Research on PSO algorithms for the rectangular packing problem

Jinmin Wang*, Yang Qi and Jing Zhang

Tianjin Key Laboratory of High Speed Cutting & Precision Machining,
Tianjin University of Technology and Education,
Tianjin 300222, China
Email: wang_jin_min@163.com
Email: marktree@sina.com
Email: 944245203@qq.com
*Corresponding author

Abstract: For packing problems that are complex optimisation problems, the solutions must rely on heuristics. Particle swarm optimisation (PSO) is an emerging technique that utilises heuristics. Because of its simplicity, PSO has been widely applied to a variety of fields, including applications to the packing problem. This paper researches the search efficiency of the PSO algorithm. Through fitting the data to the length of the solution interval and the number of iterations, a strong search capability for the PSO is proved. Then, the results of experiments with comparisons of many disturbance strategies show that a stochastic strategy is beneficial for raising the search capability of the algorithm.

Keywords: packing problem; PSO; particle swarm optimisation; ordering rule; location rule; solution interval.

Reference to this paper should be made as follows: Wang, J., Qi, Y. and Zhang, J. (2014) 'Research on PSO algorithms for the rectangular packing problem', *Int. J. Computer Applications in Technology*, Vol. 51, No. 1, pp.15–22.

Biographical notes: Jinmin Wang is a Full Professor at the Tianjin Key Laboratory of High Speed Cutting & Precision Machining, Tianjin University of Technology and Education (TUTE), China. He obtained his Bachelor degree and Master degree in Tianjin University in 1985 and 1988, respectively. He obtained his PhD in Mechanical Engineering from Tianjin University in 1996, in China. His main research interest is in the area of intelligent packing, computer aided design. He has published over 60 papers on these topics and provided consultancy to various industries.

Yang Qi received his Bachelor degree and Master degree in Tianjin University of Technology and Education in 2008 and 2011, respectively, in China. His main research areas include mechanical design, intelligent packing design.

Jing Zhang received her Bachelor degree in 2011 at Changzhou Institute of Technology, China. She is a graduate of Tianjin University of Technology and Education, China. Her main research areas include intelligent cutting and packing problem.

1 Introduction

Packing problems are encountered in a wide variety of production practices in industry. People have presented many algorithms and methods for simplifying and solving packing problems. The detailed survey about methods and application of packing problems can be found (Lodi et al., 2002; Ntene and van Vuuren, 2009). Gilmore and Gomory (1961) used a linear programming approach to solve the cutting stock problem. Valério de Carvalho and Guimarães Rodrigues (1995) exploited a simplified linear programming model to describe the two-stage cutting stock problem. Dyckhoff (1990) proposed a typology that partitioned packing problems by the dimensionality, type of object, shape of the item and more characteristics. Although this typology was revised afterwards (Wäscher et al., 2007), it was still accepted by researchers. However, the packing problem is an NP-complete problem (Garey and Johnson, 1979); thus, its intrinsic difficulty makes traditional optimisation methods unsatisfying. Since the 1990s, heuristics have developed rapidly, and many researchers have introduced heuristics into studies on packing problems; this addition improves the chances of finding better solutions. Jakobs (1996) used a genetic algorithm to evolve a sequence of pieces for a variant of the Bottom-left (BL) heuristic. Hopper and Turton (2001) combined different packing algorithms with simulated annealing, genetic algorithms, random search algorithms and evolutionary

algorithms; then, they evaluated and researched the overall performances of these algorithms on a variety of examples. Although PSO is an emerging algorithm, it has caught many researchers' attention for its simplicity and ease of use; it has been applied to the packing problem. Li et al. (2006) presented a parallel algorithm that hybridised PSO and GAs and exploited this combination to solve a packing problem. Liu et al. (2008) solved a 2D bin packing problem with a hybrid multi-objective PSO algorithm. Zhou et al. (2005) exploited PSO to optimise the circular piece packing problem. Huang et al. (2006) exploited Quantum PSO to solve an irregular polygonal packing problem. Xu et al. (2011) reviewed PSO algorithms applied to a 2D packing problem. The packing problem is an optimisation problem that has its own specific features. Whether a PSO can raise the search efficiency of a packing algorithm based on these features is a question that deserves further research.

In this paper, the two-dimensional rectangular packing problem is considered. This problem belongs to a subset of classical cutting and packing problems. The paper studies the search efficiency of the PSO. The concept of "solution interval of packing" is presented, and it is proved that PSOs have a certain search capability for packing problems by fitting the data to the length of the solution interval and the number of iterations. Through the experiments and comparisons of many disturbance strategies, this paper also proves that a stochastic strategy is beneficial for raising the search capability of a PSO on packing problems.

