\boldsymbol{x}_i^{p2}(t)$ is updated.
(h) The particle number i is initialized as $i = 1$.
(i) If $i \leq N$, the process goes to the next step. Otherwise, the process goes to the step (4m).
(j) A random number r is generated in the range $[0, 1]$.
(k) If $r \leq P_s$, the position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are updated by Eqs. (2.1) and (3.1), respectively. Otherwise, they are updated by Eqs. (2.1) and (2.2), respectively.
(l) $i = i + 1$, and the process goes to step (4i)
(m) $t = t + 1$, and the process goes to step (4d).

5. If $G(\boldsymbol{x}) = 0$, the process goes to the step 3.
6. Otherwise, the process is terminated by $z = z - 1$.

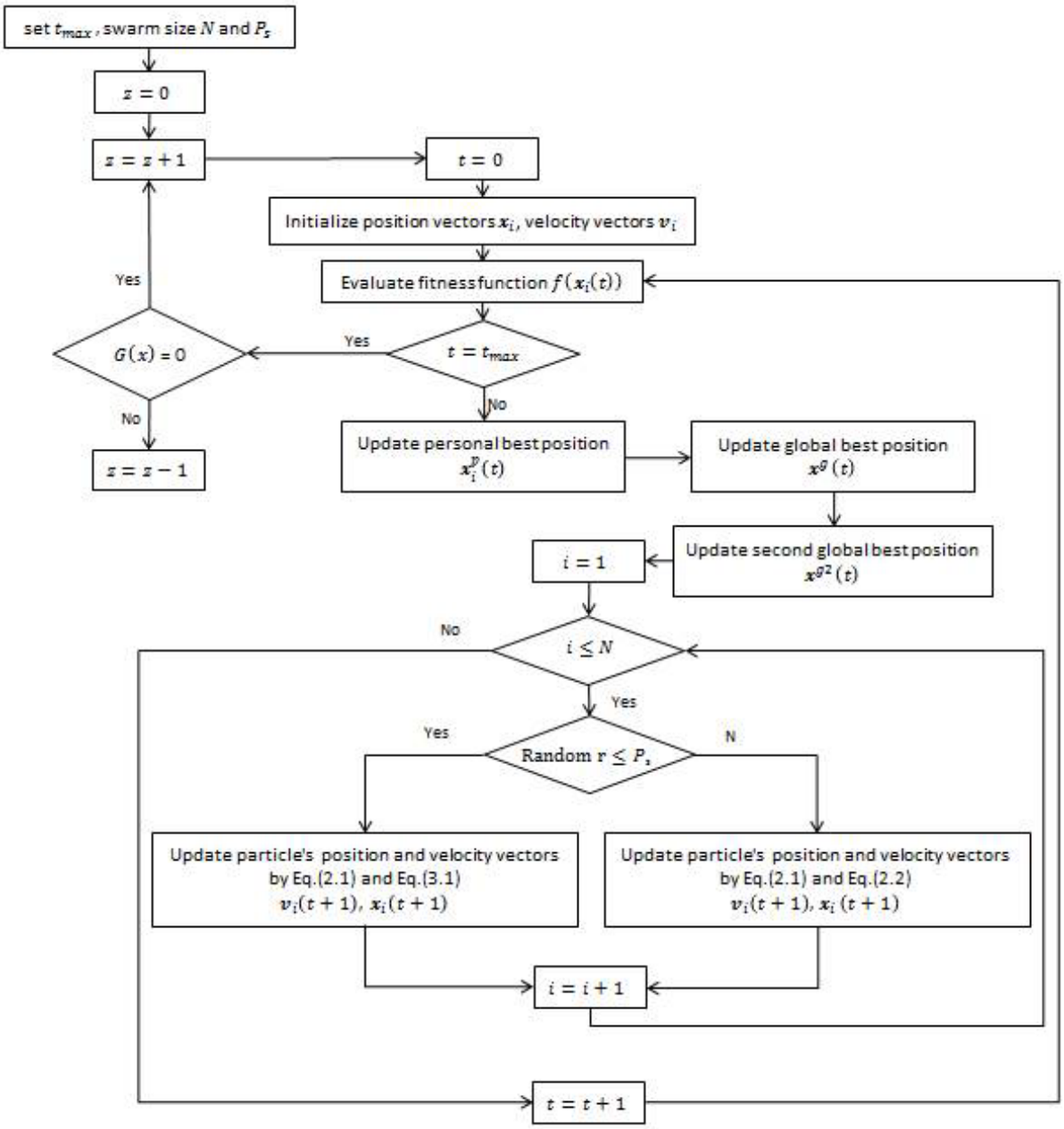


Figure 5.1: Flowchart of the parking problem by using SG-PSO.

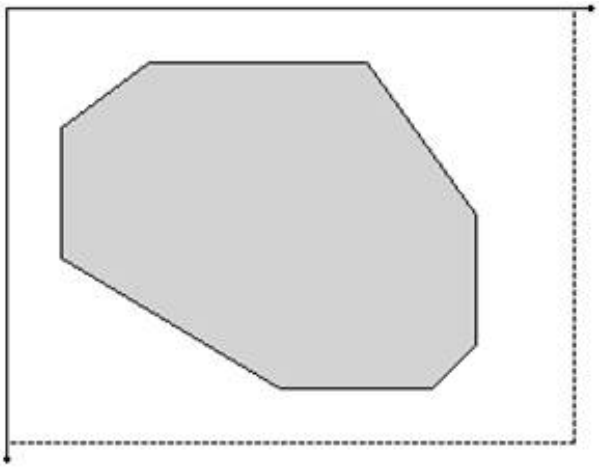


Figure 5.2: Packing region (Case A).

Table 5.1: Swarm size, maximum iteration and other parameters.

Swarm size	$N = 200$
Maximum iteration step	$t_{\max} = 2000$
Update rules parameters	$w_{\max} = 0.9, w_{\min} = 0.4, c_1 = 1.5, c_2 = 1.5, c_3 = 1.9$

