Space Layout Planning using an Evolutionary Approach[1]

Jun H. Jo and John S. Gero

Key Centre of Design Computing
Department of Architectural and Design Science
University of Sydney NSW 2006 Australia
{jun, john}@arch.usyd.edu.au

This paper describes a design method based on constructing a genetic/evolutionary-design model whose idea is borrowed from natural genetics. Two major issues from the modelling involve how to represent design knowledge for the evolutionary design model and the usefulness of the model for design problems. For the representation of design knowledge in the model, a schema concept is introduced. The utility of the model is based on its computational efficiency and its capability of producing satisfactory solutions for the given set of problem requirements. The design problem used to demonstrate the approach is a large office layout planning problem with its associated topological and geometrical arrangements of space elements. An example drawn from the literature is used.

Keywords: space layout planing, genetic algorithms, design rule schema, design gene schema, Pareto optimization.

1. INTRODUCTION

Space layout planning is one of the most interesting and difficult of the formal architectural design problems. It has been examined by many researchers over a long period (Buffa et al, 1964; Eastman, 1975; Liggett, 1980; Akin et al., 1992; Yoon and Coyne, 1992). Three major issues which have arisen from the previous research include: how to formulate this complex and nonlinear problem, how to control the combinatorial nature of the generated solutions, and how to evaluate the solutions based on the multiple criteria associated with the given requirements. A set of discrete but interdependent space elements makes the formulation of the problem difficult. This is due to the difficulties of enumeration of space relationships and the subjective nature of their interdependencies. During the synthesis stage, an enormous number of potential solutions can be generated even with a small number of space elements and this number grows exponentially as the size of the problem increases. This NP-completeness of the space layout planning problem makes it impossible for any process to guarantee to find the optimal solution within a reasonable time and there are no known algorithms for this problem. In the evaluation stage, the multiple criteria for the problem require expensive computations due to the very large number of solutions to be evaluated.

Genetic algorithms (GAs) are search methods inspired by natural genetics. The basic idea is founded on natural adaptive systems, where organisms evolve through generations to adapt themselves to the given environment. Recent work on genetic algorithms has demonstrated their success in solving optimization problems, showing their simple but

[1] To appear in *Artificial Intelligence in Engineering*

powerful search capability. The space, which will be searched by genetic algorithms, needs not be limited by restrictive assumptions concerning continuity, existence of derivatives, unimodality, and other matters (Goldberg, 1989).

Based on the advantages of GAs, genetic evolutionary concepts have been applied in the design area and have shown promising results (Gero et. al., 1994; Jo, 1993; Jo and Gero, 1994; Maher and Kundu, 1994). An issue associated with the application of the genetic analogy to design problem solving is the question of an efficient way of formulating the design problems for genetic search and an appropriate form of communication between the design and genetic evolutionary processes. Schema theory proposes that knowledge is packaged into units which embed descriptions of the contents related to the unit and information about how this knowledge is to be used (Rumelhart, 1980). The schema concept is adopted as a tool for the knowledge formulation and interpretation in the approach described here. This paper describes how an evolutionary design process model is constructed based on the genetic analogy. The utility of the model will be demonstrated by applying it to solve a space layout problem drawn from the literature and by comparing the results produced.

2. SPACE LAYOUT PLANNING

Space layout planning is the assignment of discrete space elements to their corresponding locations while the space elements have relationships among each other. The relationships include topology and geometry, and distinguish space layout planning from the classical linear assignment problem. The relationships make the design process complicated and difficult by increasing the computational cost. An optimal plan is determined by interactions and the travel cost between the space elements in the plan. This implies that space elements which are closely interrelated will tend to be located near each other on the plan.

2.1 General Approaches

Two major problems of space layout planning include the topological and geometrical assignments of space elements to meet certain criteria. Topological space planning is the process of arranging the topological relationships of space elements. The process usually results in a relationship graph in which nodes represent space elements and arcs represent the topological relationships between the elements (Miller, 1971), a bubble diagram (Korf, 1977), or a rectangular dissection (Grason, 1971; Gilleard, 1978). The dimensioning of space elements involves producing the geometric properties of the plan based on the geometric requirements of the problem (Mitchell et al., 1976; Gero, 1978; Liggett and Mitchell, 1981; Balachandran and Gero, 1987).

These topological and geometrical problems have generally been implemented separately, in which the topological problem is often implemented using grammars whereas the geometrical problem has been solved using mathematical programming or related optimization techniques. The grammatical or generative approach uses shape grammars

whose idea is based on linguistic grammar systems (Chomsky, 1957). This approach produces all possible alternatives exhaustively. Shape grammars are algorithms which use sets of composition rules for the generation of shapes and have been used to generate architectural and other spatial designs (Stiny and Mitchell, 1978; Koning and Eizenberg, 1981). Optimization techniques, including linear programming (Mitchell et al, 1976), nonlinear and dynamic programming (Gero, 1977), have been applied to produce the dimensioning of floor plans. Jo (1993) attempted solving both problems together by using the evolutionary design process model. In this approach, a set of shape rules generates a space plan and the solutions are evaluated against the given multiple criteria by using the Pareto optimization technique.

2.2 Limitations of the Conventional Approaches

Notwithstanding much of the previous research, the use of programs for space layout problem still has difficulties in problem formulation, generation and evaluation because of the complexity of the problem, the combinatorial nature of the potential solutions, and the sophisticated control required. Some of these difficulties are described below.

(1) A design problem generally has multiple criteria which are to be formulated and against which the solutions are to be evaluated. This makes the evaluation of alternative solutions very expensive computationally.

(2) The design solutions generated by a small number of space elements easily form a large solution space. As the number of elements increases, the configuration of solutions increases exponentially. Thus, the problem is a NP-complete (non-deterministic polynomial) problem which is one major drawback of the generative approach for space layout problems. Table 1 shows the number of possible solutions against the number of space elements. Currently, there are no known efficient methods which guarantee optimal solutions only approximate solution strategies. Therefore, the space layout problem has no efficient algorithms but developing methods for generating "approximate solutions that are good even if they are not precisely optimal" (Lewis and Papodimitriou, 1978) remains a useful task. In general, heuristic solution techniques which produce ‘good’ results have been developed which incorporate a variety of schemes for limiting the set of solutions explored.

n	Number of solutions	n	Number of solutions
(Feasible to solve by hand)		(Feasible to solve by computer exhaustively)	
1	1	7	5040
2	2	8	40320
3	6	9	362880
4	24	10	3628800
5	120	11	39916800
6	720	12	479001600

Table 1. Numbers of space elements ‘n’ and their possible solutions (Liggett, 1980).