2 Rectangular packing problem

The rectangular packing problem is an important type of packing problem. Although it is an abstraction and simplification of the general packing problem with a specific gap in its practice and application, it is the basis of the general packing problem, and the development of research on it can provide experience for other types of packing problems.

The rectangular packing problem is required to allocate a set of rectangular items to a given region upon request; the items cannot interfere with other items, and there is a maximum sought for the utilisation ratio of a region. A mathematical model of the rectangular packing problem is described in Wang et al. (2010).

In many solutions of the packing problem, the constructive algorithm is considered to be important. It divides the packing into two parts, one in which the items are ranked and the other in which the items are placed. Thus, the constructive algorithm is separated into the ordering rule and the location rule.

The ordering rule is used to determine the order of the rectangular items by comparing their attribute or attributes. Common rules determine the order by the area, the longest side, the feasible region or other characteristic. These rules are summarised from practice and can achieve excellent performance in applications. Because the area is an important evaluation index in the rectangular packing problem, this paper adopts the rule that generates the sequences by using the areas.

The location rule is used to determine the position of the rectangular items. Common rules include the BL routine, the down-step approach, and the location function approach. In these rules, the location function approach is more flexible than other location approaches and can be realised in computer implementations more easily. The attractive factor approach (Wang and Yang, 2005) is one of the location function approaches. In this approach, a rectangular item is moved by the attraction of attractive factors that are placed in the region to satisfy the location specifications.

The location function of the ith point is calculated as

$$\begin{aligned} G(x_i, y_i) &= \sum_{t=1}^{m} w_t g_t(x_i, y_i) \\ &= \sum_{t=1}^{m} w_t(\alpha_t \mid x_i - x_{0t} \mid + \beta_t \mid y_i - y_{0t} \mid), \end{aligned} \tag{2.1}$$

where $g_t(x_i, y_i)$ is the location function of the tth attractive factor, which is positioned in (x_{0t}, y_{0t}), m is the number of attractive factors, α_t, β_t and ω_t are weight factors, and

$$\alpha_t + \beta_t = 1, \quad \sum_{t=1}^{m} \omega_t = 1, \quad \alpha_t, \beta_t, \omega_t \in [0,1].$$

The attractive factor approach prescribes the point location function value for which the point with the smallest value is selected to be a packed-in point. In other words, if $G(x_0, y_0) = \min G(x_i, y_i)$, then the point (x_0, y_0) is a packed-in point.

In this paper, four attractive factors are placed in four corner points of a region. The parameters ω_1, ω_2, ω_3 and ω_4 are the attraction levels of the factor in the bottom-left corner, bottom-right corner, top-right corner and top-left corner, respectively. It should be noted that there are 12 parameters, which are α_t, β_t, ω_t (t = 1, 2, 3, 4), in the four attractive factors. However, there are five restrained conditions, which are $\sum_{t=1}^{4} \omega_t = 1, \alpha_t + \beta_t = 1$ (t = 1, 2, 3, 4), so that only seven parameters, specifically ω_1, ω_2, ω_3, α_1, α_2, α_3 and α_4 are defined.

Because the order of the rectangular items is defined by the area, when the parameters of the location function are defined, the packing solution has been defined. In other words, a set of parameters represents a solution to the packing, and the utilisation ratio of the region is the evaluation value. Thus, the constructive algorithm can be described as a function, the independent variables are the parameters of the location function, and the dependent variable is the utilisation ratio of the region. In this way, the packing problem is transformed into an optimisation problem. On the basis, this paper uses PSO to search for the optima in the solution space.

3 Specific swarm optimisation

Particle swarm optimisation (PSO) was presented by Kennedy and Eberhart (1995). It is an evolutionary computing technique that is based on a social psychological model. Every particle in the swarm follows a simple behaviour in which the particle imitates the successful experience of the neighbouring particles and itself. Thus, the accumulative behaviour is a search of the best region in a multi-dimensional space.

Each particle in a swarm has its own position, velocity and fitness, which is defined by an optimisation function. Each particle remembers and follows the current optimal particle and searches in the solution space.

