# Application of particle swarm optimization to the item packing problem

Y.-B. Shin & E. Kita
*Graduate School of Information Science, Nagoya University, Japan*

## Abstract

The item packing problem is a class of optimization problems which involve attempting to pack items together inside a container, as densely as possible without the item overlap. This research focuses on the application of Particle Swarm Optimization (PSO) to the item packing problem in the two-dimensional region.

PSO has the potential solutions of the problem as particles. Particles in the swarm are updated according to the update rule with the velocity and position vectors. The position vectors of the item centers are taken as the design variables. The total number of items is maximized when all items are included inside a container without the item overlap. In the original PSO, the particle position vector is updated with the best position in all particles; i.e., global best position, and the local best position in previous positions of each particle; i.e., local best position. The present PSO algorithm utilizes, in addition to them, the second best position in all particles; i.e., global second-best position.

In the numerical example, the present algorithm is applied to the item packing problem within the two-dimensional region. The region figure is not regular and the square items are packed in the region. The comparison of the original and the present PSOs show that the present algorithm can find a better solution than the original PSO.

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

## 1 Introduction

Evolutionary computations are techniques implementing mechanisms inspired by biological evolution such as reproduction, mutation, recombination, natural selection and survival of the fittest; Genetic Algorithms [1–3], Simulated Annealing [4], Evolutionary Programming [5], Genetic Programming [6, 7], Particle Swarm Optimization [8, 9] and so on.

WIT Transactions on The Built Environment, Vol 125, © 2012 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/OP120211

Genetic Algorithms (GA) [1–3] is very popular algorithm in the evolutionary computations. In GA [1–3], a population of chromosomes of candidate solutions to an optimization problem evolves toward better solutions by applying the genetic operators such as selection, crossover, mutation and so on. On the other, Genetic Programming [6, 7] and Grammatical Evolution [10–12] are designed for the different object. Their object is to find function representations for the unknown data sets and computer programs that perform a user-defined task. Particle Swarm Optimization (PSO), which has been presented in 1995 by Kennedy and Eberhart [8], 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.

In this study, we will apply PSO for solving two-dimensional packing problems. 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, packing by cost, and so on. We focus on the two-dimensional packing problems. Popular problems in two-dimensional packing are to packing circles or squares in a circle or a square. The problems are studied analytically and the maximum numbers of items are determined [13–15]. The application of PSO for solving packing problem has been presented by some researchers [16–19]. In this study, we consider that the packing regions have the arbitrarily shaped region and then, same items are packed in the region without their overlap. The design objective is to maximize the total number of the items packed in the region without the item overlap. The position vectors of the item centers are taken as the design variables. The problem is solved by the original and the present 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 [9].

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

## 2 PSO algorithms

### 2.1 Update rule of original PSO

PSO algorithm determines the potential solutions of the optimization problem as the swarm of the particles. 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 particle position vector is defined as the set of the design variables of the optimization problem. Each particle has memory and therefore, can remember the best position in search space it ever visited. The satisfaction of the particle $i$ for the design objective at time $t$ 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 $\boldsymbol{x}_i^{pbest}(t)$ and the overall best out of all particles in the swarm is as the global best position $\boldsymbol{x}^{gbest}(t)$. In the original PSO, the velocity and position vectors of the particle $i$ are updated according to the following formulas

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

$$\begin{aligned}\boldsymbol{v}_i(t+1) = w \cdot \boldsymbol{v}_i(t) + c_1 \cdot R_1 \cdot \{\boldsymbol{x}_i^{pbest}(t) - \boldsymbol{x}_i(t)\} \\ + c_2 \cdot R_2 \cdot \{\boldsymbol{x}^{gbest}(t) - \boldsymbol{x}_i(t)\}\end{aligned} \tag{2}$$

where $w$ is the inertia weight, $c_1$ and $c_2$ are acceleration coefficient, 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 varied as the algorithm progresses. The inertia weight $w$, in this study, is generally updated by self-adapting formula as

$$w = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t}{t_{\max}} \tag{3}$$

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 parameters $c_1$ and $c_2$ determine the relative pull of $\boldsymbol{x}_i^{pbest}(t)$ and $\boldsymbol{x}^{gbest}(t)$. According to the recent work done by Clerc [20], the parameters are given as

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

### 2.2 Update rule with global second-best position

The original PSO have no handling mechanism for avoiding the convergence to the local optima. Therefore, the new update rule is introduced in this section.

