Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 3650923, 17 pages
https://doi.org/10.1155/2019/3650923

Research Article

Application of Layered Coding Genetic Algorithm in Optimization of Unequal Area Production Facilities Layout

Shiwang Hou,[1,2] **Haijun Wen**,[3] **Shunxiao Feng**,[3] **Hui Wang**,[3] **and Zhibin Li**[3]

[1]*Department of Mathematics, Brunel University London, London UB8 3PH, UK*
[2]*School of Business, Huaihua University, Huaihua Hunan-418000, China*
[3]*School of Mechanical Engineering, North University of China, Taiyuan Shanxi-030051, China*

Correspondence should be addressed to Shiwang Hou; shiwang.hou@brunel.ac.uk

Received 8 April 2019; Revised 25 May 2019; Accepted 28 May 2019; Published 19 June 2019

Academic Editor: Maciej Lawrynczuk

Unequal area facilities layout problem (UA-FLP) is an inevitable problem in the process of new construction, reconstruction, and expansion of enterprises. The rationality of the facilities layout has a great influence on the operation performance of the production system. Finding the optimal solution of UA-FLP according to the requirement of production process is the main content of the plant design. The facilities were constrained by given areas and aspect ratio, respectively. By adopting the method of slicing tree, the layout space was divided into multiple regions for each facility. The genetic algorithm was developed by using layered coding to show the slicing process. Considering the production logistics cost as well as the adjacency relations between the facilities, the goal function was established and the optimal solution was obtained by running the proposed algorithm. Finally, the feasibility of the proposed approach was validated by a set of known problems. The comparison results show that it can provide decision support for rapid optimal layout of multifacilities.

1. Introduction

UA-FLP was proposed originally by Armour and Buffa in [1], and its objective is to determine the good locations for a given set of departments with different areas on some workshop floor to optimize the material handling cost and/or other objectives. The facility can be small or big according to the level of the facility layout, but it should be a physical entity with some function. There is about 20% to 50% of the processing cost used for material handling, and scientific and reasonable facilities layout can save at least 10% to 30% of the material handling fee [2, 3].

Classic FLPs tend to study equal-area facilities arrangement; i.e., all facilities have the same area and shape. In this case, the facility centroid is fixed and the overall closeness or distance between facilities will not change when switching the location of any two facilities. But, the equal-area hypothesis is impractical, and the facility area is often unequal. So, the centroid of each facility depends on its area and shape and has no regular distribution as equal-area case.

Heuristic algorithm based on the discrete model is proposed in reference [4] to deal with the UA-FLP. This approach divided the district into some small squares with a fixed area, and each facility was allocated some numbers of squares most close to its area by use of heuristic algorithm. As shown in Figure 1, the grid size determines the precision of the facility representation; the smaller the grid size, the more precise the facility representation. The facility shape was also represented well by this approach. But, the time consumption is great to calculate the interference between facilities during the detailed layout process. Also, this approach is easy to produce facility layout with irregular shape. By far, the most commonly used UA-FLP model is mainly based on block diagram with unequal area as depicted in [5].

Many researchers have studied the optimization methods for UA-FLP. The literature about the methods can be mainly divided into three categories. The first one is deterministic algorithm to calculate accurate solution, such as mathematical programming [6, 7], mixed-integer linear programming [8], and QAP as aforementioned. The second

1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	8
1	3	3	3	3	3	1	1	1	1	1	6	6	6	6	8
2	3	3	3	3	3	3	3	3	3	5	6	6	6	6	8
2	4	4	4	4	4	3	3	3	3	5	6	6	6	6	8
2	4	4	4	4	4	3	3	3	5	5	7	7	7	8	8
2	4	4	4	4	4	3	3	3	5	5	7	7	7	8	8
2	4	4	4	4	4	3	3	3	5	5	7	7	7	8	8

Figure 1: Facility layout diagram of discretization of the planar grid.

category is heuristic algorithm mentioned in reference [9–11]. Another category is intelligent algorithm, such as genetic algorithm (GA) [12, 13], particle swarm optimization (PSO) [14, 15], ant colony algorithm (ACA) [16, 17], simulated annealing algorithm (SAA) [18, 19], and tabu search algorithm [20].

Deterministic algorithm has high requirement in memory and CPU time, so it is often used to solve small-scale UA-FLP and reference [21] proved its effectiveness. Heuristic algorithm and intelligent algorithms have faster calculation speed and are suitable for large-scale UA-FLP. Furthermore, genetic algorithm is most widely used, and different genetic algorithms were developed for various UA-FLPs.

A genetic algorithm hybridized with local search to obtain the Pareto solutions set was proposed in [22], and the author adapted a random weight to combine values of two objectives. For unequal area facility layout problems, a genetic algorithm based on slicing structure was developed in [23], and four objective functions such as material handling costs, aspect ratio, closeness, and distance requests were considered simultaneously by use of a Pareto-based evolutionary approach. In order to improve the performance of premature convergence, lack of diversity, and high computational cost, an island model genetic algorithm was proposed in [24]. The compared results showed the proposed approach has great improvement on the above aspects. In [25], a multiobjective interactive genetic algorithm was proposed by considering both quantitative aspects and subjective features, which allowed the interaction between the expert designer and the algorithm. A biased random-key genetic algorithm to determine the placement order and dimensions of each facility was proposed in [26], and the results showed its better performance for 19 of the 28 benchmark facility layout problems.

For large-scale UA-FLP problems, a genetic algorithm combined with a decomposition strategy was proposed in [27]. Compared with basic genetic algorithm, the experiments in the paper showed that the proposed approach had an average solution improvement of 6% or 7% for large-scale instances with 90 or 100 facilities.

This paper put forward a method of LCGA (layered coding genetic algorithm) for slicing-based plane splitting to lay out facilities. This approach can generate feasible solution rapidly with the help of the layered coding method. It can provide larger-scale UA-FLP solving a new thinking. The rest of this paper is organized as follows: Section 2 provides a description of the UA-FLP dealt with in this paper. The design process of the proposed layered coding genetic algorithm is presented in Section 3. Section 4 compares the proposed approach with some known UA-FLPs. The conclusions are presented in Section 5.

2. UA-FLP Description

2.1. Location Relations of UA-FLP. Assume that all facilities are rectangular blocks with a given area. Considering the building module, facilities need to satisfy a given aspect ratio constraint to avoid producing an approach with long narrow shape facilities layout. All facilities must be located in a given area and cannot overlap between them. Take a UA-FLP involving facilities i and j; for example, the relative position relations are as shown in Figure 2.

Figure 3(a) shows the projection polygon of facility space. Figure 3(b) shows the projection polygon of facility and its necessary space, including transport corridor, operational space, and maintenance space for workers. To facilitate the facilities layout, Figure 3(c) considers the rectangular envelope of Figure 3(b) as the objects of UA-FLP. So, the necessary horizontal and vertical spacing between facilities shown in Figure 2 can be decomposed to the corresponding facility (as shown in Figure 4). By doing so, the amount of computation to solve UA-FLP can be decreased without losing the accuracy of result.

2.2. Objective Function and Constraints. Satisfying the abovementioned basic hypothesis, the UA-FLP becomes the following optimization problem: arranging n facilities to a specified area to minimize the material handling cost and maximize the closeness scores.

2.2.1. Minimizing the Material Handling Cost.

$$\text{Min } G_1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} d_{ij}, \tag{1}$$

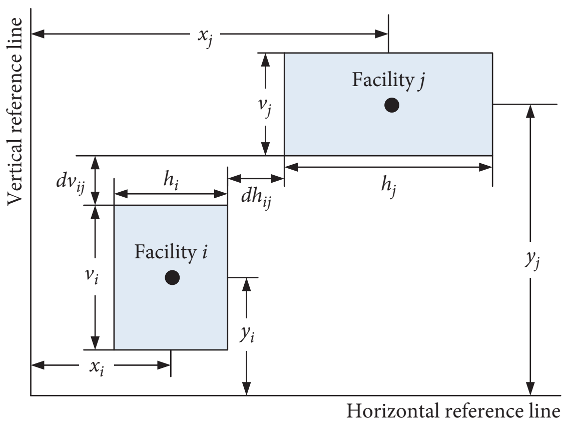


Figure 2: Location relations diagram of the facility layout. h_i: horizontal length of facility i; v_i: vertical width of facility i; x_i: x-coordinate of facility centroid relative to the ordinate origin; y_i: y-coordinate of facility centroid relative to the ordinate origin; dh_{ij}: the necessary horizontal spacing between facility i and facility j; dv_{ij}: the necessary vertical spacing between facility i and facility j.

