Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 16 (2003) 501–509
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Proposed genetic algorithms for construction site layout

Michael J. Mawdesley[a], Saad H. Al-Jibouri[b,*]

[a] School of Civil Engineering, University of Nottingham, UK
[b] Department of Construction Process Management, University of Twente, CT, M, postbus 217, 7500 AE Enschede, The Netherlands

Received 31 May 2002; received in revised form 16 April 2003; accepted 1 September 2003

Abstract

The positioning of temporary facilities on a construction site is an area of research which has been recognised as important but which has received relatively little attention. In this paper, a genetic algorithm is proposed to solve the problem in which m facilities are to be positioned to n available sites such that the total cost of construction and interactive cost due to facility layout constraints are minimised. A sequence-based genetic formulation of the problem is presented. Genetic crossover and mutation operators are developed for the problem and their performance evaluated and compared on an example project. The results obtained suggest that the different operators perform very differently but that one could be relied on to find the optimum. Overall, experiments suggest that the technique will prove useful when tackling real problems.

Keywords: Construction; Site layout; Genetic algorithms

1. Introduction

In the construction industry, site layout is a very important planning problem. The objective of site layout is to position temporary facilities both geographically and at the correct time such that the construction work can be served satisfactorily with minimal costs and improved safety and working environment. This paper starts with a general description and formulation of the problem from which a specific formulation for genetic algorithm solution is proposed. Following a brief introduction to the problem and genetic algorithms, a formulation of the site layout problem is proposed in terms of a sequencing problem that is suitable for solution using genetic algorithms. Various mutation and crossover operators are considered for use in the solution and an example is presented to illustrate the techniques developed and to allow comparison of their efficiency.

2. The construction site layout problem

Site layout problems have long been recognised as being of importance but whilst they have been written about (see for example Twort and Rees, 1995), there has been no complete solutions proposed.

The nature of the problem means that no well-defined method can guarantee a solution. At best, guidelines point out the issues that field managers must consider while laying out their project sites (Rad and James, 1983). Layout problems have however been treated using operations research (Seehof and Evans, 1967) and artificial intelligence (Hamiani, 1989; Tommelein et al., 1991).

They all have similar drawbacks; namely:

- They rely on the generation of a knowledge base which will allow choice between various geographical layouts. This has proved very difficult to produce for real projects.
- The integration of the scheduling procedures with the geographical aspects to generate the site layout has proved difficult.

The positioning of major pieces of equipment might be considered to be a sub-set of the general site layout

*Corresponding author. Tel.: +31-534-894-887; fax: +31-534-892-511.
E-mail address: jibouri@sms.utwente.nl (S.H. Al-Jibouri).

0952-1976/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2003.09.002

problem. Several authors report research into this topic by various methods. For example Warszawski and Peled (1987) describe techniques for positioning cranes on a building site. These employ knowledge-based systems. They do not look at all aspects of the positioning although it might be possible to extend them to do so.

Yeh (1995) and Zouein and Tommelein (1999) identified a construction site layout problem. In Yeh's problem, there are n resources to be positioned and n available positions and all the information about operation and set-up cost is known, an assignment of resources to positions is searched to minimise the whole cost. A neural network was used to solve the assignment type of construction site layout problem in which the problem is formulated as a discrete combinatorial optimisation problem. This is a static and special case of more general construction site layout problems.

A class of similar but different problems is facility layout problems in manufacturing industries and VLSI in electronics, which have been studied extensively. The facility layout problem is concerned with finding the most efficient arrangement of several indivisible departments with unequal area requirements within a facility. The objective of facility layout problem is to minimise the material handling cost or resource movement cost inside a facility.

In facility layout problems there are in general two sets of constraints considered:

(1) department and floor area requirements and
(2) department location restrictions (departments cannot overlay, departments must be placed within the facility and some must be fixed to a location or cannot be placed in specific regions).

Floor loading and floor-to-ceiling clear-height constraints also exist in multiple-floor facilities in which the vertical distances between departments are considered in addition to the horizontal distance.

Two sets of solution methods are employed in attacking facility layout problem: heuristics and exact methods. Exact algorithms are developed to obtain, in theory, optimal solutions, but because of the combinatorial nature of the problem, they are only applicable for small-scale problems.

Heuristic methods are usually used for larger problems. The current trend in research for facility layout problems is concentrated in three areas: developing more suitable models; extending existing models to include a time element (dynamic layout); adding uncertainty (stochastic layout); or adding multiple criteria for evaluation. There are also special cases for specific types of problems. A review of the facility layout problem can be found in Russell and Gau, (1996).

In this paper, the construction site layout problem is specified as 'to position a set of facilities on the site so that the layout objectives are optimised subject to layout constraints'. The objective functions can be of any form that might be considered by a site manager or builder.