The mathematical model of a PSO is described as follows: the position of the ith particle is represented by the D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$, and the velocity of the ith particle is represented by the D-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^T$. The particle is updated by two 'extreme values', as follows: the first 'extreme value' is the best previous solution of the particle, which is called the individual optimum, and the location of it is represented as *lbest*; the second 'extreme value' is the best solution of the particle swarm, which is called the global optimum, and the location of it is represented as *gbest*. The updated equations are as following:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (lbest_{id}^{k} - x_{id}^{k}) + c_2 r_2 (gbest_{d}^{k} - x_{id}^{k}), \quad (3.1)$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}, \quad (3.2)$$

where x_{id}^{k} is the position in the dth dimension of the ith particle in the kth iteration, c_1 and c_2 are the acceleration ratios, which are used to adjust the maximal step lengths of the movements to the individual optima and the global optima, respectively, r_1 and r_2 are random numbers from [0, 1], and w is the inertia weight, which is used to adjust the ability of the search in the global and local regions. Information on how the PSO is used to perform optimisation is described in Qi and Wang (2011).

4 Search efficiency of the algorithm

4.1 Iteration number of the optimum

The best result that can be obtained by a PSO, although it is not a certain global optimum, is still accepted. Thus, this solution is called the experimental optimum. The iteration number of the experimental optimum affects the run time of the packing problem and reflects the search efficiency of the algorithm; thus, these relationships should be researched. It should be noted that the iteration number has a relationship to the number of particles. A large population can find the result of a problem in fewer iterations (Liu et al., 2008). In this paper, the number of particles is set to a fixed value, which is 30, and the ratio factor from Engelbrecht (2009) is used to adjust the number of particles. This approach can avoid having an influence from the different number of particles on the experimental results.

The paper adopts "the iteration number of the optimum", which specifies the iteration number when an optimum is obtained in an experiment. In the experiment, "the iteration number of the optimum" is obtained as follows: every instance is calculated 10 times, and there are 100 iterations in a single time; the best result over 10 times is selected as the experiment's optimum; then, for the experimental result that obtains the optimum statistical sample, the iteration number of obtaining the optimum is the sample data, and the average of the sample data is the iteration number of the optimum.

Table 1 shows the iteration numbers for optima in examples C11–C63 in Jakobs (1996). The data in the '*N*' column of the table are the iteration numbers of the optima. As can be observed in the table, the iteration number of the optimum increases with the number of rectangular items; however, with the examples that the item numbers are the same, the iteration numbers of the optima are not the same or similar. Thus, the data show that there is not a specific relationship between these items. To research this appearance, this paper presents 'the solution interval' concept.

Table 1 Iteration numbers for optima in examples C11–C63

Example	*Number of items*	*N*
C11	16	1
C21	25	3.1
C31	28	1.3
C41	49	2.3
C51	73	10.75
C61	97	19.25
C71	196	15.4
C12	17	1
C22	25	1
C32	29	1.6
C42	49	3.6
C52	73	44
C62	97	3.4
C72	197	18.833
C13	16	1
C23	25	1
C33	28	1.7
C43	49	8.875
C53	73	23.44
C63	97	8.5
C73	196	12.444

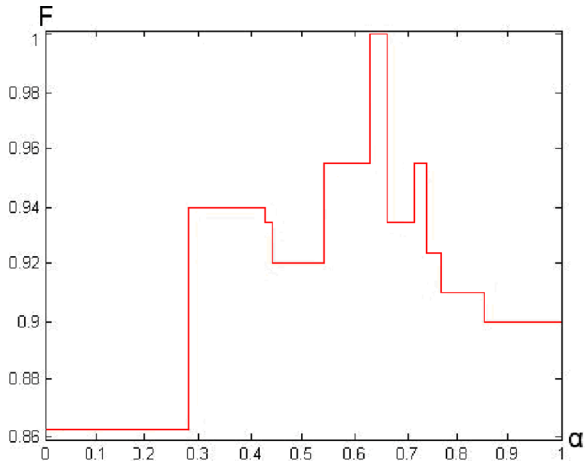
4.2 Solution interval

Through researching the properties of the attractive factor approach for solving the rectangular packing problem, we know that four attractive factors can be synthesised as an attractive factor in theory; thus, seven parameters can be simplified as one parameter. In practice, however,

considering the precision of the search and other reasons, four attractive factors are adopted in an actual computation; however, these can be simplified as a single factor in theoretical studies.

