SITE-LEVEL FACILITIES LAYOUT USING GENETIC ALGORITHMS

By Heng Li[1] and Peter E. D. Love[2]

ABSTRACT: Construction site-level facilities layout is an important activity in site planning. The objective of this activity is to allocate appropriate locations and areas for temporary site-level facilities such as warehouses, job offices, various workshops and batch plants. Depending on the size, location, and nature of the project, the required temporary facilities may vary. The layout of facilities has an important impact on the production time and cost savings, especially for large 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.

INTRODUCTION

Construction site-level facility layout concerns the allocation of locations and areas for temporary facilities such as warehouses, job offices, various workshops and batch plants. Depending on the size, location, and nature of the project, the required temporary facilities may vary. A site-level facility layout has an important impact on the production time and cost savings, especially for large projects (Hamiani and Popescu 1988). In addition, a site-level facility layout problem becomes far from trivial if a construction site is confined with available space or the site is very large in size where 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, Yeh (1995) stated that for 10 facilities, the number of possible alternatives is well above 3,628,000.

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 (Sirinaovakul and Thajchayapong 1996). The layout improvement algorithms are required to have an initial layout by which new (improved) layouts are generated through relocating the facilities to improve the initial layout. Typical examples include those algorithms developed by Buffa et al. (1964) and Moore (1976). The entire layout algorithms use the strategy of selecting and placing one facility each time according to a predetermined selecting and allocating order. The predetermined selecting and placing order represents some kind of fitness in placing a facility at a particular place. Examples include the algorithms developed by Fortenberry and Cox (1985) and Hassan and Hogg (1991). 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 (Lee and Moore 1967) and Sirinaovakul and Thajchayapong's algorithm (Sirinaovakul and Thajchayapong 1996). Very recently, artificial intelligence-based methods have been applied to solving facility layout problems. For example, knowledge-based systems have been developed to provide users with problem-specific heuristic knowledge in allocating facilities (Rad and James 1983; Tommelein et al. 1991). Yeh (1995) applied annealed neural networks to solve construction site-level facility layout problems.

This paper presents an investigation of applying the genetic algorithm (GA) system to search for the optimal solution for a construction site-level layout problem. The paper is presented as follows. The second section introduces the site-level layout problem. The third section describes the implementation of the GA system in solving the layout problem. The fourth section presents the numerical results generated from the GA system. Conclusions and discussions are given in the final section.

CONSTRUCTION SITE-LEVEL FACILITY LAYOUT PROBLEM

A construction site-level facility layout problem to be solved in this study concerns finding the most appropriate arrangement for placing a set of predetermined facilities into a set of predetermined spaces on the site. In addition, this study assumes that each of the predetermined places is capable of accommodating the largest one among the facilities. The predetermined locations are represented as rectangles, as shown in Fig. 1.

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

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

$$\text{Subject to} \quad \sum_{x=1}^{n} \delta_{xi} = 1, \quad i = 1, 2, 3, \ldots, n \qquad (1)$$

where n = number of facilities; and δ_{xi} = 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} is equal 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 centers of the two locations; otherwise, it is the sum of segmental distances between the two locations. 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). Therefore, TD reflects the total traveling distance (cost) made by construction personnel.

[1]Assoc. Prof., Dept. of Build. and Real Estate, Hong Kong Polytechnic Univ., Hung Hom, Kowloon, Hong Kong.

[2]Lect., Facu. of Sci. and Technol., School of Arch. and Build., Woolstores Campus, Deaking Univ., Geelong, Victoria 3217, Australia.

Note. Discussion open until March 1, 1999. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this paper was submitted for review and possible publication on December 9, 1997. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 12, No. 4, October, 1998. ©ASCE, ISSN 0877-3801/0004-0227–0231/$8.00 + $.50 per page. Paper No. 17160.

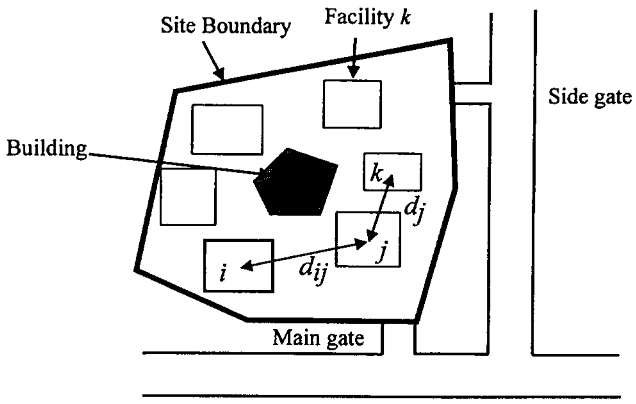

FIG. 1. Layout Space Restrictions

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.