Typically the 'best' layout is defined in terms of the cost to the project overall although this is difficult to define. It will have some relationship to the cost of setting up and removing the facility in the given position, to the amount of movement necessitated by the layout, to the extra time it adds to the overall construction, to the inconvenience caused to the site users and many other factors.

In practical terms, this is best described by means of a simple example. Consider a typical private housing estate being constructed on a green-field site by a speculative builder. At the start of the project, the builder will have to install an office, fence the site, build some roads and put in some services such as water, gas and sewers. As the work progresses, the builder will construct some of the houses starting with the foundations and moving through the walls, roofs and internal finishes to completing the external works with fences and landscaping. When some of the houses have been built, the builder will move on to others while purchasers will move into those that are complete.

Temporary facilities will be required; for example to act as welfare facilities for the operatives; to store the materials which are delivered to the site and before they are built into the works; and for the heavy equipment to move around the project. The state of the site will be continuously changing and what might have been the 'best' position for the office at the start of the project might not be the 'best' at other stages. In theory, the facilities can be of any shape and positioned anywhere on the site but in practice, offices are usually rectangular and other facilities can be approximated to that shape.

This example is in two physical dimensions and the third one is ignored. On real construction sites this is often a practical approximation. However, this is not the case on all projects. For example, if the project were the construction of a multi-storey city centre block, then the third dimension would be of considerable importance. It can also be seen that in reality, a fourth dimension, time, is important as the layout will change throughout the life of a project.

The general problem has been worked on for a considerable time (see for example Rad and James, 1983) and has been the subject of several papers in artificial intelligence, see for example (Hamiani, 1989) and (Tommelein et al., 1991). Some authors have also proposed formulations and solutions for specific site layout problems (see for example Warszawski and Peled, 1987). However, there is still no formulation and solution which is widely used in industry.

If the problem is constrained slightly by assuming that all the possible locations for the different facilities can be

identified, then the problem can be stated in mathematical terms, as follows:

- m facilities are to be positioned on a site.
- n locations are available for each facility to position, $n \geqslant m$.
- For each assignment of a facility to a candidate location, there are different set-up and removal costs. Consequently, different assignments will mean different operational costs.
- There are adjacency constraints which dictate that certain facilities must be adjacent to other facilities.
- The objective is to position all the facilities such that the total cost will be minimised subject to the constraints.

This problem can be modelled as a Quadratic Assignment Problem (QAP). This formulation requires an equal number of facilities and locations. If the number m of facilities is less than the number n of locations, $n-m$ dummy facilities can be created and zero set-up cost and transportation cost assigned to them. However, if there are fewer locations than facilities, the problem is infeasible.

Based on these ideas, the construction site layout problem can be formulated as (Yeh, 1995)

$$\text{Min} \quad F = \sum_x \sum_i \delta_{xi} C_{xi} + \sum_x \sum_i \sum_y \sum_j \delta_{xi}\delta_{yj} A_{ij} D_{xy}$$

$$\text{s.t.} \quad \delta_{yj} = 0 \text{ if } \delta_{xi} = 1 \text{ and } y \neq x,$$
$$\delta_{xj} = 1 \text{ if } \delta_{xi} = 1 \text{ and } j \neq i,$$

where F is the cost function; δ_{xi} the permutation matrix variable ($=1$ if facility x is assigned to site i); C_{xi} the construction cost of assigning facility x to site i; A_{ij} the site neighbouring index, $A_{ij} = 1$ if site i is neighbouring to site j; D_{xy} the interactive cost of assigning facility x on the site neighbouring facility y.

The solution of the problem can be represented by an $n \times n$ matrix. For example, for a problem of four facilities, a site layout could be represented by Table 1

In this, facility A is assigned to location 2; facility B to location 1; facility C to location 4 and facility D to location 3. Any specific layout can be represented by a matrix similar to Table 1.

Table 1
A solution matrix for a problem of four facilities

Facility	Location			
	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0	0	0	1
D	0	0	1	0

3. Genetic algorithms

Genetic algorithms are modelling techniques based on biological behaviour (Wilson, 1997). They rely on the speed of computers either to combine elements from two solutions, or to mutate a single solution to a complex problem to produce a third solution and evaluate it. If the third solution is 'better' than one of the others, then it 'survives' and the worst one 'dies'—along the lines of 'survival of the fittest in Darwin's theory of evolution'. The process continues through a number of iterations or 'generations' with each solution contributing to the next generation in proportion to its 'goodness'. Random factors ensure that the solution space is adequately covered. (See Dowsland, 1996 for a brief introduction to Genetic Algorithms and a discussion on their possible use to assist managers.)