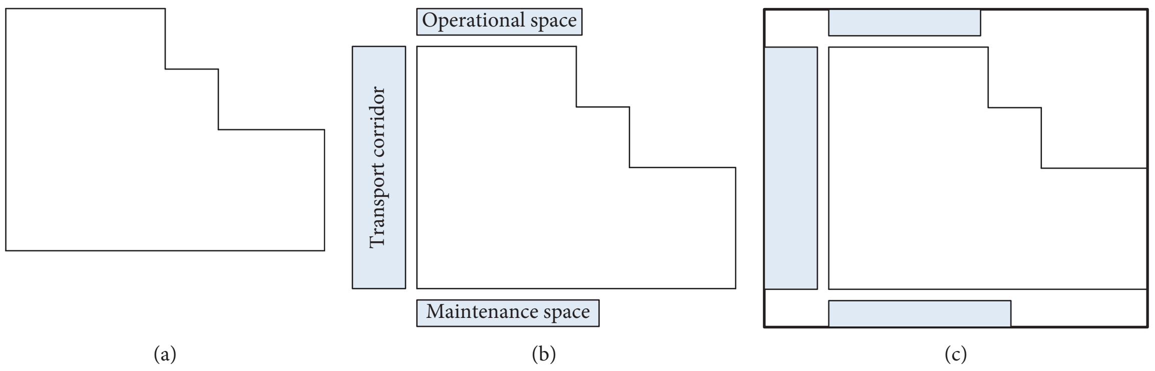


Figure 3: Diagram of facility planar space. (a) Projection polygon of facility space. (b) Projection polygon of facility and operation space. (c) Rectangular envelope of projection polygon of (b).

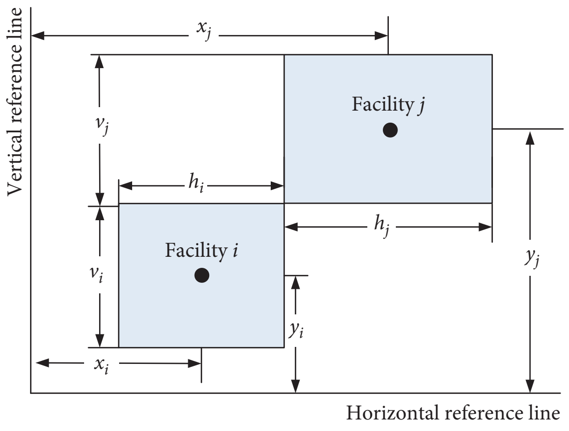


Figure 4: Simplified graphic of the location relation between facilities.

where n is the number of facilities, c_{ij} is the per unit handling cost between facility i and facility j, f_{ij} is the logistics quantity between facility i and facility j, and d_{ij} is the distance between facility i and j.

2.2.2. Maximizing the Closeness Scores.

$$\text{Max } G_2 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} r_{ij}, \tag{2}$$

where n is the number of facilities and $b_{ij} \in [0, 1]$ is the ratio of distance between facility i and facility j to the maximum distance between facilities in a given layout approach, and is used to represent their closeness factor between two facilities. r_{ij} is the quantitative score of the nonlogistics relationship level determined by the SLP method (Table 1).

This problem belongs to multiobjective optimization problem (MOOP) since there is more than one objective function to be optimized simultaneously. Optimal decisions need to be taken in the presence of tradeoffs between two or more conflicting objectives. Taking UA-FLP as example, material handling cost is minimized while closeness scores are maximized while locating facilities. For a nontrivial multiobjective optimization problem, no single solution exists that simultaneously optimizes each objective. There exist a (possibly infinite) number of Pareto optimal

Table 1: Nonlogistics relationship level and corresponding quantitative score.

Level	*A*	*E*	I	*O*	*U*	*X*
Meaning	Absolutely necessary	Especially important	Important	Ordinary important	Unimportant	Closeness undesirable
Score	4	3	2	1	0	−1

solutions; i.e., under these solutions, none of the objective functions can be improved in value without degrading some of the other objective values. All these Pareto optimal solutions are considered equally good if there is no additional subjective preference information.

There are many kinds of forms of solutions for different goals, such as a representative set of Pareto optimal solutions and/or a single solution that satisfies the subjective preferences of decision maker. In this paper, we adopt the latter in order to obtain a well-determined layout approach.

Let M be the maximum of r_{ij} and w_1 and w_2 be the weight coefficients of material handling cost and nonlogistics closeness, respectively. The above two goals can be synthesized into a minimizing objective function:

$$\begin{aligned}\text{Min } G &= w_1 G_1 + w_2 (M - G_2) \\ &= w_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} d_{ij} \\ &\quad + w_2 \left(M - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} r_{ij} \right),\end{aligned} \tag{3}$$

subject to

$$\frac{UR_i(x) - LL_i(x)}{UR_i(y) - LL_i(y)} \geq \alpha^{\min}, \tag{4}$$

$$\frac{UR_i(x) - LL_i(x)}{UR_i(y) - LL_i(y)} \leq \alpha^{\max}, \tag{5}$$

$$\frac{UR_i(x) - LL_i(x)}{UR_i(y) - LL_i(y)} = A_i, \tag{6}$$

where $UR_i(x)$, $LL_i(x)$, $LL_i(y)$, and $UR_i(y)$ denote the x-coordinate and y-coordinate of upper-right corner and lower-left corner of facility i, respectively. Formulas (4) and (5) ensure the aspect ratio of facility i in a given range $[\alpha^{\min} \ \ \alpha^{\max}]$, and formula (6) constrain the area of facility i equal to given value A_i.

Manhattan distance is adapted to calculate the distance between facilities, namely:

$$d_{ij} = \left| x_i - x_j \right| + \left| y_i - y_j \right|, \tag{7}$$

where x_i and y_i denote the x-coordinate and y-coordinate of the centroid. The exact coordinate position of facility i, $\{LL_i(x,y), UR_i(x,y)\}$, can be obtained by the following formula:

$$\begin{aligned} x_i &= LL_i(x) + \frac{[UR_i(x) - LL_i(x)]}{2}, \\ y_i &= LL_i(y) + \frac{[UR_i(y) - LL_i(y)]}{2}. \end{aligned} \tag{8}$$

3. Algorithm Design

3.1. Basic Layout Principle for UA-FLP Using Slicing Tree. In order to locate all facilities into a given area, the area should be divided into subareas with the same number of facilities. This paper adopted plane segmentation method to generate slicing tree in order to describe the relative position relationships of facilities during locating them.

For a UA-FLP with n facilities, the slicing tree contains n leaf nodes and $(n-1)$ internal nodes. The information of the plane segmentation mode is contained in internal nodes, i.e., horizontal split (labeled H) or vertical split (labeled V). Taking a UA-FLP with 5 facilities as an example, a feasible plane segmentation approach is shown in Figure 5.

The process of plane segmenting is the process of facilities layout. The change of coordinates for each partition after each plane segmenting is described as follows. The area has only two points: $O(0,0)$ and $O'(\sum h_i, \sum v_i)$, when there is no facility located. The coordinate of upper-right corner is marked with the maximum limit value; i.e., x-coordinate is the sum of width of all facilities supposing they are side-by-side arranged horizontally, and y-coordinate is the sum of length of all facilities supposing they are side-by-side arranged vertically. Taking the first partition as an example, the splitting process is as follows. Suppose AB is the first cutting line. The plane area is divided into two partitions. The upper part is for facilities 1 and 2, and the lower part is for facilities 3 to 5. After the first segmentation, the coordinates of point A and B are $A(0, \sum_{i=3,4,5} v_i)$, $B(\sum h_i, \sum_{i=3,4,5} v_i)$.

Similarly, the other four divisions can be done. Finally, all facilities are located in different parts, and the precise coordinates of the facilities are as follows: facility 1$\{A, (x_A + h_1, y_A + v_1)\}$, facility 2$\{F, (x_F + h_2, y_F + v_2)\}$, facility 3 $\{O, (x_O + h_3, y_O + v_3)\}$, facility 4$\{G, (x_G + h_4, y_G + v_4)\}$, and facility 5$\{D, (x_D + h_5, y_D + v_5)\}$. So, a feasible layout approach is provided. Based on this layout idea, this paper studies the hierarchical coding genetic algorithm in order to realize the layout scheme iterative optimization.

3.2. Genetic Algorithm Design. For a UA-FLP with n facilities, there are $n!$ possible combinations of their position and the number of combination will be larger if the shape or

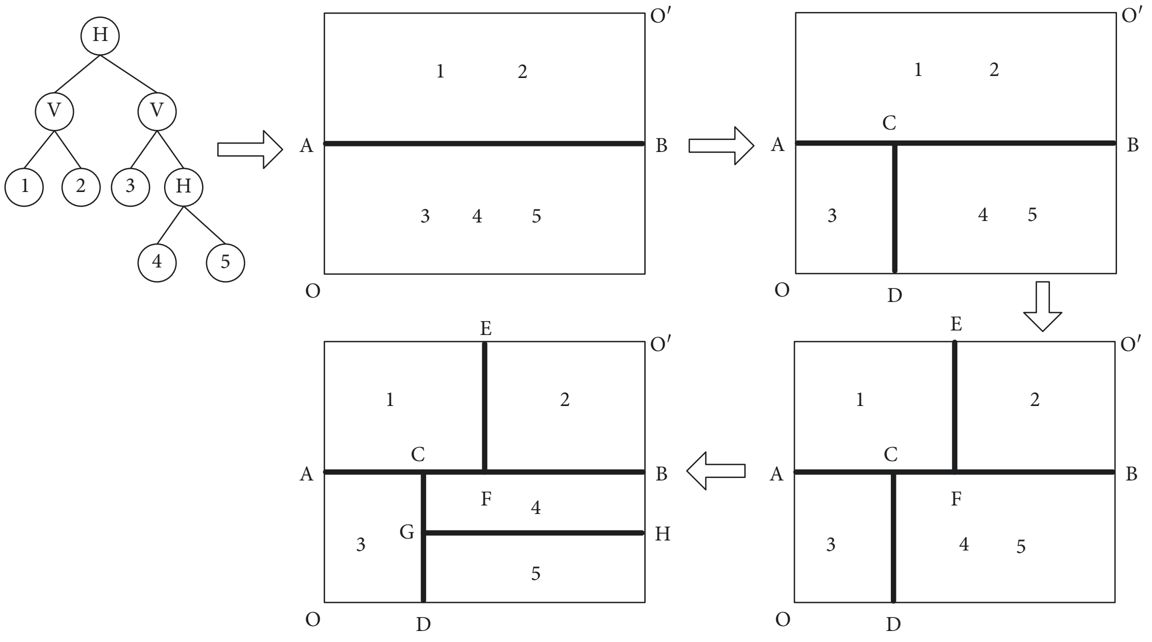


FIGURE 5: Cutting tree layout process of the plane segmentation method.

orientation of each facility is considered. Above all, there is a large number of local optimum in this huge solution space. So, it is a NP-hard problem, and it is advisable to find a suboptimal solution in acceptable cost (money, time, or computing resource) by use of some heuristics search algorithms.