In the following study, the attractive factor is only in the bottom-left corner, as an example. The corresponding conclusions can be obtained by making similar inferences on the conditions of the other corners. In this condition, the attractive degree of a factor in the bottom-left corner is $\omega_1 = 1$, and the parameter of the factor is α. Figure 1 is the fitness function graph for example C11. In this figure, the abscissa is the parameter of the attractive factor α, and the ordinate is the value of the fitness function F, which is the utilisation ratio of the packing region. As can be seen in Figure 1, the utilisation ratio of the region is not changed when the parameter of the attractive factor α is in a certain interval. This scenario occurs because the packing solution does not change in this interval. This interval is the solution interval. In the microcosmic view, the change in α cannot lead to a change in packed-in points of any rectangular item; specifically, the packed-in point is still the point for which the value of the location function is the lowest over the whole feasible region.

Figure 1 Fitness function graph for example C11 (see online version for colours)



Because of the corner-trend of the attractive factor approach, the points that could be the packed-in points are all convex points in the bottom-left corner of the feasible region. These points can be divided into three parts: the packed-in point, the top-left points related to the packed-in point and the bottom-right points related to the packed-in point. On the basis of the definition of a packed-in point, the packed-in point is the point for which the value of the location function is the lowest, that is, the values of the other points are more than or equal to the value of this point.

Suppose that the coordinate of packed-in point M is (x_0, y_0), and the coordinate of the top-left point L is $(x_0 - \Delta x_i, y_0 + \Delta y_i)$. On the basis of the location function, the following can be obtained:

$$\begin{aligned} G(L) &= G(x_0 - \Delta x_i, y_0 + \Delta y_i), \\ &= G(x_0, y_0) + \alpha(-\Delta x_i) + (1-\alpha)\Delta y_i \\ &= G(M) + \alpha(-\Delta x_i) + (1-\alpha)\Delta y_i, \\ &\because G(L) \geq G(M), \\ &\therefore -\alpha\Delta x_i + (1-\alpha)\Delta y_i \geq 0, \\ &\therefore \frac{\Delta y_i}{\Delta x_i} \geq \frac{\alpha}{1-\alpha}, \\ &\therefore \frac{\Delta y_i}{\Delta x_i + \Delta y_i} \geq \alpha. \end{aligned}$$

Thus, if the packed-in point has not changed, then α should be less than or equal to $(\Delta y_i/(\Delta x_i + \Delta y_i))$ for any top-left point, specifically, the upper limit of α is min $(\Delta y_i/(\Delta x_i + \Delta y_i))$.

The same way as for the bottom-right points, it can obtained that $(\Delta y_j/(\Delta x_j + \Delta y_j)) \geq \alpha$, specifically, the lower limit of α is max $(\Delta y_j/(\Delta x_j + \Delta y_j))$.

From the above conclusions, it can be determined that, when the parameter of the attractive factor is in the open interval

$$\left(\max\left\{ \frac{\Delta y_j}{\Delta x_j + \Delta y_j} \right\}, \quad \min\left\{ \frac{\Delta y_i}{\Delta x_i + \Delta y_i} \right\} \right),$$

the packed-in point will not change. This interval is called the stable interval of a packed-in point. Moreover, the intersection of the stable intervals of all the packed-in points is the solution interval of the packing.

On the basis of the above study, we can determine the solution interval of the packing by analysing the solution to the packing. The following is an example in which the algorithm has an attractive factor in the bottom-left corner.

Step 1. Initiate the lower limit of the interval to be $K_1 = 0$ and the upper limit of the interval to be $K_2 = 1$ and $I = 1$.

Step 2. Extract the data of the ith packed-in rectangular item, namely, the width w_i, the height h_i and the coordinate of the packed-in point (x_{0i}, y_{0i}).

Step 3. Determine the feasible region by the width w_i, the height h_i and the state of the packing and extract the convex points in the bottom-left corner. The number of convex points is N. Initiate the parameter as $j = 1$.

Step 4. Extract the coordinate (x_{ij}, y_{ij}) of the jth convex point in the bottom-left corner of the feasible region and classify it as follows:

If $x_{ij} > x_{0i}$, $y_{ij} < y_{0i}$, then the convex point is the bottom-right point; go to Step 5.

If $x_{ij} = x_{0i}$, $y_{ij} = y_{0i}$, then the convex point is a packed-in point; this point is not handled.

If $x_{ij} < x_{0i}$, $y_{ij} > y_{0i}$, then the convex point is the top-left point; go to Step 6.