(3) Various heuristic methods have been developed to improve the search efficiency. However, the quality of the solutions produced depends totally on the quality of the techniques, or heuristics employed, and the final solution is generally a local optimum. For example, Liggett (1980) employed a combination of the constructive initial placement technique with the improvement procedures for the space layout problem. The first technique finds the initial solution which is globally satisfactory. The second technique, then, improves the initial solution by a pair-wise technique. However, there is still the danger that solutions will be trapped in a local optimum because the constructive method does not guarantee the production of the global optimum boundary. Also, if no good solutions are found, there is no way to escape from the local optimum. The generation of various 'good' solutions needs a technique which can search the entire design space in a global manner. It may need a technique capable of traversing the entire design space to find more varied and better solutions. However, the classical approaches, of which the pair-wise improvement method is typical, modify a solution in a step-by-step manner and the final solution is generally 'close' to the initial solution.

3. GENETIC EVOLUTIONARY PROCESS

In nature, organisms evolve through generations to adapt themselves to a complicated and changing environment. The evolution process is continuous and cyclic, and can be described by a set of individuals and a set of biological transformations over the populations composed of these individuals. The knowledge of evolution is guided by itself and inherited from individuals. Features for self-repair, self-guidance, and reproduction are the rule in biological systems, whereas they barely exist in the most sophisticated artificial systems (Goldberg, 1989).

Genetic evolution concepts have been introduced into the artificial world as bases for constructing computational models such as genetic algorithms (Holland, 1975; Goldberg, 1989), and genetic programming (Koza, 1992). These evolutionary models use adaptive methods based loosely on the processes of biological organisms. They have mainly been applied to solve optimization and search problems, showing some superior capabilities of search and advantages over many conventional search methods. The primary concept, which is involved in the evolutionary process, is that the combination of characteristics of different individuals can sometimes produce offspring whose fitness is greater than that of either parent. Over the generations, the characteristics evolve and produce new and better solutions. In this paper, the strategy employed for the new model will be based on genetic algorithms.

The genetic algorithm is a simple routine and blind process which does not necessarily need any specific heuristic guidance. The solutions are, therefore, varied without any predetermined prejudices. These domain-independent characteristics make the genetic search process applicable in a broad range of domains. Particularly in multimodal (many-peaked)

search spaces, point-to-point search methods have the danger of locating on local optima. By contrast, the genetic search method climbs many peaks in parallel, thus there is "safety in numbers" in finding the global or near-global optimal solution and the probability of finding a false peak is reduced compared to many other methods.

As in the case of a knowledge-based design model, the representation and the process of a genetic evolution model can be considered separately. This helps not only our understanding about the model but also the application of the model within a design process model.

3.1 Genetic Representation

The terminology of genetic representation is based on natural genetics terms and provides a basis for the knowledge representation for the evolutionary process. The knowledge of an individual is represented at two levels; the genotype level and the phenotype level. A genotype is the implicit representation of an individual. Instead of dealing with the knowledge of an individual's structure, this representation at an alternate level makes transformations easy and varied. The phenotype is the decoded genotype at the physical or structure level. The behaviours of an individual can be observed on the phenotype, therefore, the evaluation task is performed at this level.

This separation of genotype and phenotype in natural systems offers clear and obvious advantages. The phenotype lives in the world and can be altered by the world. For example, a human may lose a limb in an accident, however, no such direct modification of the phenotype is transmitted to the offspring. This is a good idea otherwise offspring would have fewer and fewer limbs and organs as their parents lose them through misadventure. The genetic material is transmitted from generation to generation not the phenotype. For an excellent introduction to the concepts of natural evolution the reader could do no better than consult *The Blind Watchmaker* (Dawkins, 1987). There is a rather different advantage in a computational model of design in utilising this genotypic representation of the design: it is the ease with which the representation can be manipulated and the large scale consequential effects on the structure of the designs which can result from that manipulation.

In genetics, the whole information of an individual structure is stored in a genotype string or a chromosome as genetic codes. The genotype string(G) is composed of a finite set of genes and their values, called alleles. In the artificial world, a gene(g_i) can be considered as an instruction in a recipe and is represented as a particular character or a set of characters in a string. A locus is the position of a gene and is identified separately from the gene's function (Goldberg, 1989). A genotype combines the separate information of a set of genes and constitutes an entire individual structure. All the genetic transformations (crossover and mutation) happen at the genotype level.

$$G = \{g_i\} \qquad (1)$$

where,

G genotype
g_i i-th gene,

The genetic search process works with a population of genotype strings and the expression of a population of the t-th generation is described below (a generation in genetic algorithms results when applying the evolutionary operators to produce a population containing a new mix of genes and those genotypes are transformed into phenotypes):

$$p(t) = \{G_1, G_2, G_3, ..., G_n\} \quad (2)$$

where

p(t) population of the t-th generation.

A phenotype is the outward, visible expression of a genotype string or an individual. The interpretation of a genotype to its phenotype allows realization of the structure of an individual. Because a phenotype is tangible and confronts the environment, the behaviours or the fitness of a structure can be observed through the phenotype, P. The fitness, F, is defined as the performance of an individual structure in its environment.

$$P = m(G) \quad (3)$$

where

P phenotype,
m mapping or interpretation operator,

$$\text{and } F = \varphi\{P\} \quad (4)$$

where

F fitness,
φ operator which transforms the phenotype into its fitness,
P phenotype.

3.2 Genetic Operations

The biological genetic operations include recombination and natural selection. The recombination operations transform individuals and produce a new population carrying different features from those of the former population. Some individuals are then selected based on their fitness in the given environment. If an individual has better behaviour than those of others, it has more chances to be selected. By using these operations, organisms evolve through generations to adapt themselves to the environment.