5.3 Numerical Examples

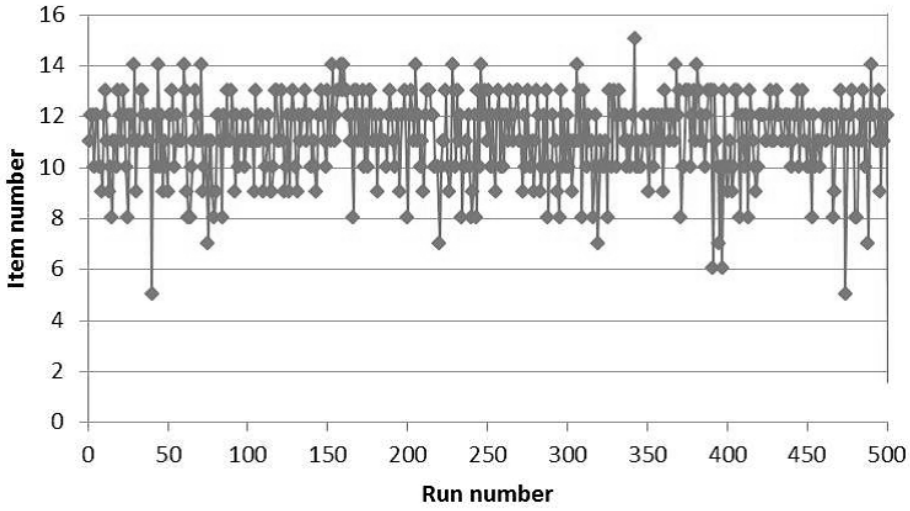
5.3.1 Case A

The packing problem in two-dimensional polygonal regions is considered as a numerical example. The packing region of case A is shown in Fig. 5.2. PSO parameters are shown in Table 5.1. Number of particles and maximum iteration steps are specified as $N = 200$ and $t_{\max} = 2000$, respectively. The other parameters are taken as $w = 0.9$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$, and $P_s = 0.1$.

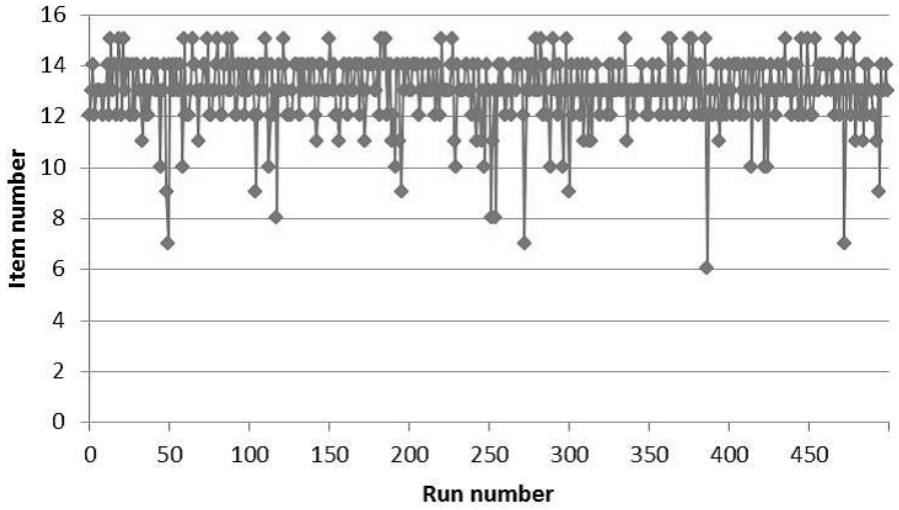
Five hundred simulations are performed from different initial conditions. Maximum item numbers for case A are shown in Fig. 5.3. The figures are plotted with the run

Table 5.2: Comparison of original PSO and SG-PSO in case A.

	Original PSO	SG-PSO
Average item number	11.144	12.984
Average CPU time (seconds)	35.007	59.753
Success rate in $z_{\max} \geq 13$	18.4%	73.2%



(a) Original PSO



(b) SG-PSO

Figure 5.3: Maximum item numbers in Case A.

Table 5.3: Effect of parameter P_s in case A.

P_s	0.1	0.2	0.3	0.4	0.5	0.6
Average item number z	12.97	12.81	12.69	12.06	12.72	11.46
Average CPU time (seconds)	58.82	67.11	64.99	72.47	86.94	144.53

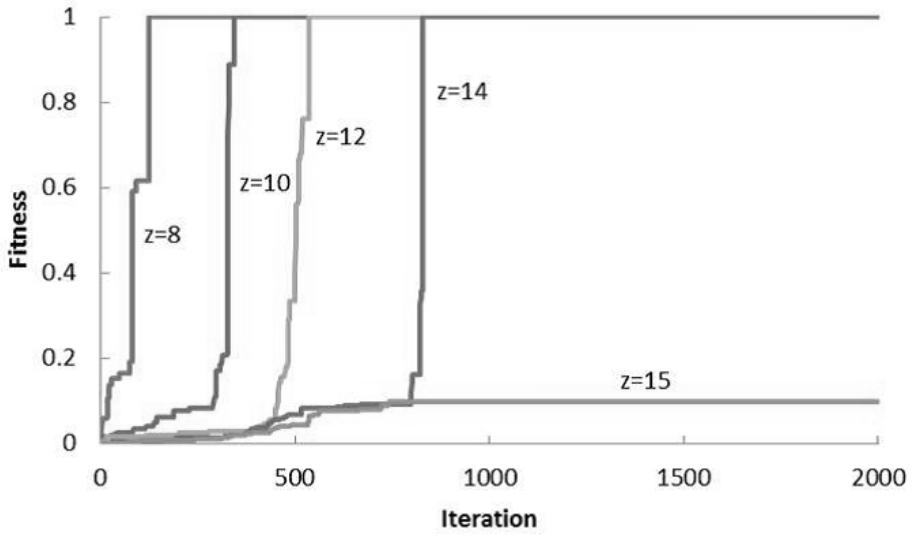
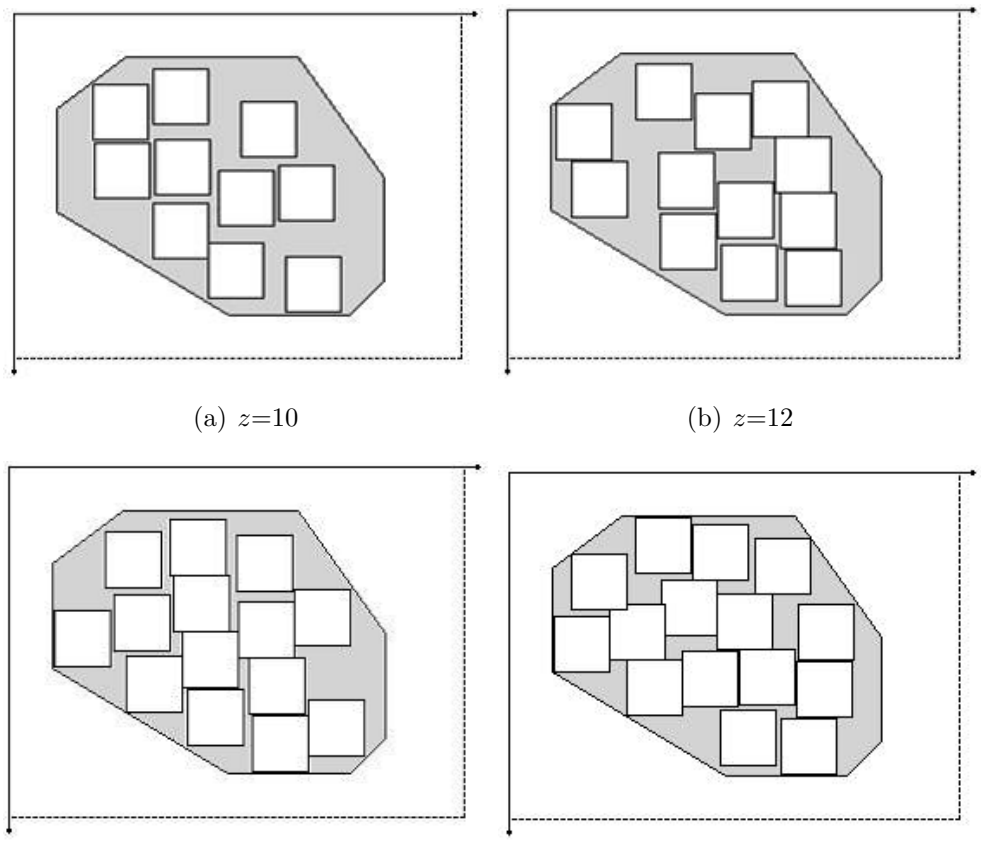