IMPLEMENTATION OF GAs

GAs 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 (Holland 1975; Goldberg 1989). In this study, the algorithms are examined as a possible method for solving site-level facility layout as combinatory optimization problems. The primary advantage of GAs lies in their capacity to move randomly from one feasible layout to another, without getting into local optima in which other algorithms often get trapped.

GAs 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 a chromosome whose length varies with each application. Some measure of fitness is applied to the solutions to construct better solutions. There are three basic operators in the basic GA system: (1) Reproduction (or selection); (2) crossover; and (3) 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 are randomly coupled, and each couple of the string partially exchanges information. Mutation is the occasional random alteration of the value of one of the bits in the string.

Layout Representation as Genetic Coding

Selecting an appropriate representation is an important step in applying gas to an optimization problem. A number of representation schemes such as binary strings (Koumousis and Georgiou 1994), permutation representation (Goldberg and Lingle 1985), and ordinal representation (Grefenstette 1986) have been experimented with for combinatorial optimization problems. In this study, the representation for a site-level layout is a permutation type, because other representation schemes cannot represent the problem under this study. 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.

	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 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

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. 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 the 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, the gates are usually 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.

The frequencies of trips (in 1 day) made between facilities are assumed, as listed in (2)

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

TABLE 2. Initial Population of Layouts

Layout (1)	Chromosomes											Objective function (13)
	L1 (2)	L2 (3)	L3 (4)	L4 (5)	L5 (6)	L6 (7)	L7 (8)	L8 (9)	L9 (10)	L10 (11)	L11 (12)	
1	7	5	3	11	8	6	4	1	2	9	10	19,894
2	7	4	2	9	11	8	3	1	5	6	10	20,387
3	4	8	5	3	7	11	6	1	9	2	10	19,250
4	2	8	6	4	5	9	11	1	3	7	10	18,223
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
30	5	8	6	3	9	4	7	1	2	11	10	18,479

The distances of the 11 locations are listed in (3). The distances are measured in meters

$$D = \begin{bmatrix} 0 & 15 & 25 & 33 & 40 & 42 & 47 & 55 & 35 & 30 & 20 \\ 15 & 0 & 10 & 18 & 25 & 27 & 32 & 42 & 50 & 45 & 35 \\ 25 & 10 & 0 & 8 & 15 & 17 & 22 & 32 & 52 & 55 & 45 \\ 33 & 18 & 8 & 0 & 7 & 9 & 14 & 24 & 44 & 49 & 53 \\ 40 & 25 & 15 & 7 & 0 & 2 & 7 & 17 & 37 & 42 & 52 \\ 42 & 27 & 17 & 9 & 2 & 0 & 5 & 15 & 35 & 40 & 50 \\ 47 & 32 & 22 & 14 & 7 & 5 & 0 & 10 & 30 & 35 & 40 \\ 55 & 42 & 32 & 24 & 17 & 15 & 10 & 0 & 20 & 25 & 35 \\ 35 & 50 & 52 & 44 & 37 & 35 & 30 & 20 & 0 & 5 & 15 \\ 30 & 45 & 55 & 49 & 42 & 40 & 35 & 25 & 5 & 0 & 10 \\ 20 & 35 & 45 & 53 & 52 & 50 & 40 & 35 & 15 & 10 & 0 \end{bmatrix} \tag{3}$$

The use of GAs requires an initial population of seeds (layouts) that will be used as the basis for GA operations. The size of the initial population is specified by the user. In this study, the GA system generates the initial population of 30 layouts using a random number generator under the constraint that two gates (Facilities 8 and 11) are not subject to relocation. Table 2 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.

Crossover—Edge Recombination Operator

A number of crossover operators have been developed for scheduling problems, such as partially matched crossover, order crossover, cyclic crossover, and edge recombination operator (Whitley 1989; Starkweather et al. 1991; Lee et al. 1993). Among the operators developed for sequencing recombination problems, the edge recombination operator, developed by Whitley (1989), proved to be more efficient than the rest (Starkweather et al. 1991). The edge recombination operator is modified in this study for solving the site-level facility layout problem by 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 has two edges (8, 7). Therefore, the edge list for Location 2 is (9, 8, 7, 4).

