Genetic search for solving construction site-level unequal-area facility layout problems

Heng Li [a,*], Peter E.D. Love [b]

[a] *Department of Building and Real Estate, Hong Kong Polytechnic University Hung Hom, Kowloon, Hong Kong, China*
[b] *Australian AGILE Construction Initiative, School of Architecture and Building, Woolstores Campus, Deakin University, Geelong, Victoria 3217, Australia*

Abstract

A construction site represents a conflux of concerns, constantly calling for a broad and multi-criteria approach to solving problems related to site planning and design. As an important part of site planning and design, the objective of site-level facility layout is to allocate appropriate locations and areas for accommodating temporary site-level facilities such as warehouses, job offices, workshops and batch plants. Depending on the size, location and nature of the project, the required temporary facilities may vary. The layout of facilities can influence on the production time and cost in projects. In this paper, a construction site-level facility layout problem is described as allocating a set of predetermined facilities into a set of predetermined places, while satisfying layout constraints and requirements. A genetic algorithm system, which is a computational model of Darwinian evolution theory, is employed to solve the facilities layout problem. A case study is presented to demonstrate the efficiency of the genetic algorithm system in solving the construction site-level facility layout problems. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Site-level facilities layout; Permutation representation; Genetic algorithm

1. Introduction

A construction site represents a conflux of concerns, constantly calling for a broad and multi-criteria approach to solving problems related to site planning and design. As an important part of site planning and design, the objective of site-level facility layout is to allocate appropriate locations and areas for temporary site-level facilities such as warehouses, job offices, workshops and batch plants. Unless the site is a part of large development, it is often confined by a boundary/edges. The edges represent constraints by which the facility layout problem has to comply with.

The layout of site-level facilities can have an important impact on the production time and cost, especially in the case of large projects [9]. In addition, such a problem becomes far from trivial if a construction site is confined due to the lack of available space, or if the site is very large, then traveling between facilities can be considerably time-consuming. To arrange a set of predetermined facilities into appropriate locations, while satisfying a set of layout constraints, is a difficult problem as there are many possible alternatives. For example, [28] stated that for 10 facilities, the number of possible alternatives is well above 3,628,000.

---

* Corresponding author. Tel.: +852-2766-5879; fax: +852-2764-5131; e-mail: bshengli@polyu.edu.hk

Due to the complexity of facility layout problems, many algorithms have been developed to generate solutions for the problems. The algorithms can be classified as layout improvement, entire layout and partial layout categories [22]. The layout improvement algorithms require an existing layout by which new (improved) layouts are generated through relocating the facilities in order to improve the layout. Typical examples include those algorithms developed by Buffa et al. [1] and Moore [18]. The entire layout algorithms use the strategy of selecting and placing one facility each time according to a predetermined selecting and allocating order. Examples include the algorithms developed by Fortenberry and Cox [3] and Hassan and Hogg [10]. In the partial improvement algorithms, a facility is placed at all possible locations to generate all possible partial layout alternatives. Then, the best layout is selected from the alternatives. Another facility is added onto the best layout by repeating the same procedure until all facilities are located. Examples of the partial improvement algorithms include CORELAP [12] and Sirinaovakul and Thajchayapong's algorithm [22]. Recently, artificial intelligence based methods have been applied to solving facility layout problems. For example, knowledge based systems, which have been developed to provide users with problem-specific heuristic knowledge so that facilities can be allocated accordingly [21,26]. Moreover, Yeh [28] applied annealed neural networks to solve construction site-level facility layout problems.

Although no studies of applying genetic algorithms (Gas) in solving site-level facility layout problems have been reported, GAs have been applied to a diverse range of engineering and construction management searching problems, which include:

Structural optimization [11,19];
Resource scheduling [2,14];
Optimizing labor and equipment assignment [16]
Determination of laying sequence for a continuous girder reinforced concrete floor system [20], and Space allocation [4].

Genetic algorithm systems have also been applied to solve space layout planning problems. For example, Gero and Kazakov [4] incorporated the concept of genetic engineering into the GA system for solving building space layout problems. Similarly, Tate and Smith [25] compared different methods with the GA system in the application of space layout problems.

