Exploration of An Optimized Comprehensive Algorithm for Goods Stacking Based on The Simulated Annealing Algorithm and Genetic Algorithm

Peng Li*, Yong Tang, Bin Yang
College of Automobile and Traffic Engineering, Nanjing Forestry University, Nanjing, Jiangsu, 210037, China
*Corresponding author (E-mail: lipengqf@nuaa.edu.cn)

Abstract
Along with intensification of market competition, and in order to guarantee steady and efficient operation of production and to obtain the maximum economic interests, the original simple, partial and conventional control and experience-based management can no longer meet requirements of modern production. All corporate managers and control engineers are faced with the following problems, including how operation decision-making and organization of production should change along with changes of the raw material supply and product demands on the market; how the production process should be controlled under the condition that the production plan is changed so that production flexibility can be maximized; how management and decision-making should be carried out under the prerequisite of not dramatically changing the production process so that comprehensive economic interests of corporate production can be maximized. In the highly automatic system, reasonable and efficient operation of the production process has become extremely complex. An effective set of computer scheduling and control strategies is required. This justifies the necessity of studying the issue of scheduling. Since most scheduling issues are non-deterministic polynomial complete problems (NP), there have not yet been any efficient solution strategies. Thus, research into the issue of scheduling also holds huge theoretical significance. This paper finds out that the simulated annealing algorithm (SAA) can give full play to advantages of the simulation algorithm and the genetic algorithm. This can not only narrow down the search range of feasible solutions, but also avoid getting stuck in the partial optimal solution. During the practical operation process, the SAA can effectively improve the space utilization rate of the tank, and provide vigorous theoretical basis for reduction of corporate costs and human resource waste.

Key words: Annealing Algorithm, Genetic Algorithm, Optimized Comprehensive Algorithm

1. INTRODUCTION
Along with development of science and technology, the production scale has been expanding and becoming increasingly complex (Hui, 2010). The market competition has been intensifying (Ghezavati and Nia, 2015). All this has raised a higher requirement of corporate management and monitoring of the production process (Jonathan, Bruno and Hamid, 2010). In the recent decades, various production processes have undergone dramatic changes (Nallakumarasamy, Srinivasan and Raja, 2011). Large-scale production and continuity of the production process have become two striking characteristics (Liu and Ye, 2014). In order to solve the above problems, Dr. Harrington proposed the concept of computer integrated manufacturing in 1973 (Liu, Sun and Yan, 2011). Computer integrated manufacturing or CIM is a new philosophy to organize, manage and operate corporate production (Lin, Chang and Lie, 2010). It relies on computer software and hardware, and makes a comprehensive utilization of modern management techniques, manufacturing techniques, information techniques, automation techniques, and systematic engineering techniques to organically integrate and optimize operation of elements related to humans, technology and operation as well as information flow and material flow (Abdi, Fathian and Safari, 2012). All in all, CIM can help improve product quality and reduce consumption, and help enterprises win the market competition. According to the above definition of CIM, CIMS or computer integrated manufacturing system is a practical system integrated based on philosophy.

A method to obtain the maximum of the utility function and the profit function, \( u(x) \), is to adopt the objective function as the fitness function. However, many optimization issues aim at obtaining the minimum of the cost function, \( g(x) \); while the genetic algorithm requires the fitness function to be positive. Meanwhile, the higher the degree of fitness is, the better the individual will be. Therefore, under many occasions, the objective function of problems is adopted as the measure of the objective function. Apart from transforming the objective function into the form of maximum, the fitness function should be non-negative. Below is a major method for conversion:

\[
 f' = af + b
 \]
Obviously, there are many ways to choose the coefficient, \( C_{\text{max}} \). \( C_{\text{max}} \) can be a suitable input or the maximum, \( g(x) \), during the evolution process or the maximum, \( g(x) \), in the current group. Of course, \( C_{\text{max}} \) should better be irrelevant to the group.

