Computer Assisted Methods in Engineering and Science, **19**: 241–255, 2012.

Solving two-dimensional packing problem using particle swarm optimization

Young-Bin Shin, Eisuke Kita
Graduate School of Information Science, Nagoya University
Japan
e-mail: kita@is.nagoya-u.ac.jp

Particle swarm optimization is one of the evolutionary computations which is inspired by social behavior of bird flocking or fish schooling. This research focuses on the application of the particle swarm optimization to two-dimensional packing problem. Packing problem is a class of optimization problems which involve attempting to pack the items together inside a container, as densely as possible. In this study, when the arbitrary polygon-shaped packing region is given, the total number of items in the region is maximized. The optimization problem is defined not as the discrete-value optimization problem but as the continuous-value optimization problem. The problem is solved by two algorithms, original and improved PSOs. In the original PSO, the particle position vector is updated by the best particle position in all particles (global best particle position) and the best position in previous positions of each particle (personal best position). The improved PSO utilizes, in addition to them, the second best particle position in all particles (global second best particle position) in the stochastic way. In the numerical example, the algorithms are applied to three problems. The results show that the improved PSO can pack more items than the original PSO and therefore, number of the successful simulations is also improved.

Keywords: packing problem, particle swarm optimization, global best position, global second best position, personal best position.

1. Introduction

For solving nondeterministic polynomial time (NP)-hard problems, many researchers have been studying several evolutionary computations such as genetic algorithm (GA) [1], simulated annealing (SA) [2], particle swarm optimization (PSO) [3] and so on. This study focuses on the application of PSO to two-dimensional packing problems, which is one of the popular NP-hard problems. PSO, which has been presented in 1995 by Kennedy and Eberhart [3], is based on a metaphor of social interaction such as bird flocking and fish schooling. PSO is a population-based optimization algorithm, which could be implemented and applied easily to solve various function optimizations problem, or the problems that can be transformed to the function minimization or maximization problem.

Two-dimensional packing problems are a class of optimization problems in mathematics which involve attempting to pack objects together in the packing region as densely as possible. There are many variations of this problem, such as two-dimensional packing, linear packing, packing by weight, packing by cost, and so on. 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.

This study focuses on two-dimensional packing problems. Two-dimensional packing problems are traditionally defined as the optimization problems that pack definite-shaped (circle or square) items in a definite-shaped (circle or square) packing regions. Therefore, analytical solutions can be obtained in some cases [4–6]. However, in some actual industrial applications, it is not assumed

that the packing regions have definite shapes such as circle or square. When the packing regions do not have the definite shapes, the solutions should be obtained by numerical way. In this study, the packing problem is defined as the maximization of the items in the two-dimensional packing regions without their overlap when the packing regions with arbitrary polygon-shapes are given. The problems are solved by using particle swarm optimization.

The application of PSO for solving packing problem has been presented by some researchers [7–10]. Liu et al. [7] presented evolutionary PSO for solving bin packing problem. Zhao et al. [8, 9] applied the discrete PSO for solving rectangular packing problem. Thapatsuwan et al. [10] 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 sizes of the storage and the ship cabin are designed in conformity to the container sizes, the optimization problem can be defined as the discrete-valued problem and therefore, the special PSOs such as discrete PSO are applied. In this study, arbitrarily polygon-shaped packing regions are assumed. Therefore, the problems are defined as the continuous-valued optimization problems.

The design objective function is to maximize the total number of the items packed into the packing region without their overlap. The total number of items and the position vectors of the item centers are used as the design variables. The problem is solved by the original and the improved PSOs. In the PSO, the potential 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 best position of all particles (global best position) and the local best position in previous positions of each particle (personal best position). The improved PSO utilizes, in addition to them, the second best position of all particles (global second best position). The use of the global second best particle position has been already presented in [11]. Its numerical discussions and applications were not described in this reference. Besides, the stochastic use of the global second-best particle position $\boldsymbol{x}^{g2}$ is presented in this study.

The remaining part of this paper is organized as follows. The PSO algorithms and the optimization problem are explained in Sec. 2 and 3, respectively. In Sec. 4, the packing problem in two-dimensional regions is solved. Finally, the conclusions are summarized again in Sec. 5.

2. PSO ALGORITHM

2.1. Original PSO

2.1.1. Update rule

In the PSO algorithm, the particles represent potential solutions of the optimization problem. Each particle in the swarm has a position vector $\boldsymbol{x}_i(t)$ $(i = 1, 2, \ldots, N)$ and a velocity vector $\boldsymbol{v}_i(t)$ in the search space at time t. The parameter N denotes the total number of particles in the swarm. The particle position vector is defined by the design variable set for the optimization problem. Each particle also has memory and hence, can remember the best position in search space it has ever visited. The satisfaction of the particle i for the design objective is estimated by the objective function or the fitness function $f(\boldsymbol{x}_i(t))$.

The position at which each particle takes the best fitness function is known as the personal best position $\boldsymbol{x}_i^p(t)$ and the overall best out of all particles in the swarm is a global best position $\boldsymbol{x}^g(t)$. Then its velocity and position are updated according to the following formulas:

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

$$\boldsymbol{v}_i(t+1) = \omega \cdot \boldsymbol{v}_i(t) + c_1 \cdot r_1 \times (\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 \cdot r_2 \times (\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)), \tag{2}$$

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

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

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \times \frac{t}{t_{\max}}, \tag{3}$$

where the parameter $\omega_{\max}$ and $\omega_{\min}$ 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 parameters c_1 and c_2 denote the effect of $\boldsymbol{x}_i^p(t)$ and $\boldsymbol{x}^g(t)$. According to the recent work done by Clerc [12], the parameters are given as

$$c_1 = c_2 = 1.5. \tag{4}$$

2.1.2. Algorithm

Total number of the particles in the swarm is fixed as N. The original PSO algorithm is shown in Fig. 1 and summarized as follows.

1. Set $t = 0$.
2. For $i = 1, \cdots, N$, set $\boldsymbol{x}_i^p(t) = \mathbf{0}$.
3. Initialize the position vector $\boldsymbol{x}_i(t)$ and the velocity vector $\boldsymbol{v}_i(t)$ of each particle with random numbers.
4. Set $t = t + 1$.