Each particle has three memories and thus, can remember, the global best position $\boldsymbol{x}^{gbest}(t)$, the local best position $\boldsymbol{x}_i^{pbest}(t)$, and the second global best position $\boldsymbol{x}^{sgbest}(t)$. The use of the second global bestposition $\boldsymbol{x}^{sgbest}(t)$ attempts

to search diversity and change movement of particles. The velocity and position vectors are updated according to the following formulas.

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t+1) + \boldsymbol{v}_i(t+1) \tag{5}$$

$$\begin{aligned}\boldsymbol{v}_i(t+1) = w \cdot \boldsymbol{v}_i(t) &+ c_1 \cdot R_1 \cdot \{\boldsymbol{x}_i^{pbest}(t) - \boldsymbol{x}_i(t)\} \\ &+ c_2 \cdot R_2 \cdot \{\boldsymbol{x}^{gbest}(t) - \boldsymbol{x}_i(t)\} \\ &+ c_3 \cdot R_3 \cdot \{\boldsymbol{x}^{sgbest}(t) - \boldsymbol{x}_i(t)\}\end{aligned} \tag{6}$$

where $w$ is the inertia weight, $c_1, c_2$ and $c_3$ are acceleration coefficient, 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 $c_3 = 1.9$.

### 2.3 Algorithm

PSO share the information of $\boldsymbol{x}_i^{pbest}$, $\boldsymbol{x}^{gbest}$ and $\boldsymbol{x}^{sgbest}$. Obviously, $\boldsymbol{x}^{sgbest}$ is worse than $\boldsymbol{x}^{gbest}$. If only equation (6) is used for updating particle velocity and position vectors, the results must be worse than that by the original PSO. Therefore, the present algorithm uses both update rules; equations (2) and (6). The switching of the update rules (2) and (6) is controlled according to the probability $P_s$. The process is as follows:

1. Initialize the position and velocity vectors of particles with random numbers.
2. Set $t = 1$.
3. For $i = 1, 2, \cdots, N$:
   (a) Evaluate fitness functions $f(\boldsymbol{x}_i(t))$ for the particle $i$.
   (b) If $f(\boldsymbol{x}_i(t)) > f(\boldsymbol{x}_i^{pbest})$, set $\boldsymbol{x}_i^{pbest} = \boldsymbol{x}_i(t)$.
4. Find the first- and second-best particles $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ among $\boldsymbol{x}^{gbest}$, $\boldsymbol{x}^{sgbest}$ and $\boldsymbol{x}_i(t)$ $(i = 1, 2, \ldots, N)$.
5. If $\boldsymbol{x}^1 > \boldsymbol{x}^{gbest}$, set $\boldsymbol{x}^{gbest} = \boldsymbol{x}^1$.
6. If $\boldsymbol{x}^2 > \boldsymbol{x}^{sgbest}$, set $\boldsymbol{x}^{sgbest} = \boldsymbol{x}^2$.
7. Generate random number $r$ in the interval$[0, 1]$.
8. If $r > P_s$, update the velocity and position vectors of all particles according to equations (1) and (2), respectively.
9. If $r \leq P_s$, update the velocity and position vectors of all particles according to equations (5) and (6), respectively.
10. Set $t = t + 1$ and go to step 3 if $t \leq t_{\max}$.

## 3 Packing problem

### 3.1 Optimization problem

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

The objective function is defined as follows.

$$\max z \tag{7}$$

The design variables vector is defined as the set of the center position vectors of all items as follows.

$$\boldsymbol{x}_i = \{p_x^1, p_y^1, \cdots, p_x^k, p_y^k, \cdots, p_x^z, p_y^z\}^T \tag{8}$$

where the vector $\{p_x^k, p_y^k\}$ denotes the center position vector of the item $k$,

The constraint conditions are as follows.