A commonly-used method is to convert the fitness function into the reciprocal of the objective function, namely:

\[
f(x) = \frac{1}{g(x)}
\]  

When objective function used for problem-solving is a profit function, the following transform can be adopted to ensure the non-negativity:

\[
f' = f - (\bar{f} - c \sigma)
\]  

Where, the coefficient, \( C_{\text{max}} \), can be a proper input value or a maximum of the current generation or the former generation or the function of the group variance.

2. SCALING CALCULATION OF FITNESS

In designing the genetic algorithm, one will have to cope with a population scale ranging from dozens to hundreds. The scale is seriously deviated from the species scale in the real world. Therefore, adjustment of the individual reproduction quantity is extremely important to genetic operation. If there is a super individual observed in the group, or that the fitness of the individual far exceeds the average fitness of the population, the individual will take up an absolute proportion in the population if the selection is based on the fitness proportion. Consequently, the algorithm can converge at a local optimal point at an earlier date. The phenomenon is called premature convergence. Under the condition, the fitness of the individual should be shrunk to reduce the competitiveness of the super individual and to prevent premature convergence. On the other hand, in the later period of the search process, though the population has an adequate degree of diversity, the average fitness of the population might get close to the optimal fitness of the population. Under the condition, there is almost no competition in the population. As a result, target searching can hardly get improved and even stops. In response to the situation, the fitness of the individual should be amplified to strengthen competitiveness of the individual.

The shrinking adjustment of fitness is known as scaling of the fitness function. After introduction of the fitness function scaling by De Jong, scaling has become an important technique to maintain the competitiveness during the evolution process. So far, methods of scaling mainly include the following ones:

2.1. Linear scaling

Assume that the original fitness function is \( f \), and that the fitness function, after scaling, is \( f' \). Then, linear scaling can be written as below:

\[
f' = af + b
\]  

Where, coefficients, \( a \) and \( b \), can be set by different means, but should meet the following two conditions:

1) The mean value of fitness should be equal to the mean value of fitness after scaling;
2) The maximum of the fitness function, after scaling, should be equal to the designated multiple of the mean value of the original fitness function.

In other words,

\[
f'_{\text{max}} = C_{\text{mul}} \cdot f_{\text{avg}}
\]  

Where, \( C_{\text{mul}} \) denotes the copied number of the expected optimal population. The experiment suggests that, in a not too typical group \( n = 50 \sim 100 \), the value of \( C_{\text{mul}} \) can range between 1.2 and 2.0.

2.2. \( \sigma \) interrupt

\( \sigma \) interrupt is a pre-processing method before the above linear scaling. Its mainly purpose is to avoid the negative value of fitness after scaling. Below is the mathematic expression of \( \sigma \) interrupt:

\[
f' = f - (\bar{f} - c \sigma)
\]  

Where, the constant, \( c \), should be properly chosen.

2.3. Power scaling

Power scaling can be directly defined as below:

\[
f' = f^k
\]  

Where, the power index, \( K \), is related to the problem solving, and can be rectified during the algorithm process.
3. CROSSOVER OPERATOR SIMULATION

3.1. Single-point crossover model

A crossover point is randomly set in the individual string. During the crossover, part of the structure of the two individuals before or after the point are exchanged to form two new individuals. See Table 1 and Fig. 1.

The method is mainly applied to the real number encoding. The two male parents are set as below:

\[ S_1 = \left( v_1^{(0)}, v_2^{(0)}, \ldots, v_m^{(0)} \right) \]  
\[ S_2 = \left( v_1^{(2)}, v_2^{(2)}, \ldots, v_m^{(2)} \right) \]

Part of the male parent vector is chosen to generate a random number, \( a \), in the section of \((0,1)\). Then, its offspring can be written as below:

\[ S_1' = \left( v_1^{(0)}, v_2^{(0)}, \ldots, v_k^{(0)}, ak^{(0)} + (1-a)v_k^{(1)} + \ldots + (1-a)v_m^{(0)} \right) \]

\[ S_2' = \left( v_1^{(2)}, v_2^{(2)}, \ldots, v_k^{(2)}, ak^{(2)} + (1-a)v_k^{(1)} + \ldots + (1-a)v_m^{(2)} \right) \]