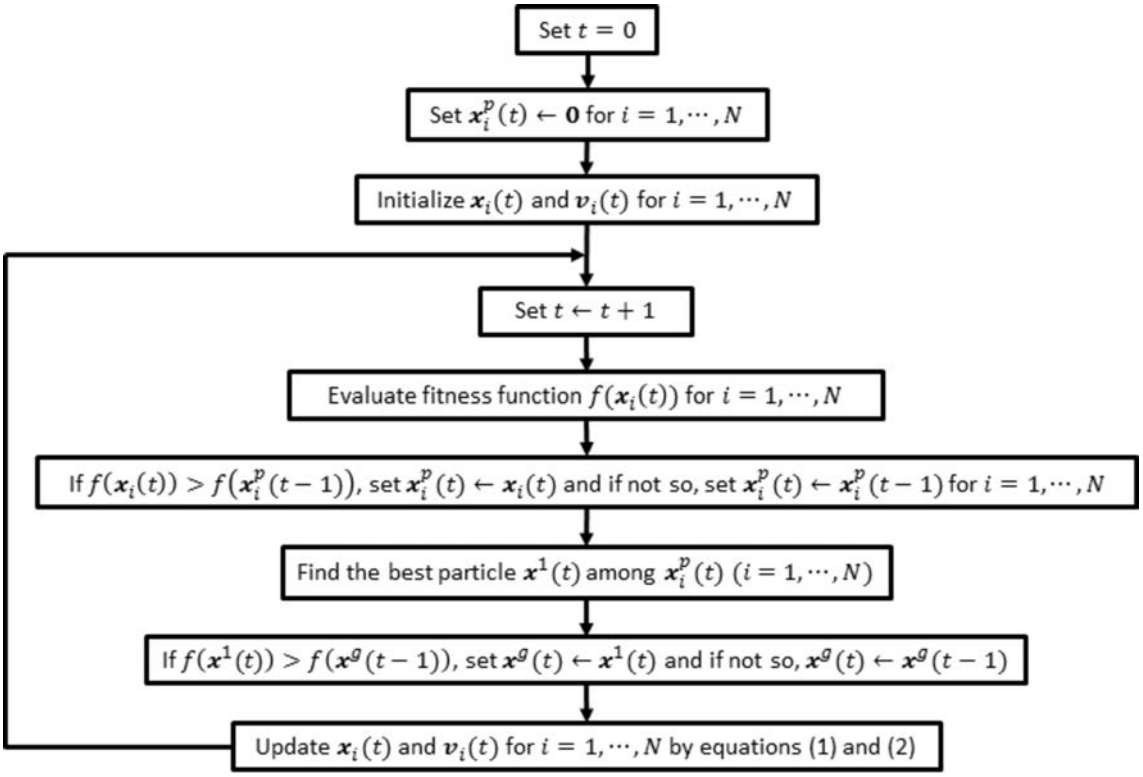

Fig. 1. Flowchart of original PSO algorithm.

5. For $i = 1, \cdots, N$,
   (a) Evaluate fitness functions $f(\boldsymbol{x}_i(t))$ for each particle.
   (b) If $f(\boldsymbol{x}_i(t)) > f(\boldsymbol{x}_i^p(t-1))$, set $\boldsymbol{x}_i^p(t) = \boldsymbol{x}_i(t)$.
   (c) If $f(\boldsymbol{x}_i(t)) \leq f(\boldsymbol{x}_i^p(t-1))$, set $\boldsymbol{x}_i^p(t) = \boldsymbol{x}_i^p(t-1)$.
6. Find the best particle position $\boldsymbol{x}^1(t)$ among $\boldsymbol{x}_i^p(t)$ $(i = 1, 2, \cdots, N)$.
7. If $f(\boldsymbol{x}^1(t)) > f(\boldsymbol{x}^g(t-1))$, set $\boldsymbol{x}^g(t) = \boldsymbol{x}^1(t)$.
8. If $f(\boldsymbol{x}^1(t)) \leq f(\boldsymbol{x}^g(t-1))$, set $\boldsymbol{x}^g(t) = \boldsymbol{x}^g(t-1)$.
9. For $i = 1, \cdots, N$, update the velocity vector $\boldsymbol{v}_i(t)$ and position vector $\boldsymbol{x}_i(t)$ of each particle according to Eqs. (2) and (1), respectively.
10. Go to step 5 if $t \leq t_{\max}$.

2.2. Improved PSO

2.2.1. Update rule with global second-best position

The original PSO has no handling mechanism for the local optimization. The original PSO reduces the chance of local optimization to make use of $\boldsymbol{x}_i^p(t)$ in one way. In the improved PSO, in addition to the global best position $\boldsymbol{x}^g(t)$ and the personal best position $\boldsymbol{x}^p(t)$, each particle can remember the position of the global second-best particle $\boldsymbol{x}^{g2}(t)$. The use of $\boldsymbol{x}^{g2}(t)$ can reduce the chance of local optimization of PSO. In the improved PSO, the position vector and the velocity vector of the particle are updated according to Eq. (1) and the following formula, respectively.

$$\boldsymbol{v}_i(t+1) = w \cdot \boldsymbol{v}_i(t) + c_1 \cdot r_1 \times (\boldsymbol{x}_i^p(t) - \boldsymbol{x}_i(t)) + c_2 \cdot r_2 \times (\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)) + c_3 \cdot r_3 \times (\boldsymbol{x}^{g2}(t) - \boldsymbol{x}_i(t)), \quad (5)$$

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; $c_1 = c_2 = 1.5$. The parameter c_3 is determined from some numerical experiments, which is given as $c_3 = 1.9$.

The update rule (5) has been already presented in [11]. Its numerical discussions and applications were not described in this reference. The stochastic use of the global second-best particle position $\boldsymbol{x}^{g2}(t)$ is presented in this study.

2.2.2. Algorithm

The improved PSO shares the information of $\boldsymbol{x}_i^p(t)$, $\boldsymbol{x}^g(t)$ and $\boldsymbol{x}^{g2}(t)$. Obviously, $\boldsymbol{x}^{g2}(t)$ is worse than $\boldsymbol{x}^g(t)$. If only Eq. (5) is used for updating particle velocities, the obtained result is similar to worse than that of original PSO. Therefore, both update rules of the original and improved PSOs are used in the present algorithm. Here, we present the following update rule in which the PSO of Eq. (2) and the PSO of Eq. (5) are randomly used in probability P_s. The improved PSO algorithm is shown in Fig. 2 and summarized as follows.

1. Set $t = 0$.
2. For $i = 1, \cdots, N$, set $\boldsymbol{x}_i^p(t) = \boldsymbol{0}$ and $\boldsymbol{x}^g(t) = \boldsymbol{0}$.
3. Initialize the position vector $\boldsymbol{x}_i(t)$ and the velocity vector $\boldsymbol{v}_i(t)$ of each particle with random numbers.
4. Set $t = t + 1$.