Since genetic algorithms are generic and flexible and need little knowledge and information about the problem domain, they have found wide application in diverse areas (Goldberg, 1989). There have been many applications of genetic algorithms in civil engineering. Soh and Yang, (1996) used a genetic based search technique for shape design of structures in civil engineering in the least weight design of truss structure. Navon and McCrea (1997) established a genetic algorithm to optimise a construction robot's kinematics based on collision avoidance, percentage of coverage, dexterity, unit cost and total cost. Al-Tabtabai and Alex (1997) applied a genetic algorithm to manpower scheduling optimisation. There are also many other application cases for using genetic algorithms in scheduling, planning and management.

The main aspects to be considered in the development and use of genetic algorithms are encoding, fitness function, the selection procedure, crossover, mutation operations and termination.

4. Construction site layout using genetic algorithms

There are several possible genetic algorithm formulations of the construction site layout problem. In this paper, a sequence-based formulation is proposed and investigated.

4.1. Sequence-based encoding

A specific solution to the site layout problem as shown by Table 1 is a very sparse matrix and would therefore consume considerable computing resources if it were used for large, practical problems. However, because of the property of one to one correspondences between facilities and locations, a sequence can be used as a more efficient alternative.

For example, for a problem of m facilities that are labelled as A, B, C,…, a sequence S can be established which contains a permutation of all the labels. S can be interpreted as: assigning facility $S[i]$ to site i, where $i \in [1, m]$. Different sequences mean different layout solutions and any manipulation of the sequence S will correspond to a new layout. The sequence of facilities can consequently be used as a chromosome representation scheme. If this is done, the simple forms of crossover and mutation described in classical papers on genetic algorithms are not directly applicable and possible alternatives are described below.

4.2. Crossover operators

Sequence-based representation has been used in genetic algorithms for many problems (Davis, 1991) and many related crossover operators have been developed. In this section, four sequence-based crossover operators, chosen from the literature, are described and their performance for the site layout problem compared.

Crossover 'a' (*order crossover 1*). The order crossover operator was originally developed by Davis (1985) and Davis (1991). In this operation, two crossover locations are generated randomly and an offspring inherits the elements between the two crossover points from a selected parent in the same order and position as they appeared in that parent. The remaining elements are from the other parent in the order in which they appear in that parent, beginning with the first position following the second crossover point and skipping over all elements already present in the offspring. The operation can be represented as

Step 1: Randomly generate two integers $t1$ and $t2$, $t1 < t2$, $t1$, $t2 \in [1, N]$
Step 2: For $i := t1$ to $t2$ do
Child1[i]: = Parent1[i];
Remove $j \in$ Child from Parent2;
Step 3: for $i := t2 + 1$ to N and $i := 1$ to $t1 - 1$
Child1[i]: = Parent2[j], $j \in [1, N]$ and Parent2[j] $\notin$ Child1.

This is illustrated in Table 2.

Table 2
Generation of children by order crossover 1

Parent 1	1	2	3	4	5	6	7
Cross at			*		*		
Parent 2	3	6	2	5	1	7	4
Child 1	1	7	3	4	5	6	2
Child 2	6	7	2	5	1	3	4

Table 3
Generation of child by order crossover 2

Parent 1	1	2	3	4	5	6	7
Cross at		*		*	*		
Parent 2	3	6	2	5	1	7	4
Child 1	6	2	3	4	5	1	7

This operator was meant to preserve the relative order of elements in the sequences to be combined. However, through this operation, the offspring actually inherits part of the order, adjacency and absolute position from one parent and the relative order from the other.

Crossover 'b' (*order crossover 2*). Introduced by Syswerda and Palmucci, (1991), this operator differs from the order crossover 1 in that in this method several positions are chosen randomly from a parent and the order of the activities in the corresponding positions in the other parent is imposed on the same elements in the first parent to make an offspring.

Suppose positions 2,4,5 have been chosen for the operation. Child1 inherits elements 2 4 5 from parent 1 directly, child1 = (, 2, ,4,5, , ,) and in the positions 2 4 5 in parent 2, the elements are 6 5 1 and their positions in parent 1 are 1 5 6 so child1 = (6, *2, ,4,5*,1, ,). Copy the other elements from parent 1 and child1 becomes (6 *2* 3 *4 5* 1 7) as shown in Table 3.

Crossover 'c' (partially mapped crossover (*PMX*)). This operation was proposed by Goldberg and Lingle (1987). In this method, two crossover sites are generated randomly and the offspring inherits the elements between these two sites of a parent directly. Each element between the two crossover points in the alternate parent is mapped to the position held by this element in the first parent. Then the remaining elements are inherited from the alternate parent. As in the order crossover 1 the direct transfer of elements of the first parent to offspring preserves order, adjacency and position for that section.

Suppose $w_S(i)$ = the position of activity i in sequence S.