GA is a bionic algorithm for searching the optimal solution based on the principle of biological evolution. It simulates the natural process of gene recombination and evolution and compiles the parameters into binary-code or decimal-code (or other codes) genes. Several genes constitute a chromosome (individual), and many chromosomes carry out operations similar to natural selection, pairing crossover, and mutation. The final optimization result is obtained after repeated iterations (that is, generation inheritance).

3.2.1. Layered Coding Approach. Coding is to map the phenotype data in solution space into genotype data in genetic structure. During iterations of GA, a coding string represents a solution and genetic operations are done by operating the bits of this string. So, the coding method also affects the genetic operators.

There are mainly two coding methods: real number coding and binary coding. The former uses a real number as a gene, is easy to understand, and does not need decoding process, but it is also easy for premature convergence, thus falling into local optimum. The latter uses a binary string with specific length as a gene and has higher stability, larger population diversity, and better performance for global search. In this paper, we adopted the binary coding method. The number of bit is determined by the accuracy of the solution to be achieved.

For example, suppose an x-coordinate ranging in [0, 4] and the solution is exactly 4 decimal places behind the decimal point. The solution space is divided into (0-1)* (1e + 4) = 10,000 equal fractions. It takes 14 bits of binary to represent a solution; i.e., the coding of a solution is a 14 bit binary string since $2^{13} < 10000 < 2^{14}$. The decoding process is as follows:

$$x_{\text{coordinate}} = 0 + \text{decimal}(\text{chromosome}) * \frac{(4-0)}{2^{14}-1}. \tag{9}$$

Generally, for $x \in [\text{lower\_bound}, \text{upper\_bound}]$, the value of x after decoding is

$$x = \text{lower\_bound} + \text{decimal}(\text{chromosome}) * \frac{(\text{upper\_bound} - \text{lower\_bound})}{2^{\text{chromosome\_length}} - 1}. \tag{10}$$

The coded chromosome string should represent the following information simultaneously: facility sequence for layout, splitting point sequence, and splitting mode. The coding approach will be detailed below.

Facility sequence code is in the first layer. N facilities are coded by n different integers in the interval [1, n] by use of integer coding. The coding string can be any sequences of n integers in the interval [1, n] to allow different facility locating orders. Figure 6 shows a code string [1–13] of 13 facilities, f1, . . ., f_{13}, with a random order of integers from 1 to 13, and the value of each bit of code string denotes the number of the facility it represents.

Splitting point code lies in the second layer. A feasible splitting point lies between every two bits of facility coding string. For every two adjacent bits of facilities coding string, there is a splitting point. For a n-bit facility coding string, there are $n-1$ splitting points. We denote every splitting

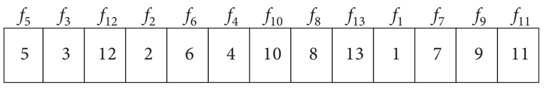

f_5 f_3 f_{12} f_2 f_6 f_4 f_{10} f_8 f_{13} f_1 f_7 f_9 f_{11}
5 3 12 2 6 4 10 8 13 1 7 9 11

Figure 6: Facility sequence code string of 13 facilities.

position as an integer between 1 to $n-1$. For the facility code string in Figure 6, its corresponding splitting position is shown in Figure 7.

The facility set is divided into two different parts at the splitting point selected firstly. The remainder splitting points are contained in these two subsets of facilities. After each splitting operation, the number of subsets of facilities increases by 1. After $n-1$ splitting, the facility set with n facilities will be divided into n single facilities locating in n different area blocks. For the code string of splitting point, we code them as a sequence from 1 to $(n-1)$ with random order corresponding to different plane segmentation approaches. The value of each bit of code string denotes the number of the splitting position in the facility code string. Taking the string shown in Figure 6 as an example, there are 12 positions that can be set as splitting points when carrying out plane segmentation. Suppose we produce a splitting point code string as [1–12], the splitting operation will begin with the 7th splitting position, then 4th, and finally, 6th. The splitting process is shown in Figure 8.

The last layer of coding provides the information of splitting mode, horizontally or vertically. The splitting result of these two ways is different, and the facility layout is also different. We use 0 for horizontal and 1 for vertical. So, the splitting mode code is a binary string that has the same number of bits as the splitting point code. For the example as shown in Figure 8, assume a splitting mode code is [0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0]. Figure 9 shows a complete three-layer coding string or a chromosome of a feasible solution for a UA-FLP with 13 facilities. For the first step, the splitting mode is horizontal and the process of plane segmentation is similar to the description in Section 3.1.

So, for a UA-FLP with n facilities, the three-layer coding string can be expressed as shown in Figure 10.

3.2.2. Crossover Operation. In genetic algorithms and evolutionary computation, crossover, also called recombination, is a genetic operator used to combine the genetic information of two parents to generate new offspring. It is one way to stochastically generate new solutions from an existing population. Newly generated solutions are typically mutated before being added to the population.

The coding method determines the data structures to store genetic information and also affects the crossover operators.

Due to the aforementioned layered coding structure, genetic operation must be carried out by the segment to ensure the feasibility of the new code string. Taking the chromosome gene string in Figure 7 as an example, crossover points are selected in three layers, respectively. The two parent individuals swap the gene segments before and after the crossover points in three layers, respectively.

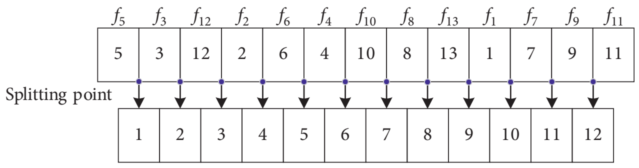

f_5 f_3 f_{12} f_2 f_6 f_4 f_{10} f_8 f_{13} f_1 f_7 f_9 f_{11}
5 3 12 2 6 4 10 8 13 1 7 9 11
Splitting point
1 2 3 4 5 6 7 8 9 10 11 12

Figure 7: Bit value of splitting point sequence code.

A crossover operation of layered coding string of 5 facilities is shown in Figure 11.

It can be found that some bit value of new code strings will lose or repeat in the first two layers after crossover operation. So, repair operation is necessary. Sort the missing value in ascending order and replace the repeated value of corresponding layer code strings. Taking the above offspring as example, the layer 1 code of offspring is [5 3 1 5 1]. The reappeared values are 5 and 1, and the missing values are 2 and 4. We repair the offspring code of layer 1 as [5 3 1 2 4]. After similar repairment, the offspring code of layer 2 is [4 1 2 3].

3.2.3. Mutation. Mutation changes one or more gene values in a chromosome from its initial state in order to maintain genetic diversity from one generation of a population to the next. This can also prevent the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. For the layered code, we also adopted different mutation operators.

A mutation operator involves a probability P_{m} that an arbitrary bit in a genetic sequence will mutate from its original state. A common method of implementing the mutation operator involves generating a standard uniform-distributed random number R for each bit in a sequence.

The corresponding bit will be modified if $R > P_{\mathrm{m}}$. This single-point mutation is suitable for third-layer binary code of an individual.

For the first- and second-layer code, mutation operation is implemented by interchanging two genes of code string in order to ensure the feasibility of new individual after mutation. The gene bits to mutate are selected randomly and also use a standard uniform-distributed random number R to determine whether or not the selected bits will be swapped.

3.3. Processing for UA-FLP with Empty Space. The method proposed herein split the area into subareas that are consistent with the number of facilities, each of which accommodates a facility. For a UA-FLP with empty space, we firstly need to turn it into a UA-FLP without empty space. The detailed process is explained as follows.