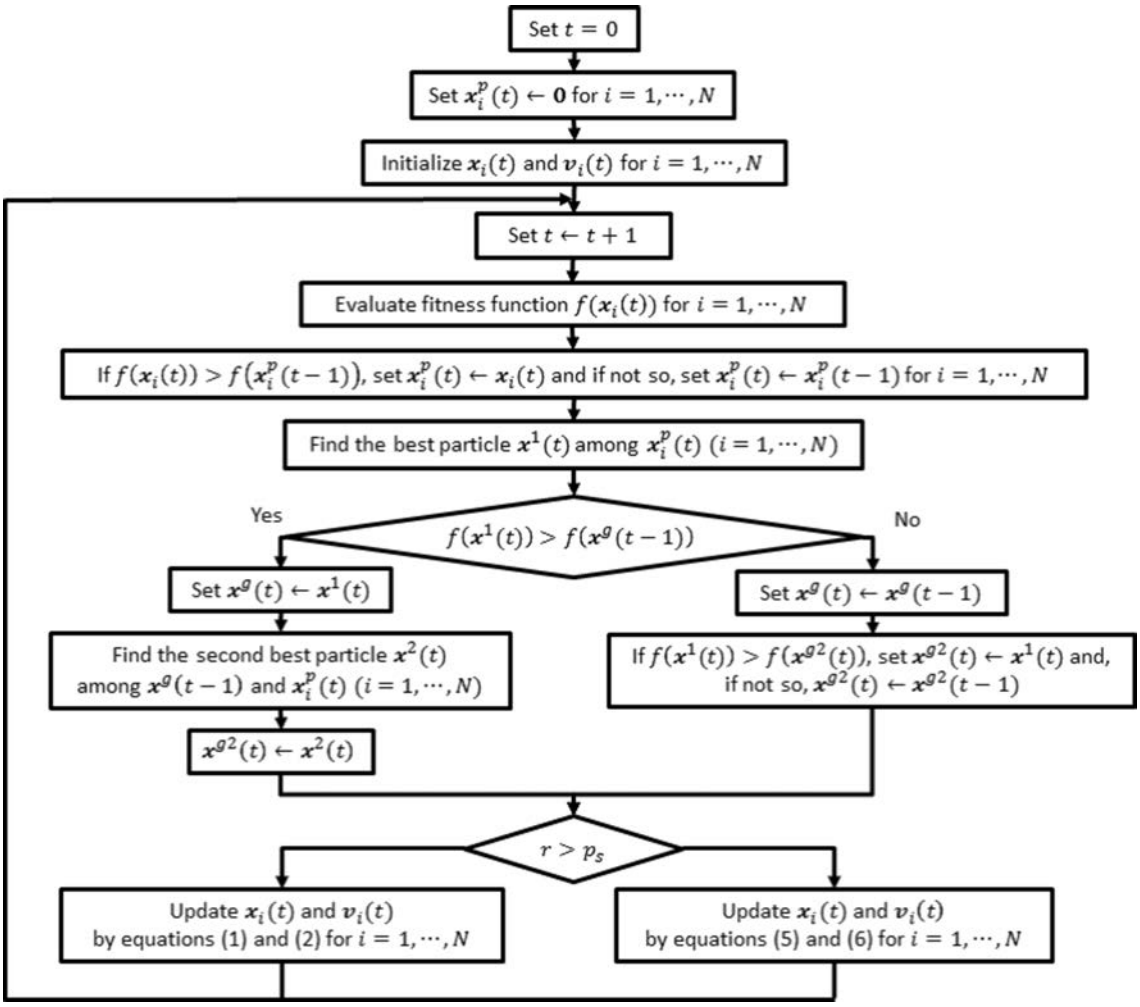

Fig. 2. Flowchart of improved PSO algorithm.

5. For $i = 1, 2, \cdots, N$:
   (a) Evaluate fitness functions $f(\boldsymbol{x}_i(t))$ for each particle.
   (b) If $f(\boldsymbol{x}_i(t)) > f(\boldsymbol{x}_i^p(t-1))$, set $\boldsymbol{x}_i^p(t) = \boldsymbol{x}_i(t)$.
   (c) If $f(\boldsymbol{x}_i(t)) \leq f(\boldsymbol{x}_i^p(t-1))$, set $\boldsymbol{x}_i^p(t) = \boldsymbol{x}_i^p(t-1)$.
6. Find the best particle position $\boldsymbol{x}^1(t)$ among $\boldsymbol{x}_i^p(t)$ $(i = 1, 2, \ldots, N)$.
7. If $f(\boldsymbol{x}^1(t)) > f(\boldsymbol{x}^g(t-1))$, go to next step. Otherwise, go to step 12.
8. Set $\boldsymbol{x}^g(t) = \boldsymbol{x}^1(t)$.
9. Find the second best particle position $\boldsymbol{x}^2(t)$ among $\boldsymbol{x}^g(t-1)$ and $\boldsymbol{x}_i^p(t)$ $(i = 1, 2, \ldots, N)$.
10. Set $\boldsymbol{x}^{g2}(t) = \boldsymbol{x}^2(t)$.
11. Go to step 14.
12. If $f(\boldsymbol{x}^1(t)) > f(\boldsymbol{x}^{g2}(t))$, set $\boldsymbol{x}^{g2}(t) = \boldsymbol{x}^1(t)$.
13. If $f(\boldsymbol{x}^1(t)) \leq f(\boldsymbol{x}^{g2}(t))$, set $\boldsymbol{x}^{g2}(t) = \boldsymbol{x}^{g2}(t-1)$.
14. Generate random number r in the interval$[0, 1]$.

15. If $r > P_s$, update the velocity and position vectors of all particles according to Eqs. (1) and (2), respectively.

16. If $r \leq P_s$, update the velocity and position vectors of all particles according to Eqs. (1) and (5), respectively.

17. Go to step 5 if $t \leq t_{\max}$.

3. Packing problem

3.1. Optimization problem

The packing problem can be formulated to maximize the number of items N compacted into a two-dimensional polygonal region P as follows:

$$\begin{aligned} &\max \quad z \\ &\text{s.t.} \quad g_1(i,P) = 0, \\ &\qquad g_2(i,j) = 0, \\ &\qquad 0.5w \leq p_x^i \leq W - 0.5w, \\ &\qquad 0.5h \leq p_y^i \leq H - 0.5h, \\ &\qquad i = 1,2,\ldots,z; \quad j = 1,2,\ldots,z, \end{aligned} \tag{6}$$

where the vector $\{p_x^i, p_y^i\}$ denotes the center position vector of the item i, w and h are item sizes, and W and H are feasible space sizes. The function $g_1(i,P)$ estimates the inclusion of the item i in the region P, which is defined as follows:

$$g_1(i,P) = \begin{cases} 0 & \text{The item } i \text{ is included in the region } P. \\ 1 & \text{The item } i \text{ is not included in the region } P. \end{cases} \tag{7}$$

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

$$g_2(i,j) = \begin{cases} 0 & \text{The item } i \text{ and } j \text{ are not overlapped.} \\ 1 & \text{The item } i \text{ and } j \text{ are overlapped.} \end{cases} \tag{8}$$

3.2. PSO implementation

When the number of 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

$$f(\boldsymbol{x}_i) = \frac{1}{1 + \sum_{i=1}^{z} \left\{ g_1(i,P) + \sum_{j=1}^{z, i \neq j} g_2(i,j) \right\}}; \tag{9}$$

- design variable vector

$$\boldsymbol{x}_i = \{p_x^1, p_y^1, \cdots, p_x^i, p_y^i, \cdots, p_x^z, p_y^z\}^T; \tag{10}$$

- side constraints for design variable

$$\begin{aligned} &0.5w \leq p_x^i \leq W - 0.5w \quad (i = 1,2,\cdots,z); \\ &0.5h \leq p_y^i \leq H - 0.5h \quad (i = 1,2,\cdots,z). \end{aligned} \tag{11}$$

3.3. Optimization process

The process of the present algorithm is summarized as follows.