This paper presents an investigation of applying the GA system to search for the optimal solution for a construction site-level layout problem. The paper is presented as follows. Section 1 introduces the site-level layout problem. Section 2 describes the implementation of the GA system for solving the layout problem. Section 3 presents the numerical results generated from the genetic algorithm system. Conclusions and discussions are given in Section 5.

2. Site-level facility layout as unequal-area facility layout problem

A construction site-level layout problem is concerned with assigning a number of predetermined facilities into a number of predetermined places. In this paper, we have assumed that the number of predetermined places should be equal to or greater than the number of predetermined facilities. If the number of predetermined places is greater than the number of predetermined facilities, then a number of 'dummy' (fictitious) facilities will be added to make both numbers equal. By assigning both the distance and frequency as 0, the 'dummy' facilities will not affect the layout results.

Construction site-level facility layout problems can be modeled as a quadratic assignment problem in which costs associated with the flow between facilities are linear with respect to distance traveled and quantity of flow. If each of the predetermined places is capable of accommodating any of the facilities, then the facility layout problem can be modeled as an equal-area facility layout problem. If some of the predetermined places are only able to accommodate some of the facilities, then the problem becomes an unequal-area facility layout problem, where predetermined places have differing areas. Generally, unequal-area layout problems are more difficult to solve than equal-area layout problems, primarily because unequal-area layout problems introduce additional constraints into the problem formulation.

In this study, we introduce a method to solve site-level unequal-area facility layout problems. In these problems, a subset of discrete locations L_p ($L_p \in N$, $N = \{1, 2, 3, \ldots n\}$, $0 < p < n$, where N is

the total set of the facilities, and n is the total number of facilities) are smaller than the rest of the locations, and these smaller locations are unable to accommodate a subset of the predetermined facilities F_v ($F_v \in N$, $N = \{1, 2, 3, \ldots n\}$, $0 < v < n$). In Fig. 1, the predetermined locations are represented as rectangles. Two of the locations are relatively smaller than the rest of the locations as they are located at a narrower edge of the site, indicating that these two locations are only capable of accommodating smaller facilities, whilst the larger locations are able to accommodate any of the facilities.

The objective of site-level facility layout is to minimize the total traveling distance of site personnel between facilities. The total distance is defined as in Eq. (1):

$$\text{Minimize TD} = \sum_{i=1}^{n} \sum_{x=1}^{n} \sum_{j=1}^{n} \delta_{xi} f_{xi} d_{ij} \qquad (1)$$

Subject to,

$$\sum_{x=1}^{n} \delta_{xi} = 1, \{i = 1,2,3,\ldots,n\} \text{ and}$$

$$\left\{ \text{if } \delta_{xi} = 1 \text{ then } \left(x \notin L_p \text{ or } i \notin F_v \right) \right\} \qquad (2)$$

Where n is the number of facilities; δ_{xi} is the permutation matrix variable. Coefficient f_{ij} is the frequencies of trips made by construction personnel between facilities i and j. From the definition, it can be observed that f_{ij} equals to f_{ji}. The frequency is expressed as the number of trips per time period, in this study it is defined as the number of trips per day. Coefficient d_{ij} is the distances between locations m and n. If the two locations are next to each other, then the distance is defined as the distance between center of both locations, otherwise, it is the sum of segmental distances between them. For example, d_{ik} is the sum of d_{mj} and d_{jk}. If there are two paths linking the two locations, then the shorter path is selected for calculating the distance (see Fig. 1). Symbols L_p and F_v denote the subsets containing the smaller locations and the larger facilities that cannot be accommodated in the smaller locations, respectively. Therefore, total distance, TD, reflects the total traveling distance made by construction personnel.
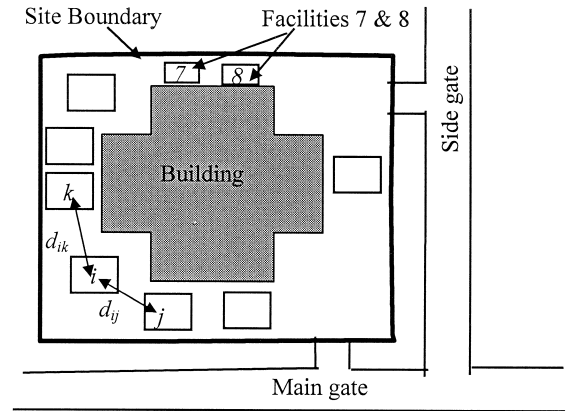

Fig. 1. Layout space restrictions.