The recombination operators comprise crossover and mutation. Crossover allows the combination of different individuals (parents), in the form of genotype strings, to swap their

information with each other and therefore to produce hybrid information (children) which may have better performances than their parents, Figure 1. The crossing site can be chosen at random and specifies the location where the information is swapped in the parent genotypes.

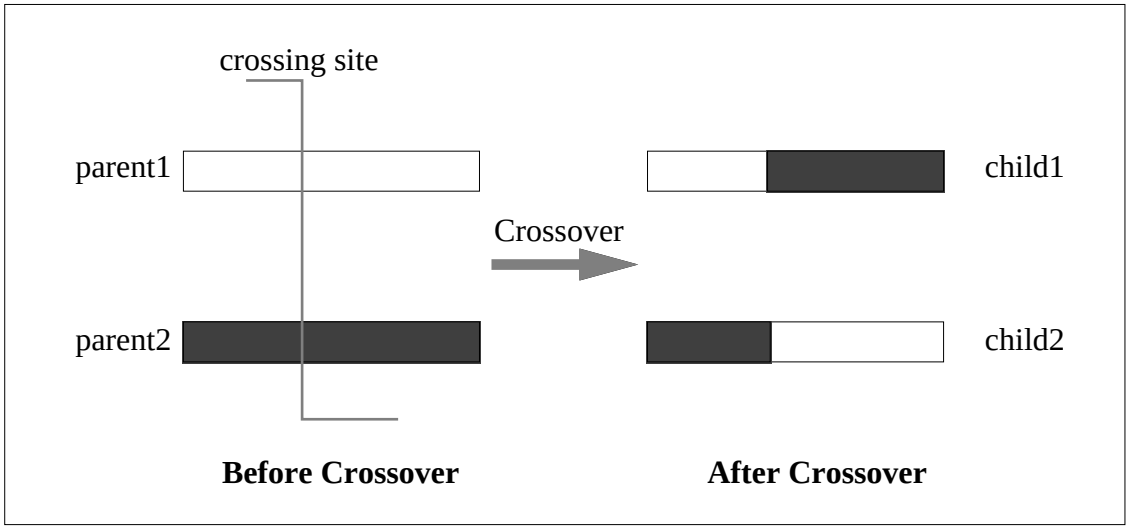


Figure 1 A schematic of simple crossover shows the alignment of two parent strings and their two children generated by the partial exchange of genetic information.

Mutation is an alteration of the value at a random position in the genotype string. If the genes are represented by binary digits then the genotype is a bit string. The example below presents the mutation operation being applied to the second bit (underlined) of a genotype string.

1<u>0</u>10 --> 1<u>1</u>10

The use of the mutation operator together with crossover provides insurance against the development of a uniform population incapable of further evolution (Holland, 1992). For example in the binary function optimization which is to maximize the value of a binary string, crossover between string1, 1010 (fitness = 10), and string2, 1001 (fitness = 9) never attains the global optimum which is 1111 (fitness = 15) without mutation on the second bit of string1 or string2. Figure 2 shows how the crossover and mutation operations move the current states to some others on the search space. In this example, the crossover operation occurs between parent1 (0010), and parent2 (1100), where the crosspoint is between the second and third bits on each parent. The mutation happens on the fourth bit on the string, 1100, and produces a new string, 1101. This string could never be produced by crossover alone.

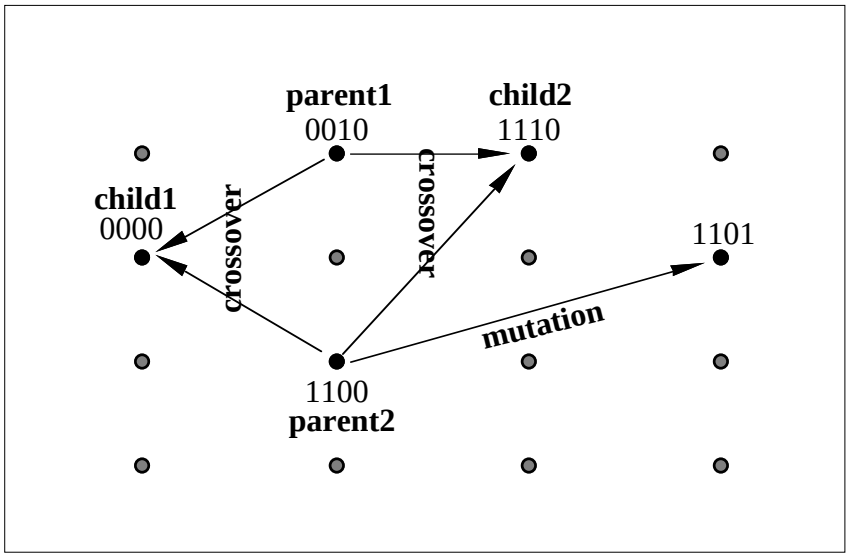


Figure 2 Crossover and mutation operations in a search space. Each dot represents an individual state in the search space.

The selection is carried out based on each individual's fitness in the environment. The goals and evaluation devices form a simulated environment in which individuals either 'live' or 'die' (Goldberg, 1989). A probabilistic method allows the more highly fit individuals to have more chances to be selected than the lowly fit ones. This means that individuals with a higher value have a higher probability of contributing one or more offspring in the next generation. Here the inferior solutions also have a chance of being selected and therefore the process explores a wide range of the search space including regions which may not be considered by other methods. The offspring replace the parent strings which have low fitness which are discarded at each generation so that the total population remains the same size. The selected solutions, or new population, are then sent to the mating pool and are manipulated by the recombination operators.

The selection operation can be implemented in a number of ways. One of them employs a roulette wheel where each individual in the population has a slot on the roulette sized in proportion to its fitness function values compared to others. By spinning the weighted roulette wheel, more highly fit individuals are selected and they will have a higher number of offspring in the succeeding generation.

3.3 Genetic Evolutionary Process in Design

The application of the evolutionary approach in the design area provides a complementary way of overcoming some limitations of design problems. The parallel search with a fixed population prevents the combinatorial explosion problem and helps the process to escape from local optima traps. The alternative, or genotypic, representation of design knowledge allows the formulation of many design problems as homogeneous strings which makes the

transformation operations efficient. The representation also does not necessarily need any strong mathematical formulation and is often easy to formulate. The random but probabilistic method of selection does not guarantee the global optimum. However, the use of simple genetic operations, crossover and mutation, helps to improve the resulting solutions irrespective of the initial population.