The algorithm for PMX can be as

Step 1: Randomly generate two integers $t1\}$ and $t2$, $t1 < t2$, $t1$ and $t2 \in [1, N]$;
Step 2: for $i := t1$ to $t2$
Child1[I]: = Parent1[i];
Step 3: For $i := t1$ to $t2$
If Parent1[i] $\neq$ Parent2[j] and $j \in [t1, t2]$
Child1[$w_{\text{Parent2}}(i)$]: = Parent2[j]; $j \in [1, N]$ and Parent2[j] $\notin$ Child1.
Step 4: For the rest of elements
Child1[I]: = Parent1[i].

Suppose $t1 = 3$ and $t2 = 4$, then using this method, child1[i]: = Parent1[i], $i \in [3, 4]$. For the first selected

Table 4
Generation of child by PMX

Parent 1	1	2	3	4	5	6	7
Cross at			*		*		
Parent 2	3	6	2	5	1	7	4
Child 1	2	6	3	4	1	7	5

Table 5
Generation of child by cycle crossover

Parent 1	1	4	3	2	5	6	7
Start point			*				
Parent 2	3	6	2	5	1	7	4
Child 1	1	6	3	2	5	7	4

element 3 of parent 1, it is the first element in parent 2, so child[1]:= Parent2[3]:=2; Off[7]:= Parent2[4]:=5, the rest of the elements of child are inherited from parent 2 directly and the child is as shown in Table 4.

It should be noted that PMX is influenced by position and in some cases it results in a mutation where neither adjacency, position nor relative order is preserved.

Crossover 'd' (*cycle crossover*). This method, originally developed by Oliver et al. (1987), preserves the absolute position of elements in the present sequence. In this method, a starting position is randomly selected. The offspring inherits the starting element in the starting position of a parent. The element in the same position of the other parent is located in the first parent and the offspring inherits the element from this located position in the other parent. The procedure continues until the starting element is reached again. The rest of the positions of the offspring are filled with elements from the other parent. This method preserves the position of elements from one or the other parent without any disruption.

In the example shown in Table 5, position 3 is chosen as the starting point. Child1[3]:=3; the element of parent 2 in this position is 2 and element 2 appears in position 4 of parent 1, so child1[4]:=2; Parent2[4]=5= Parent1[5], so child1[5]:=5; Parent2[5]=1= Parent1[1] so child1[1]:=1; Parent2[1]=3= Parent1[3], the starting point is reached. We have child1=(*1, ,3,2,5,* ,) and fill the empty positions with elements in the same positions of parent 2 thus we get child1=(1,6,3,2,5,7,4).

4.3. Mutation operations

For order based chromosomes, the conventional mutation of bit switch cannot be applied directly. In the following sections, four order specific mutation operators are described.

Mutation '*a*'. Davis (1991) proposed a so-called scramble sub-list mutation for his order based genetic algorithm. In this mutation, the operator selects a sub-list of the items from a parent chromosome and permutes them in the child, leaving the rest of the chromosome as it was in the parent. Scramble sub-list mutation has no parameters although a variation can be made involving limiting the length of the section of the chromosome that can be scrambled. An example is shown as

Parent: 1 4 *3 2 5* 6 7

Child: 1 4 *2 3 5* 6 7

In this example, a subsection containing 3, 2, 5 is selected and a permuted order 2, 3, 5 is generated and chosen. The rest of the chromosomes remain unchanged.

Mutation '*b*'. The reversal of the sequence of activities has been used as mutation for TSP and proves to be effective and useful. Here it is applied in this GA as a mutation operator as well. In this operation, the whole sequence is reversed. For example, after the operation, a chromosome (1,2,3,4,5) will become (5,4,3,2,1).

This operation can be generalised as to randomly select a subsection of the chromosome sequence and then reverse the subsection only and remain the orders of the rest activities of the chromosome.

Mutation '*c*'. In this operation, a sequence is randomly parted into two sections and then the two sections are reconnected with swapped positions. So if the parting position is k, $k \in (1,n)$, then $S[i-k] := S[i]$, for $i \in [k+1,n]$; and $S[i+k] := S[i]$ for $i \in [1,k]$.

Mutation '*d*'. In this operation, the start of the sequence to be mutated is connected to its end to form a cycle. This cycle is cut randomly to produce a new sequence.

4.4. Fitness function

Fitness values are used to represent the relative goodness of chromosomes. Traditionally, better chromosomes have larger fitness values. In site layout problems, facilities are typically positioned to reduce cost and the objective is usually to minimise the construction cost. In this work, the construction cost is used as the fitness function and the genetic algorithm is designed to produce a minimum rather than a maximum.

5. Numerical examples

In this section, a numerical example is taken from Yeh (1995) and used to demonstrate the genetic algorithm formulation and to compare the performance of the operators.