It is necessary to note that as the objective function is the total traveling distance, it conveniently integrates time and cost into one consideration. Because time and cost are two primary concerns in construction, we believe it is appropriate to use the objective function for optimizing the site-level facility layout problems.

3. Implementation of genetic algorithms

Genetic algorithms are a set of optimization algorithms that seek to improve performance by sampling areas of the parameters space that are more likely to lead to better solutions [5,8]. In this study, the algorithms are examined as a possible method for solving site-level facility layout as a combination of optimization problems. The primary advantage of GAs lies in their capacity to move randomly from one feasible layout to another, without being drawn into local optima in which other algorithms are often trapped.

Genetic algorithms employ a random, yet directed, search for locating the globally optimal solution. Typically, a set of GAs requires a representation scheme to encode feasible solutions to the optimization problem. Usually a solution is represented as a linear string called chromosome whose length varies with each application. Some measure of fitness is applied to the solutions in order to construct better solutions. There are three basic operators in the basic GA system: Reproduction (or Selection), Crossover, and Mutation. Reproduction is a process

in which strings are duplicated according to their fitness magnitude. Crossover is a process in which the newly reproduced strings is randomly coupled, and whereby each couple of string partially exchanges information. Mutation is the occasional random alteration of the value of one of the bits in the string.

3.1. Layout representation as genetic coding and satisfaction of unequal-area constraint

Selecting an appropriate representation is an important step in applying GAs to an optimization problem. A number of representation schemes such as binary strings [11], permutation representation [6], and ordinal representation [7] have been experimented for combinatorial optimization problems. In this study, the representation for a site-level layout is a permutation type. Each facility layout can be represented by an $n \times n$ permutation matrix (n is the number of facilities, or locations) in which rows and columns are labeled by facilities and locations, respectively. There is only one '1' in each row and column, and the rest of the elements are 0. The corresponding row and column number of '1' indicate the location where the facility is placed. Fig. 2 shows a permutation matrix with 11 facilities and locations.

The permutation matrix can also be represented in a string form, as shown in Table 1. In the string representation, the position of a cell represents the facility number, and the value of the cell represents the location of the facility. For example, the location of facility 3 is 8, thus 8 is placed under facility 3 in

Table 1
String layout representation

Facility	1	2	3	4	5	6	7	8	9	10	11
Location	5	3	8	7	4	2	11	1	6	9	10

Table 1. In later sections, only the locations of the layout are displayed in the string representation, and the order of facilities is assumed to be sequential from 1 to 11.

In Table 1, an example of layout indicates that 11 facilities are located to 11 locations. The 11 facilities are:

1. Site office
2. Falsework workshop
3. Labor residence
4. Storeroom 1
5. Storeroom 2
6. Carpentry workshop
7. Reinforcement steel workshop
8. Side gate
9. Electrical, water and other utilities control room
10. Concrete Batch workshop
11. Main gate

In a construction site, the main gate and side gate are important for transport and access. Thus the relative position to the gates affects the facility layout decision. However, usually the gates are determined before the construction starts, and they are not subject to change during the allocation process. Thus, the two gates are treated as special facilities which have been 'clamped' on the predetermined locations. However, as the relative position to the gates will affect the layout of other facilities, they are included in the facility list as a special constraint to the layout problem.

Among the 11 facilities, 3 are too large to be accommodated in the 2 smaller locations indicated in Fig. 1. The 3 facilities include,

    1—Site office
    3—Labor residence
    10—Concrete Batch workshop

Thus, the subset of facilities F_v includes {1, 3, 10} which cannot be allocated in the smaller locations. The subset of smaller locations L_p includes {7, 8}, as indicated in Fig. 1. Using the unequal-area constraint stated in (2), the permutation matrix is verified to ensure that no large facilities are allocated

	Location										
Faci	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0	0	0	0	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1
8	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	1	0

Fig. 2. Permutation matrix with 11 facilities.