Table 1. Analysis coordinates and cargo demands

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Coordinate</th>
<th>Quantity demanded</th>
<th>Serial number</th>
<th>Coordinate</th>
<th>Quantity demanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1302,3313)</td>
<td>30</td>
<td>17</td>
<td>(3514,1756)</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>(3738,1314)</td>
<td>80</td>
<td>15</td>
<td>(3816,3158)</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>(2155,3322)</td>
<td>80</td>
<td>16</td>
<td>(2071,3350)</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>(3513,1388)</td>
<td>70</td>
<td>18</td>
<td>(3560,3313)</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>(3266,1434)</td>
<td>50</td>
<td>30</td>
<td>(3757,3456)</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>(3337,1447)</td>
<td>50</td>
<td>31</td>
<td>(2038,3636)</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>(3336,1338)</td>
<td>20</td>
<td>33</td>
<td>(2373,3831)</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>(2187,1022)</td>
<td>80</td>
<td>33</td>
<td>(3238,1806)</td>
<td>60</td>
</tr>
<tr>
<td>9</td>
<td>(2313,580 )</td>
<td>80</td>
<td>32</td>
<td>(3405,3357)</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>(2367,450 )</td>
<td>50</td>
<td>34</td>
<td>(3382,3723)</td>
<td>60</td>
</tr>
<tr>
<td>11</td>
<td>(3005,1850)</td>
<td>70</td>
<td>37</td>
<td>(3238,3301)</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>(3473,1547)</td>
<td>20</td>
<td>35</td>
<td>(3834,3320)</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>(3566,1281)</td>
<td>20</td>
<td>36</td>
<td>(3120,3440)</td>
<td>70</td>
</tr>
<tr>
<td>14</td>
<td>(3361,1757)</td>
<td>20</td>
<td>38</td>
<td>(3424,3345)</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>(1333,784 )</td>
<td>30</td>
<td>30</td>
<td>(3556,3637)</td>
<td>40</td>
</tr>
</tbody>
</table>

Figure 1. Optimal location and delivery result of logistics distribution center: a. Site selection results of standard genetic algorithm; b. Site selection results of speed update + adaptive mutation

3.2. Crossover simulation of two points

Similar with one-point crossover, only two cross points are set in the process of two-point crossover randomly. The code strings are exchanged between the two cross points, as shown in Figure 2.
The improvement of goods combination efficiency in logistics warehousing can not only effectively save storage space and reduce warehousing costs, but also enhance the competitiveness of enterprises. However, the traditional linear programming and heuristic methods used to realize the optimization of large scale goods combination cannot meet the needs of the increasingly information-based logistics industry. Therefore, this paper studies the advantages and disadvantages of simulated annealing algorithm and genetic algorithm in solving large-scale combination optimization. According to the research results, the simulated annealing algorithm has a strong local search ability to prevent the whole search process from falling into local optimal solution. However, its ability to grasp the whole search process is relatively weak, which is not conducive to bringing the whole process into the most promising area, leading to the low computing efficiency of the simulated annealing algorithm. In contrast, the local search ability of genetic algorithm is weak. Premature convergence is prone to occur, and the evolution is slow. However, it has excellent global searching ability, high parallel processing ability and strong robustness. It can be found that the combination of the two algorithms can give full play to their advantages, overcome the shortcomings of their independent operation, and achieve a perfect combination. Based on the combination of the advantages of the simulated annealing algorithm and the genetic algorithm, this paper examines the validity of the combined simulated annealing genetic algorithm by transforming storage shelves into encasement problems, which provides theoretical guidance for practical application.

Assume that there is a regular rectangular box with a known size and a batch of goods with different sizes. The question is how to realize the largest space volume rate highest or the highest load utilization rate on the premise that the box meets certain constraints so as to achieve the best economic benefits.