The recombination algorithm includes an iterative process that starts by randomly selecting a location from the parental strings as the current location (assuming that Location 9 is selected as the current location), then the current location is removed from the entire edge 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. The process 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. The strings that are not selected for crossover are copied into the next generation. Thus, the total size of the population remains constant during the entire GA operation. Table 3 indicates the edge recombination operator and its results.

TABLE 3. Procedure of Modified Edge Recombination Algorithm

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

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

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

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

• *Create the edge table,*

Edge Table			
Location	Edges/Links	Location	Edges/Links
2	9, 8, 7, 4	7	6, 4, 3, 2
3	11, 8, 7, 5	8	11,6, 3, 2
4	7, 6, 5, 2	9	11, 5, 2
5	9, 6, 4, 3	11	9, 8, 3
6	8, 7, 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	5	9	11	8	7	3	2	4

• *Add Facility 8 and Facility 11 to the layout table,*

Offspring	6	5	9	11	8	7	3	■	2	4	■

Note: Shaded areas are clamped facilities that are not subject to change.

Mutation—Symmetric Genes Exchange

Mutation facilitates to randomly alter certain genes of a string, to prevent the loss of good genes in crossover. 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 the population. For the range of population size between 30 and 150, the appropriate value of probability of mutation is 0.01 (Srinivas and Patnaik 1994). Because facilities 8 and 11 will not be relocated, they are first 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 facilities 8 and 11 back to the string, the mutated string is produced. Table 4 shows the procedure with an example.

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

TABLE 4. Procedure of 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 elect a gene, and exchange position with its symmetric gene,*

5	8	7	3	9	4	6	2	11

- *Add Facility 8 and Facility 11 to the layout table,*

5	8	7	3	9	4	6	▨ 1	2	11	▨ 10

Note: Shaded areas are clamped facilities that are not subject to change.

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, i.e., 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 a constant; otherwise, the number of strings may increase considerably, causing a phenomenon known as combinatorial explosion. As stated previously, in this study, the number of strings in each generation remains a constant. Therefore, the problem of combinatorial explosion is avoided.

In this study, although the initial population size was set at 30, 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 experimented with different population sizes ranging from 30, 50, 75, 100, and 150. Detailed results of the experiments are described in the next section.

With the reproduction numbers determined, reproduction follows a procedure where strings with nonzero 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 GA model for the site-level layout problem is organized as follows:

1. *Chromosome representation.* Facility layout is represented as a list of locations.
2. *Initialization of population.* A size of population is created either manually or random generation by the computer.
3. *Crossover.* Apply the edge recombination operator to produce new strings by crossover.
4. *Mutation.* Symmetric exchange of cells.
5. *Reproduction.* Evaluate the current generation to assign reproduction numbers to each string according to its fitness.
6. *Evaluation.* Evaluate the variation of the fitness among the 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 3.)

COMPUTATIONAL RESULTS

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

From the results (see Fig. 3), it was observed that the GA system performed poorly with small population sizes as they provide an insufficient 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, resulted 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 15,090. The optimal layout solution is indicated in Table 5.

In Fig. 3, the effect of the population size on the convergence of the whole process is presented (computational results are presented in Table 6). The curves corresponded to the minimum objective function values of every generation with a population size of 30, 50, 70, 100, or 150, under the parameters of $P_c = 0.6$ and $P_m = 0.01$.