Table 2
String layout representation

Facility	1	2	3	4	5	6	7	8	9	10	11
Location	5	**8**	**3**	7	4	2	11	1	6	9	10

to the smaller locations. To do so, the GA system firstly identifies the locations of facilities included in F_v, and then check if their locations are included in L_p. In this example, the locations included in F_v are (5, 8, 9), and it is apparent that facility 3, which is the labor residence, is allocated to a smaller location 8. Therefore, the constraint is violated at location 8. To overcome the violation, the GA system searches the neighboring allocations of the violated location, and swaps locations of the violated facility with the neighboring facility. After this operation, the layout represented in Table 1 becomes the string representation in Table 2 in which the shaded cells are swapped.

The frequencies of trips (in one day) made between facilities are assumed, as listed in Eq. (3).

$$F = \begin{bmatrix} 0 & 5 & 2 & 2 & 1 & 1 & 4 & 1 & 2 & 9 & 1 \\ 5 & 0 & 2 & 5 & 1 & 2 & 7 & 8 & 2 & 3 & 8 \\ 2 & 2 & 0 & 7 & 4 & 4 & 9 & 4 & 5 & 6 & 5 \\ 2 & 5 & 7 & 0 & 8 & 7 & 8 & 1 & 8 & 5 & 1 \\ 1 & 1 & 4 & 8 & 0 & 3 & 4 & 1 & 3 & 3 & 6 \\ 1 & 2 & 4 & 7 & 3 & 0 & 5 & 8 & 4 & 7 & 5 \\ 4 & 7 & 9 & 8 & 4 & 5 & 0 & 7 & 6 & 3 & 2 \\ 1 & 8 & 4 & 1 & 1 & 8 & 7 & 0 & 9 & 4 & 8 \\ 2 & 2 & 5 & 8 & 3 & 4 & 6 & 9 & 0 & 5 & 3 \\ 9 & 3 & 6 & 5 & 3 & 7 & 3 & 4 & 5 & 0 & 5 \\ 1 & 8 & 5 & 1 & 6 & 5 & 2 & 8 & 3 & 5 & 0 \end{bmatrix} \tag{3}$$

The distances of the 11 locations are listed in Eq. (4). The distances are measured in meters.

$$D = \begin{bmatrix} 0 & 15 & 25 & 33 & 40 & 42 & 47 & 55 & 35 & 30 & 20 \\ 15 & 0 & 10 & 18 & 25 & 27 & 32 & 42 & 50 & 45 & 35 \\ 25 & 10 & 0 & 8 & 15 & 17 & 22 & 32 & 52 & 55 & 45 \\ 33 & 18 & 8 & 0 & 7 & 9 & 14 & 24 & 44 & 49 & 53 \\ 40 & 25 & 15 & 7 & 0 & 2 & 7 & 17 & 37 & 42 & 52 \\ 42 & 27 & 17 & 9 & 2 & 0 & 5 & 15 & 35 & 40 & 50 \\ 47 & 32 & 22 & 14 & 7 & 5 & 0 & 10 & 30 & 35 & 40 \\ 55 & 42 & 32 & 24 & 17 & 15 & 10 & 0 & 20 & 25 & 35 \\ 35 & 50 & 52 & 44 & 37 & 35 & 30 & 20 & 0 & 5 & 15 \\ 30 & 45 & 55 & 49 & 42 & 40 & 35 & 25 & 5 & 0 & 10 \\ 20 & 35 & 45 & 53 & 52 & 50 & 40 & 35 & 15 & 10 & 0 \end{bmatrix} \tag{4}$$

The use of GAs requires an initial population of 'seeds' (layouts) which will be used as the basis for GA operations. The user specifies the size of the initial population. In this example, the GA system generates the initial population of 30 layouts using a random number generator under the constraint that two gates (facility 8 and 11) are not subject to relocation. Table 3 lists some of the seeding layouts used as the initial population. Facility 8 (side gate) and facility 11 (main gate) are clamped at locations 1 and 10, respectively.

3.2. Crossover-adjusted edge recombination operator

A number of crossover operators have been developed for scheduling problems, such as the partially matched crossover (PMX), order crossover (OX), cyclic crossover (CX) and edge recombination operator [13,24,27]. Among the operators developed for sequencing recombination problems, the edge recombination operator, developed by Whitley [27], proved to be more efficient than the rest [4]. The edge recombination operator is modified in this study for solving the site-level facility layout problem by using GAs.