Step 5. If

$$\frac{y_{0i}-y_{ij}}{x_{0i}-x_{ij}+y_{0i}-y_{ij}}>K_1, \text{ then } K_1=\frac{y_{0i}-y_{ij}}{x_{0i}-x_{ij}+y_{0i}-y_{ij}};$$

otherwise, K_1 does not change. If $j < N$, then $j \to j+1$ and go to Step 4; if $j=N$, then $i \to i+1$ and go to Step 7;

Step 6. If

$$\frac{y_{ij}-y_{0i}}{x_{ij}-x_{0i}+y_{ij}-y_{0i}}<K_2, \text{ then } K_2=\frac{y_{ij}-y_{0i}}{x_{ij}-x_{0i}+y_{ij}-y_{0i}};$$

otherwise, K_2 does not change. If $j < N$, then $j \to j+1$ and go to Step 4; if $j = N$, then $i \to i+1$;

Step 7. Examine whether the ith packed-in rectangular item exists. If it exists, then go to Step 2. If it does not exist, then end the programming and output the upper limit K_1 and the lower limit K_2 of the interval.

For the condition of the attractive factor in the other corner, the corresponding algorithms can be designed by imitating the above algorithm.

The difference between the upper limit of the solution interval and the lower limit of the solution interval is the length of the solution interval. Because the length of the total solution interval is 1, the value is also regarded as the search probability. The longer the solution interval is, the larger the probability that the interval is searched, and vice versa.

4.3 Data fitting of the length of the solution interval and the iteration number of the optimum

The solution interval of the experimental optima for Examples T2a–T6e and C21–C63 in Hopper (2000) can be obtained by the above algorithm. In Tables 2 and 3, the data in the 'N' column are the iteration numbers of the optima, and the data in the 'λ' column are the lengths of the solution intervals (or the search probability).

The length of the solution interval and the iteration number of the optima are plotted as dots in Figure 2. Only the data for which the number of iterations is more than 1 are selected in the figure. This selection is made because the minimum iteration number of the optima is 1. If the data of which the number of iterations is 1 are selected, then the analysis would be inappropriately altered. In this figure, the abscissa expresses the iteration number of the optima (N), and the ordinate expresses the length of the solution interval (λ).

As can be observed in Figure 2, these points present a curvilinear distribution. Here, we use Matlab to fit these points, and the fitting curve is shown in Figure 2. The four parameters that are used to evaluate the fitting degree are as follows:

SSE: 0.0007682, R^2: 0.6233,

Adjusted R^2: 0.6062, RMSE: 0.005909.

Table 2 N and λ from examples T2a–T6e

Example	N	λ
T2a	1.2	0.020653
T3a	1.1	0.021575
T4a	1	0.029140
T5a	1.6	0.014127
T6a	13.71	0.002567
T2b	1	0.135880
T3b	1	0.068702
T4b	10.29	0.001764
T5b	2.6	0.018706
T6b	10.8	0.002866
T2c	1	0.104489
T3c	1	0.034541
T4c	2.1	0.006173
T5c	1.2	0.017576
T6c	8.7	0.003529
T2d	1	0.072541
T3d	1	0.405063
T4d	1.1	0.022139
T5d	6.29	0.002457
T6d	15.5	0.000562
T2e	1	0.161161
T3e	1	0.059347
T4e	1	0.016585
T5e	9.57	0.001150
T6e	14.67	0.000906

The function of the fitting curve is

$$f(t)=\frac{0.03202}{t+0.3732}. \tag{4.1}$$

This function cannot reflect the search efficiency of the algorithm completely, the number of particles should be considered. The number of particles is 30, and the product of the iteration number of the optimum and the number of particles is the number of searched particles. Therefore, the experiential function that reflects the number of searched particles and the length of the solution interval is calculated as:

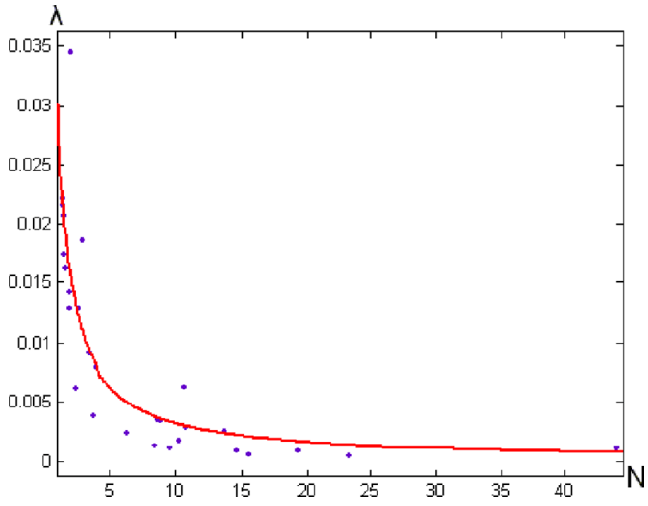
$$y=\frac{0.9606}{x+11.196}. \tag{4.2}$$