Suppose the ratio of the original region for all the facilities is $r = W/H$ and the area of i-th facility is A_i, we can arrange all the facilities in a new region, marked by the red border in Figure 12, with the area equal to the sum of all facilities since all necessary spaces for each facility has been included in its area as illustrated in Figure 3. The width W'

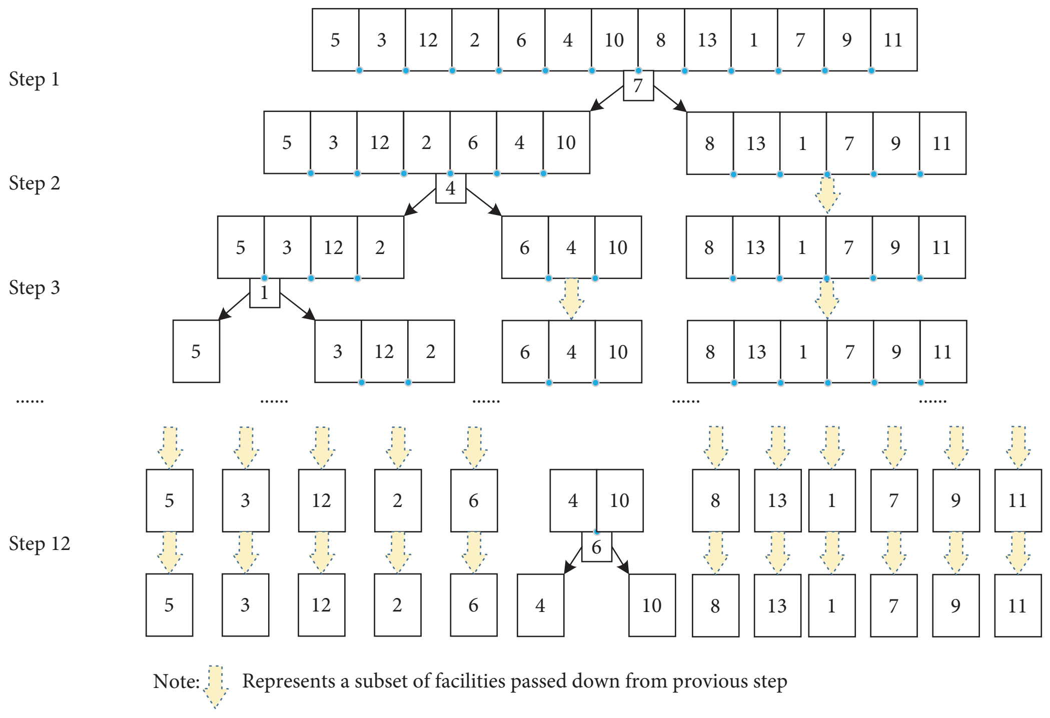


Figure 8: Splitting process according to a given splitting point sequence.

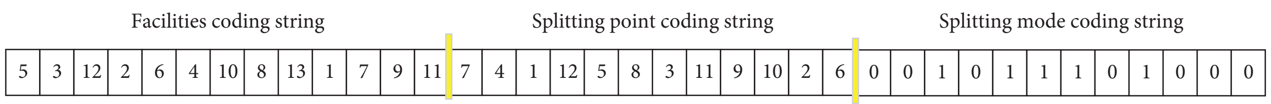


Figure 9: A complete three-layer coding string of UA-FLP with 13 facilities. (a) n-bits facilities coding string, (b) $(n-1)$-bits splitting point coding string, (c) $(n-1)$-bits splitting mode coding string.

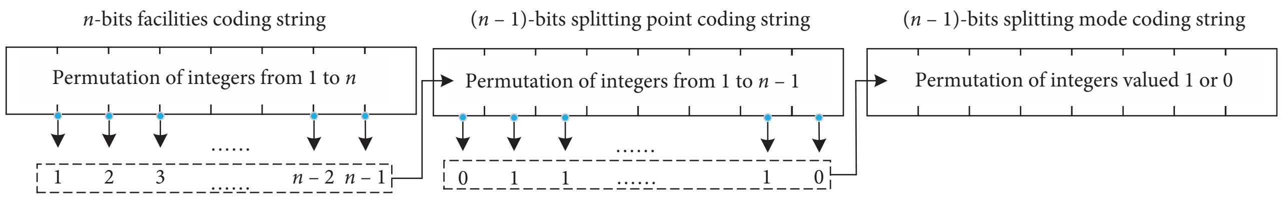

n-bits facilities coding string
(n – 1)-bits splitting point coding string
(n – 1)-bits splitting mode coding string
Permutation of integers from 1 to n
Permutation of integers from 1 to n – 1
Permutation of integers valued 1 or 0
1 2 3 ...... n – 2 n – 1
0 1 1 ...... 1 0

Figure 10: Layered coding chromosome gene string.

and height H' of this new region should satisfy the following equations:

$$\begin{cases} \dfrac{W}{H} = \dfrac{W'}{H'}, \\ W' * H' = \displaystyle\sum_{i=1}^{n} A_i. \end{cases} \tag{11}$$

In general, solutions found in this way will be better than solutions found in other methods because the horizontal and vertical coordinates of each facility will be smaller.

4. Case Study

In order to validate the proposed approach, a set of problems described in the literature was used in this section. All the tested problems are shown in Table 2.

The algorithm is coded in Matlab 2015b. The computer's configuration running the algorithms was as follows: Intel Core i5-4460 (3.20 GHz), 8 GB RAM, and a Windows 10 operating system.

The algorithm parameters settings for the tested problems are listed in Table 3, and the algorithm proposed in this paper is described in detail in Section 4.1.

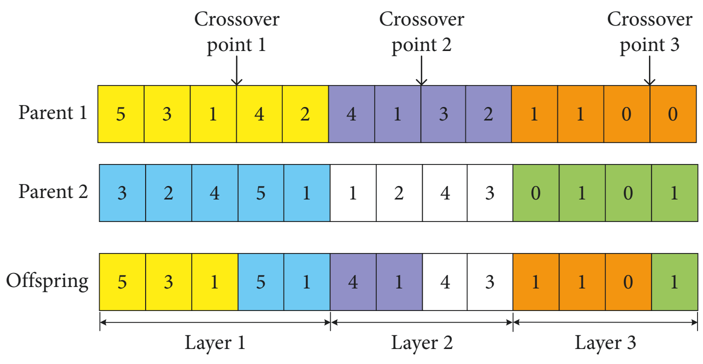


FIGURE 11: Layered coding crossover operation. (a) Parent 1. (b) Parent 2. (c) Offspring.

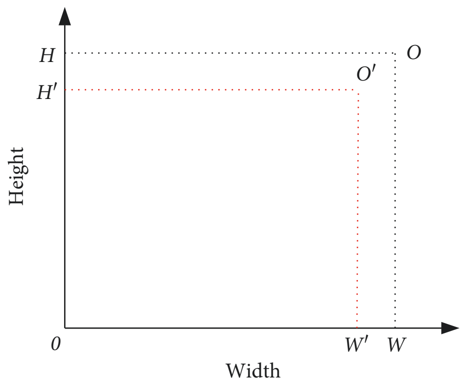


FIGURE 12: Layout diagram for UA-FLP with empty space.

TABLE 2: Results of comparison of tested problems between LCGA and other approach.

Problems	Facility number	Best known results of reference	Facility dimension $[W, H]$	Data reference
O7	7	134.19 [24]	[8.54, 13]	Meller et al. [28]
O8	8	245.51 [24]	[11.31, 13]	Meller et al. [28]
O9	9	241.06 [29]	[12, 13]	Meller et al. [28]
F10	10	8567.00 [30]	[90, 95]	Montreuil et al. [30]
VC10	10	22899.65 [24]	[51, 25]	Van Camp et al. [31]
MB11	11	1171 [32]	[6, 6]	Bozer et al. [33]
BA12	12	8021 [20]	[6, 10]	Bazaraa [34]
BA14	14	4665.93 [20]	[7, 9]	Bazaraa [34]
AB20-ar7	20	4793.47 [20]	[2, 3]	Armour and Buffa [1]
SC30	30	3563.95 [35]	[12, 15]	Liu and Meller [32]
SC35	35	3814.98 [35]	[15, 16]	Liu and Meller [32]
P62	62	3720521 [35]	[100, 137.18]	Komarudin and Wong [36]

TABLE 3: Results of comparison of tested problems between LCGA and other approach.

Problems	Population Size	Max generations	Crossover Probability	Mutation Probability	Selection method
O7	80	1000	0.7	0.05	Fitness-based
O8	80	1000	0.7	0.05	Fitness-based
O9	100	1000	0.7	0.05	Fitness-based
F10	100	1000	0.6	0.08	Fitness-based
VC10	100	1000	0.6	0.08	Fitness-based
MB11	120	1500	0.6	0.08	Fitness-based
BA12	120	1500	0.6	0.08	Fitness-based
BA14	200	2000	0.6	0.08	Fitness-based
AB20-ar7	300	2000	0.5	0.08	Fitness-based
SC30	500	2000	0.4	0.08	Fitness-based
SC35	500	2000	0.4	0.08	Fitness-based
P62	600	2000	0.3	0.1	Fitness-based

```
    Input: Popu_size, $n\_f$
    Output: $F$(facility population), Sp[Popu_size $\times$ $n\_f$] (Splitting position string), Sm[$1 \times n\_f - 1$] (splitting mode string)
(1) for $I \longleftarrow 1$ to Popu_size
(2)     $F(i,:) \longleftarrow$ permutation of $n\_f$ integers valued from $1 \sim n\_f$;
(3)     $Sp(i,:) \longleftarrow$ permutation of $(n\_f - 1)$ integer valued from $1 \sim (n\_f - 1)$;
(4)     $Sm(i,:) \longleftarrow$ permutation of $(n\_f - 1)$ number valued 1 or 0;
(5) end for
(4) Establishment of fitness function.
```

Algorithm 1: Coding of facility population, splitting position and splitting mode layer.