4. EVOLUTIONARY DESIGN PROCESS MODEL

An evolutionary design process model is constructed using the concepts of the evolutionary search process (Jo, 1993; Jo and Gero, 1994). The framework for this new design model is the design analysis-synthesis-evaluation process, while the synthesis stage is implemented using the genetic search operation. Since the design model is composed of heterogeneous processes, a means of interpretation based on the schema idea is introduced. The model is domain independent; it can be applied to any design synthesis composition problem by using appropriate design knowledge including design elements and evaluation functions, and by modifying schemas.

4.1 Genetic Representation of Design Knowledge

One of the major issues from the use of genetic search process in the design model is how to formulate design knowledge and make both representations of designs and genetics communicate with each other. The design schemas used in this model include the design rule schema and the design gene schema. The design rule schema plays the role of formulating design knowledge as design elements manageable by the design process. The design gene schema is the translated design rule schema for the genetic search mechanism to recognise and manipulate the design elements in genetic codes.

4.1.1 Design rule schema for knowledge formulation

The design rule schema is defined as a class of design transformations (Jo, 1993). It formulates the relevant design knowledge as a homogeneous set of design rules. Then a set of design rules is instantiated from the design rule schema. A design rule schema includes a target situation (LHS), described by its components, and a transformation operator (τ). The result (RHS) of the rule application is not included in the schema because it is not manipulated by the design process but only appears in the phenotypic structure. The general form of a design rule schema is:

$$S_r = \{LHS, \tau\} \qquad (5)$$

where

LHS	left hand side
S_r	design rule schema
τ	transformation operator

4.1.2 Design gene schema for knowledge interpretation

Because the design model adopts the genetic engine as a search mechanism, the design rules need to be expressed in their genetic terms in order to be manipulated by the genetic engine. While keeping the original semantics of the design rule schema, a design gene schema is produced by restructuring the design rule schema based on the following principle: if a component of a design rule schema needs to be transformed, the component is active and translated into the design gene schema. The other components are inactive. They are not included in the design gene schema and kept in the interpretation knowledge (K_i). The fact that the actual translation does not happen for every design rule, or instance, but on its schema, or class, makes the translation task efficient, especially for the case with a large number of design rules. Design genes are instances of the design gene schema and are genotypic representations of design rules. They can often be expressed as binary numbers or symbols. The design gene schema allows for consistent information maintenance during the genetic transformation process and the transformed solutions can be recognised and translated into the design world correctly. The translation of a design rule schema into a design gene schema is:

$$S_g = \tau_s(S_r, K_i) \qquad (6)$$

where

K_i	interpretation knowledge
S_g	design gene schema
S_r	design rule schema
τ_s	schema translator

The interpretation knowledge (K_i) is specified by the user. It provides the information required for the translation between the representations of the design and the genetics. The information concerns the design rule schema, the design gene schema, design variables and their possible values, active/inactive elements, and the initial/terminal rules, etc. When genotypes are decoded in order to be evaluated, the interpretation knowledge guides the process by providing such information.

In Figure 3, for example, there are four design rules instantiated from a design rule schema. Several components in each design rule always have constant values, such as same square shapes and a marker and do not need to be transformed. Therefore the active component in the example is only the transformation action and this becomes the component of the design gene schema. The inactive components and the design rule schema are kept in the interpretation knowledge and will be recalled when the design genes are mapped into their phenotype. A design gene schema can, then, instantiate any number of design genes by assigning possible values to the components.

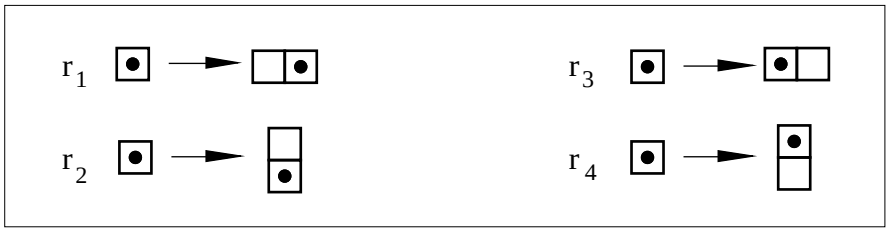


Figure 3 Simple design rules with a marker •. Only the transformation action among the components of each design rule identifies each rule and becomes a component of the design gene schema.

Based on the strategy of the translation between the design rule schema and the design gene schema, the rules of Figure 3 can be transformed to the design genes:

S_r = {LHS, τ}
= {E_x, (E_n, α)}

where
E_x existing design element,
E_n new design element
α transformation action and possible values are {$\rightarrow$, $\downarrow$, $\leftarrow$, $\uparrow$}

The instantiated design rules are:

r_1 = {□, (□, $\rightarrow$)},
r_2 = {□, (□, $\downarrow$)},
r_3 = {□, (□, $\leftarrow$)},
r_4 = {□, (□, $\uparrow$)},

where
$\rightarrow$ puts a new element to the right side of the marked element,
$\downarrow$ puts a new element below the marked element,
$\leftarrow$ puts a new element to the left side of the marked element,
$\uparrow$ puts a new element above the marked element.

There is only one active component to be transformed: transformation action. Therefore,

$S_g = \{\alpha\}$

4.1.3 Representation of genotypic information

A design gene carries a set of active design elements in its genetic language. The design rules in Figure 3 can be represented in two digit binary codes.

Binary representations	Symbolic representations
00	$\rightarrow$
01	$\downarrow$
10	$\leftarrow$
11	$\uparrow$

A genotype is a finite set of design genes, combining separate design information into an entire individual structure. Figure 4 shows an example of a genotype 011010111100 composed using the design gene schema, $S_g = \{\alpha\}$, where α is defined using the four rules in Figure 3. Figure 4 shows six applications of S_g and their phenotypes.