Figure 5.4: Fitness convergence of SG-PSO in case A.



(a) z=10 (b) z=12

(c) z=14 (d) z=15

Figure 5.5: Placement conditions of using SG-PSO in case A.

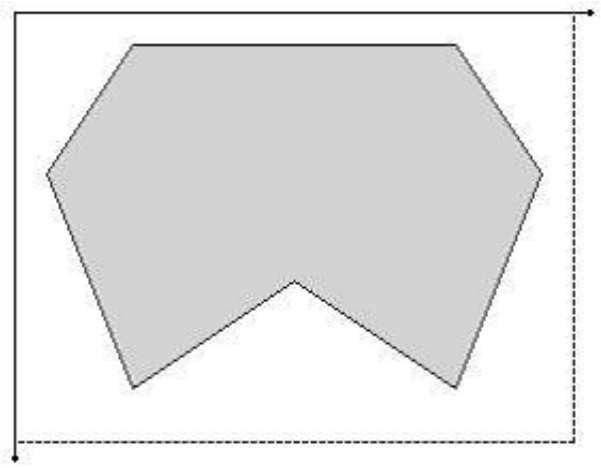


Figure 5.6: Packing region (Case B).

Table 5.4: Comparison of original PSO and SG-PSO in case B.

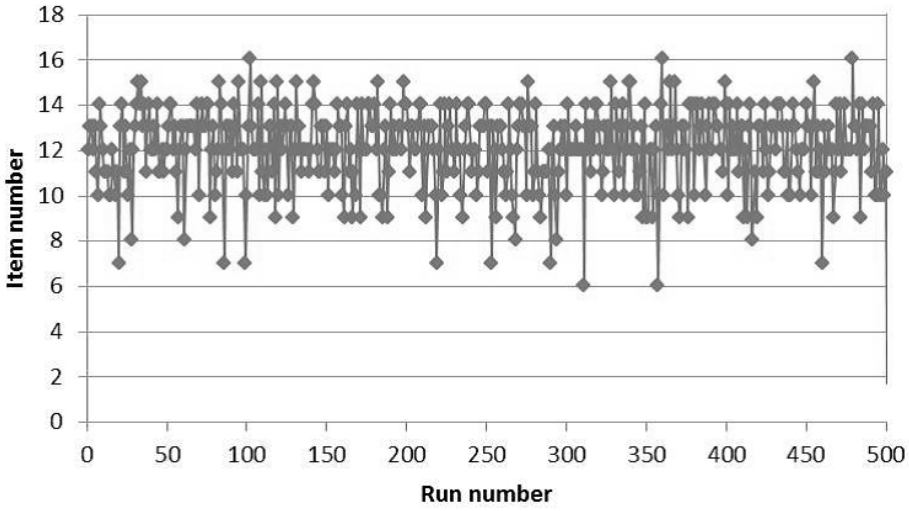
	Original PSO	SG-PSO
Average item number z	12.07	13.99
Average CPU time (seconds)	35.198	65.124
Success rate in $z \geq 14$	19.8%	75.2%

number as the horizontal axis and the item number z as the vertical axis, respectively.

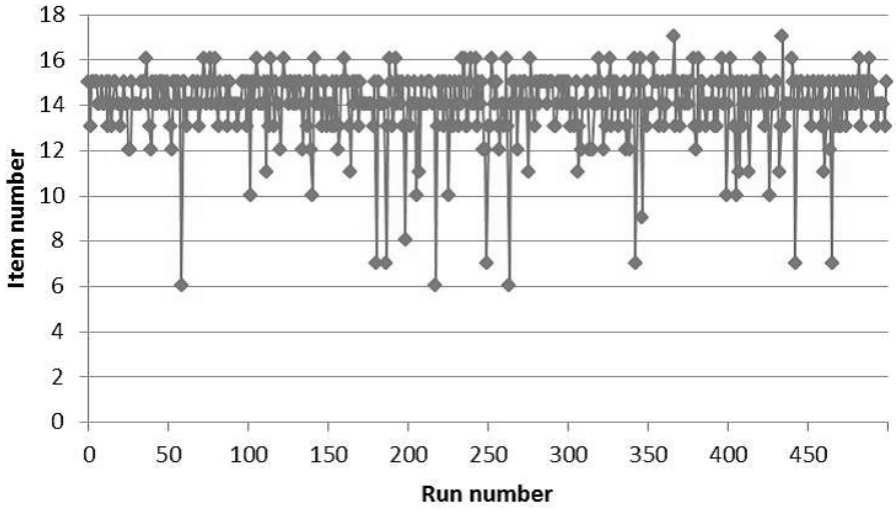
The results by the original PSO and the SG-PSO are compared in Table 5.2. The average item number and the average CPU time denote the average values of the maximum item numbers and CPU time in five hundred runs, respectively. The success rate means the percentage of the runs in which the maximum item number $z_{\max}$ is greater than 13. The average item numbers are 11.144 in case of the original PSO and 12.984 in SG-PSO. The average CPU times are 35.007 and 59.753, respectively. The success rates are 18.4% and 73.2%, respectively. The use of the SG-PSO can increase the item number and improve the success rate although the CPU time is increased.

Fig. 5.4 shows the fitness function $f(\boldsymbol{x}^g)$ at $z = 8, 10, 12, 14$ and 15. In case of item numbers $z = 8, 10, 12$ and 14, fitness functions almost converge to 1 at 400, 700, 800 and 1000 iterations, respectively. In case of item number $z = 15$, fitness cannot converge to 1. Therefore, in this case, maximum item number is concluded to be $z = 14$. Fig. 5.5 shows the item placement in case of the SG-PSO and the items overlap in case of $z = 15$.

Next, the effect of the parameter P_s is discussed. Table 5.3 shows the maximum number of items and the CPU times for the different parameter P_s. The results show that the item number is maximized at $P_s = 0.1$ and CPU time is also shortest.