1. Set the threshold P_s and the maximum step size $t_{\max}$.
2. Set $z = 0$.
3. Set $z = z + 1$.
4. Apply PSO algorithm.
5. If $f(\boldsymbol{x}^g(t)) = 1$, return to Step 3. If not so, go to next step.
6. Stop by $z = z - 1$.

4. Numerical examples

4.1. 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. 3. PSO parameters are shown in Table 1. Number of particles and maximum iteration steps are specified as $N = 200$ and $t_{\max} = 2000$, respectively. The other parameters are taken as $\omega = 0.9$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$, and $P_s = 0.1$.

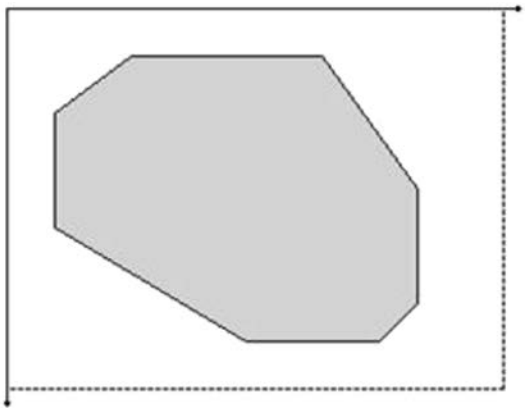

Fig. 3. Packing region in case A.

Table 1. Parameters.

Number of particles	$N = 200$
Maximum iteration step	$t_{\max} = 2000$
Update rules parameters	$\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$

Five hundred simulations are performed from different initial conditions. Maximum item numbers for case A are shown in Fig. 4. The figures are plotted with the run number as the horizontal axis and the item number z as the vertical axis, respectively.

The results by the original and the improved PSOs are compared in Table 2. The average maximum item number and the average CPU time denote the average values of the maximum item numbers and CPU time 500 simulations, respectively. The average maximum item number is 11.144 in case of the original PSO and 12.984 in improved PSO. The average CPU time is 35.007

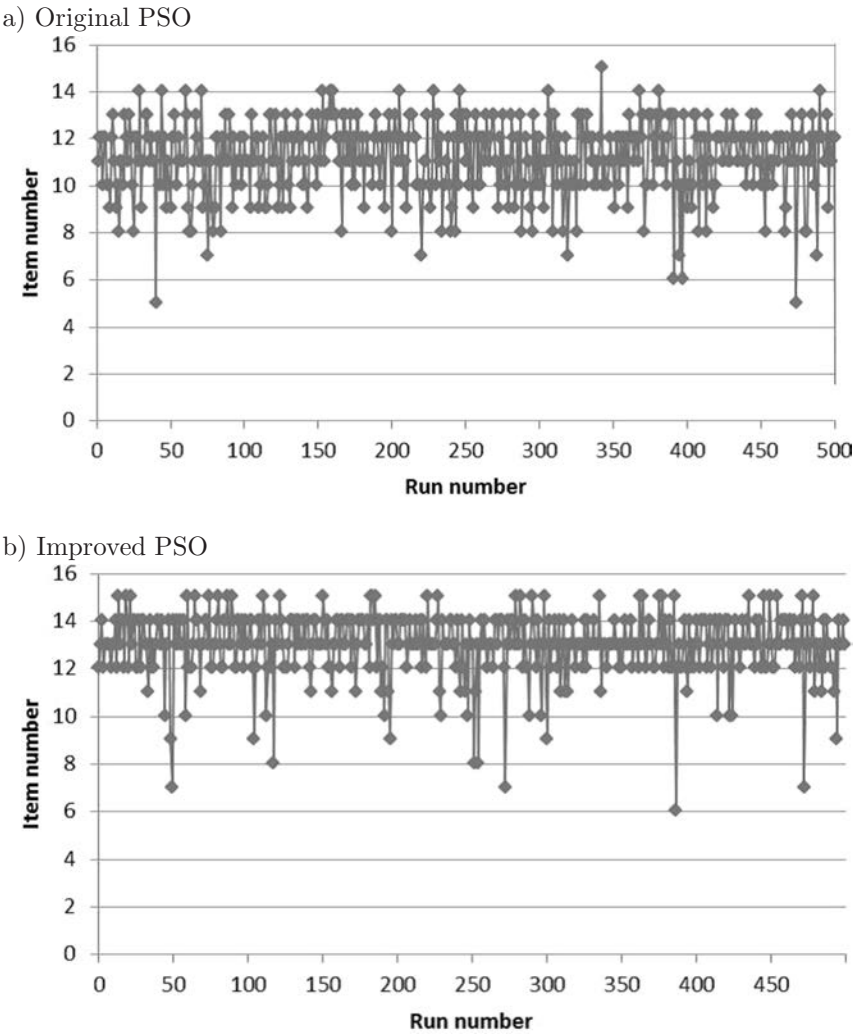

Fig. 4. Maximum item numbers in case A.

Table 2. Comparison of original and improved PSO in case A.

	Original PSO	Improved PSO
Average maximum item number $(z_{\max})_{ave}$	11.144	12.984
Average CPU time (seconds)	35.007	59.753
Number of success simulations ($z_{\max} \geq 14$)	14	123
Number of success simulations ($z_{\max} = 15$)	1	33

and 59.753, respectively. The average maximum item number is defined as the average values of the maximum item numbers in the 500 simulations. Although the item numbers are integer numbers, the average maximum item numbers are shown as decimal numbers in order to compare the algorithms, effectiveness. The maximum number of the items which can be packed in the region is manually estimated as 15. The term "number of success simulations ($z_{\max} = 15$)" and "number of success simulations ($z_{\max} \geq 14$)" denote the number of the simulations which could reach $z_{\max} = 15$ and $z_{\max} \geq 14$, respectively. Therefore, the former and the latter means the number of simulations which could reach optimal and quasi-optimal solutions, respectively. Unfortunately, both algorithms cannot reach the optimal solution because number of success simulations ($z_{\max} = 15$) is 1 in the original PSO and 33 in the improved PSO, respectively. Number of success simulations ($z_{\max} \geq 14$) is 14 in the original PSO and 123 in the improved PSO, respectively. Therefore, it is concluded that

the use of the improved PSO can increase the item number and the number of success simulations although the CPU time is increased.