In the pure random search algorithm, the number of searched particles and the probability of the final solution, which is the ratio of the length of the solution interval to the length of the total region, are reciprocal, i.e., $y = 1/x$. Throughout the comparison with function (4.2), we can see that the value of x (the number of search algorithms) is much less than that of the random search algorithm, for the same value of y (the search probability). This scenario proves that PSOs have strong search capabilities.

Table 3 N and λ from examples C21–C63

Example	N	λ
C21	3.1	0.009219
C31	1.3	0.016380
C41	2.3	0.012821
C51	10.75	0.006203
C61	19.25	0.000928
C22	1	0.060606
C32	1.6	0.012821
C42	3.6	0.008059
C52	44	0.001166
C62	3.4	0.003832
C23	1	0.218182
C33	1.7	0.034483
C43	8.875	0.003462
C53	23.44	0.000461
C63	8.5	0.001316

Figure 2 Data points and fitting curve (see online version for colours)



5 Disturbance strategy

Although PSOs have strong search capabilities, the algorithm can easily become stuck in a local optimum when the solution interval for an excellent packing is very narrow. Thus, disturbance strategies should be added to the algorithm to interfere with the search of the PSO.

This paper presents seven disturbance strategies: eliminate-inferior-particle routine (EIPR), restart routine (RR), vary-inferior-particle routine (VIPR), quasi-GA routine (QGAR), rotate-moving routine (RMR), quasi-gradient-method routine (QGMR) and rotate-disturbance routine (RPR).

EIPR: A specific proportion (the algorithm selects 10%) of the lower-fitness particles is selected in a swarm, and these particles are initialised at random.

RR: All of the particles are initialised.

VIPR: A specific proportion (the algorithm selects 10%) of the lower-fitness particles is selected in the population; then, the partial dimensions of the selected particles are initialised at random. These dimensions are selected by a random number. For example, the variation threshold is set at 2/3. To decide whether the dimension is initialised, a random number from 0 to 1 is generated by the computer. If the random number is more than or equal to 2/3, the dimension is initialised. If the random number is <2/3, then the dimension is not handled.

QGAR: Many pairs (the algorithm uses 10% of the total) of the lower-fitness particles are selected in the population; then, the partial dimensions of these pairs are exchanged. These dimensions are selected by a random number.

RMR: Because a particle in a swarm is seven-dimensional, in the course of execution, the movement of the particles is divided into seven steps, from α_1 to ω_3, successively. At every step, the particle moves only in this dimension.

QGMR: In the course of execution, the next direction of a particle's movement is defined by the following function:

$$q_j = \frac{f_1 - f_0}{p_{1j} - p_{0j}} + \frac{f_1 - f_b}{p_{1j} - p_{bj}}, \tag{5.1}$$

where p_{1j}, p_{0j}, p_{bj} are the current and previous positions of this particle and the jth dimensional position of the best particle, respectively, and f_1, f_0, f_b are the current and previous fitting values of the particle and the fitness of the best particle, respectively. This paper prescribes the length of the velocity vector to be the maximum step length that is prescribed in the PSO. Then, the velocity in every dimension can be defined by the length of the velocity vector and the direction vector.

The upper six routines are the individual disturbance strategies, but the rotate-disturbance routine is a combined disturbance strategy.

RPR: EIPR, RR, VIPR and QGAR are used in rotation.

In this paper, examples C61–C63 are used for disturbance strategy comparison experiments. In an experiment, each disturbance strategy is used 10 times, and there are 100 iterations in a single instance. Furthermore, the numbers of each disturbance strategy in the calculations are maintained to be approximately equal. Tables 4–6 show the data from the experiments. In the tables, the data in the '*Average number*' are obtained by the method as follows: the number of iterations to obtain the final solution, regardless of whether it is the optimum, is the sample data. The average of the sample data are the average number of iterations. The data in the '*Max ratio*' column are the maximum utilisation ratio. The data in the '*Min ratio*' column are the minimum utilisation ratio. The data in the '*Ave ratio*' column are the average utilisation ratio. The data in the '*Ts*' column are the number of times required to obtain the maximum utilisation ratio.