(a) Original PSO

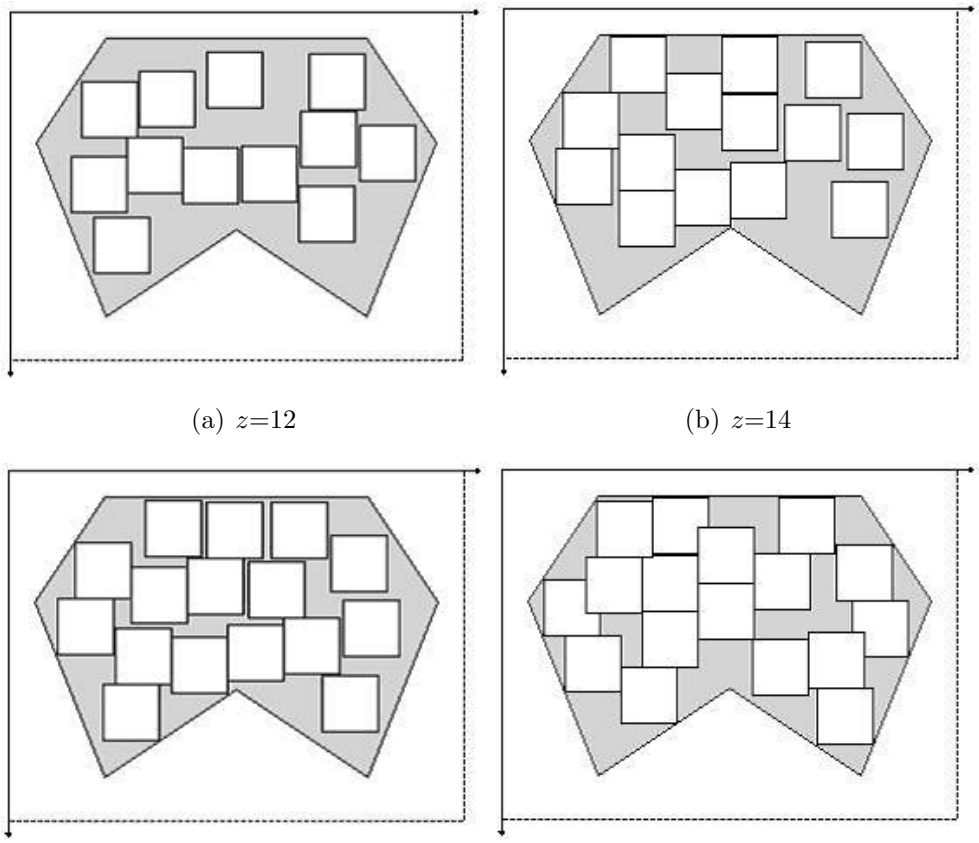


(b) SG-PSO

Figure 5.7: Maximum item numbers in Case B.

Table 5.5: Effect of parameter P_s in case B.

P_s	0.1	0.2	0.3	0.4	0.5	0.6
Average item number	14.1	13.83	13.97	13.76	13.37	13.39
Average CPU time (seconds)	66.57	77.00	81.80	85.56	95.18	108.65



(a) z=12 (b) z=14

(c) z=16 (d) z=17

Figure 5.8: Placement conditions of using SG-PSO in case A.

5.3.2 Case B

The packing region of case B is shown in Fig. 5.6. PSO parameters are identical to the case A (Table 5.1). Number of particles and maximum iteration steps are specified as $N = 200$ and $t_{\max} = 2000$, respectively. The other parameters are taken as $w = 0.9$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$, and $P_s = 0.1$.

The results are shown in Fig. 5.7 and Table 5.4. The average item numbers are 12.07 in case of the original PSO and 13.99 in SG-PSO. The average CPU times are 35.198 and 65.124, respectively. The success rates are 19.8% and 75.2%, respectively. The use of the SG-PSO can increase the item number and improve the success rate although the CPU time is increased.

Next, the effect of the parameter P_s on the convergence property is discussed. The maximum number of items and the CPU times for the different parameter P_s are listed in Table 5.5. The results show that, at $P_s = 0.1$, the item number is largest and CPU time is shortest.

5.4 Conclusion

PSO solution of the two-dimensional packing problem was presented in this study. Since the storage and the ship cabin are designed so that their sizes are equal to the integral multiple of the container sizes, it is assumed that the items are placed every certain interval. The author considered in this study that the packing region is arbitrarily polygon-shaped. The problem was solved by the original PSO and SG-PSO. In the original PSO, the particle position vectors are updated by the global and the personal best positions. The SG-PSO utilizes, in addition to them, the second global best position of all particles. The use of the second global best position is determined in the probabilistic way.

The algorithms were compared using two numerical examples. The design objective is to maximize the number of items contained in the packing region without the item overlap. The results showed that the SG-PSO algorithm could find better solutions than the original PSO. The maximum item number in the SG-PSO is bigger by one or two items than that in the original PSO. In case of the SG-PSO, the CPU time and the maximum item number depend on the probability to switch the original PSO and the PSO with second global best position of particles. Therefore, the adequate parameter design will be discussed for future work.

Chapter 6

Application to Truss Structure Design

6.1 Introduction

Structural optimization is an important field in engineering applications [57]. Structural optimization problems are mainly classified into size, shape and topology optimizations [42, 60]. In the size optimization problem, the size of the structural element is used as design variables. In the shape and topology optimization problems, the boundary profile and/or the topology of the structures are changed during the optimization process. In this chapter, the truss structure design is considered as a size optimization problem. The aim of this optimization problem is to minimize the weight of truss structure so as not to violate the stress and strain constraint conditions. This problem can be solved by several algorithms; the gradient-type algorithm such as Newton method and steepest descent method, the evolutionary algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Fourie and Groensold have firstly applied PSO to truss structure design [4]. They have focused on the application of the PSO to the size and shape design of truss structures. Since then, many PSO algorithm variants have been proposed to apply for truss design [18, 6, 28, 40, 59]. In this chapter, the PSO with second global and personal best particles are applied to the design of ten bar truss structure and the results are compared with the original PSO.