Figure 5 shows the fitness function $f(\boldsymbol{x}^g)$ converges to 1 in case of the item number $z \leq 15$ although it dose not converge to 1 in case of $z = 16$. Therefore, in this case, maximum item number is concluded to be $z = 15$. Figure 6 shows the item placement in case of the improved PSO. We can determine from Fig. 6 that the items overlap in case of $z = 16$.

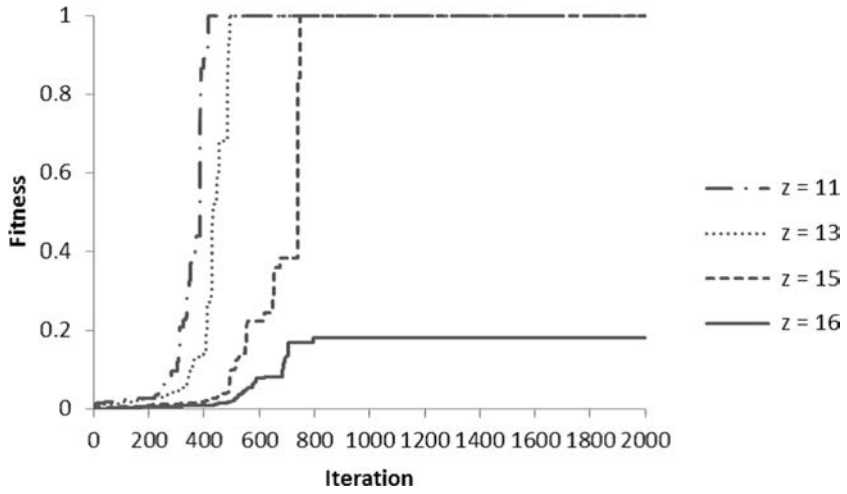

Fig. 5. Fitness convergence of improved PSO in case A.

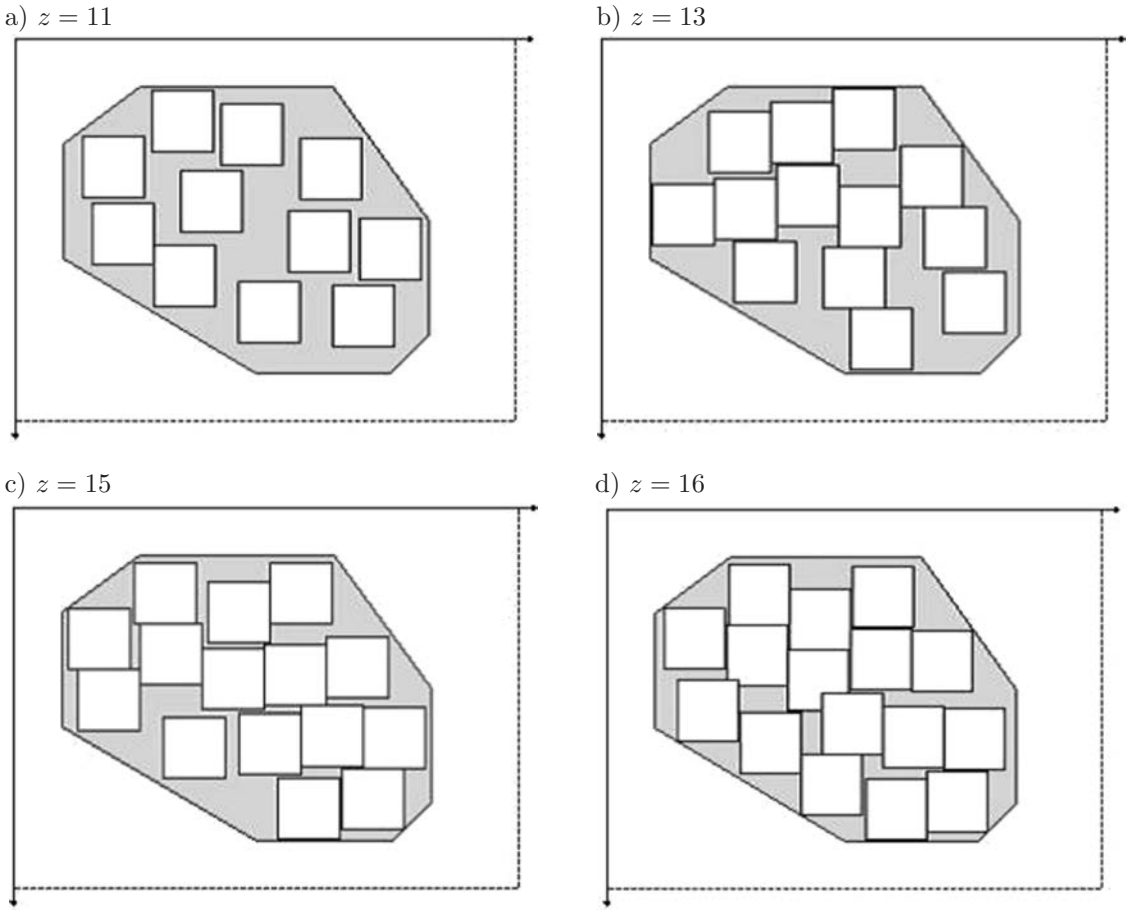

Fig. 6. Best item placement determined by improved PSO in case A.

Next, the effect of the parameter P_s is discussed. Table 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.

Table 3. Effect of parameter P_s in case A.

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

4.2. Case B

The packing region of case B is shown in Fig. 7. PSO parameters are identical to case A (Table 1). Number of particles and maximum iteration steps are specified as $N = 200$ and $t_{\max} = 2000$, respectively. The other parameters are taken as

$$\omega = 0.9, \quad c_1 = 1.5, \quad c_2 = 1.5, \quad c_3 = 1.9, \quad \text{and} \quad P_s = 0.1.$$

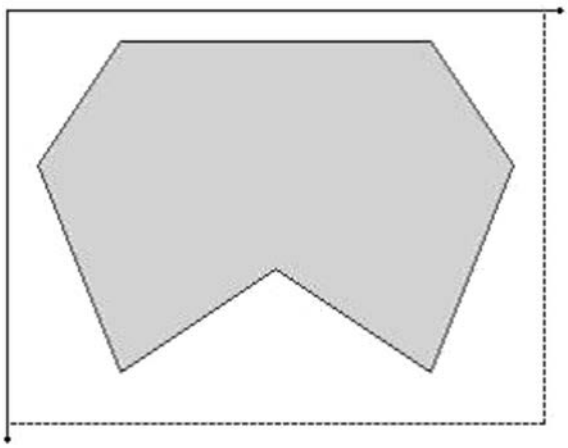

Fig. 7. Packing region in case B.