The modified edge recombination operator uses an edge table to construct an offspring that inherits as much genetic information as possible from the parental strings. The procedure of the edge recombination starts with selecting two layout strings from the population as parents. Facility 8 (side gate) and facility 11 (main gate) are removed from the parental strings as they are not subject to change. An edge table is created by listing all edges (genes near to a particular cell) of a location. For example, for location 2, Parent 1 has two edges (9, 4), and Parent 2 also has two edges (8, 7). Therefore, the edge list for location 2 is (9, 8, 7, 4). The edge lists for rest of the locations are constructed in a similar manner, as shown in Table 4.

The recombination algorithm includes an iterative process which starts by randomly selecting a location from the parental strings as the current location, (Let us assume that location 9 is selected as the current location.), then the current location is removed from the entire edge table (location 9 is then removed

Table 3
Initial population of layouts

Chromosomes	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Objective function
Layout 1	5	7	3	11	8	6	4	1	2	9	10	19537
Layout 2	4	7	2	3	11	8	9	1	5	6	10	19731
Layout 3	4	8	5	3	7	11	6	1	9	2	10	19250
Layout 4	2	8	6	4	5	7	11	1	3	9	10	18682
Layout 30	5	8	6	3	9	4	7	1	2	11	10	18479

from the edge lists of the table). Thereafter, the replacement for the current location is selected by determining which location in the edge list has the fewest entries in its own edge list (location 2 has the fewest entries: (8, 7), therefore, location 9 is replaced by location 2 in the new string). The process

Table 4
The procedure of the modified edge recombination algorithm

· *Select two strings as parents (layout 2 and layout 4) in Table 3*

Parent 1:	4	7	2	9	11	8	3	**1**	5	6	**10**
Parent 2:	2	8	6	4	5	7	11	**1**	3	9	**10**

· *Remove Facility 8 and 11 from the parental strings, as they are not subject to allocation*

Parent 1:	4	7	2	9	11	8	3	5	6
Parent 2:	2	8	6	4	5	7	11	3	9

· *Create the edge table*

Edge table

Location	Edges/links	Location	Edges/links
2	9, 8, 7	7	11, 5, 4, 2
3	11, 9, 8, 5	8	11, 6, 3, 2
4	7, 6, 5	9	11, 5, 3, 2
5	7, 6, 4, 3	11	9, 8, 7, 3
6	8, 5, 4		

· *The edge recombination algorithm*

1. Select a location from one of the parents as 'current location'.
2. Remove all occurrence of 'current location' from the edge table.
3. If the current location has entries in its edge list, then go to step 4; otherwise, go to step 5.
4. Determine which of the locations in the edge list of the current location has the fewest entries in its own edge list.
The location with the fewest entries becomes the current location, which replaces the initially selected location. Randomly select one entry from the edgelist of the current location. Go to step 2.
5. If there is no remaining location 'unvisited' then stop. Otherwise, randomly select a 'unvisited' location and go to step 2.

· *From applying the edge recombination algorithm, the following offspring can be obtained*

Offspring	6	3	11	8	7	5	2	4	9

· *Satisfaction of the unequal-area constraint*

The above offspring is a check to ensure that smaller locations in L_p are not allocated to larger facilities included in F_v. In this offspring, none of the facilities 1, 3 and 10 are allocated to locations 7 and 8. Therefore, the unequal-area constraint is satisfied. If the constraint is violated, then adjustment has to be made by swapping the violated location with the neighboring facility, as described in the previous section.

· *Add Facility 8 and Facility 11 to the layout table*

Offspring	6	5	9	11	8	7	3	**1**	2	4	**10**

repeats until all locations are visited. The probability of crossover (P_c) determines whether crossover will be applied to the selected pair of parental strings or not. In this study, the probability of crossover was set at 0.6. Table 4 indicates the adjusted edge recombination operator and its results.

3.3. Mutation-adjusted symmetric genes exchange

Mutation facilitates are used to randomly alter certain genes of a string, so that the loss of good genes during the crossover can be prevented. Because of the nature of the problem, we designed the 'symmetric genes exchange' operator for the site-level layout problem. The operator randomly selects a string from the population by the probability of mutation (P_m). The probability of mutation is related to the size of population. For the range of population size between 30–150, the appropriate value of probability of mutation is 0.01 [23]. Because facility 8 and facility 11 will not be relocated, they are firstly removed from the layout string. The operator then randomly chooses a gene from the string and swaps it with its symmetric gene in the string. After adding facility 8 and facility 11 back to the string, the mutated string is produced. Table 5 demonstrates this procedure with an example.