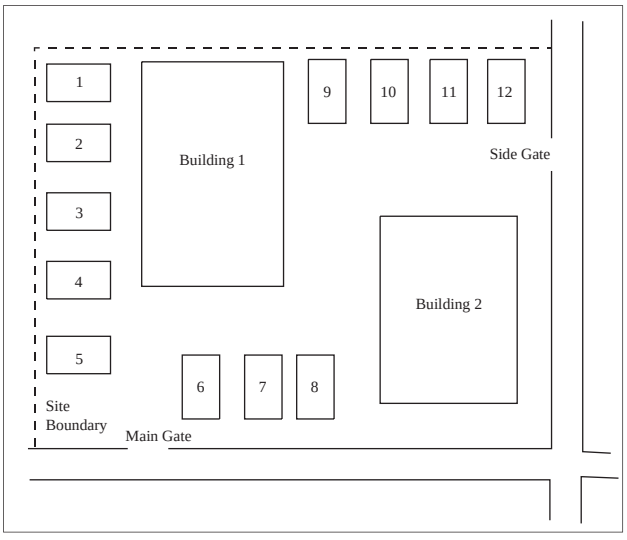

Fig. 1. The example site layout problem (after Yeh, 1995).

In this example, there are two permanent buildings to be constructed. The description of the site including 12 locations where temporary facilities may be placed is shown in Fig. 1. The following 12 facilities are to be positioned:

1. Reinforcing steel shop 1	R1
2. Reinforcing steel shop	R2
3. Carpentry shop 1	C1
4. Carpentry shop 2	C2
5. Falsework shop 1	F1
6. Falsework shop 2	F2
7. Concrete batch plant 1	B1
8. Concrete batch plant 2	B2
9. Job office	JO
10. Labour residence	LR
11. Electricity equipment and water-supply shop	E
12. Warehouse	W

Their data, construction cost matrix (C), site neighbouring index matrix (A) and interactive cost matrix (D) (the unit of all costs in the test case is £1000) are shown in Tables 6–8 respectively.

It should be noted here that the site neighbouring index matrix is slightly different to that proposed by Yeh since on the site, location 1 is not adjacent to location 12 as suggested in the original.

The interactive cost matrix shows that there is a penalty of 100 for positioning facility 1 next to facility 10 or 11 and a similar penalty for positioning facility 2 next to facility 10 or 11. This penalty is large compared to the cost of setting up the facilities in position and so it would be expected to act as an absolute constraint.

All the solutions obtained by the techniques imply that this constraint is, in fact, not tight and thus this part of the fitness function could be removed for part of the testing.

Table 6
Construction cost matrix (C)

	1	2	3	4	5	6	7	8	9	10	11	12
R1	35	35	30	30	35	15	10	15	6	6	7	10
R2	35	30	9	9	13	30	30	35	15	18	12	7
C1	18	15	15	15	15	8	14	10	8	10	15	15
C2	13	7	12	18	18	15	15	15	15	8	8	12
F1	18	15	15	20	15	8	10	8	8	7	15	15
F2	14	8	10	17	12	15	15	15	15	8	7	9
B1	32	35	15	15	15	10	9	13	7	10	15	15
B2	31	30	9	8	15	18	15	16	15	15	15	15
JO	39	35	13	8	8	15	18	15	8	18	9	18
LR	18	8	8	8	15	10	15	15	13	15	15	15
E	7	10	8	19	15	10	10	8	15	10	6	15
W	9	10	6	7	7	7	15	15	18	15	15	12

Table 7
Site neighbouring index matrix (A)

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	0
8	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	1	0	1	0	0
10	0	0	0	0	0	0	0	0	1	0	1	0
11	0	0	0	0	0	0	0	0	0	1	0	1
12	0	0	0	0	0	0	0	0	0	0	1	0

Table 8
Interactive cost matrix (D)

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	100	100	0	0
2	0	0	0	0	0	0	0	0	100	100	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	100	100	0	0	0	0	0	0	0	0	0	0
10	100	100	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0

6. Experiments

A package was written specifically to solve the problem under consideration. It was written in Delphi running under Windows NT. A generation of the

Table 9
Possible choice of parameters

Crossover type	Parameter values	Mutation type	Mutation proportion
a—order crossover 1		a—mutation 1	0, 0.2, 0.4
b—order crossover 2	2,4,6,8,10	b—mutation 2	0, 0.2, 0.4
c—partially mapped crossover		c—mutation 3	
d—cycle crossover		d—mutation 4	

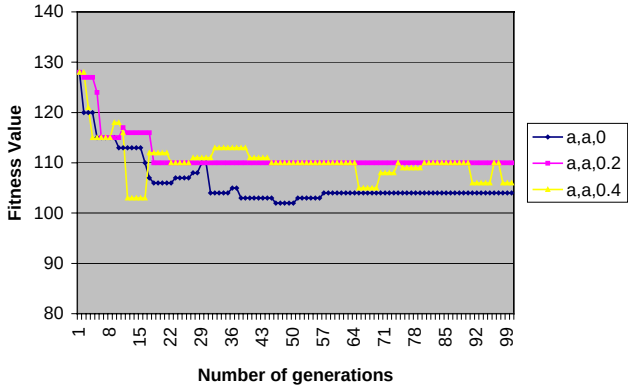

Fig. 2. Typical values of fitness for different generations for order crossover 1, mutation type 1.