$$g_1(k, P) = 0 \tag{9}$$

$$g_2(k, l) = 0 \tag{10}$$

$$0.5w \leq p_x^k \leq W - 0.5w \tag{11}$$

$$0.5h \leq p_y^k \leq H - 0.5h \tag{12}$$

where $k = 1, 2, \cdots, z; l = 1, 2, \cdots, z$. The parameters $w$ and $h$ are item sizes, and $W$ and $H$ are feasible space sizes. The function $g_1(k, P)$ estimates the inclusion of the item $k$ in the region $P$, which is defined as follows:

$$g_1(k, P) = \begin{cases} 0 & \text{The item } k \text{ is included in the region } P. \\ 1 & \text{The item } k \text{ is not included in the region } P. \end{cases} \tag{13}$$

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

$$g_2(k, l) = \begin{cases} 0 & \text{The item } k \text{ and } l \text{ are not overlapped.} \\ 1 & \text{The item } k \text{ and } l \text{ are overlapped.} \end{cases} \tag{14}$$

### 3.2 PSO implementation

When the number of the items $z$ is given, PSO solves the item packing problem within the packing region without violating the constraint conditions. In the optimization problem to be solved by PSO, the fitness function is defined as follows.

$$f(\boldsymbol{x}_i) = \frac{1}{1 + \sum_{k=1}^{z} \left\{ g_1(k, P) + \sum_{l=1}^{z, k \neq l} g_2(k, l) \right\}} \tag{15}$$

### 3.3 Optimization process

The process of the present algorithm is shown in Fig.1 and summarized as follows.

1. $z = 0$.
2. Set the threshold $P_s$ and the maximum step size $t_{\max}$.

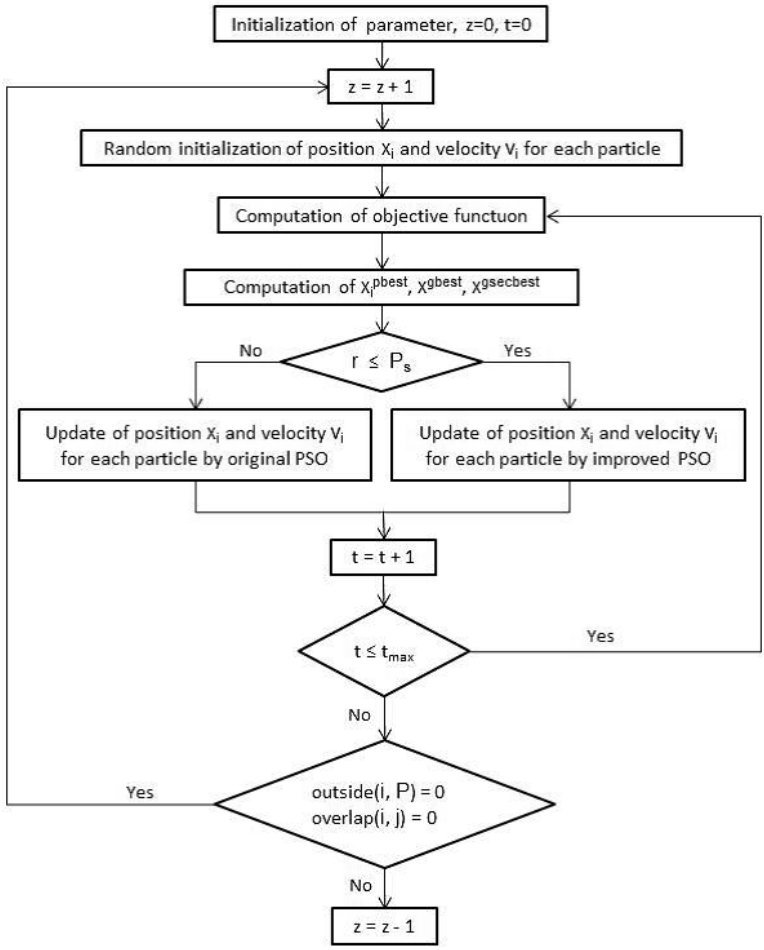



Figure 1: Flow chart of the parking problem by using improved PSO.

3. Update the item number by $z = z + 1$.
4. Perform PSO algorithm:
   (a) $t = 0$
   (b) Initialize a particle population with random position and velocity vectors.
   (c) Evaluate the fitness function for each particle $f(\boldsymbol{x}_i)$.
   (d) Estimate the $\boldsymbol{x}_i^{pbest}$, $\boldsymbol{x}^{gbest}$ and $\boldsymbol{x}^{gsbest}$.
   (e) If $f(\boldsymbol{x}^{gbest}) = 0$, go to Step 3.
   (f) Generate the random number $r$ in $[0, 1]$.
   (g) If $r \le P_s$, update particles by equation (6), otherwise updated by equation (2).