3.4. New generation of population

The 'fitness' of each layout is evaluated using the objective function. The best string is the layout with the lowest value of the objective function, and the worst string is the one with the highest value from the objective function. Each string is then assigned with an integer, namely a reproduction number, which represents the number of replicas of the string that will be included in the next generation of population. The reproduction number is in proportion to the fitness of the string: the best string is assigned with the maximum reproduction number; while the worst string is assigned with 0. In assigning reproduction numbers, one concern is to ensure that the total number of strings in the next generation will remain constant, otherwise, the number of strings may increase considerably, thus causing a phenomenon known as 'combinatorial explosion'.

The initial population size was set at 30 and extensive experiments have been conducted to obtain the effect of population size on the performance of the GA system in searching for optimal facility layout. We have experimented with different population sizes ranging from 30, 50, 75, 100 and 150. Detailed results of the experiments are described in Section 4.

Table 5
The procedure of the symmetric genes exchange operator

· *Select a string from the population*										
5	8	6	3	9	4	7	**1**	2	11	**10**
· *Remove Facility 8 and 11 from the parental strings, as they are not subject to allocation*										
5	8	6	3	9	4	7	2	11		
· *Randomly select a gene, and exchange position with its symmetric gene*										
5	8	7	3	9	4	6	2	11		

· *Satisfaction of the unequal-area constraint*
The above offspring is checked to ensure that smaller locations in L_p are not allocated to larger facilities included in F_v. In this offspring, facility 3, which is included in F_v, is allocated to location 7 which is included in L_p. Therefore, the unequal-area constraint is violated. An adjustment is made by swapping the violated location with the neighboring facility, which is shown below by the data in bold.

5	8	**3**	**7**	9	4	6	2	11	
· *Add Facility 8 and Facility 11 to the layout table*									
5	8	3	7	9	4	6	**1**	2	11	**10**

With the reproduction numbers determined, reproduction follows a procedure where strings with non-zero values of reproduction numbers are selected and duplicated to the total copies given by their reproduction numbers. For example, if a string has a reproduction number of 2, then it is selected and two exact replicas are made and entered into the next generation. If a string has a reproduction number of 0, then the string is an inferior solution, thus it will die out as no replica will be made to enter the next generation. To summarize the previous descriptions, the genetic algorithm model for the site-level layout problem is organized as follows:

(i) *Chromosome representation*—The facility layout is represented as a list of locations.

(ii) *Initialization of population*—A size of population is created either manually or random generation by the computer.

(iii) *Crossover*—Apply the adjusted edge recombination operator to produce new strings by 'crossover'.

(iv) *Mutation*—Adjusted symmetric exchange of cells.

(v) *Reproduction*—Evaluate the current generation to assign reproduction numbers to each string according to its 'fitness'.

(vi) *Evaluation*—Evaluate the variation of the fitness among population to determine the next step. If the difference between the highest fitness and the lowest fitness is smaller than an allowable range, then stop and the string with the highest fitness becomes the solution of the site-level facility layout problem. Otherwise, go to step (iii).

4. Computational results

A Visual Basic program called GenePlan-1 was developed on a personal computer to implement the GA system. The tests were run on the problem presented and described. In the experiment, 5 levels of population sizes (30, 50, 75, 100 and 150) were used to investigate the effect of population size on the performance of genetic algorithm system using the adjusted edge recombination operator. At each generation, the objective function values of the best and the worst layouts were compared. The GA system carried out with optimization until the difference between the best and worst values was within a predetermined range.