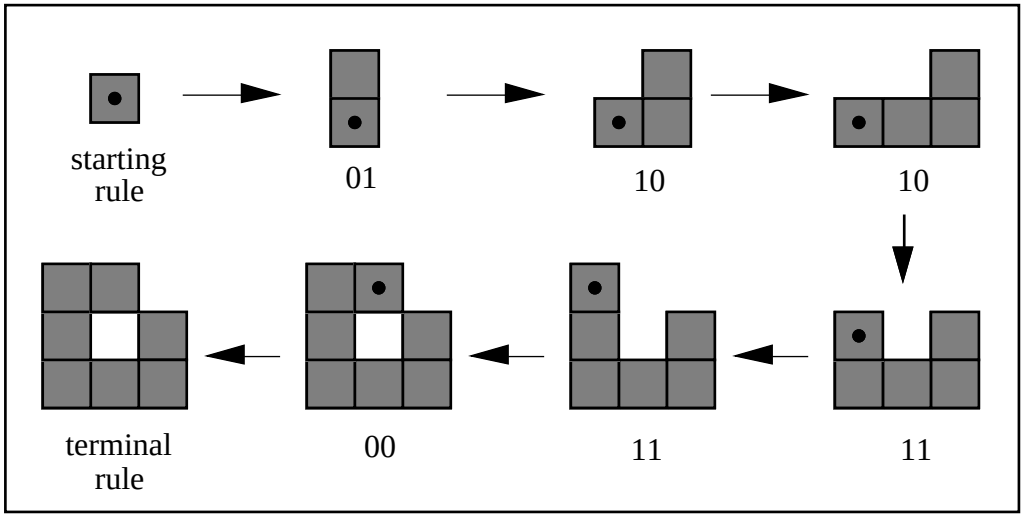


Figure 4 An example of a genotype, 011010111100, and its phenotype. The interpretation knowledge provides necessary information including the starting and terminal rules.

4.2 Evolutionary Design Process

The whole process of the evolutionary design model includes a design analysis process (not formalised here), a genetic search process and a design evaluation process. The design analysis process analyses the given design problems, and retrieves and formulates design elements which will be manipulated by the following processes. This process uses the design rule schema for formulating the design knowledge. The genetic search process transforms the design elements and generates new design solutions. This process is continuous and cyclic, and concludes when the termination conditions are met. The design evaluation process evaluates the final solutions, often by the user. Among the three processes, only the genetic search process is implemented with the computer here.

The genetic search process is composed of an initialization stage and three iterative operations: evaluation, selection and recombination. During the initialization stage, a population of genotype strings is generated by randomly seeding the genotype where each genotype string represents a potential solution. Values for a number of parameters affecting the process are assigned during this stage. Thereafter the evaluation-selection-recombination loop runs iteratively until some specified termination conditions are met.

In the evaluation operation, the performance of a newly generated individual is evaluated against the imposed constraints which are specified by the given requirements. Since the generated individuals are represented as genotypes, they need to be decoded to their phenotypes in the beginning of this operation so that their fitness values can be derived. For

the evaluation a Pareto optimization technique is employed. Pareto optimal solutions, which are often called non-dominated or non-inferior, are solutions whose behaviours are not dominated by others (Radford and Gero, 1988). This method transforms the fitness values of solutions and returns their vector values. The Pareto optimization technique allows the efficient evaluation of multicriteria solutions.

For the selection operation, an imaginary roulette wheel is used. The slot angles of the roulette wheel are sized based on the Pareto value of each individual. Sometimes, the size of the slot angles need to be adjusted in order to reduce or increase the prejudice among solutions. Several methods of assigning the slot angles have been elaborated (Jo, 1993). Table 2 and Figure 5 show how to translate a Pareto optimal value (B_{p_i}) to a slot angle (α_i) where both are inversely proportional to each other. A variable A_i is introduced to invert the Pareto optimal value.

$$A_i = \sum B_{p_i} / B_{p_i}$$
$$\alpha_i = A_i / \sum A_i * 360°$$

where,

A_i inverse value of each Pareto optimal value,
B_{p_i} individual fitness or the Pareto optimal value,
α_i a slot angle.

	B_{p_i}	A_i	α_i
s_1	2	6.5	74.5°
s_2	3	4.3	49.3°
s_3	1	13.0	149.1°
s_4	4	3.3	37.8°
s_5	3	4.3	49.3°
	$\sum B_{p_i} = 13$	$\sum A_i = 31.4$	$\sum \alpha_i = 360.0°$

Table 2 An example of slot angle assignment.

There are five solutions with their Pareto optimal values shown in Table 2. The solution s_3 has the best Pareto optimal value, s_1, has the second best , s_2 and s_5 have the third best value and so on. Then each Pareto optimal value (B_{p_i}) is transferred to the slot angles (α_i) according to its fitness ratio (A_i) against the total fitness. Figure 5 shows a graphic representation of the Pareto values and the transferred slot angles in the imaginary roulette.

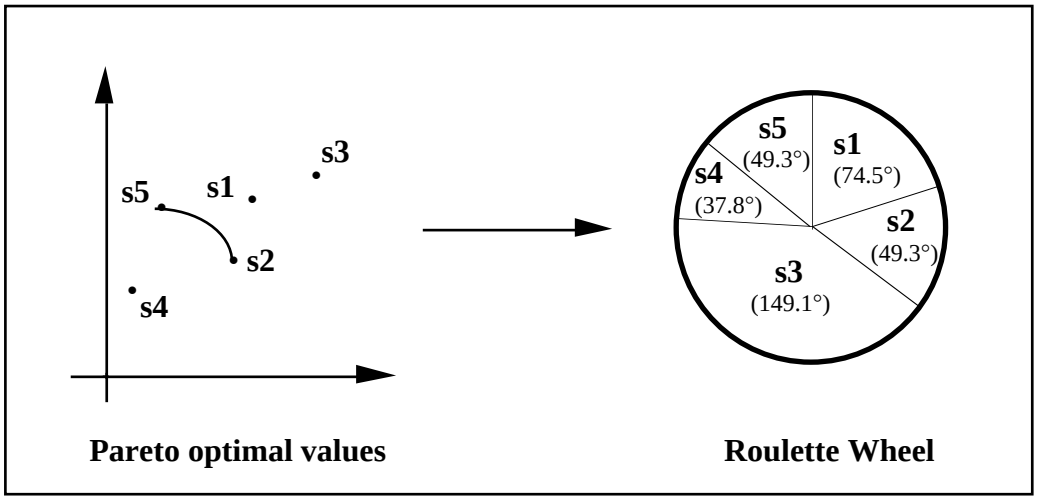


Figure 5 An individual's Pareto optimal value will produce its slot angle on the roulette.