The results are shown in Fig. 8 and Table 4. The average maximum item number is 12.07 in case of the original PSO and 13.99 in improved PSO. The average CPU time is 35.198 and 65.124, respectively. The maximum number of the items which can be packed in the region is manually estimated as 18. The term "number of success simulations ($z_{\max} = 18$)" and "number of success simulations ($z_{\max} \geq 17$)" denote the number of the simulations which could reach $z_{\max} = 18$ and $z_{\max} = 17$ or 18, respectively. Therefore, the former and the latter means the number of simulations which could reach optimal and quasi-optimal solutions, respectively. Unfortunately, both algorithms cannot reach the optimal solution because "number of success simulations ($z_{\max} = 18$)" is zero in both algorithms. Number of success simulations ($z_{\max} \geq 17$) is 0 in the original PSO and 2 in the improved PSO, respectively. Therefore, it is shown that case B is much more difficult to be solved than case A and that the use of the improved PSO can increase the item number and number of success simulations although the CPU time is increased.

Figure 9 shows the item placement in case of the improved PSO. We notice that the items overlap in case of $z = 18$ although they do not overlap in case of $z \leq 17$.

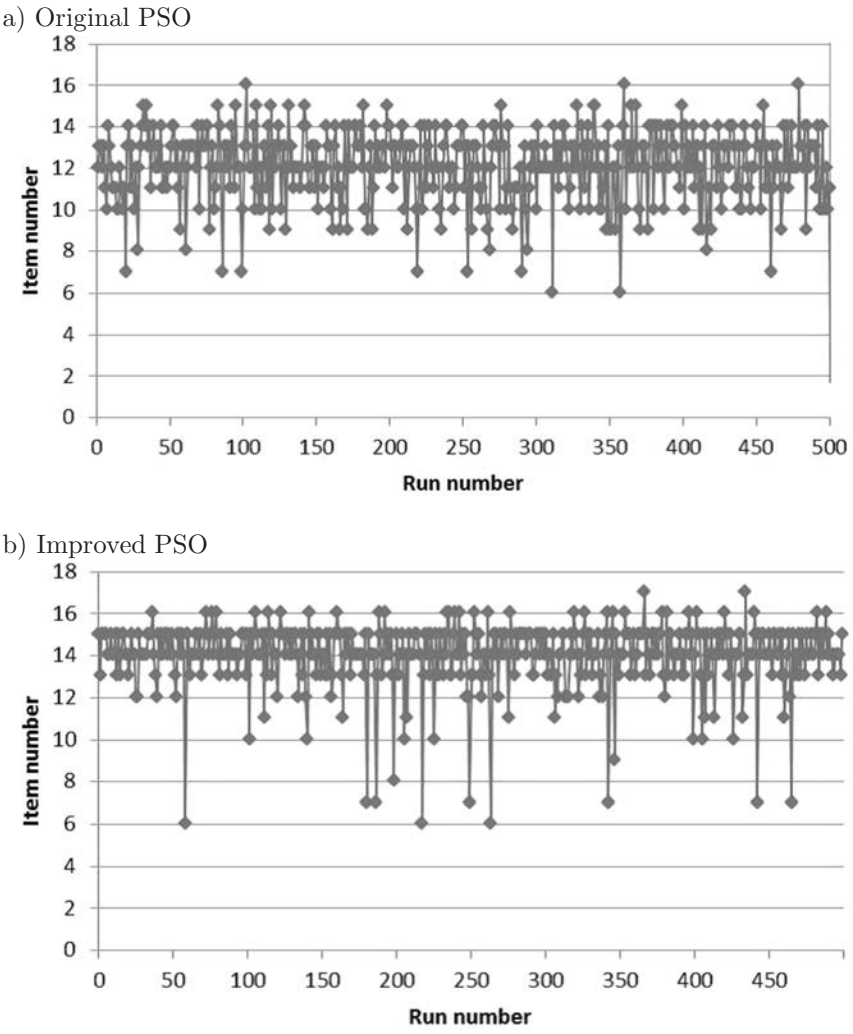

Fig. 8. Maximum item numbers in case B.

Table 4. Comparison of original and improved PSO in case B.

	Original PSO	Improved PSO
Average maximum item number $(z_{\max})_{ave}$	12.07	13.99
Average CPU time (seconds)	35.198	65.124
Number of success simulations ($z_{\max} \geq 17$)	0	2
Number of success simulations ($z_{\max} = 18$)	0	0

Next, the effect of the parameter P_s to 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. The results show that, at $P_s = 0.1$, the item number is largest and CPU time is shortest.

4.3. Case C

The packing region of case C is shown in Fig. 11. PSO parameters are identical to case A (Table 1). Number of particles and maximum iteration steps are specified as $N = 200$ and $t_{\max} = 2000$, respectively. The other parameters are taken as $\omega = 0.9$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$, and $P_s = 0.1$.