The results were obtained after 100 runs. From the results (see Fig. 3), it was observed that the GA system generally performed better with large population sizes as they provide a sufficient number of strings for GA operations. However, it was also observed that a considerably large population size (e.g., 150) also led to poor performance. A medium population size, i.e., a population size of 100, re-
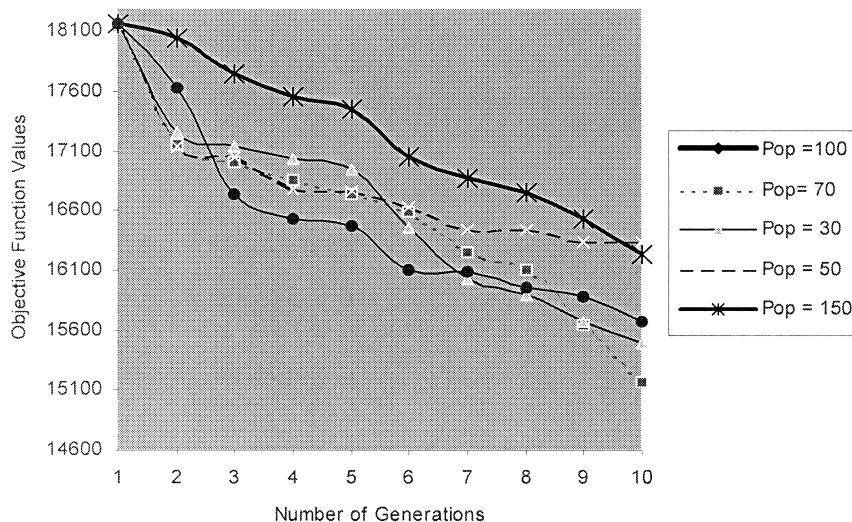

Fig. 3. Variation of objective function values with different population sizes.

Table 6
Optimal layout solution

Facility	1	2	3	4	5	6	7	8	9	10	11
Location	11	5	9	7	2	8	3	1	6	4	10

sulted in the best performance. For the population size of 100, the GA operations were converged and the optimal solution was obtained after 90 generations. The optimal objective function value is 15160, and the optimal facility layout is shown in Table 6.

In Fig. 3, the effect of the population size on the convergence of the whole process is presented. The curves corresponded to the minimum objective function values of every generation with population size of 30, 50, 70, 100, or 150, under the parameters of $P_c = 0.6$ and $P_m = 0.01$. The crossover and mutation probabilities (P_c, P_m) are very important in the GA operations inasmuch they are used to diverge from obtaining a local optimum. However, the incorrect selection of them may delay the convergence process. Further information of the effect of these probabilities on the convergence process can be found in [17]. However, in order to understand the effect of crossover and mutation probabilities, future research will be conducted on experimenting the effect of different combinations of crossover and mutation probabilities on the convergence of the GA operations.

Compared to the equal-area facility layout problem studied in [15], results from the unequal-area layout problem achieved slightly higher objective function values, indicating that the unequal-area constraint has some effect on the layout solution. Specifically, the neighborhood swapping heuristic used to overcome the violation of space constraint in crossover and mutation might have delayed the convergence of the GA operation.

5. Conclusions

This paper has introduced a representation scheme for representing construction site-level facility layout problems into strings suitable for GA operations. The paper then demonstrated the robustness of the GA approach in solving layout problems as combinatorial optimization problems, which are difficult to be

solved by conventional methods. The strength of the GA system lies in its ability of locating the global optimum using a random yet directed searching operators. Therefore, the GA system is less likely to restrict the search to a local optimum.

Experiments on the effect of different population sizes on the GA convergence indicated that the GA system converged earlier with a medium sized population. One limitation of this study is that it did not investigate the effect of other parameters, such as the probability of crossover (P_c) and the probability of mutation (P_m), on the convergence. Another limitation is that in the problem description, locations have to be predetermined. This problem may be solved by assigning all available locations as predetermined locations, and adding a number of dummy facilities to ensure that the number of locations equates to the number of facilities. In summary, this study is novel the following aspects: Firstly, the string representation proposed for representing the site-level facility layout problems; Secondly; the improved crossover and mutation operators; and Thirdly, the investigation on the effect of population size on the convergence of the GA system; and Fourthly; the paper introduced a new method for dealing with unequal-area facility layout problems.

Acknowledgements

References

[1] E.S. Buffa, G.C. Armour, T.E. Vollmann, Allocating facilities with CRAFT, Harvard Business Review 42 (1964) 136–157.

[2] W.-T. Chan, D.K.H. Chua, G. Kannan, Construction resource scheduling with genetic algorithms, ASCE Journal of Construction Engineering and Management 122 (2) (1996) 125–132.