By spinning the roulette wheel, a population of solutions is selected for the new generation. They are sent to the mating pool for the further transformations by the recombination operators.

The recombination operators, crossover and mutation, transform solutions and generate new design solutions. The crossover operator mixes up the design information of different individuals to produce hybrid design information, whereas the mutation operator modifies a part of design information to create new design information. The number of positions for crossover or mutation in a genotype string is one or more.

5. EXAMPLE

The Evolutionary Design based on Genetic Evolution system, called EDGE (Jo, 1993), is a computational design system based on this evolutionary design model. The EDGE system is implemented in C on a Unix platform. In this section, we show how the system is applied to solve the space layout problem which was attempted by Liggett(1985), in order to see how the design model performs for a practical design problem.

5.1 Problem Specifications for the EDGE System

The given problem is a topological and geometrical assignment problem in which office departments are to be placed in a four-level terraced building divided into 17 zones, Figure 6(a). In general, this type of problem consists of certain fixed locations to which a number of discrete facilities are to be assigned. The design elements for the current problem include a set of activities or space elements, a space in which to locate the activities, an operator to locate a specific activity to a specific location, a strategy to control the operation, and evaluation criteria. The solution produced by Liggett's approach (Liggett, 1985) is shown in Figure 6(b), in terms of activity labels in zones.

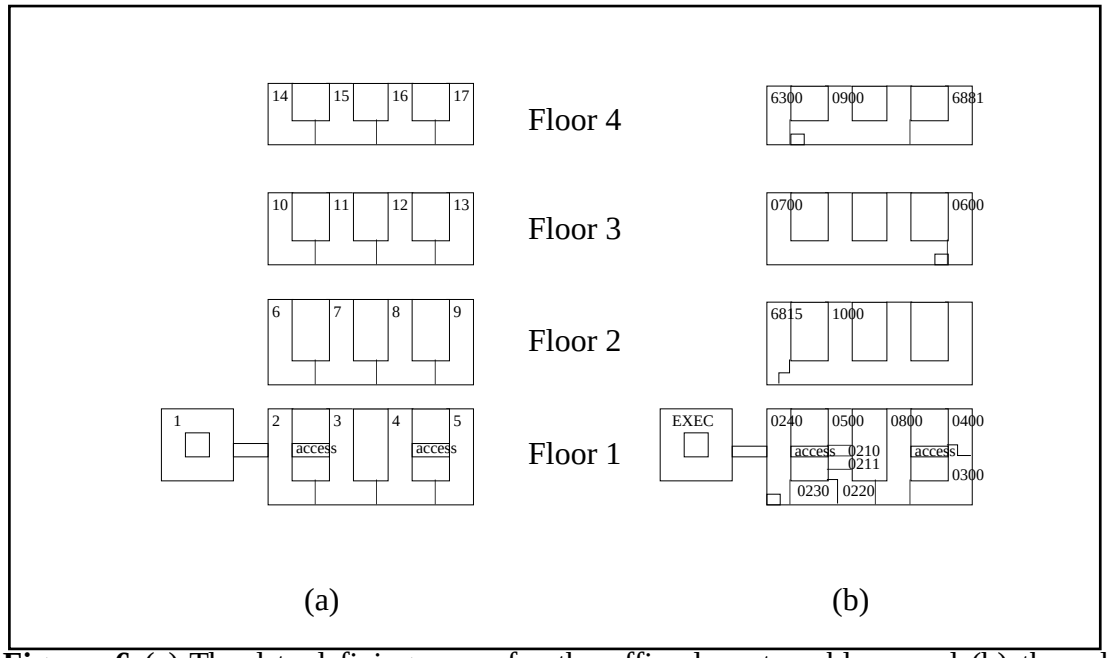


Figure 6 (a) The data defining zones for the office layout problem, and (b) the solution produced by Liggett's system.

5.1.1 Activities and locations

Each activity is defined in terms of area requirements or as a number of equal-sized modules, and the location is considered as a uniform grid. At the beginning of the process, selected activities can be preassigned to specific locations. The basic activities used in Liggett's approach include 21 departments including 3 preassigned activities thus the number of possible solutions is 19! Figure 6(a) and Table 3 show the zone definitions, whilst Table 4 shows the activity definitions.

ID#	Area (sq feet)	Description	# of modules
1	35100.	1st floor - exec wing	39
2	18000.	1st floor - west	20
3	19800.	1st floor - west/central	22
4	18000.	1st floor - east/central	20
5	18000.	1st floor - east	20
6	16200.	2nd floor - west	18
7	18000.	2nd floor - west/central	20
8	16200.	2nd floor - east/central	18
9	16200.	2nd floor - east	18
10	14400.	3rd floor - west	16
11	16200.	3rd floor - west/central	18
12	14400.	3rd floor - east/central	16
13	14400.	3rd floor - east	16
14	12600.	4th floor - west	14
15	14400.	4th floor - west/central	16
16	12600.	4th floor - east/central	14
17	12600.	4th floor - east	14
18	2700.	public access - west	3
19	2700.	public access - east	3
Total :	292500.		325

Table 3. Zone definitions for office layout problem used by Liggett (Liggett, 1985).

ID#	Area (sq feet)	Description	# of modules
1	28800.	dept. exec	32
2	1737.	dept. 0210	2
3	1527.	dept. 0211	2
4	7129.	dept. 0220	8
5	7537.	dept. 0230	8
6	13366.	dept. 0240	15
7	11952.	dept. 6815	13
8	13409.	dept. 0300	15
9	6227.	dept. 0400	7
10	5423.	dept. 0500	6
11	10712.	dept. 0600	12
12	47796.	dept. 0700	53
13	8857.	dept. 6300	10
14	14495.	dept. 6881	16
15	16593.	dept. 0800	18
16	28293.	dept. 0900	31
17	54848.	dept. 1000	61
18	2700.	public access - west	3
19	2700.	public access - east	3
20	3600.	exec garden	4
21	2700.	exec access	3
Total :	290401.		322
			% of total space available : 99%

Table 4. Activity definitions for office layout problem (Liggett, 1985).