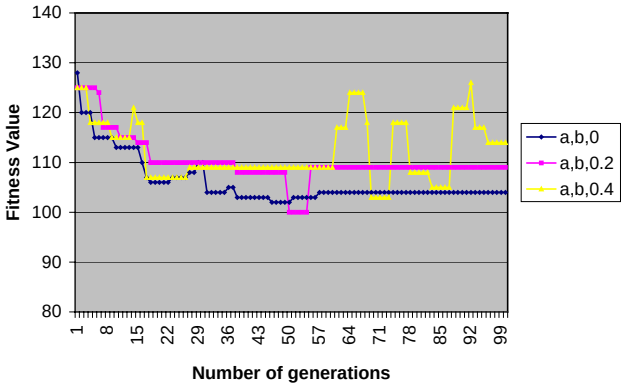

Fig. 3. Typical values of fitness for different generations for order crossover 1, mutation type 2.

problem with a population of 100 took approximately 0.4 s to run on a P2-233 processor with 64 MB of RAM. This timing was taken by averaging and no attempt has been made to optimise the programming.

In order to determine the performance of the operators in the particular problem under consideration, several experiments have been devised. The choice of the parameters is summarised in Table 9.

The behaviour of any genetic algorithm on any problem depends on many factors, some of which are random and, consequently, it is not totally predictable. Two types of experiments were therefore devised.

In the first, the different combinations of the possible values of the parameters from Table 5 were selected and the genetic algorithm was run for 100 generations with a maximum number in the population of 80. In the second, a single combination of parameters was used for multiple runs of the genetic algorithm.

Space does not allow presentation of all the results for the first set of experiments in this paper since there is a large number of combinations possible. Typical results of the first set of experiments are shown in Figs. 2–6.

It can be seen that the behaviour of the algorithm varied widely with different 'optima' being found and with varying amounts of stability in the solution.

Of the examples shown, in the 100 generations allowed, the combination to produce the lowest value for the cost was the order crossover 2 with no mutation. This can be seen in both Figs. 4 and 5. This crossover

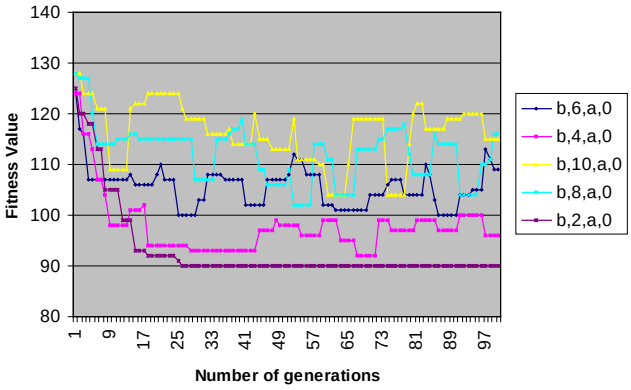

Fig. 4. Typical values of fitness for different generations for order crossover 2, mutation type 1.

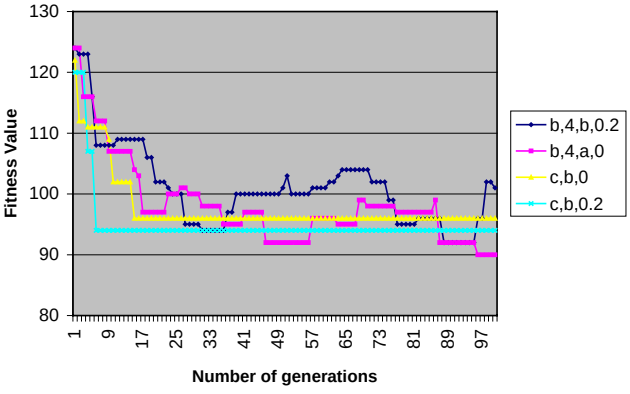

Fig. 5. Typical values of fitness for different generations for order crossover 2 and partially mapped crossover and various mutation types.

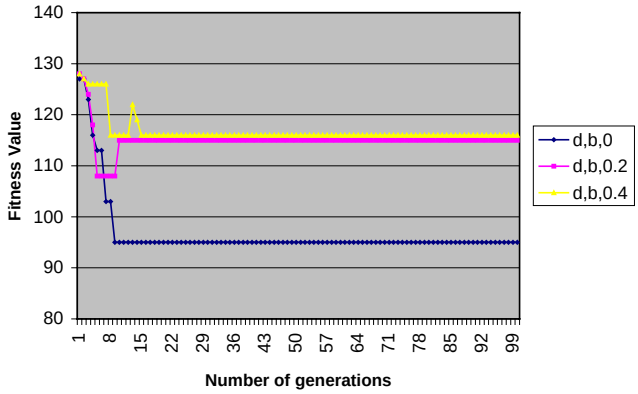

Fig. 6. Typical values of fitness for different generations for cycle crossover, mutation type 2.