4.1. Parameters Setting and Pseudocode of Algorithm

(1) *Coding Method.* Integer coding was adopted in this case. The number n of facilities was assigned to n_f. The layout approach and the splitting point sequence were denoted by two interpermutations which took values from $[1, n]$ and $[1, n-1]$, respectively. The splitting mode of each splitting operation was represented by a thirteen bits encoded string, and each bit can take value 0, splitting horizontally, or 1, splitting vertically.

(2) *Parameters Setting.* See Table 3.

(3) *The Generation of Initial Population.* According to the above approach, initial facility population, splitting position string, and splitting mode string can be generated by use of Algorithm 1.

Algorithm 2 carries out the establishment of fitness function and its value calculation.

The crossover operation and the corresponding repair approach of facility level are presented in Algorithm 3.

As for the crossover and repair operation of the splitting point level, the method is similar to the above ideas; just replace the *f_popu* and *n_f* with *so_popu* and ($n\_f - 1$). There is no need for repair operation in the splitting mode level, and its crossover operation is the same as the facility level.

The mutation operation of the facility level is shown in Algorithm 4.

The mutation operation of the splitting point level or splitting mode level can be completed by replacing the *f_popu* and *N_f* of Algorithm 4 with *so_popu* and ($n\_f - 1$) or *sp_popu* and ($n\_f - 1$).

4.2. Computational Results and Analysis. According to the above parameters setting, all the tested problems were solved by the proposed layout algorithm; the average optimal value of objective function of each tested problems was obtained as shown in Table 4 by running the proposed algorithm 30 times for each problems.

The optimal layout approach of all tested cases obtained by the proposed approach is presented in Figures 13(a)–13(l).

From Table 4, we found that the proposed approach can search better solutions than did the methods from literature for 8 of 12 tested problems. As shown, the proposed approach has more improvement when the facility number becomes bigger; for example, the improvement present can be up to 4–7% when the facility number reaches 30 and 35. Moreover, the average running time for finding the best solutions and the average total CPU running times have reduced drastically than did the approach of other literature. But, for the test case of P62 with no spare space for all the facilities, the proposed approach has a large gap from the optimal result of [35].

5. Conclusions and Prospects

Considering the constraints of area and aspect ratio of facilities, this paper proposed a slicing-tree-based binary plane segmentation method. The given area is divided into some blocks whose number equal to the number of facilities waiting for arrangement. The optimal solution was found by use of the layered coding genetic algorithm with the goal of maximizing the closeness relationship score and minimizing the material handling cost between facilities at the same time. The results of comparison of above 12 known problems between other literature methods and proposed approach show the effectiveness of the plane segmentation layout strategy and the reliability of layered coding genetic algorithm for solving the problems.

We draw the following conclusions based on the above study:

(i) It is reasonable and effective to partition the facilities layout area by use of the binary plane segmentation method, which can arrange a reasonable block area for each facility. So, the feasibility of solution was guaranteed during the iterative process of genetic algorithm.

(ii) By use of the layered coding genetic algorithm, the optimal splitting approach, i.e., optimal facility layout scheme, will be found during multiple iterative process. The result can provide decision support for actual production facility layout.

(iii) The plane splitting process was expressed well by the layered coding approach. When the UA-FLP problem changed, i.e., the facility number, facility area, and also the aspect ratio, these changed values can be assigned in the form of parameters; as a result, the corresponding optimal layout approach can be output quickly. Also, the layout approaches of different parameters can be compared easily in order to find the key influence factors.

```
     Input: $(x, y) \longleftarrow$ coordinate of facility centroid
       $D \longleftarrow$ Matrix of logistics quantity between facilities;
       $C \longleftarrow$ Matrix of unit logistics cost between facilities;
       $r \longleftarrow$ Matrix of closeness score between facilities;
       $r\_max \longleftarrow$ maximum of closeness score between facilities;
       $a1 \longleftarrow$ weight of logistics cost; $a2 \longleftarrow$ weight of nonlogistics factors;
     Output: goal_v (value of fitness function)
(1)  for $i \longleftarrow 1$ to n_f
(2)    for $j \longleftarrow 1$ to n_f
(3)      $\mathbf{d}(\mathbf{i},\mathbf{j}) = |\mathbf{x_i} - \mathbf{x_j}| + |\mathbf{y_i} - \mathbf{y_j}|$
(4)    end for
(5)  end for
(6)  $G1 \longleftarrow$ sum according to the row (sum according to the column(D.*flow_f.*flow_c));
(7)  $G2 \longleftarrow$ sum according to the row (sum according to the column(D./D_max.*relation_v));
(8)  $goal\_v \longleftarrow a1^{*}G1 + a2^{*}(r\_max\text{-}G2)$;
(5)  Genetic manipulation
```

Algorithm 2: Calculating the value of fitness function.

```
      Input: Popu_size, Pc, n_f
      Output: newPop_f (newfacility population after crossover peration and repairment)
(1)   f_popu_cro $\longleftarrow$ individuals in facility level for crossover;
(2)   so_popu_cro $\longleftarrow$ individuals in splitting position level for crossover;
(3)   sp_popu_cro $\longleftarrow$ individuals in splitting mode level for crossover;
(4)   newPop_f $\longleftarrow$ zeros(popu_size*Pc, n_f);
(5)   $r1 \longleftarrow$ sorted array of r1 from lowest to highest;
(6)   while $r1(1) = r1(2)$
(7)     Do $r1 \longleftarrow$ regenerate two random integers within [1, n_f];
(8)   end while
(9)   $r1 \longleftarrow$ sorted array of r1 from lowest to highest;
(10)  for $j \longleftarrow 1$ TO popu_size*Pc/2
(11)    newPop_f(2*j − 1,:) $\longleftarrow$ [f_popu_cro(j + popu_n*0.8/2,1 : r1(1)),f_popu_cro(j, r1(1) + 1 : r1(2) − 1),
f_popu_cro(j + popu_n*0.8/2, r1(2) : end)];
(12)    newPop_f(2 * j,:) $\longleftarrow$ [f_popu_cro(j, 1 : r1(1)), f_popu_cro(j + popu_n*0.8/2, r1(1) + 1 : r1(2) − 1),
f_popu_cro(j, r1(2) : end)];
(13)  end for
(14)  ch_temp $\longleftarrow$ integer order sequence taking value within [1, N_f];
(15)  for $i \longleftarrow 1$ TO popu_n*Pc
(16)    ind1 $\longleftarrow$ locating the position of element with multiple occurrences in newPop_f;
(17)    temp_re1 $\longleftarrow$ all elements that appear in newPop_f;
(18)    temp_re2 $\longleftarrow$ temp_re1(ind1);
(19)    ind2 $\longleftarrow$ locating the position of element in ch_temp that do not appear in newPop_f;
(20)    for $j \longleftarrow 1$ TO length(temp_re2)
(21)      ind3 $\longleftarrow$ locating the position of element in ith row of newPop_f that equals to the jth
element of temp_re2;
(22)      newPop_f (i,ind3) $\longleftarrow$ ch_temp(ind2(j));
(23)    end for
(24)  end for
```

Algorithm 3: Crossover operation and repair in facility layer.

The optimal layout approach of UA-FLP can be obtained fast by use of the plane segmentation method and layered coding genetic algorithm; the corresponding objective function value and the minimal area needed for locating facilities can also be provided. For multiproduct facility layout problem, different layout approaches for different products can be output by running the proposed algorithm in terms of the different requirement of product for logistics cost and closeness relationship between facilities, i.e., realizing a dynamic optimization for multiproduct facility layout. The facility number has a significant impact on performance of proposed algorithm, and there is the

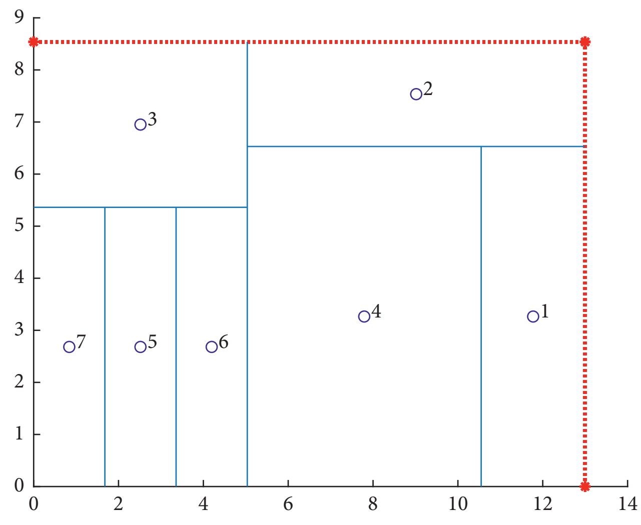
```
     Input: $f\_popu$, $n\_f$
     Output: $f\_popu$ after mutation operation
(1)  $Pm \longleftarrow$ Mutation probability;
(2)  for $i \longleftarrow 1$ TO $N\_f$
(3)    if $Pm >$ random in [0,1];
(4)      $f\_popu\_mu \longleftarrow f\_popu(i,:)$;
(5)      $r4 \longleftarrow$ unidrnd($N\_f$, 1, 2);
(6)      while $r4(1) == r4(2)$
(7)        Do $r4 \longleftarrow$ unidrnd($N\_f$, 1, 2);
(8)      end while
(9)      $f\_popu\_mu(1, r4(1)) \longleftarrow$ f_popu(i, $r4(2)$);
(10)     $f\_popu\_mu(1, r4(2)) \longleftarrow f\_popu(i, r4(1))$; $f\_popu(i,:) \longleftarrow f\_popu\_mu$;
(11)   end if
(12) end for
```

Algorithm 4: Mutation operation.