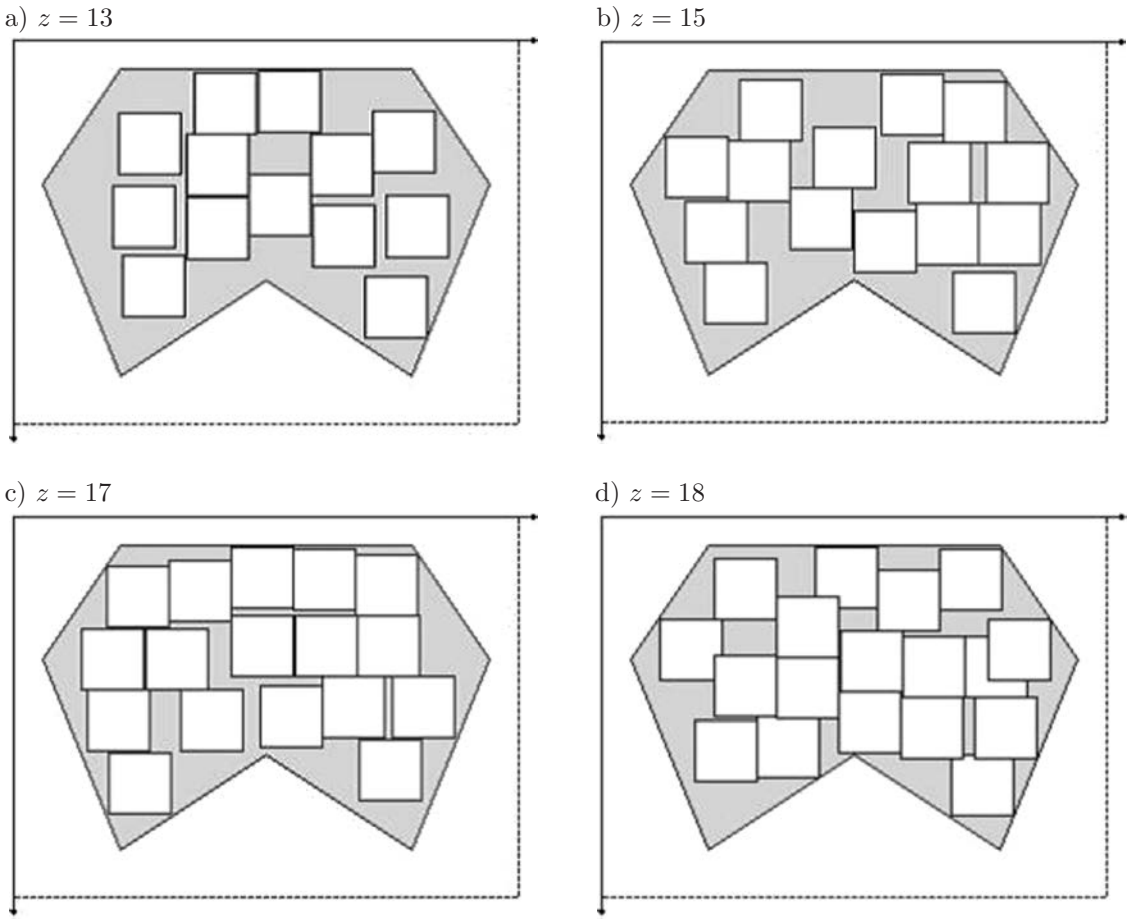

Fig. 9. Best item placement determined by improved PSO in case B.

Table 5. Effect of parameter P_s in case B.

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

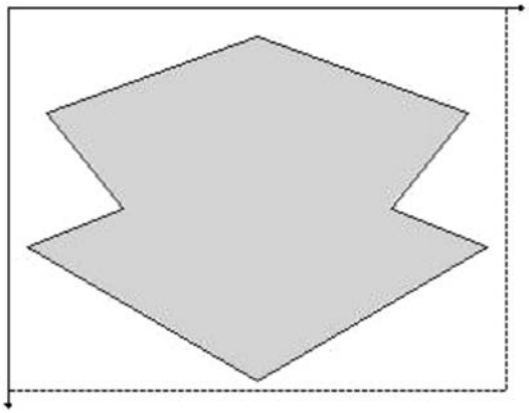

Fig. 10. Packing region in case C.

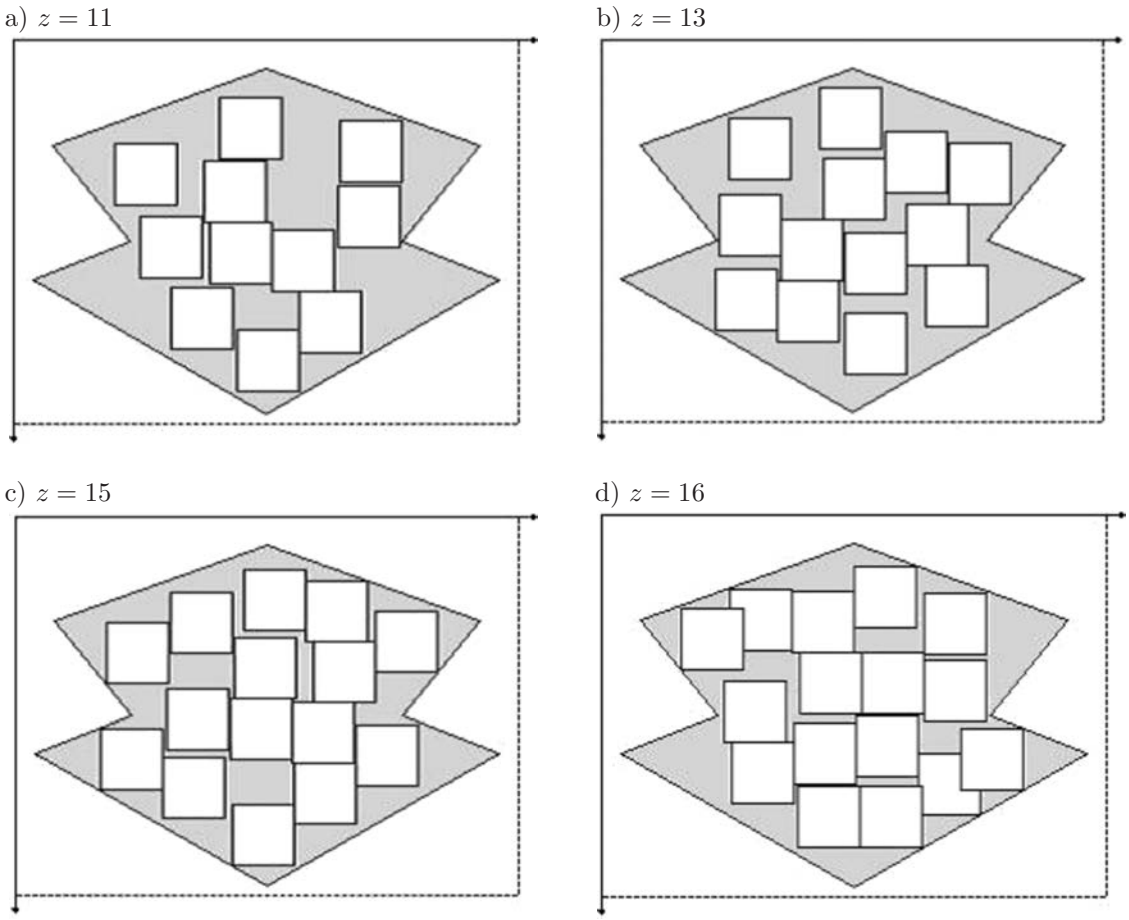

Fig. 11. Best item placement determined by improved PSO in case C.

Figure 11 shows the item placement in case of the improved PSO. We notice that the items overlap in case of $z = 16$ although the items do not overlap at $z \leq 15$.

The results are shown in Fig. 12 and Table 6. The average maximum item number is 12.07 in case of the original PSO and 13.99 in improved PSO. The average CPU time is 35.198 and 65.124, respectively. The maximum number of the items, which can be packed in the region, is manually estimated as 16. The term "number of success simulations ($z_{\max} = 16$)" and "number of success simulations ($z_{\max} \geq 15$)" denote the number of the simulations which could reach $z_{\max} = 16$ and $z_{\max} = 15$ or 16, respectively. Therefore, the former and the latter means the number of simulations which could reach optimal and quasi-optimal solutions, respectively. Unfortunately, both algorithms cannot reach the optimal solution because "number of success simulations ($z_{\max} = 16$)" is zero in both algorithms and "number of success simulations ($z_{\max} \geq 15$)" is 2 in the original PSO and 11 in the improved PSO. Therefore, also in this case, the use of the improved PSO can increase the item number although the CPU time is increased.