The maximum loading volume and maximum loading quality of the box are assumed as \( V \) and \( G \). The length, width, height and mass of the goods \( i \) to be loaded are \( l_i, w_i, h_i, m_i \). The objective function of the question is maximization function.

\[
\text{Max } z = \lambda (l_i w_i h_i \varphi_i) / V + (1-\lambda) \sum_{i=1}^{n} (m_i \varphi_i) / G
\]  

Where,

- \( z \) —— Objective maximization function
- \( \lambda \) —— \( 0 \sim 1 \) variable. When the objective function is the largest volume rate of the box, \( \lambda = 1 \). When the objective function is the maximum utilization rate of loading mass, \( \lambda = 0 \).
- \( \varphi \) —— \( 0 \sim 1 \) variable. When \( \varphi_i = 1 \), goods \( i \) has been loaded. When \( \varphi_i = 0 \), goods \( i \) is not loaded.

4. THE CONSTRAINTS OF THE PACKING PROBLEM

Volume constraint of goods loading: The total volume of the goods to be loaded shall not exceed the maximum loading capacity of the box.

Mass constraint of goods loading: The total mass of the goods to be loaded shall not exceed the maximum mass loaded by the box.

Direction constraint of goods loading: The goods to be loaded should be restrained in the direction of loading. For example, some goods should not be loaded upside down or diagonal. At present, direction constraint of goods loading is generally divided into three categories: arbitrary rotation, horizontal rotation, and no rotation.

Sequence constraint of goods loading: Different goods are required to be delivered in and out of the warehouse at different time, and the priorities are different. For example, the goods strict in time should be placed near the exit to guarantee the priority of access.

The maximum function can be described as:

![Figure 2](image-url)
\[
\exp\left(-\frac{\Delta f}{T_0}\right) \approx 1
\]  
(13)

Where, \(\Delta f = f(j) - f(i)\). In other words, as long as the temperature is high enough, the initial requirements of the acceptance probability of 1 can be satisfied. However, the initial temperature \(T_0 \to \infty\) cannot be realized in practical application. According to the principle of compromise, the temperature range is selected between 100 and 300. Although this practice has some empirical basis, in the application of this question, this paper takes \(\exp\left(-\frac{\Delta f}{T_0}\right) \geq 95\%\) as the principle determining the initial temperature. \(a, a_1, b_1, b_2, c\) are barycentric coordinate boundary of goods after loading.

A reasonable cooling mechanism can effectively control the attenuation of \(T\). Smaller attenuation is conductive to expanding the search range and find the optimal solution with higher quality. In this paper, the simplest attenuation function is used:

\[
t_{s+1} = a t_s, \quad k \geq 0, \quad 0 < a < 1
\]  
(14)

In genetic algorithm, the search process depends on the fitness function value, instead of external information. Fitness function directly affects the convergence speed of genetic algorithm, and even the generation of optimal solution. Therefore, selecting a suitable fitness function is of great importance to the success of an algorithm. Here we select the objective function as the fitness function.

\[
f(i) = \frac{\sum_{i=1}^{n} m_i \phi_i}{G} \times 100\% \quad \text{or} \quad f(i) = \frac{\sum_{i=1}^{n} l_i \phi_i}{V} \times 100\%
\]  
(15)

In order to reflect the difference between the optimized genetic algorithm and the traditional genetic algorithm, this paper simulates several models, and the simulation parameters of each model are as shown in Table 2, Figure 3 and Figure 4.