Table 4 Test of the disturbance strategies for example C61

Name	*Max ratio*	*Min ratio*	*Ave ratio*	*Average number*	*Ts*
No	0.985313	0.983229	0.9844626	22.1	4
EIPR	0.985313	0.983229	0.9845835	54.8	5
RR	0.985313	0.983854	0.9849169	8	6
VIPR	0.985313	0.983854	0.9851671	48.1	9
QGAR	0.985313	0.983229	0.9845001	7	4
RMR	0.985313	0.983229	0.9840782	67.7	3
QGMR	0.985313	0.967708	0.9798568	18	3
RPR	0.985313	0.981250	0.9838925	20.4	5

Table 5 Test of the disturbance strategies for example C62

Name	*Max ratio*	*Min ratio*	*Ave ratio*	*Average number*	*Ts*
No	0.991771	0.991771	0.991771	3.4	10
EIPR	0.991771	0.991771	0.991771	5.4	10
RR	0.991771	0.991771	0.991771	5.8	10
VIPR	0.991771	0.991771	0.991771	8.8	10
QGAR	0.991771	0.991771	0.991771	8.5	10
RMR	0.991771	0.991771	0.991771	39.5	10
QGMR	0.991771	0.991771	0.991771	18.2	10
RPR	0.991771	0.991771	0.991771	7.2	10

Table 6 Test of the disturbance strategies for example C63

Name	*Max ratio*	*Min ratio*	*Ave ratio*	*Average number*	*Ts*
No	0.981667	0.974479	0.9802294	10.2	8
EIPR	0.981667	0.981667	0.981667	10.5	10
RR	0.981667	0.981667	0.981667	16.8	10
VIPR	0.981667	0.981667	0.981667	18.3	10
QGAR	0.981667	0.981667	0.981667	13.6	10
RMR	0.980625	0.974479	0.9785426	30.7	5
QGMR	0.981667	0.981667	0.981667	18	10
RPR	0.981667	0.975521	0.9810524	15.8	9

As can be seen in Tables 4–6, each strategy has definite superiority; however, none of the strategies has absolute superiority in every example. If the 'average utilisation ratio' and the 'times' are the two indices that are compared, then the vary-inferior-particle routine is outstanding. In the three test examples, the number of times to obtain the maximum utilisation ratio is 9 times, 10 times and 10 times, and these are maxima in terms of the strategies. However, the average numbers of iterations are large, specifically 48.1, 8.8 and 18.3, respectively. These values are relatively large for the corresponding strategies. Inspecting the other strategies, the following phenomenon is roughly shown: for solutions that are relatively good, the number of iterations is large; for solutions that are not so good, the number of iterations is less. Therefore, if the two aspects are considered together, then the restart routine is excellent. In the three test examples, the strategy rankings of the "average utilisation ratio" are second, first and first, respectively, and the rankings of the "average number of iterations" are second, third and fifth, respectively. It is observed that this strategy not only can obtain a good solution but also can obtain a lower number of iterations compared with the other strategies.

Through the descriptions of strategies, we know that, for some strategies, the result of which the particles are handled by the strategy relate to the global best particle, such as the rotate-moving routine and the quasi-gradient-method routine, while in other strategies, the result of which the particles are handled by strategy is random and does not relate to the globally best particle, such as the eliminate-inferior-particle routine and the vary-inferior-particle routine. Therefore, whether the result of a particle is random, the above seven strategies can be divided into two types: the random-type strategy and the non-random-type strategy. The random-type strategies are the eliminate-inferior-particle routine, the restate routine, the vary-inferior-particle routine, the quasi-GA routine, and the rotate-disturbance routine. The non-random-type strategies have a rotate-moving routine and a quasi-gradient-method routine. Through the data of the above experiments, we can see that the experimental results of these two non-random-type strategies are not good, and the results of the restart routine and the eliminate-inferior-particle routine, for which the level of randomness is relatively high, are excellent. These results show that the randomise strategy helps in the searched region for this algorithm. This result arises from the complexity of the packing problem. Through the graph of the fitness function (Figure 1), we can see that the lengths of the solution interval are different, and the values of the fitness are not regular. For such a fitness function, the algorithm should have a strong capability for a global search. The randomise strategy can increase the capability to a certain extent.