Activities are assigned from the top to the bottom floor, following the instructions which are given via a genotype string. Since the perimeter of a building is fixed, solutions often include a department being over two different floors. This may increase the travel cost between modules of the same department. One strategy employed in this work is that, if one floor is full, the assignment is continued from a position which is on the next floor and is just below the previous assignment. Figure 7(a) shows how activities are assigned over different floors. Figure 7(b) shows how modules of activities are mapped on to locations over one floor. In a zone, units of a department are often forced to segment. In this case it is recommended that the designer participates and modifies the solution manually in order to complete it.

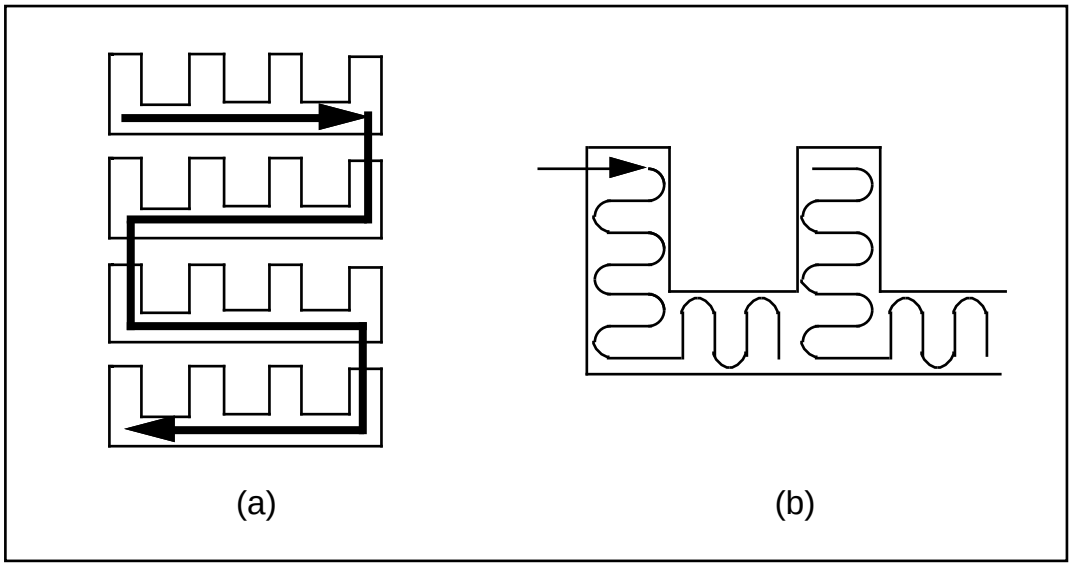


Figure 7 Assignment of activities over different floors (a) and over a floor (b) in the EDGE system.

This system employs the single stage assignment, whereas Liggett's system used a multi-stage assignment which includes the floor, zone and block assignment levels. There are three reasons why the EDGE system does not use multi-stage assignment in this example. First, the relationships between activities on the same floor are not always necessarily better than distributing them between different floors. This is because the vertical movement through a stairway, or an elevator, is considered to be one unit and therefore is cheaper than having the activities on the other side of the same floor. This aspect makes the priority of the floor assignment level less useful. Second, the actual travel cost between two modules over two different and adjacent zones is cheaper than the cost between the same two on opposite sides of the same zone, Figure 6 (a). This makes the zone assignment level less useful. Third, the genetic operations do not necessarily need problems to be formulated by any complicated and sophisticated strategies, because of the random nature of the operations and the transformations on the genotype level.

5.1.2 Evaluation criteria

The objective is to produce an assignment of activities to locations that minimizes an overall cost measure, subject to meeting specified space needs requirements. The evaluation criteria for the implementation are the same as those used in Liggett's system. The cost measure considers both interactive costs which are calculated as the product of some measure of interaction between pairs of activities and the distance or travel time between their assigned locations. The travel matrix, Table 5, specifies distance between locations. The travel cost between a pair of activities is calculated by multiplying the distance between their locations by the interaction between them.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		4	9	21	27	9	10	23	28	10	11	24	29	11	12	25	30	6	24
2			5	17	23	5	6	19	24	6	7	20	25	7	8	21	26	2	20
3				13	18	6	5	18	23	7	6	19	24	8	7	20	25	2	15
4					5	23	18	5	6	24	19	6	7	25	20	7	8	15	2
5						24	19	6	5	25	20	7	6	26	21	8	7	20	2
6							5	18	23	5	6	19	24	6	7	20	25	3	21
7								13	18	6	5	18	23	7	6	19	24	3	16
8									9	23	18	5	6	24	19	6	7	16	3
9										24	19	6	5	25	20	7	6	21	3
10											5	18	23	5	6	19	24	4	22
11												13	18	6	5	18	23	4	17
12													5	23	18	5	6	22	4
13														24	19	6	5	17	4
14															5	18	23	5	23
15																13	18	5	18
16																	5	18	5
17																		23	5
18																			18

Table 5. Travel matrix (Liggett, 1985).

The interaction matrix is based on subjective judgements of the client, Table 6. The client rates adjacency needs on an ordinal scale (3 – most important, to 0 – not important).

	0210	0211	0220	0230	0240	6815	0300	0400	0500	0600	0700	6300	6881	0800	0900	1000	ACCESS	
0210	3		3	2	2	2			3								2	3
0211		3		2	2	2			3								2	
0220			3		2	2			3								2	
0230				3		2			3								2	
0240					3				3								2	
6815						2			3								2	3
0300					3										3		2	
0400															3		2	
0500																	2	
0600																	2	3
0700																3	2	
6300																3	2	
6881																3	2	
0800																	2	3
0900																	2	
1000																		3

Table 6. Activity interactions (Liggett, 1985).