Table 4: Results of comparison of tested problems between LCGA and other approaches.

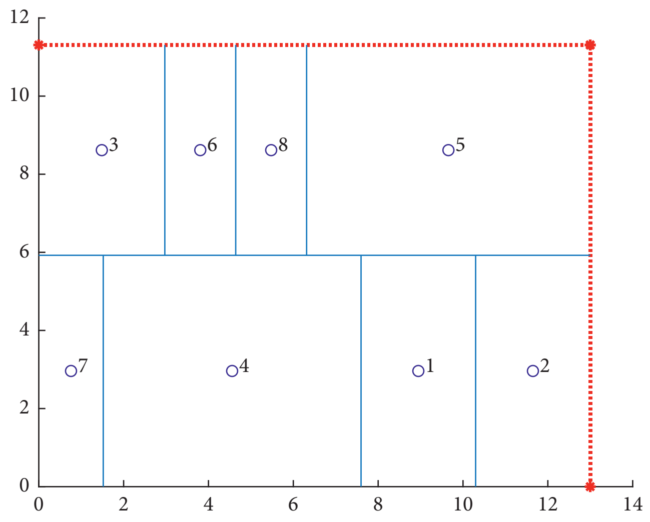
Problems	Best results of LCGA	Imp (%)	Average optimal iteration number	Average optimal searching time	Total CPU time (s)
O7	131.6773	−1.87	85	10.8	136.2
O8	245.5002	0.00	78	13.2	144.5
O9	241.0616	0.00	101	15.1	164
F10	8449.7	−1.37	152	29.5	177.4
VC10	22845	−0.24	196	37.6	180
MB11	1278.1	9.15	619	112.3	187.06
BA12	8040.8	0.25	210	38.4	190
BA14	4592.24	−1.58	235	80.1	356.1
AB20-ar7	4805.47	0.25	510	384.6	1543.2
SC30	3412.87	−4.24	780	398.7	1895.3
SC35	3519.9	−7.73	856	465.4	1931
P62	439080	18.02	1120	3178.6	7605.28

Note: Imp (%) = (the result of the proposed approach in this paper−the best solution found in the literature in Table 1)/(the best solution found in the literature in Table 1)*100.

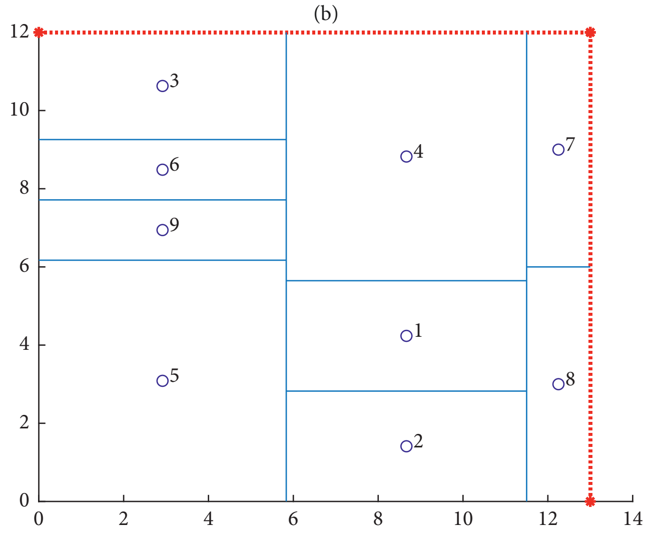


(a)

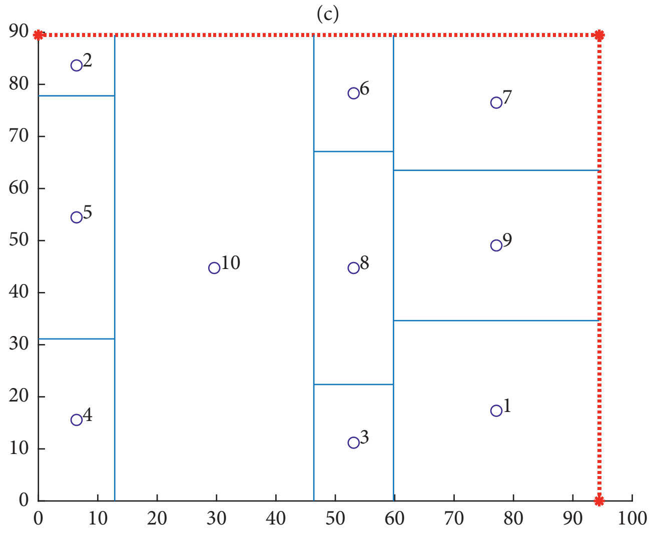
Figure 13: Continued.



(b)

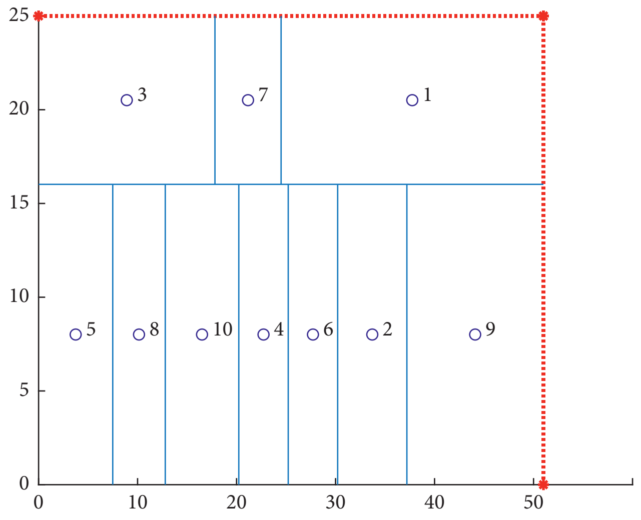


(c)

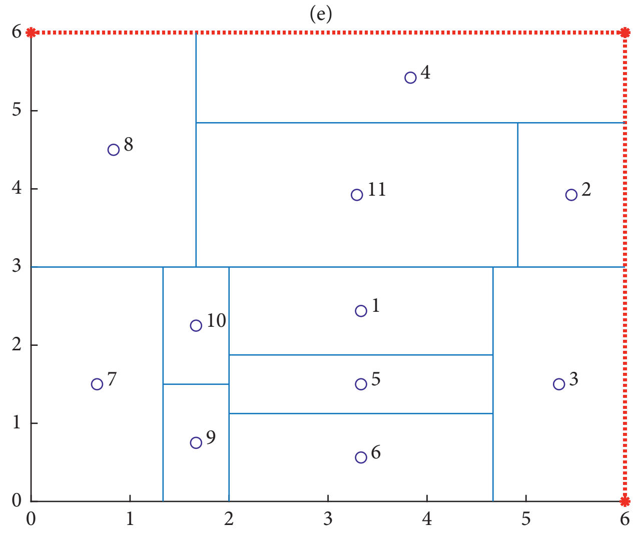


(d)

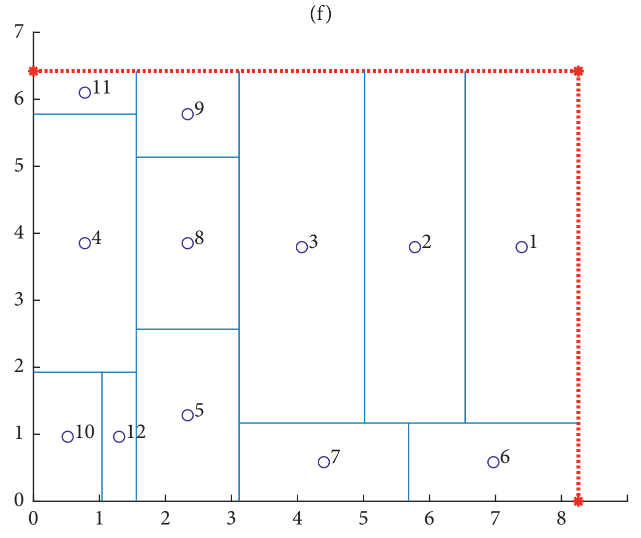
Figure 13: Continued.