Table 6. Comparison of original and improved PSO in case C.

	Original PSO	Improved PSO
Average maximum item number $(z_{\max})_{ave}$	11.7	12.864
Average CPU time (seconds)	60.754	80.931
Number of success simulations ($z_{\max} \geq 15$)	2	11
Number of success simulations ($z_{\max} = 16$)	0	0

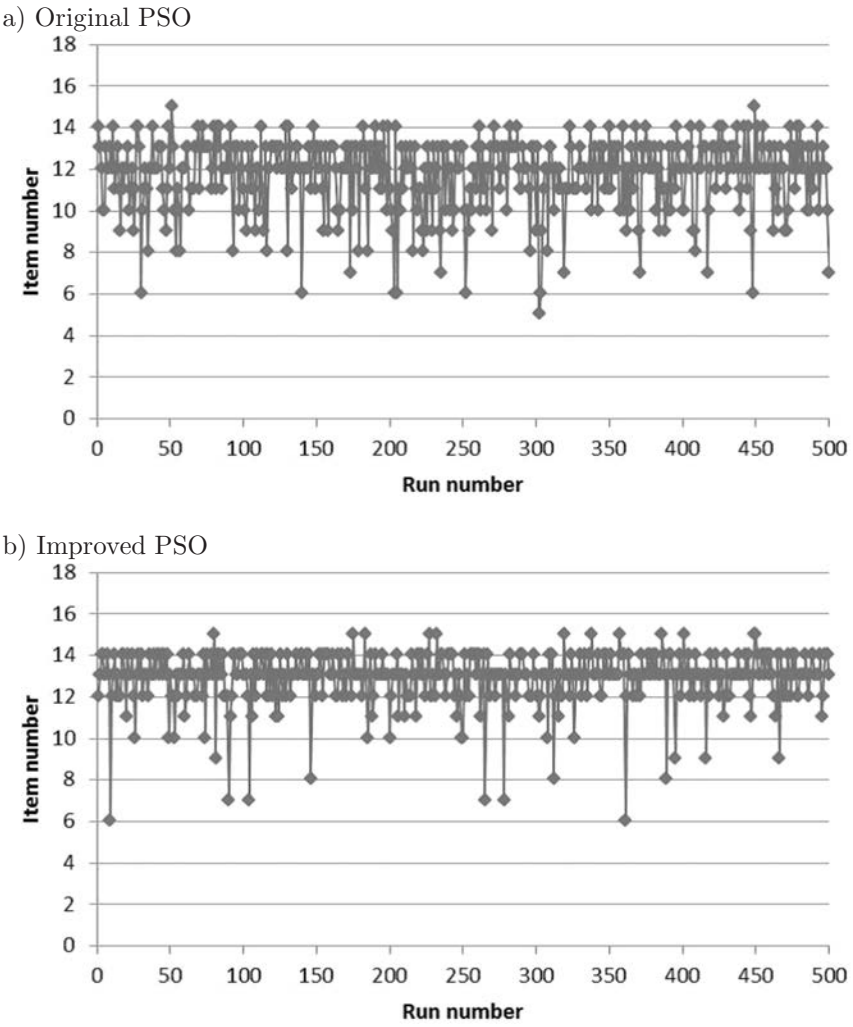

Fig. 12. Maximum item numbers in case C.

Table 7. Effect of parameter P_s in case C.

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

Next, the effect of the parameter P_s to 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. The results show that, at $P_s = 0.1$, the item number is largest and CPU time is shortest.

5. Conclusions

The application of PSO for solving two-dimensional packing problems 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, usually it is assumed that the items are placed every certain intervals. In this study, we considered the packing region that was arbitrarily polygon-shaped. The problem was solved by the original and improved PSOs. In the original PSO, the particle position vectors are updated by the global and the personal best positions. The improved PSO utilizes, in addition to

them, the global second best position of all particles. The use of the global second best position is determined in the probabilistic way.

The algorithms were compared in three numerical examples. The design, objective is to maximize the number of items contained in the packing region without their overlap. In case of the original PSO, the average values of the maximum number of the packed items are 11.1 in case A, 12.07 in case B and 11.7 in case C, respectively. In case of the improved PSO, they are 12.98 in case A, 13.99 in case B and 12.86 in case C, respectively. It is concluded that the use of the improved PSO can increase the maximum item number by two items.

By the way, we would like to discuss the disadvantages of the present algorithm. Firstly, the performance of the improved PSO depends on the threshold P_s. In this study, the threshold P_s was determined from the numerical experiments. Therefore, we would like to discuss the adequate parameter design. Secondly, from the viewpoint of engineering application of the packing problem, the closer arrangement of the items is important. The aim of this study was to estimate the maximum item number in the packing domain without their overlap. Therefore, the closer arrangement of items is not taken into consideration in this study. Thus, in the next study we would like to present the closer arrangement of the items by defining the additional objective function or constraint conditions.

References

[1] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1st. edition, 1975.

[2] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

[3] J. Kennedy, R.C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE the International Conference on Neural Networks*, volume 6, pages 1942–1948, 1995.

[4] Hallard T. Croft, Falconer Kenneth J., Guy Richard K. *Unsolved Problems in Geometry.* Springer-Verlag, 1991.

[5] J. Melissen. Packing 16, 17 or 18 circles in an equilateral triangle. *Discrete Mathematics*, 145: 333–342, 1995.

[6] Erich Friedman. Packing unit squares in squares: a survey and new results. *The Electronic Journal of Combinatorics*, DS7, 2005.

[7] D.S. Liu, K.C. Tan, S.Y. Huang, C.K. Goh, W.K. Ho. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, 190(2): 357–382, 2008.

[8] Chen Zhao, Liu Lin, Cheng Hao, Liu Xinbao. Solving the rectangular packing problem of the discrete particle swarm algorithm. In *Business and Information Management, 2008. ISBIM '08. International Seminar on*, volume 2, pages 26–29, 2008.

[9] Chuan He, Yuan-Biao Zhang, Jian-Wen Wu, Cheng Chang. Research of three-dimensional container-packing problems based on discrete particle swarm optimization algorithm. In *Test and Measurement, 2009. ICTM '09. International Conference on*, volume 2, pages 425–428, dec. 2009.

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

[11] Ryan Forbes, Mohammad Nayeem Teli. Particle swarm optimization on multi-funnel functions. http://www.cs.colostate.edu/nayeem/papers/pso_paper.pdf.

[12] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proceedings of 1999 Congress on Evolutionary Computation*, volume 3, pages 1951–1957, 1999.