[3] J.C. Fortenberry, J.F. Cox, Multiple criteria approach to facilities layout problem, Journal of Production Research 23 (1985) 773–782.

[4] J.S. Gero, V. Kazakov, Learning and reusing information in space layout planning problems using genetic engineering, Artificial Intelligence in Engineering 11 (3) (1997) 329–334.

[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Massachusetts, 1989.

[6] D.E. Goldberg, R. Lingle, Alleles, Loci and the Travelling Salesman Problem, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, Pittsburgh, PA, 1985, pp. 154–159.

[7] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics SMC 16 (1986) 122–128.

[8] J.H. Holland, Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.

[9] A. Hamiani, G. Popescu, CONSITE: a knowledge-based expert system for site layout, in: K.M. Will (Ed.), Computing in Civil Engineering: Microcomputers to Supercomputers, ASCE, New York, 1988, pp. 248–256.

[10] M.M.D. Hassan, G.L. Hogg, One constructing a block layout by graph theory, Journal of Production Research 6 (1991) 1263–1278.

[11] V.K. Koumousis, P.G. Georgiou, Genetic algorithms in discrete optimization of steel truss roofs, ASCE Journal of Computing in Civil Engineering 8 (3) (1994) 309–325.

[12] R.C. Lee, J.M. Moore, CORELAP—computerised relationship layout planning, Industrial Engineering 18 (1967) 195–200.

[13] I. Lee, R. Sikora, M.J. Shaw, Joint lot sizing and sequencing with genetic algorithms for scheduling evolving the chromosome structure, Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, Korgan Kaufman Publishers, San Mateo, CA, 1993, pp. 383–389.

[14] H. Li, P.E.D. Love, Improved genetic algorithms for time-cost optimisation, ASCE Construction Engineering and Management 123 (3) (1997) 233–237.

[15] H. Li, P.E.D. Love, Site-Level Facilities Layout Using Genetic Algorithms, ASCE Computing in Civil Engineering (Forthcoming) 1998.

[16] H. Li, P.E.D. Love, S. Ogunlana, Comparing Genetic Algorithms and Non-Linear Optimisation for Labor and Equipment Assignment, Building Research and Information (Forthcoming) 1998.

[17] F. Maturana, P. Gu, A. Naumann, D.H. Norrie, Object-oriented job-shop scheduling using genetic algorithms, Computers in Industry 32 (1997) 281–294.

[18] J.M. Moore, Facilities design with graph theory and strings, Omega 4 (1976) 193–203.

[19] S. Nagendra, D. Jestin, R.T. Haftka, L.T. Watson, Improved genetic algorithm for the design of stiffened composite panels, Computers and Structures 58 (3) (1996) 543–555.

[20] Y. Natsuaki, S. Mukandai, K. Yasuda, H. Furuta, Application of Genetic Algorithms to the Problem of Determining the Laying Sequence for a Continuous Girder Reinforced Concrete Floor System, in: Y.C. Loo (Ed.), EASEC-5, Building for the 21st Century, pp. 811–816, Gold Coast, Australia, July, 1995.

[21] P.F. Rad, B.M. James, The layout of temporary construction facilities, Cost Engineering 25 (2) (1983) 19–27.

[22] B. Sirinaovakul, P. Thajchayapong, An analysis of computer-aided facility layout techniques, Journal of Computer-Integrated Manufacturing 9 (4) (1996) 260–264.

[23] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transaction on Systems, Man and Cybernetics 24 (4) (1994) 656–667.

[24] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, C. Whitley, A comparison of genetic sequencing operators, Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, CA, 1991, pp. 69–75.

[25] Tate and Smith, 1995.

[26] I.D. Tommelein, R.E. Levitt, T. Confrey, SightPlan experiments: alternate strategies for site layout design, ASCE Journal of Computing in Civil Engineering 5 (1) (1991) 42–63.

[27] D. Whitley, The GENITOR algorithm and selective pressure: why rank-based allocation of reproductive trials is best, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufman, Palo Alto, CA, 1989, pp. 116–121.

[28] I.-C. Yeh, Construction-site layout using annealed neural network, ASCE Journal of Computing in Civil Engineering 9 (3) (1995) 201–208.