6 Conclusions

We researched the search efficiency of a PSO in the packing problem, and this paper presents the concept of the "solution interval of packing" for the phenomenon that the iteration numbers of equal level examples are different. The solution interval also reveals the complexity of the rectangular packing problem because the whole solution region consists of many independent solution intervals and there is no regularity between the solution intervals. The results determine that the optimisation algorithm has a strong capability for performing global searches. Nevertheless, we have the goal to discover how to increase the capabilities of a global search. First, the disturbance strategies and combinations of strategies that this paper has presented can increase these capabilities. However, there are still many strategies and combinations that should be tested and researched. Additionally, the algorithm has much room for improvement. For example, the ordering rule that sequences

by the area is too rigid for the ordering of rectangular items and restricts the searching scope of the algorithm. The ideas in this paper are expected to be helpful for further research on this algorithm.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China under Grant 60975046. The authors wish to express their appreciation to the supporter. Similarly, the authors would like to thank the anonymous referees for their constructive comments and suggestions.

References

Dyckhoff, H. (1990) 'A typology of cutting and packing problems', *European Journal of Operational Research*, Vol. 44, pp.145–159.

Engelbrecht, A.P. (2009) *Fundaments of Computational Swarm Intelligence*, Tsinghua University Press, Beijing.

Garey, M. and Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York.

Gilmore, P.C. and Gomory, R.E. (1961) 'A linear programming approach to cutting-stock problem', *Operational Research*, Vol. 9, pp.849–859.

Hopper, E. (2000) *Two-Dimensional Packing Utilising Evolutionary Algorithms and Other Meta-Heuristic Methods*, Cardiff University, UK.

Hopper, E. and Turton, B. (2001) 'An empirical investigation of meta-heuristic and heuristic algorithm for a 2D packing problem', *European Journal of Operational Research*, Vol. 128, pp.34–57.

Huang, J., Xu, W., Sun, J. and Dong, H. (2006) 'Study on layout problem using quantum-behaved particle swarm optimization algorithm', *Chinese Journal of Computer Applications*, Vol. 26, No. 12, pp.3015–3018.

Jakobs, S. (1996) 'On genetic algorithms for the packing of the polygons', *European Journal of Operational Research*, Vol. 88, pp.165–181.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', *Proceedings of IEEE International Conference on Neural Networks*, IEEE Service Center, Piscataway, pp.1942–1948.

Li, G., Zhao, F., Guo, C. and Teng, H. (2006) 'Parallel hybrid PSO-GA algorithm and its application to layout design', *Lecture Notes in Computer Science*, Vol. 4221, pp.749–758.

Liu, D., Tan, K., Huang, S., Goh, C. and Ho, W. (2008) 'On solving multi-objective bin packing problems using evolutionary particle swarm optimization', *European Journal of Operational Research*, Vol. 190, pp.357–382.

Lodi, A., Martello, S. and Monaci, M. (2002) 'Two-dimensional packing problems: a survey', *European Journal of Operational Research*, Vol. 141, pp.241–252.

Ntene, N. and van Vuuren, J.H. (2009) 'A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem', *Discrete Optimization*, Vol. 6, pp.174–188.

Qi, Y. and Wang, J. (2011) 'The particle swarm optimization algorithm for solving rectangular packing problem', *Advanced Materials Research*, Vol. 186, pp.479–483.

Valério de Carvalho, J.M. and Guimarães Rodrigues, A.J. (1995) 'An LP-based approach to a two-stage cutting stock problem', *European Journal of Operational Research*, Vol. 84, pp.580–589.

Wang, J. and Yang, W. (2005) 'Dynamic attractive factors applied in packing problems', *Chinese Journal of Computer Aided Design & Computer Graphics*, Vol. 17, No. 8, pp.1725–1729.

Wang, J., Wang, B. and Zhu, Y. (2010) 'Solving rectangular packing problem using a hybrid SAGA algorithm', *Chinese Journal of Tianjin University of Technology and Education*, Vol. 20, No. 2, pp.5–9.

Wäscher, G., Haußner, H. and Schumann, H. (2007) 'An improved typology of cutting and packing problems', *European Journal of Operational Research*, Vol. 183, pp.1109–1130.

Xu, Y., Yang, G., Bai, J. and Pan, C. (2011) 'A review of the application of swarm intelligence algorithms to 2D cutting and packing problem', *Lecture Notes in Computer Science*, Vol. 6728, pp.64–70.

Zhou, C., Gao, L. and Gao, H. (2005) 'Particle swarm optimization based algorithm for constrained layout optimization', *Chinese Control and Decision*, Vol. 20, pp.36–40.