6.2 10-Bar Truss Structure Design

10-bar truss structure is considered as the problem. The geometry structure of 10-bar truss is shown in Fig. 6.1. The labels (1) to (6) denote the node numbers. The numbers by the bars denote the bar numbers. The load P_1 and P_2 are applied at the node (2) and

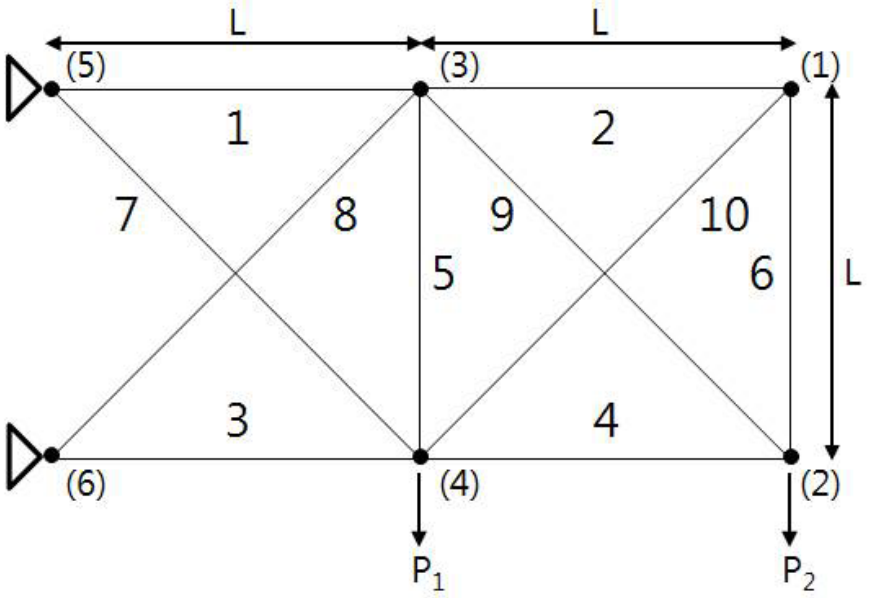


Figure 6.1: 10-bar truss structure.

(4).

The design objective is to minimize the structure weight. The objective function is given as

$$\min \boldsymbol{W} = \rho \sum_{i=0}^{10} L_i A_i \tag{6.1}$$

where ρ, A_i and L_i denote the material density, the cross-sectional area and the length of the bar i, respectively.

The stress at each bar should be smaller than the the allowable stress. The constraint condition is given as

$$\frac{\sigma_i}{\sigma_{allow}} - 1 \leq 0 \tag{6.2}$$

where σ_i and σ_{allow} denote the stress at the bar i and the allowable stress of the material, respectively.

The cross-sectional areas are taken as the design variables.

$$\{A_1, A_2, \cdots, A_{10}\} \tag{6.3}$$

The side constraint condition for the design variable A_i is given as

$$A_{\min} \leq A_i \leq A_{\max} \tag{6.4}$$

where $A_{\min}$ and $A_{\max}$ denote the upper and the lower bounds of the design variable, respectively.

The geometry parameters and material properties are shown in Table 6.1.

Table 6.1: Geometry parameters and material properties.

L	$360in$
P_1, P_2	$100kips$
σ_{allow}	$25ksi$
E(Young's modulus)	10^4ksi
ρ(Material density)	$0.1lb/in^3$
Swarm size	10
Max iteration number	3000

6.3 PSO Implementation

The fitness function is identical to the objective function.

$$Fitness = \rho \sum_{i=0}^{10} L_i A_i \rightarrow \min \tag{6.5}$$

The particle position vector $\boldsymbol{x}$ is defined as the set of the design variables.

$$\boldsymbol{x} = \{A_1, A_2, \cdots, A_{10}\} \tag{6.6}$$

The optimization process is summarized as follows.

1. The maximum iteration step t_{max}, swarm size N and the design conditions are specified.
2. The iteration number is initialized as $t = 0$.
3. The particle position vector $\boldsymbol{x}_i(t)$ and velocity vector $\boldsymbol{v}_i(t)$ are initialized with random numbers.
4. The fitness function for each particle $f(\boldsymbol{x}_i(t))$ is evaluated.
5. If $t \leq t_{max}$, the process goes to the next step. Otherwise, the process is terminaed.
6. Personal best particle position $\boldsymbol{x}^g(t)$ is updated.
7. Global best particle position $\boldsymbol{x}_i^p(t)$ is updated.
8. Second global best particle position $\boldsymbol{x}_i^{p2}(t)$ is updated.
9. The particle number i is initialized as $i = 1$.
10. If $i \leq N$, the process goes to the next step. Otherwise, the process goes to the step 14.
11. A random number r is generated in the range $[0, 1]$.
12. If $r \leq P_s$, in case of SG-PSO, the position vector $\boldsymbol{x}_i(t+1)$ and the velocity vector $\boldsymbol{v}_i(t+1)$ of the particle i are updated by Eqs. (2.1) and (3.1), respectively. In case of SP-PSO, the position vector and the velocity vector of the particle i are updated by Eqs. (2.1) and (3.2), respectively. If $r > P_s$, they are updated by the other rules.

13. $i = i + 1$, and the process goes to step 10.
14. $t = t + 1$, and the process goes to step 4.

6.4 Numerical Results

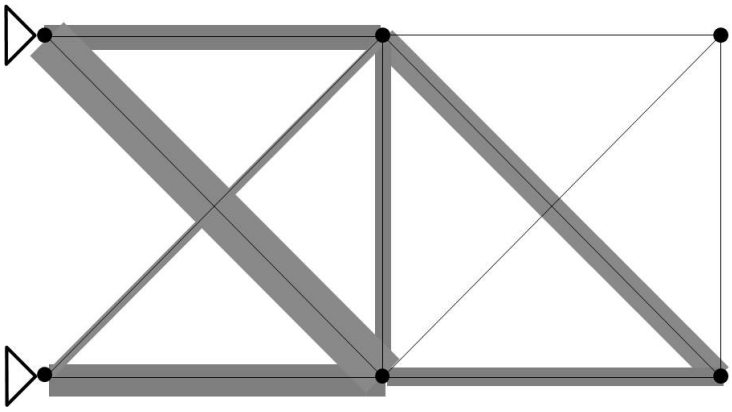
10-bar truss structures for the optimized using original PSO, SG-SPO and SP-PSO are shown in Fig 6.2. The detail results of the best search performance in 20 trials are shown in Table 6.2. The results are compared with cross-sectional area of each element and the weight for the optimized truss using original PSO, SG-SPO and SP-PSO. In addition, Table 6.2 shows the constraint satisfaction that the stress of each element should be less than the allowable stress. The final weights obtained by original PSO, SG-PSO and SP-PSO were 1621.09, 1566.89 and 1567.68, respectively. There were almost equal between SG-PSO and SP-PSO, however, for using original PSO optimized weight was worse than SG-PSO and SP-PSO.

Fig 6.3 shows a convergence history of the original PSO, SG-PSO and SP-PSO for 10 bar truss. The SG-PSO and SP-PSO showed a fast convergence to minimize weight in early iterations while the original PSO converged slowly. The SG-PSO slightly converged, in addition, more fast than the SP-PSO since 1000 iteration.