**Table 2. Simulation parameter setting results of each model**

<table>
<thead>
<tr>
<th></th>
<th>Standard genetic algorithm</th>
<th>Speed update + adaptive mutation</th>
<th>Field mean + speed update + adaptive mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial inertia weight (W_1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Final initial inertia weight (W_2)</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Initial acceleration constant (c_{11}, c_{21})</td>
<td>1, 1</td>
<td>1, 1</td>
<td>1, 1</td>
</tr>
<tr>
<td>Final acceleration constant (c_{1F}, c_{2F})</td>
<td>1, 1</td>
<td>2.5, 2.5</td>
<td>2.5, 2.5</td>
</tr>
<tr>
<td>(c_j)</td>
<td>--</td>
<td>--</td>
<td>1.5</td>
</tr>
<tr>
<td>Maximum Iterations</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Mutation rate (k)</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Number of particles</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

**Figure 3. Model iterative fitness variation:**

(a) Field mean + speed update + adaptive mutation; (b) Speed update + adaptive mutation
There are many kinds of encoding forms in genetic algorithm, and the most commonly used is binary encoding. However, binary encoding is somewhat weak to solve the packing problem. Firstly, when the 3D coordinate of an object is represented in binary system, the encoding is too long, which rapidly expands the search space and decreases the algorithm performance. Secondly, the adjustment of coordinate values only describes the change of the position of the goods to be transferred, leading to the cross phenomenon and substantially increasing binary numerical disk to be processed. In view of the disadvantage of binary code in dealing with bin packing problem, this paper adopts the coding form based on genetic gene. The goods are sorted in descending order according to the volume, $i = 1, 2, 3, \ldots, n$, so that the same kind of goods can be put together as much as possible in loading to avoid producing too much space debris, which is conductive to improving the utilization ratio of space. Each loading scheme must consider the constraints of the loading direction and loading sequence of the goods in encoding, and a symbol string $S = (s_1, s_2, s_3 \ldots s_n, s_{n+1}, \ldots, s_{2n})$ with the coding length of $2n$ is obtained.

Where $s_i \sim s_n$ is an integer representing the serial number of goods. It is a sequence of non repeated integers generated by $[1, n]$ randomly, and represents the loading sequence of the goods corresponding to the serial number. $s_{n+1} \sim s_{2n}$ is the rotation direction of goods, and represented by numbers 1, 2 and 3. 1 represents random rotation, 2 represents horizontal rotation, and 3 represents no rotation.

There is a one-to-one correspondence between $s_i$ and $s_{n+i}$. That is, when the ith goods $s_i$ is to be loaded, the constraint of loading direction $s_{n+i}$ should be considered. After decoding the initial solution, $S = (s_1, s_2, s_3 \ldots s_n, s_{n+1}, \ldots, s_{2n})$ will be converted into a set of feasible solutions $a = (a_1, a_2, a_3 \ldots a_m, a_{m+1}, \ldots, a_{2m})$. $m$ represents the quantity of goods to be loaded, and it can be found that not all the goods will be loaded into the box. When the loaded goods are $a = (a_1, a_2, a_3 \ldots a_m, a_{m+1}, \ldots, a_{2m})$, we first judge whether $a_i$ can be loaded in the box. If it cannot be loaded in the box, we will go to the next one $a_{i+1}$. If it can be loaded in the box, we need to determine if the space inside the box is enough to load the goods until the box cannot load any goods.

5. CONCLUSION

The selection mechanism of genetic algorithm aims to ensure that the individuals with large fitness function values have higher survival rate, so as to improve the global convergence and search efficiency of genetic algorithm. Different selection methods have various effects on the performance of genetic algorithms. At present, the most widely used two methods are random traversal method and optimal individual preservation method. This paper selected the optimal individual preservation method. That is, the maximum fitness of the offspring generated by parent genetic manipulation is compared with that of the parent. If it is smaller than the parent highest fitness, the individuals with the highest fitness value is directly replaced by the worst individual in the offspring. It can be found that the simulated annealing genetic algorithm gives full play to the advantages of simulation algorithm and genetic algorithm. It not only narrows the search range of the feasible solution, but also avoids the possibility of falling into the local optimum solution. In the actual operation process, the algorithm effectively improves the space utilization rate of the box so as to provide a strong theoretical basis for enterprises to reduce costs and save manpower.

ACKNOWLEDGEMENTS

National Natural Science Foundation of China, Study on three-dimensional air cargo loading problem of
REFERENCES