(h) $t = t + 1$.
(i) If $t \leq t_{max}$, return to Step 4c.
5. If $g_1(i, P) = g_2(i, j) = 0$, return to Step 3.
6. Stop by $z = z - 1$.

## 4 Numerical examples

The packing region is shown in Figure 2. PSO parameters are listed 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 $w = 0.9$, $c_1 = 1.5$, $c_2 = 1.5$, $c_3 = 1.9$, and $P_s = 0.1$.

The results are shown in Figure 3 and Table 2. The average values of the maximum item numbers are 12.07 in the original PSO and 13.99 in the present

Table 1: Parameters.

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

Table 2: Comparison of original and improved PSOs.

| | Original PSO | Improved PSO |
|---|---|---|
| Average item number $z$ | 11.7 | 12.864 |
| Average CPU time (seconds) | 60.754 | 80.931 |
| Success rate in $z \geq 13$ | 36.8% | 73.8% |

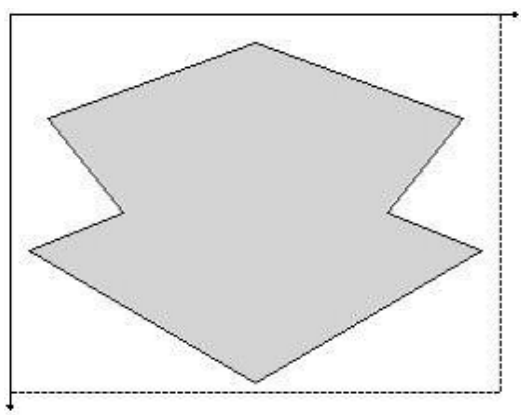


Figure 2: Packing region.

Table 3: Effect of parameter $P_s$ in case B.

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

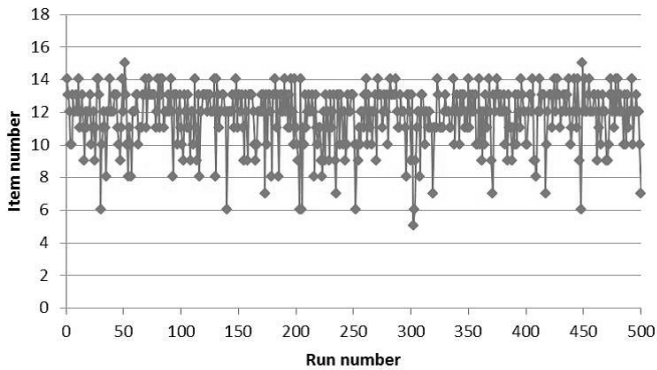



Figure 3: Maximum item numbers by original PSO.

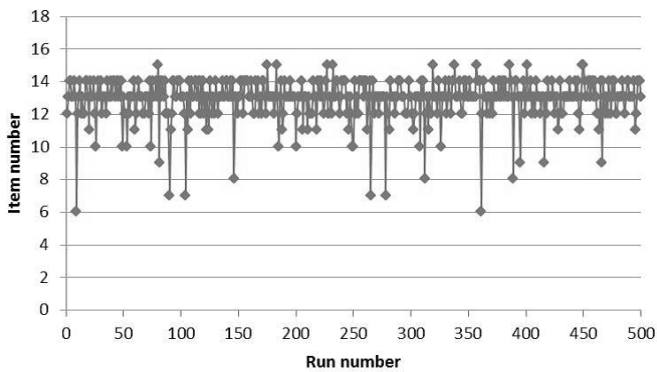
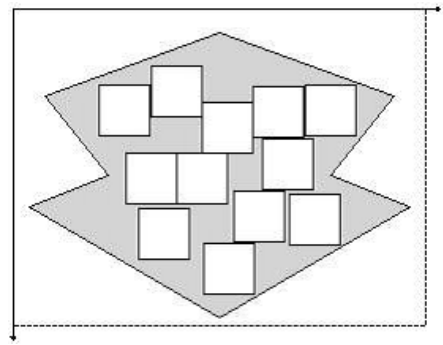
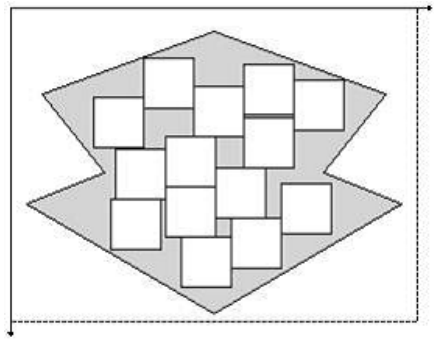