(e)

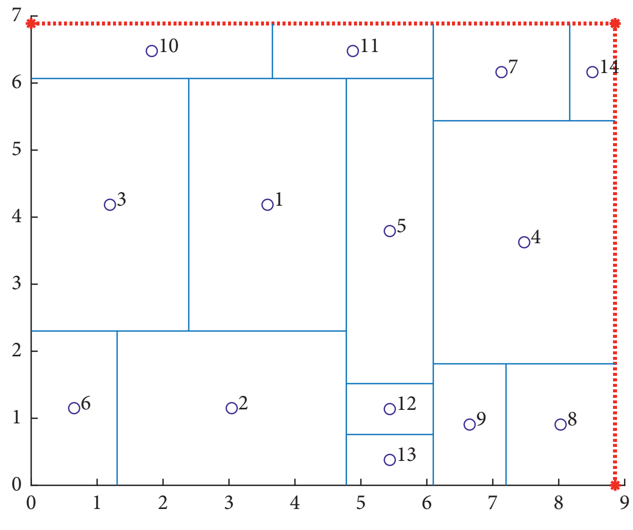


(f)

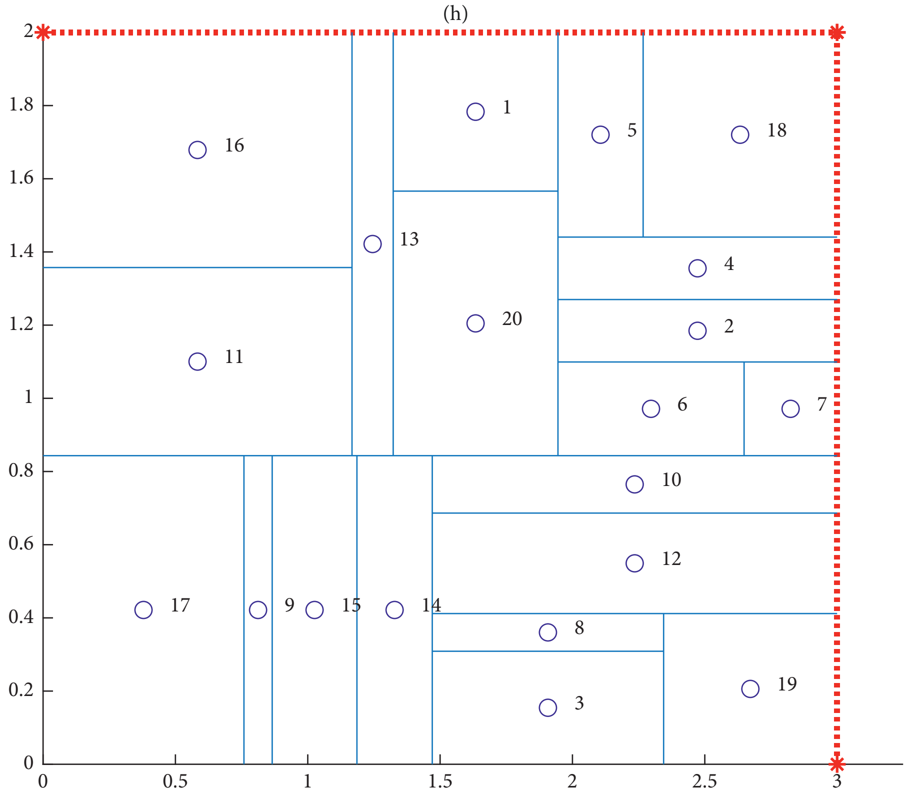


(g)

Figure 13: Continued.

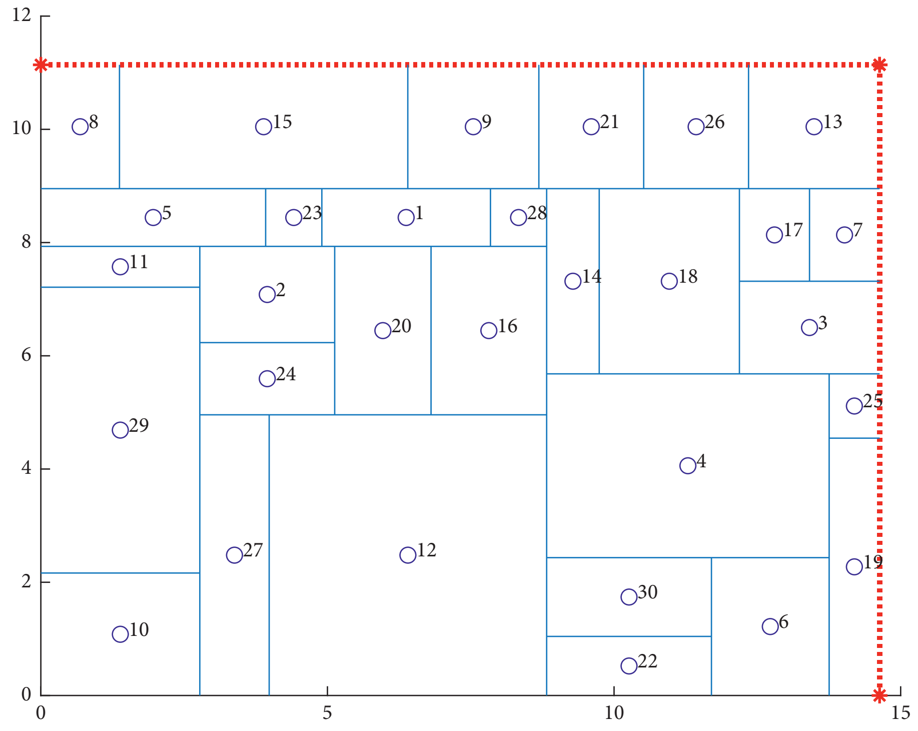


(h)

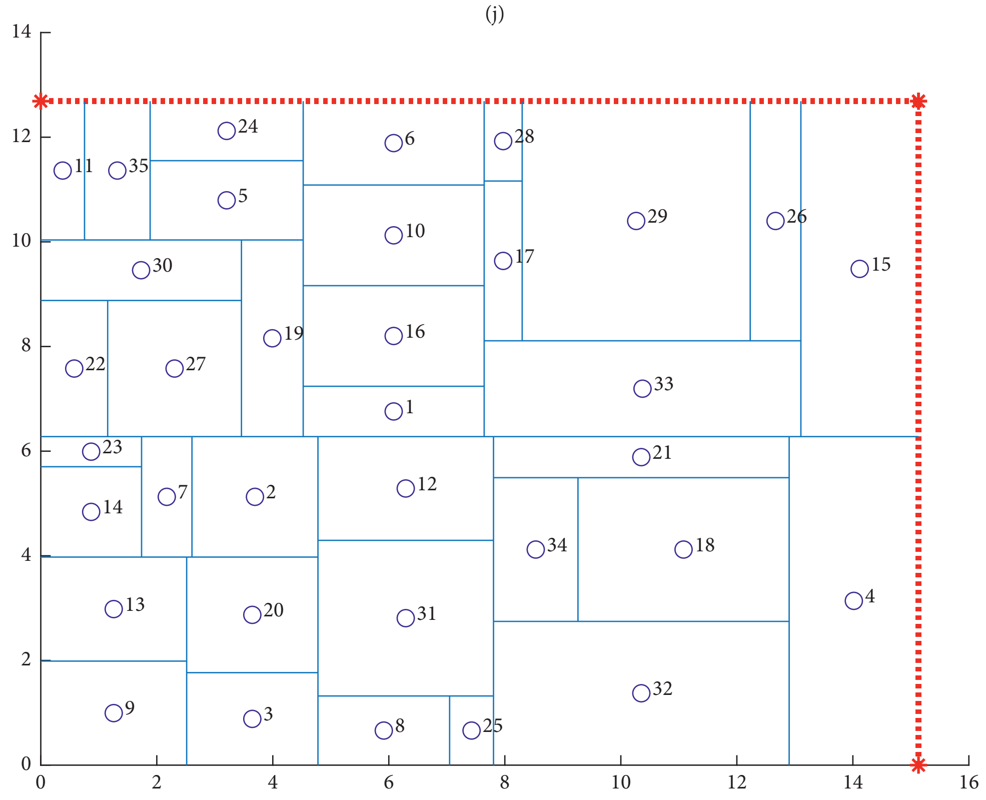


(i)

Figure 13: Continued.



(j)



(k)

Figure 13: Continued.