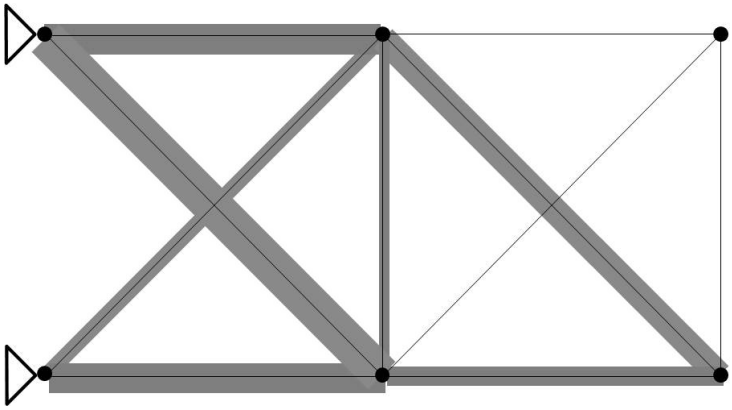
6.5 Conclusion

In this chapter 10-bar truss optimization with allowable stress constraint by using SG-PSO and SP-PSO was addressed. The obtained results for optimal weight of truss structure were compared with the original PSO. SG-PSO and SP-PSO can reduce more lightly than the original PSO, since optimal weights of 10 bar truss structure obtained using the SG-PSO and SP-PSO are better solutions than obtained using the original PSO. In addition, the SG-SPO and SP-PSO converged to near optimal earlier than original PSO.

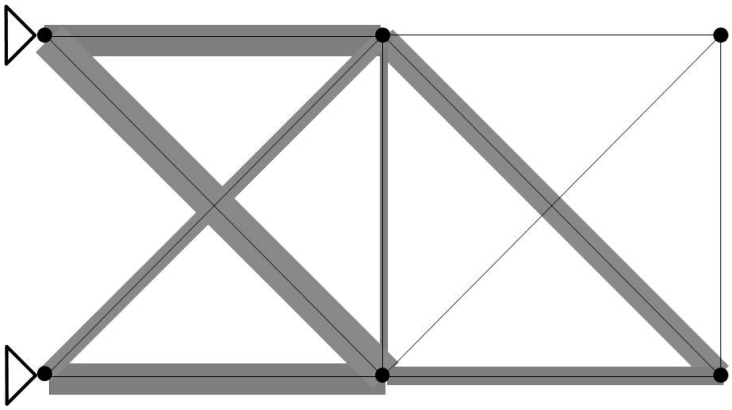
In this study, acceleration coefficient parameters of SG-PSO and SP-PSO were not considered as truss optimization, and thus future work is to implement and investigate the parameters of SG-PSO and SP-PSO on truss design. In addition, the extension of the design variable is necessary in order to effectiveness of SG-PSO and SP-PSO in complex truss design.



(a) 10 bar Truss optimized in the original PSO



(b) 10 bar Truss optimized in the SG-PSO



(c) 10 bar Truss optimized in the SP-PSO

Figure 6.2: 10 bar truss optimized in the original PSO, SG-PSO and SP-PSO.

Table 6.2: Cross-sectional area and constraint satisfaction.

A_{num}	Original PSO	SG-PSO	SP-PSO
A_1	5.378328	6.413477	6.585532
A_2	0.11	0.1	0.1
A_3	6.677631	6.386833	6.63111
A_4	3.945894	4.012937	3.931209
A_5	3.425103	2.205183	1.893459
A_6	0.1	0.1	0.1
A_7	10.32967	8.291609	7.801332
A_8	1.948008	3.21489	3.633728
A_9	5.585255	5.580647	5.651379
A_{10}	0.1	0.1	0.1
$\frac{\sigma_{num}}{\sigma_{allow}}$	Original PSO	SG-PSO	SP-PSO
$\frac{\sigma_1}{\sigma_{allow}}$	1.00	0.98	0.99
$\frac{\sigma_2}{\sigma_{allow}}$	0.54	0.60	0.99
$\frac{\sigma_3}{\sigma_{allow}}$	0.81	0.98	0.98
$\frac{\sigma_4}{\sigma_{allow}}$	1.00	0.98	1.00
$\frac{\sigma_5}{\sigma_{allow}}$	0.75	0.76	0.76
$\frac{\sigma_6}{\sigma_{allow}}$	0.54	0.60	0.69
$\frac{\sigma_7}{\sigma_{allow}}$	0.91	0.98	1.00
$\frac{\sigma_8}{\sigma_{allow}}$	1.00	1.00	0.97
$\frac{\sigma_9}{\sigma_{allow}}$	0.33	0.33	0.33
$\frac{\sigma_{10}}{\sigma_{allow}}$	0.77	0.84	0.97
Weight	1621.09	1566.89	1567.68

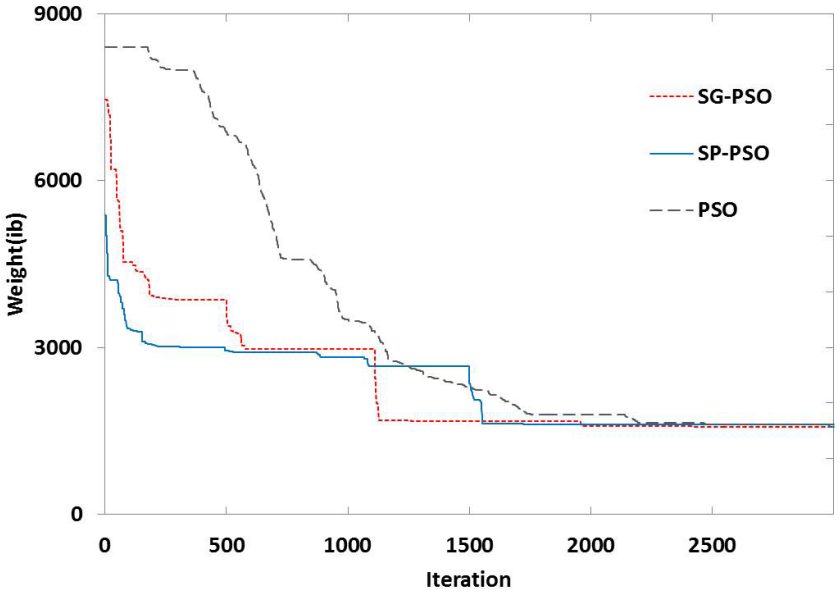


Figure 6.3: Convergence history of the original PSO, SG-PSO and SP-PSO.

Chapter 7

Conclusion

In this thesis, a new PSO algorithm was studied to improve the search performance of the original PSO. In the original PSO, the most notable scheme is that the particle positions are updated from the personal and global best particle positions that the particles have ever found. Unlike this scheme, two PSO algorithms were introduced in this study. The proposed PSO algorithms employ the second global best and the second personal best particle positions in order to improve the search performance of the original PSO. In the first proposed PSO algorithm, the update rules with second global best positions are randomly used. This algorithm is referred as SG-PSO. In the second proposed PSO algorithm, the update rule with second personal best positions are randomly used. This algorithm is referred as SP-PSO. It is considered that the use of the second best position is effective in avoiding converging to the local optimum.