Table 10
Location of the facilities in the three optimal solutions

Solution	Facility											
	R1	R2	C1	C2	F1	F2	B1	B2	JO	LR	E	W
1	9	12	6	10	8	11	7	4	5	2	1	3
2	10	12	6	2	8	11	7	4	9	3	1	5
3	10	12	9	2	8	11	7	4	5	3	1	6

performed differently depending on the position selected for the crossover, with an early position giving by far the best results. Crossover 'a' (order crossover 1) generally performed worse than the others on this problem, never producing values of fitness function below 100.

Figs. 4 and 5 illustrate the effects of the initial random population on the performance of the procedure since one of the lines in each figure is for order crossover 2 with crossover at 4 and no mutation. In Fig. 4, the lowest value attained is 92 whilst in Fig. 5 it is 90.

In fact, analysis of the results indicated that three 'optimal' solutions were found as shown in Table 10. The optimal value of the fitness function was 90, which is 4 above the infeasible lower bound solution of all facilities in their optimal position. It was also 3 below the answer suggested by Yeh although, as mentioned earlier, the problem used here was slightly changed from that in the original paper.

Figs. 7 and 8 show the results of 50 runs of the algorithm all for a maximum of 100 generations but terminating if the fitness function attained the value 90 (which was taken to be the optimum). The optimum value was attained in 19 runs of the 50 runs. Values above 92 only occurred in 4 runs. The number of generations taken to attain the 'optimum' value appears not to be related to the value of the optimum.

Fig. 9 shows the results of using the optimum value as the only termination criterion in 50 runs of the algorithm. The parameters used were order crossover 2 with the crossover at 2 and no mutation. The algorithm always found the optimum value although in one case it did take nearly 1000 generations. In more than half of the runs (26), the optimum was found in less than 300 generations.

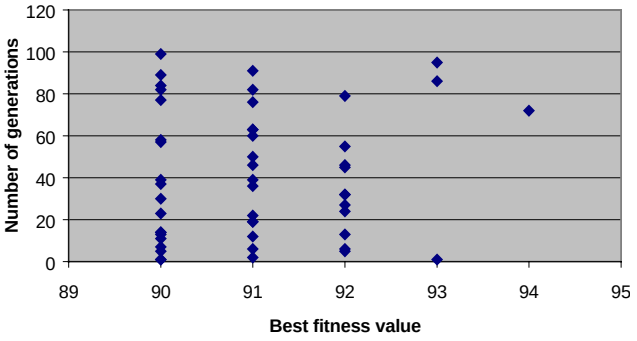

Fig. 7. Number of generations to attain the 'best fitness value' in 50 runs of the algorithm.

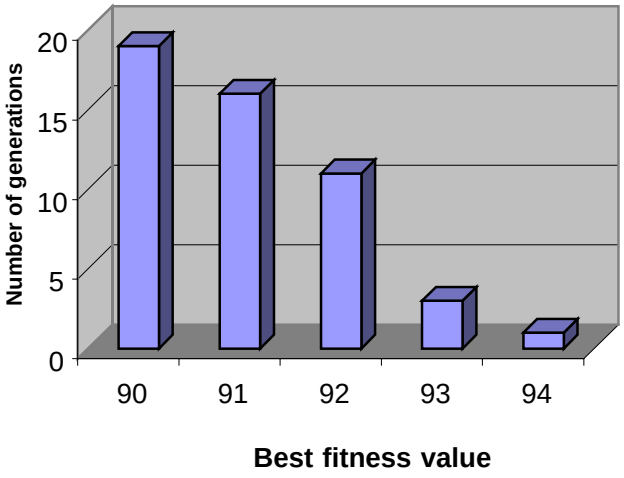

Fig. 8. Frequency of occurrence of different values of best fitness in 50 runs of the algorithm.

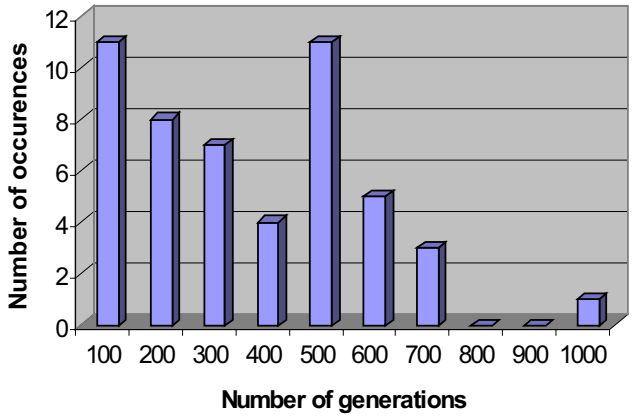

Fig. 9. Number of generations required to achieve the optimum fitness value.