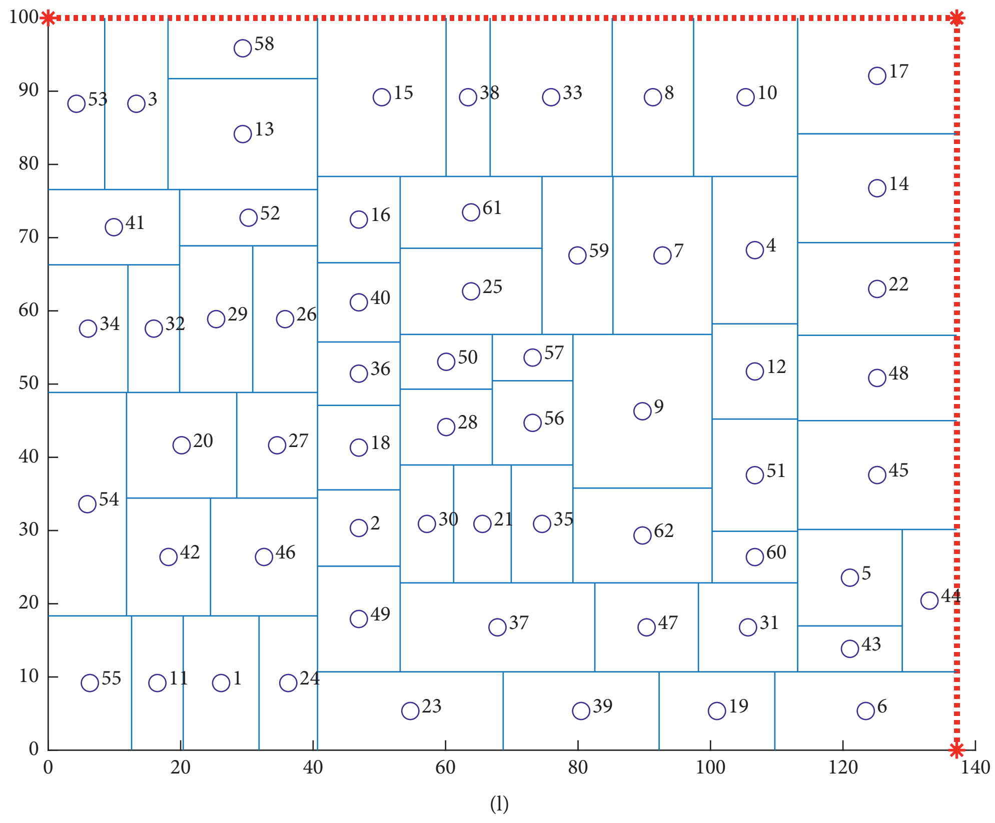


Figure 13: Optimal layout approach for the test cases. (a) Optimal layout approach for O7. (b) Optimal layout approach for O8. (c) Optimal layout approach for O9. (d) Optimal layout approach for F10. (e) Optimal layout approach for VC10. (f) Optimal layout approach for MB11. (g) Optimal layout approach for Ba12. (h) Optimal layout approach for Ba14. (i) Optimal layout approach for AB20. (j) Optimal layout approach for SC30. (k) Optimal layout approach for SC35. (l) Optimal layout approach for P62.

possibility of fall into local optimum, so the large-scale facility layout optimization algorithm and the combination scheme with the local search algorithm can be chosen as the direction of further research.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by China Scholarship Fund (CSC No. 201708430274); Scientific Research Fund of Hunan Provincial Education Department (Grant No. 16B208); Humanities and Social Sciences Project of Hunan Province (Grant No. 16YBX018); and MOE (Ministry of Education in China) Project of Humanities and Social Sciences (Grant No. 13YJC630049).

References

[1] G. C. Armour and E. S. Buffa, "A heuristic algorithm and simulation approach to relative location of facilities," *Management Science*, vol. 9, no. 2, pp. 294–309, 1963.
[2] J. A. Tompkins, *Facilities Planning*, John Wiley, New York, NY, USA, 4th edition, 2010.
[3] F. Liu, M. Dong, W. H. Hou, and Y. Zhai, "Multi criteria evaluation for facility layout problems," *Journal of Shanghai Jiaotong University*, vol. 4, pp. 664–668, 2007.
[4] M.-J. Wang, M. H. Hu, and M.-Y. Ku, "A solution to the unequal area facilities layout problem by genetic algorithm," *Computers in Industry*, vol. 56, no. 2, pp. 207–220, 2005.
[5] U. Haktanirlar and K. B. S. Kulturel, "An artificial immune system based algorithm to solve unequal area facility layout problem," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5384–5395, 2012.
[6] T.-Y. Wang, H.-C. Lin, and K.-B. Wu, "An improved simulated annealing for facility layout problems in cellular manufacturing systems," *Computers & Industrial Engineering*, vol. 34, no. 2, pp. 309–319, 1998.
[7] K. L. Mak, Y. S. Wong, and F. T. S. Chan, "A genetic algorithm for facility layout problems," *Computer Integrated Manufacturing Systems*, vol. 11, no. 1-2, pp. 113–127, 1998.
[8] M. C. Georgiadis, G. Schilling, G. E. Rotstein, and S. Macchietto, "A general mathematical programming

approach for process plant layout," *Computers & Chemical Engineering*, vol. 23, no. 7, pp. 823–840, 1999.

[9] K. R. Kumar, G. C. Hadjinicola, and T. I. Lin, "A heuristic procedure for the single-row facility layout problem," *European Journal of Operational Research*, vol. 87, no. 1, pp. 65–73, 1995.

[10] A. Taghavi and A. Murat, "A heuristic procedure for the integrated facility layout design and flow assignment problem," *Computers & Industrial Engineering*, vol. 61, no. 1, pp. 55–63, 2011.

[11] R. Kothari and D. Ghosh, "Insertion based Lin-Kernighan heuristic for single row facility layout," *Computers & Operations Research*, vol. 40, no. 1, pp. 129–136, 2013.

[12] K. Krishnakumar and S. N. Melkote, "Machining fixture layout optimization using the genetic algorithm," *International Journal of Machine Tools and Manufacture*, vol. 40, no. 4, pp. 579–598, 2000.

[13] S. Vallapuzha, E. C. De Meter, S. Choudhuri, and R. P. Khetan, "An investigation into the use of spatial coordinates for the genetic algorithm based solution of the fixture layout optimization problem," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 2, pp. 265–275, 2002.

[14] H. Samarghandi, P. Taabayan, and F. F. Jahantigh, "A particle swarm optimization for the single row facility layout problem," *Computers & Industrial Engineering*, vol. 58, no. 4, pp. 529–534, 2010.

[15] S. Önüt, U. R. Tuzkaya, and B. Doğaç, "A particle swarm optimization algorithm for the multiple-level warehouse layout design problem," *Computers & Industrial Engineering*, vol. 54, no. 4, pp. 783–799, 2008.

[16] M. Solimanpur, P. Vrat, and R. Shankar, "Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing," *European Journal of Operational Research*, vol. 157, no. 3, pp. 592–606, 2004.

[17] Y. H. Chen, "A new data structure of solution representation in hybrid ant colony optimization for large dynamic facility layout problems," *International Journal of Production Economics*, vol. 142, no. 2, pp. 362–371, 2013.

[18] R. Matai, "Solving multi objective facility layout problem by modified simulated annealing," *Applied Mathematics and Computation*, vol. 261, pp. 302–311, 2015.

[19] R. Şahin, "A simulated annealing algorithm for solving the bi-objective facility layout problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4460–4465, 2011.

[20] S. Kulturel-Konak, "A linear programming embedded probabilistic tabu search for the unequal-area facility layout problem with flexible bays," *European Journal of Operational Research*, vol. 223, no. 3, pp. 614–625, 2012.

[21] X. H. Suo and Z. Q. Liu, "Modeling and solution algorithms for facility layout of manufacturing systems," *Computer Integrated Manufacturing*, vol. 13, pp. 1941–1951, 2007.

[22] M. J. Ye and G. G. Zhou, "The application of genetic algorithm in the Bi-criteria layout problem with aisles," *Journal of Systems Engineering—Theory & Practice*, vol. 10, pp. 101–107, 2005.

[23] G. Aiello, G. La Scalia, and M. Enea, "A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10352–10358, 2012.

[24] J. M. Palomo-Romero, L. Salas-Morera, and L. García-Hernández, "An island model genetic algorithm for unequal area facility layout problems," *Expert Systems with Applications*, vol. 68, pp. 151–162, 2017.

[25] L. García-Hernández, A. Arauzo-Azofra, L. Salas-Morera, H. Pierreval, and E. Corchado, "Facility layout design using a multi-objective interactive genetic algorithm to support the DM," *Expert Systems*, vol. 32, no. 1, pp. 94–107, 2015.

[26] F. G. Paes, A. A. Pessoa, and T. Vidal, "A hybrid genetic algorithm with decomposition phases for the unequal area facility layout problem," *European Journal of Operational Research*, vol. 256, no. 3, pp. 742–756, 2016.

[27] J. F. Gonçalves and M. G. C. Resende, "A biased random-key genetic algorithm for the unequal area facility layout problem," *European Journal of Operational Research*, vol. 246, no. 1, pp. 86–107, 2015.

[28] R. D. Meller, V. Narayanan, and P. H. Vance, "Optimal facility layout design," *Operations Research Letters*, vol. 23, no. 3–5, pp. 117–127, 1998.

[29] K. Y. Wong and Komarudin, "Solving facility layout problems using flexible bay structure representation and ant system algorithm," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5523–5527, 2010.

[30] B. Montreuil, N. Ouazzani, and E. Brotherton, "Coupling zone-based layout optimization, ant colony system and domain knowledge," in *proceedings of the 8th International Material Handling Research Colloquium*, pp. 301–331, Graz, Cincinnati, OH, USA, June 2004.

[31] D. J. van Camp, M. W. Carter, and A. Vannelli, "A nonlinear optimization approach for solving facility layout problems," *European Journal of Operational Research*, vol. 57, no. 2, pp. 174–189, 1992.

[32] Q. Liu and R. D. Meller, "A sequence-pair representation and MIP-model-based heuristic for the facility layout problem with rectangular departments," *IIE Transactions*, vol. 39, no. 4, pp. 377–394, 2007.

[33] Y. A. Bozer and R. D. Meller, "A reexamination of the distance-based facility layout problem," *IIE Transactions*, vol. 29, no. 7, pp. 549–560, 1997.

[34] M. S. Bazaraa, "Computerized layout design: a branch and bound approach," *A I I E Transactions*, vol. 7, no. 4, pp. 432–438, 1975.

[35] S. Kulturel-Konak and A. Konak, "Unequal area flexible bay facility layout using ant colony optimisation," *International Journal of Production Research*, vol. 49, no. 7, pp. 1877–1902, 2011.

[36] Komarudin and K. Y. Wong, "Applying ant system for solving unequal area facility layout problems," *European Journal of Operational Research*, vol. 202, no. 3, pp. 730–746, 2010.