In this thesis, a general introduction and the PSO algorithm were described in chapter 1 and 2. In chapter 3, the SG-PSO and SP-PSO algorithms were described in detail, and the effect of the acceleration coefficient parameters on SG-PSO and SP-PSO was discussed through five test functions. Suitable parameters for all the test functions were obtained. The optimal parameters were used for the benchmark functions in chapter 4. The trajectory of the global best particle position was illustrated to observe its motion. While in the original PSO, the global best particle is attracted to the local optimum, SG-PSO and SP-PSO avoid the local optimum, and they finally find the global optimum.

In chapter 4, the applicability of SG-PSO and SP-PSO was discussed in various numerical examples. The SG-PSO and SP-PSO were compared with the other PSO variants using 11 benchmark functions. The other PSO variants were described in chapter 2. The search performance comparison was discussed in terms of the average error values and the convergence histories. The SP-PSO algorithm showed the best search performance in 10 benchmark functions except for the Shifted Rotated Weierstrass function. In the case of the Shifted Rotated Weierstrass function, the SG-PSO and SP-PSO algorithms were the first and the second best ones, respectively. Regarding the convergence histories, the

author observed that SP-PSO more rapidly converged to zero than other PSO variants. This is related to the trajectory of the global best particle position, mentioned in chapter 3. The SP-PSO algorithm maintains the diversity of position of the particles, while other PSO algorithms tend to reach the local optimal solution rapidly.

In chapter 5, the two-dimensional packing problem was addressed. The packing region was a two-dimensional arbitrarily polygon-shaped and then, rectangular items were packed in it. The algorithms were compared with the original PSO in two different shape regions. The objective of this problem was to maximize the number of contained items in the packing region without item overlap. The results showed that the SG-PSO algorithm found better solutions than the original PSO. The maximum item number in the SG-PSO was bigger by one or two items than that in the original PSO.

In chapter 6, the weight optimization of the 10-bar truss structure with allowable stress constraint was addressed. The numerical results were compared with the original PSO. The structures optimized by SG-PSO and SP-PSO were slightly lighter than that optimized by the original PSO; thus, SG-PSO and SP-PSO could find better solutions than the original PSO.

From the above results, the author would like to discuss the features of the present PSO algorithms. The results in chapter 4 showed that SP-PSO was better than the other PSO algorithms and the SG-PSO. Since the SP-PSO algorithm gives a different additional weight to the cognitive component of the update rule for each particle, it can enhance the local search performance of the original PSO. Therefore, it showed the good search performance for test functions with multiple local optima. When the search space is simple, on the other hand, particles need to search around the global optimum. In the SG-PSO algorithm, the particles search for the optimal solution around the global and second global best position particles. Therefore, the SG-PSO algorithm is applicable to the problems with a simple search space. For higher dimensional functions, a performance comparison is required in the next step. In addition, this study considered the single objective functions, so use of the multi-objective functions remains a future topic.

PSO is a stochastic optimization method, and the search performance depends on the parameter values. In the case of SG-PSO, when the difference in distance between the global best position and the second global best position is small, the second global best position has a small influence on each particle. Therefore, the study of varying parameters using the difference in distance between the best and the second best positions is an important future research.

In this thesis, the author applied SG-PSO and SP-PSO to the packing problem and truss structure design as application problems. Since the optimal parameters of SG-PSO and SP-PSO were not considered for truss optimization, the implementation and investigation of optimal parameters for truss design will be discussed in the next research

work. The extension of the design variable is necessary to evaluate the effectiveness of SG-PSO and SP-PSO for complex problems. In the near future, the author will study the applicability of SG-PSO and SP-PSO to other interesting engineering applications.

Reference

[1] Jaszkiewicz A. Multiple objective metaheuristic algorithms for combinatorial optimization. Citeseer, 2001.

[2] Liu B., Wang L., Jin Y. H., Tang F., and Huang D. X. Improved particle swarm optimization combined with chaos. *Chaos Solitons and Fractals*, Vol. 25, pp. 1261–1271, 2005.

[3] Melissen J. B. and Schuur P. C. Packing 16, 17 or 18 circles in an equilateral triangle. *Discrete Mathematics*, Vol. 145, pp. 333–342. Elsevier, 1995.

[4] Fourie P. C. and Groenwold A. A. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, Vol. 23, pp. 259–267. Springer, 2002.

[5] He C., Zhang Y. B., Wu J. W., and Chang C. Research of three-dimensional container-packing problems based on discrete particle swarm optimization algorithm. *Test and Measurement, 2009. ICTM'09. International Conference on*, Vol. 2, pp. 425–428. IEEE, 2009.

[6] Luh G. C., Lin C. Y., and Lin Y. S. A binary particle swarm optimization for continuum structural topology optimization. *Applied Soft Computing*, Vol. 11, pp. 2833–2844. Elsevier, 2011.

[7] Ratnaweera A. C., Halgamuge S. K., and Watson H. C. Particle swarm optimiser with time varying acceleration coefficients. *Proceedings of the International Conference on Soft Computing and Intelligent Systems*, pp. 240–255, 2002.

[8] Ratnaweera A. C., Halgamuge S. K., and Watson H. C. Particle swarm optimization with self-adaptive acceleration coefficients. *Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 264–268, 2003.

[9] Trelea I. C. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, Vol. 85, pp. 317–325. Elsevier, 2003.

[10] Zhao C., Lin L., Hao C., and Xinbao L. Solving the rectangular packing problem of the discrete particle swarm algorithm. *Business and Information Management, 2008. ISBIM'08. International Seminar on*, Vol. 2, pp. 26–29. IEEE, 2008.

[11] Henderson D., Jacobson S. H., and Johnson A. W. The theory and practice of simulated annealing. *Handbook of metaheuristics*, pp. 287–319. Springer, 2003.

[12] Zhang D., Kang Y., and Deng A. A new heuristic recursive algorithm for the strip rectangular packing problem. *Computers & Operations Research*, Vol. 33, pp. 2209–2217. Elsevier, 2006.

[13] Van den Bergh F. and Engelbrecht A. P. Effect of swarm size on cooperative particle swarm optimisers. *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 892–899, 2001.

[14] Aarts E. and Lenstra J. K. Local search in combinatorial optimization, 1997. *Wiley, Chichester [2] L. Cox, L. Ernst, Controlled Rounding, INFOR*, Vol. 20, pp. 423–432.

[15] Friedman E. Packing unit squares in squares: A survey and new results. Citeseer, 2002.

[16] Goldberg D. E. and Linge R. L. Alleles, loci and the traveling salesman problem. *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Vol. 3, pp. 154–1959, 1985.

[17] Parsopoulos K. E. and Vrahatis M. N. Upso: A unified particle swarm optimization scheme. *Lecture series on computer and computational sciences*, Vol. 1, pp. 868–873, 2004.

[18] Perez R. E. and Behdinan K. Particle swarm approach for structural design optimization. *Computers and Structures*, Vol. 85, pp. 1579–1588. Elsevier, 2007.