7. Conclusions

It has been shown to be possible to formulate the site layout problem as a sequence-based genetic algorithm.

Four crossover and four mutation operators have been tested for their efficiency at finding the optimum layout for a problem proposed and solved by Yeh.

For the twelve-facility problem, the best crossover operator was found to be order crossover 1 with crossover at position 2. In the tests, it was found to be best to have no mutation. This combination always found the optimum in the particular problem being studied. This was surprising since, without mutation, it would be expected to sometimes converge to local optima rather than always finding the global optimum. This might have happened because there were three equally good solutions to the problem.

Some of the operators never found the optimum in any of the tests carried out.

The example problem formulation provided in this paper can be extended to include many other costs without significant increase in computer power requirement.

Although this problem did not require it, the dynamic nature of a site can be taken into account by the development of several different layouts for the different phases of the work. If this is done, the benefits and costs of moving a facility during the execution of the project can be balanced in the fitness function.

It may be possible to include more complex, less tangible but equally important aspects into the fitness function. Thus, work is ongoing to extend these ideas to include safety and environmental aspects and balance these with the finances of a project.

References

Al-Tabtabai, H., Alex, A.P., 1997. Manpower scheduling optimisation using genetic algorithms. In: Proceedings of the Fourth congress on Computing in Civil Engineering, Philadelphia, June, pp. 702–709.

Davis, L., 1985. Applying Adaptive Algorithms to Epistatic Domains. In: Proceedings of International Joint Conference on Artificial Intelligence, Los Angeles, CA, USA. Morgan Kaufmann Publishers, Los Altos, CA, pp. 234–243.

Davis, L., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

Dowsland, K.A., 1996. Genetic algorithms—a tool for OR? Journal of the Operational Research Society 47 (4), 550–561.

Goldberg, D., Lingle, R., 1987. Alleles, Loci, and the Travelling Salesman Problem. In: Grefenstette, J.J. (Ed.), Proceedings of Second International Conference on Genetic Algorithms and Their Applications MIT. Lawrence Erlbaum, London, pp. 121–134.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimisation, and Machine Learning. Addison-Wesley, Reading, MA, ISBN 0-201-15767-5.

Hamiani, A., 1989. Knowledge representation for the site layout problem. Proceedings of Computing in Civil Engineering, ASCE, Reston, VA, pp. 283–289.

Navon, R., McCrea, A.M., 1997. Selection of optimal construction robot using genetic algorithms. Journal of computing in civil engineering 11 (3), 175–183.

Oliver, I., Smith, D., Holland, J., 1987. A study of permutation crossover operators on travelling salesman problems. In: Grefenstette, J.J. (Ed.), Proceedings of Second International Conference on Genetic Algorithms and Their Applications, MIT, Cambridge, MA.

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

Russell, D.M., Gau, K.Y., 1996. Trends and perspectives: the facility layout problem: recent and emerging trends and perspectives. Journal of Manufacturing Systems 15 (5), 351–366.

Seehof, J.M., Evans, U.O., 1967. Automated layout design program. Industrial Engineering 18, 690–695.

Soh, C.K., Yang, J., 1996. Fuzzy controlled genetic algorithm search for shape optimisation. Journal of computing in civil engineering 10 (2), 143–150.

Syswerda, G., Palmucci, J., 1991. The application of genetic algorithms to resource scheduling. In: Belew, R.K., Booker, L.B. (Eds.), Proceedings of the fourth international Conference on Genetic Algorithms. Morgan Kaufman, San Mateo, CA, pp. 502–508.

Tommelein, I.D., Levitt, R.E., Hayes-Roth, B., Confrey, T., 1991. Sightplan experiments: alternate strategies for site layout design. ASCE Journal of Computing in Civil Engineering 5 (1), 42–63.

Twort, A.C., Rees, J.G., 1995. Civil Engineering Supervision and Management 3rd Edition. Arnold, London.

Warszawski, A., Peled, N., 1987. An expert system for crane selection and location. Proceedings of the Fourth international symposium on robotics and artificial intelligence in building construction, Vol. 1, Haifa, Israel.

Wilson, J.M., 1997. A genetic algorithm for the generalised assignment problem. Journal of the Operational Research Society 48 (8), 804–809.

Yeh, I.-C., 1995. Construction-site layout using annealed neural network. Journal of Computing in civil engineering 9 (3), 201–208.

Zouein, P.P., Tommelein, I.D., 1999. Dynamic layout planning using a hybrid incremental solution method. Journal of Construction Engineering and Management 125 (6), 400–408.