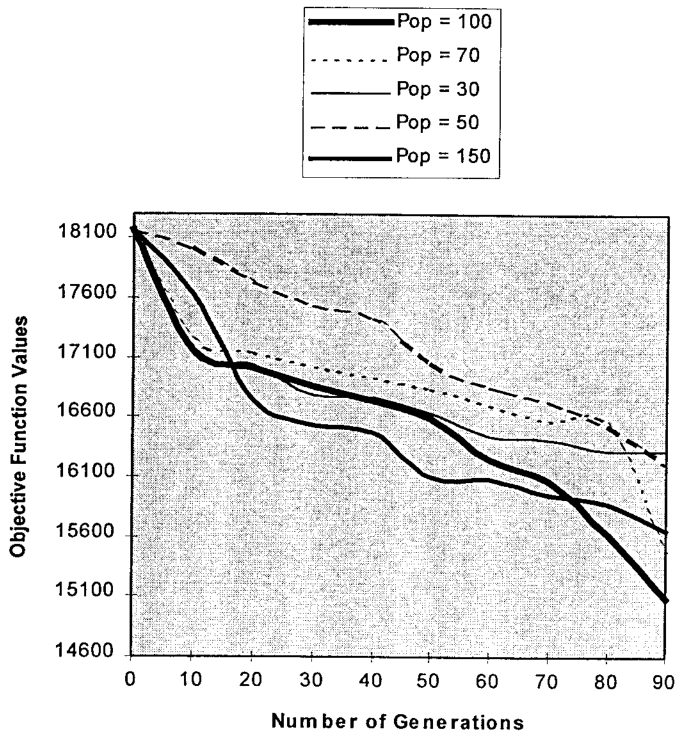

FIG. 3. Variation of Objective Function Values with Different Population Sizes

TABLE 5. Optimal Layout Solution

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

TABLE 6. Computational Results from Experiments

Generation number (1)	Facility											Objection function value (13)
	1 (2)	2 (3)	3 (4)	4 (5)	5 (6)	6 (7)	7 (8)	8 (9)	9 (10)	10 (11)	11 (12)	
(*a*) Population size = 100												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	2	8	11	3	7	5	6	1	9	4	10	17,162
20	7	9	3	2	5	6	4	1	8	11	10	17,013
30	4	2	6	11	7	9	8	1	5	3	10	16,857
40	2	8	7	9	5	4	6	1	11	3	10	16,742
50	6	9	8	11	5	3	2	1	7	4	10	16,586
60	5	11	8	6	3	9	2	1	4	7	10	16,443
70	8	4	7	6	11	5	2	1	9	3	10	16,268
80	7	4	6	11	2	9	5	1	8	3	10	16,022
90	11	5	8	7	2	9	3	1	6	4	10	15,090
(*b*) Population size = 70												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	7	4	5	6	9	3	2	1	11	8	10	17,256
20	7	11	4	3	5	8	6	1	2	9	10	17,139
30	5	9	2	6	3	11	7	1	4	8	10	17,021
40	3	4	7	9	5	11	6	1	2	8	10	16,932
50	4	9	3	6	7	8	2	1	5	11	10	16,846
60	7	4	5	11	8	6	3	1	2	9	10	16,715
70	2	6	4	11	7	9	8	1	5	3	10	16,582
80	9	4	3	7	2	5	8	1	11	6	10	16,563
90	5	9	11	8	4	3	2	1	7	6	10	15,482
(*c*) Population size = 30												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	6	5	4	2	7	11	9	1	8	3	10	17,130
20	6	8	11	4	9	5	7	1	3	2	10	17,041
30	11	3	5	2	7	6	8	1	9	4	10	16,784
40	5	7	4	11	8	6	3	1	9	2	10	16,768
50	2	3	5	9	4	11	7	1	8	6	10	16,639
60	6	4	8	5	9	11	2	1	7	3	10	16,438
70	8	11	2	9	6	4	3	1	5	7	10	16,412
80	9	2	3	7	8	5	6	1	4	11	10	16,325
90	6	8	2	7	3	9	4	1	5	11	10	16,320
(*d*) Population size = 50												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	4	7	3	5	6	2	9	1	11	8	10	18,025
20	6	2	3	4	7	5	9	1	8	11	10	17,734
30	6	4	7	3	11	2	5	1	9	8	10	17,549
40	4	11	7	3	9	6	2	1	5	8	10	17,438
50	4	11	3	9	2	7	8	1	6	5	10	17,046
60	8	3	11	2	6	7	9	1	5	4	10	16,852
70	7	4	5	3	6	11	2	1	8	9	10	16,739
80	11	9	3	6	5	2	8	1	4	7	10	16,532
90	9	6	8	2	7	3	4	1	5	11	10	16,213
(*e*) Population size = 150												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	4	11	5	2	7	6	8	1	9	3	10	17,638
20	5	6	7	9	3	2	8	1	4	11	10	16,735
30	11	9	3	6	5	2	8	1	4	7	10	16,532
40	11	2	5	6	8	3	7	1	4	9	10	16,470
50	9	7	2	5	3	11	8	1	4	6	10	16,103
60	11	9	2	7	5	6	8	1	4	3	10	16,089
70	11	8	3	4	2	9	6	1	7	5	10	15,948
80	9	3	4	6	7	11	5	1	8	2	10	15,872
90	8	5	6	9	7	11	3	1	4	2	10	15,656

CONCLUSIONS

This paper has introduced a representation scheme for representing construction site-level facility layout problems into strings suitable for GA operations. This paper then demonstrated the robustness of the GA approach in solving layout problems as combinatorial optimization problems that are difficult to solve by conventional methods. The strength of the GA system lies in its ability of locating the global optimum using random yet directed searching operators. Therefore, the GA system is less likely to restrict the search to a local optimum.