[19] Glover F. and Kochenberger G. A. Handbook of metaheuristics. Springer, 2003.

[20] Juang C. F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, Vol. 34, pp. 997–1006. IEEE, 2004.

[21] Venter G. and Sobieszczanski-Sobieski J. Particle swarm optimization. *AIAA*, Vol. 41, pp. 1583–1589, 2003.

[22] Shi X. H., Liang Y. C., Lee H. P., HP, Lu C., and Wang L. M. An improved ga and a novel pso-ga-based hybrid algorithm. *Information Processing Letters*, Vol. 93, pp. 255–261. Elsevier, 2005.

[23] Zhan Z. H., Zhang J., Li Y., and et al. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B:Cybernetics*, Vol. 39, pp. 1362–1381, 2009.

[24] Holand and John H. Adaptation in natural and artificial systems. *Ann Arbor: The University of Michigan Press*, 1975.

[25] Angeline P. J. Using selection to improve particle swarm optimization. *Proceedings of IEEE International Conference on Evolutionary Computation*, Vol. 89, 1998.

[26] Kennedy J. and Eberhart R. Particle swarm optimization. *Proceedings of IEEE the International Conference on Neural Networks*, 1995. 1942–1948.

[27] Kennedy J. and Mendes R. Population structure and particle swarm performance. *Computational Intelligence, Proceedings of the World on Congress on*, Vol. 2, pp. 1671–1676. IEEE, 2002.

[28] Li L. J., Huang Z. B., Liu F., and Wu Q. H. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures*, Vol. 85, pp. 340–349. Elsevier, 2007.

[29] Liang J. J., Qin A. K., Suganthan P. N., and Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, Vol. 10, pp. 281–295. IEEE, 2006.

[30] Jansen K. and Solis-Oba R. A polynomial time approximation scheme for the square packing problem. *Integer Programming and Combinatorial Optimization*, pp. 184–198. Springer, 2008.

[31] Mishra S. K. Global optimization by differential evolution and particle swarm methods: Evaluation on some benchmark functions. 2006.

[32] Tang K., Yáo X., Suganthan P. N., MacNish C., Chen Y. P., Chen C. M., and Yang Z. Benchmark functions for the cec ' 2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007.

[33] Davis L. Applying adaptive algorithms to epistatic domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 162–164, 1985.

[34] Davis L., Lawrence, and et al. Handbook of genetic algorithms. Van Nostrand Reinhold, New York, 1991.

[35] Morten L., Thomas K. R., and Thiemo K. Hybrid particle swarm optimiser with breeding and subpopulations. *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 1, pp. 469–476. Citeseer, 2001.

[36] Lü Z. and Huang W. Perm for solving circle packing problem. *Computers and Operations Research*, Vol. 35, pp. 1742–1755. Elsevier, 2008.

[37] Clerc M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, Vol. 3. IEEE, 1999.

[38] Clerc M. Think locally, act locally: The way of life of cheap-pso, an adaptive pso. 2001.

[39] Clerc M. and Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, Vol. 6, pp. 58–73. IEEE, 2002.

[40] Gomes H. M. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Systems with Applications*, Vol. 38, pp. 957–968. Elsevier, 2011.

[41] Molga M. and Smutnicki C. Test functions for optimization needs. *Test functions for optimization needs*, 2005.

[42] Xie Y. M. and Steven G. P. A simple evolutionary procedure for structural optimization. *Computers and structures*, Vol. 49, pp. 885–896. Elsevier, 1993.

[43] Yagiura M. and Ibaraki T. On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan*, Vol. 32, pp. 33–55. Citeseer, 2001.

[44] Metropolis N., Arianna W. R., Marshall N. R., Augusta H. T., and Edward T. Equation of state calculations by fast computing machines. *The Journal of chemical physics*, Vol. 21, pp. 1087–1092. AIP Publishing, 1953.

[45] Suganthan P. N. Particle swarm optimiser with neighborhood operator. *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. IEEE Press, pp. 1958–1962, 1999.

[46] Suganthan P. N., Hansen N., Liang J. J., Deb K., Chen Y. P., Auger A., and Tiwari S. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL Report*, 2005.

[47] Thapatsuwan P., Sepsirisuk J., Chainate W., and Pongcharoen P. Modifying particle swarm optimisation and genetic algorithm for solving multiple container packing problems. *Computer and Automation Engineering, 2009. ICCAE'09. International Conference on*, pp. 137–141. IEEE, 2009.

[48] Baraglia R., Hidalgo J. I., and Perego R. A hybrid heuristic for the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, Vol. 5, pp. 613–622. IEEE, 2001.

[49] Collins C. R. and Stephenson K. A circle packing algorithm. *Computational Geometry*, Vol. 25, pp. 233–256. Elsevier, 2003.

[50] Eberhart R. and Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, Vol. 1, pp. 84–88. IEEE, 2000.

[51] Forbes R. and Nayeem M. T. Particle swarm optimization on multi-funnel functions. *Computer Aided Optimum Design in Engineering XII*, Vol. 255.

[52] Baskar S., Alphones A., Suganthan P. N., and Liang J. J. Design of yagi–uda antennas using comprehensive learning particle swarm optimisation. *IEE Proceedings-Microwaves, Antennas and Propagation*, Vol. 152, pp. 340–346. IET, 2005.

[53] Kirkpatrick S., Gelatt Jr. C. D., and Vecchi M. P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[54] Liu D. S., Tan K. C., Huang S. Y., Goh C. K., and Ho W. K. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, Vol. 190, pp. 357–382. Elsevier, 2008.

[55] Yang X. S. Nature-inspired metaheuristic algorithms. Luniver press, 2010.

[56] Croft H. T., Falconer K. J., and Guy R. K. Unsolved problems in geometry. Springer, 1991.

[57] Haftka R. T. and Gurdal Z. Elements of structural optimization. Vol. 11. Springer, 1992.

[58] Leung J. Y. T., Tam T. W., Wong C. S., Young G. H., and Chin F. Y. L. Packing squares into a square. *Journal of Parallel and Distributed Computing*, Vol. 10, pp. 271–275. Elsevier, 1990.

[59] Camp C. V., Meyer B. J., and Palazolo P. J. Particle swarm optimization for the design of trusses. *Proceeding of 2004 ASCE Structures Congress*, 2004.

[60] Toropov V. V. Simulation approach to structural optimization. *Structural Optimization*, Vol. 1, pp. 37–46. Springer, 1989.

[61] Jiang Y., Hu T., Huang C., and Wu X. An improved particle swarm optimization algorithm. *Applied Mathematics and Computation*, Vol. 193, pp. 231–239. Elsevier, 2007.

[62] Shi Y. and Eberhart R. A modified particle swarm optimizer. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73. IEEE, 1998.

[63] Shi Y. and Eberhart R. Parameter selection in particle swarm optimizer. *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 591–600, 1998.