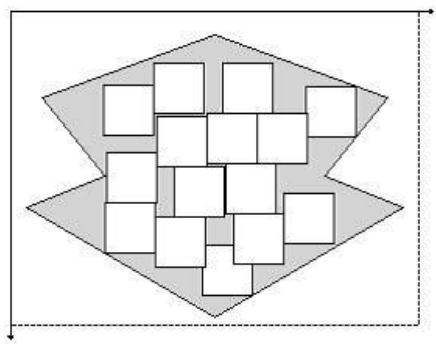

Figure 4: Maximum item numbers by present PSO.



(a) $z$=12



(b) $z$=14



(c) $z$=15

Figure 5: Item placements in case of improved PSO.

PSO. The average CPU time is 35.198 and 65.124, respectively. The success rate is 19.8% and 75.2%, respectively. The use of the improved PSO can increase the item number improve the success rate although the CPU time is increased.

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 3. The results show that, at $P_s = 0.1$, the item number is largest and CPU time is shortest.

## 5 Conclusions

Application of Particle Swarm Optimization (PSO) for the two-dimensional packing problem was presented in this study. It was assumed that the packing region was not regularly but arbitrarily shaped. The square items were packed in the region as dense as possible without their overlapping. PSO was applied for solving the arrangement problem of the items in the region. The original PSO updates the particle position vectors by two information; the global best position and the local best position. The present algorithms uses, in addition to them, the second global best position. The present algorithm was applied for solving the item packing problem to the two-dimensional concave region. The results were compared with them by the original PSO. The average values of the maximum item numbers are 12.07 in the original PSO and 13.99 in the present PSO. Therefore, the present PSO can find better solution than the original PSO.

## References

[1] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1 edition, 1975.
[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1 edition, 1989.
[3] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1 edition, 1991.
[4] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
[5] D. B. Fogel and J. W. Atmar. *Proc. 1.st annual Conference on Evolutionary Programming*. Evolutionary Programming Society, 1992.
[6] J. R. Koza, editor. *Genetic Programming II*. The MIT Press, 1994.
[7] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, editors. *Genetic Programming III*. Morgan Kaufmann Pub., 1999.
[8] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE the International Conference on Neural Networks*, volume 6, pages 1942–1948, 1995.
[9] Ryan Forbes and Mohammad Nayeem Teli. Particle swarm optimization on multi-funnel functions.

[10] C.Ryan, J.J.Collins, and M.O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of 1st European Workshop on Genetic Programming*, pages 83–95. Springer-Verlag, 1998.
[11] C.Ryan and M.O'Neill. Crossover in grammatical evolution: A smooth operator? In *Proceedings of the European Conference on Genetic Programming*, pages 149–162. Springer-Verlag, 2000.
[12] C.Ryan and M.O'Neill. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer-Verlag, 2003.
[13] Hallard T Croft, Falconer Kenneth J., and Guy Richard K. *Unsolved Problems in Geometry*. Springer-Verlag, 1991.
[14] J. Melissen. Packing 16, 17 or 18 circles in an equilateral triangle. *Discrete Mathematics*, 145:333–342, 1995.
[15] Erich Friedman. Packing unit squares in squares: a survey and new results. *The Electronic Journal of Combinatorics*, DS7, 2005.
[16] D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, and W. K. Ho. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, 190(2):357 – 382, 2008.
[17] Chen Zhao, Liu Lin, Cheng Hao, and 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.
[18] Chuan He, Yuan-Biao Zhang, Jian-Wen Wu, and 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.
[19] P. Thapatsuwan, J. Sepsirisuk, W. Chainate, and 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.
[20] 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.