For medium or large construction projects, it is not unusual to have up to 40 temporary facilities that need to be located on site. It is expected that the system developed in this study can easily handle the problem size. However, extensive tests will be conducted to ensure the usefulness of the system in dealing with facility allocation problems with larger sizes. In addition, future research is needed to compare the performance of the GA system with other methods used in solving facility allocation problems.

Experiments on the effect of different population sizes on the GA convergence indicated that the GA system converged earlier with the medium size of the 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 the number of facilities. In summary, this study is novel in at least the following three aspects: (1) The string representation proposed for representing the facility layout problems; (2) the improved crossover and mutation operators; and (3) the investigation on the effect of population size on the convergence of the GA system.

ACKNOWLEDGMENTS

This research project has been supported by a Research Grants Committee grant from the Hong Kong Special Administrative Region government.

APPENDIX. REFERENCES

Buffa, E. S., Armour, G. C., and Vollmann, T. E. (1964). "Allocating facilities with CRAFT." *Harvard Business Rev.*, 42, 136–157.
Fortenberry, J. C., and Cox, J. F. (1985). "Multiple criteria approach to facilities layout problem." *J. Production Res.*, 23, 773–782.
Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley Publishing Co., Reading, Mass.
Goldberg, D. E., and Lingle, R. (1985). "Alleles, loci and the travelling salesman problem." *Proc., 1st Int. Conf. on Genetic Algorithms and Their Applications*, Carnegie-Mellon Univ., Pittsburgh, Pa., 154–159.
Grefenstette, J. J. (1986). "Optimization of control parameters for genetic algorithms." *IEEE Trans. on Sys., Man, and Cybernetics*, SMC-16, 122–128.
Hamiani, A., and Popescu, G. (1988). "CONSITE: A knowledge-based expert system for site layout. Computing in civil engineering." *Microcomputers to supercomputers*, K. M. Will, ed., ASCE, New York, N.Y., 248–256.
Hassan, M. M. D., and Hogg, G. L. (1991). "One constructing a block layout by graph theory." *J. Production Res.*, 6, 1263–1278.
Holland, J. H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, Mich.
Koumousis, V. K., and Georgiou, P. G. (1994). "Genetic algorithms in discrete optimization of steel truss roofs." *J. Comp. in Civ. Engrg.*, ASCE, 8(3), 309–325.
Lee, I., Sikora, R., and Shaw, M. J. (1993). "Joint lot sizing and sequencing with genetic algorithms for scheduling evolving the chromosome structure." *Proc., 5th Int. Conf. on Genetic Algorithms*, Univ. of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers Inc., San Mateo, Calif., 383–389.
Lee, R. C., and Moore, J. M. (1967). "CORELAP-computerised relationship layout planning." *Industrial Engrg.*, 18, 195–200.
Moore, J. M. (1976). "Facilities design with graph theory and strings." *Omega*, 4, 193–203.
Rad, P. F., and James, B. M. (1983). "The layout of temporary construction facilities." *Cost Engrg.*, 25(2), 19–27.
Sirinaovakul, B., and Thajchayapong, P. (1996). "An analysis of computer-aided facility layout techniques." *J. Computer-Integrated Manufacturing*, 9(4), 260–264.
Srinivas, M., and Patnaik, L. M. (1994). "Adaptive probabilities of crossover and mutation in genetic algorithms." *IEEE Trans. on Syst., Man and Cybernetics*, 24(4), 656–667.
Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., and Whitley, C. (1991). "A comparison of genetic sequencing operators." *Proc., 4th Int. Conf. on Genetic Algorithms*, 69–75.
Tommelein, I. D., Levitt, R. E., and Confrey, T. (1991). "SightPlan experiments: Alternate strategies for site layout design." *J. Comp. in Civ. Engrg.*, ASCE, 5(1), 42–63.
Whitley, D. (1989). "The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best." *Proc., 3rd Int. Conf. on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., San Mateo, Calif., 116–121.
Yeh, I.-C. (1995). "Construction-site layout using annealed neural network." *J. Comp. in Civ. Engrg.*, ASCE, 9(3), 201–208.