The program associates the interaction value specified for a pair of activities in the matrix with each pair of individual activity modules in the plan. The problem can be stated as:

$$\min \left(\sum a_{ij} + \sum\sum q_{i_1 i_2} c_{j_1 j_2} \right)$$

where,

a_{ij}	fixed cost of assigning element i to location j
$c_{j_1j_2}$	distance measure from location j_1 to location j_2
i	spatial unit to be located
j	possible location
$q_{i_1i_2}$	interaction between spatial unit i_1 and spatial unit i_2

In order to eliminate the effect of activity size on the criterion function, each interaction value is standardized by dividing it by the number of modules associated with the two activities. Then each resulting value is converted to its slot angle by using the Pareto optimization technique.

5.2 Liggett's Approach

Since the early attempts to deal with the floor plan layout problem, for example CRAFT (Buffa, et al., 1964), numerous computer programs have been developed to automate the architectural spatial allocation problem. Liggett's system (1985) is one of the most well known approaches. Two typical strategies in the system include constructive placement and pair-wise improvement. The constructive placement algorithm was originally developed by Graves and Whinston (1970) and is used in her system in order to produce an initial solution. This strategy is a kind of n-stage decision process and locates activities one by one commencing with an empty set. The next element to be assigned is chosen on the basis of the expected value of the objective function. Then as an improvement procedure, "pair-wise" change is used. Starting from the initial solution, the procedure consists of systematically evaluating possible exchanges between pairs of activities and making an exchange if it improves the value of the criterion towards the neighbourhood for the "best" solution. This iterative improvement is a kind of hill-climbing strategy. A solution of the improvement technique is very dependent on the initial solution. In the process, the constructive procedure sets the general tone of the solution, while the improvement procedure refines the details. Figure 6(b) shows the solution produced by these procedures.

The pair-wise change is applied only on pairs of neighbouring units. The exchange between a pair over 3 or 4 units is impossible even though the exchange may produce better solutions. For example, if the units consist of "1 2 2 1" and the objective is to locate the same kind of units together, the solution may never get "1 1 2 2 " or "2 2 1 1" because any change can make the performance worse immediately. This aspect shows a limitation of the pair-wise method in overcoming local optima.

Another strategy used by Liggett is the multi-stage assignment which includes the floor, zone and block assignment stages. The activities are assigned to each floor, to each zone of a single floor, then to specific location modules. Zones are specified by defining their physical perimeters in terms of area modules. Each floor is divided into four zones with the addition of an extra wing on the bottom floor, Figure 6(a). The zoning was selected to match vertical

circulation patterns. Also the largest spaces are placed on the plan first in order to result in fewer activities split between zones.

5.3 Design Formulation using the Design Rule Schema

A solution satisfying the given requirements is obtained by manipulating design elements which include activity modules and their locations in the bounded floors. Since the perimeter of the building is fixed and the size of each activity is given from the initial requirements, the genotype needs only the order of activities. While Liggett's approach uses one unit to one cell assignment, our approach uses a topological rule-based assignment. The manipulable design elements are formulated in a design rule schema as "assigning modules of an activity by a topological transformation action in the plan." The structure of the design rule schema is:

S_r = {LHS, τ}
= {marker, new_activity}

Then the semantics of this design rule schema is "if the module has a marker, assign a set of modules of a chosen activity on the location specified by the assigning rule." Here the chosen activity indicates an activity which has not been assigned on the floors yet. Since there are 15 manipulable activities, excluding the fixed activities, and 3 extra modules, 18 design rules can be instantiated from the design rule schema to fill the whole plan. A design rule which is instantiated from the design rule schema is of the form:

r_1 = {•, (□, d, →)}

where,

•	marker
d	a department
□	a unit size
→	locate a new unit on the right side of the marker and move the marker to the new unit.

5.4 Translation of the Design Elements to the Genetic Codes

A set of design elements, or the order of activities in this example, is in the form of design rules and needs to be interpreted into the language of the genetic search system. The principle for the schema translation was earlier defined as "only components of a design rule schema which are distinct and need to be manipulated by the genetic search process are active and translated to those of the design gene schema. The other components are inactive and not translated but kept in the interpretation knowledge (K_i)." Among the components of the

design rule schema, introducing a new activity (L_n) is manipulated by the genetic search process and therefore is included in the design gene schema.

S_r = {marker, new_activity}

S_g = {new_activity}

The total number of these design rules, or design genes in this example, is the same as the total number of the space plan zones excluding those for fixed activities, ie the same number as the number of activities to be located. Each gene must include a distinct activity to prevent an activity being used twice. However, the mutation or crossover operations of the genetic search process mix up the activities and easily produce duplicate ones. To prevent an activity from being used more than once, a reordering function is necessary to make all activities of a genotype unique.

This example adopts binary codes for the computation and symbolic codes for the explanation of the genetic search operation. An alternative design representation can utilize symbolic codes (Jo, 1993). The activities need five bits per gene to represent their elements, Table 7. The codes between 10011 and 11111 are unoccupied by any activities.

Activities (symbolic codes)	**Binary (Digital) codes**	**Number of modules**
Dept. 0210	00000 (0)	2
Dept. 0211	00001 (1)	2
Dept. 0220	00010 (2)	8
Dept. 0230	00011 (3)	8
Dept. 0240	00100 (4)	15
Dept. 6815	00101 (5)	13
Dept. 0300	00110 (6)	15
Dept. 0400	00111 (7)	7
Dept. 0500	01000 (8)	6
Dept. 0600	01001 (9)	12
Dept. 0700	01010 (10)	53
Dept. 6300	01011 (11)	10
Dept. 6881	01100 (12)	16
Dept. 0800	01101 (13)	18
Dept. 0900	01110 (14)	31
Dept. 1000	01111 (15)	61
Extra module1	10000 (16)	1
Extra module2	10001 (17)	1
Extra module3	10010 (18)	1
Fixed activities		
Dept. Exec		32
Public Access - West		3
Public Access - East		3

Table 7. Interpretation of design elements as their genetic codes and their modules.

A genotype string, which represents a sequence of design rules in terms of design genes, combines the separate design information to constitute a complete individual structure which produces a single solution. The length of a genotype string is calculated by

multiplying the number of design genes in the genotype, number of activities in a design gene and number of bits in a gene. In this example, the length of a genotype is 95 bits, where a genotype is composed of 19 design genes, each gene contains one activity and each activity has 5 bits.

5.5 Results

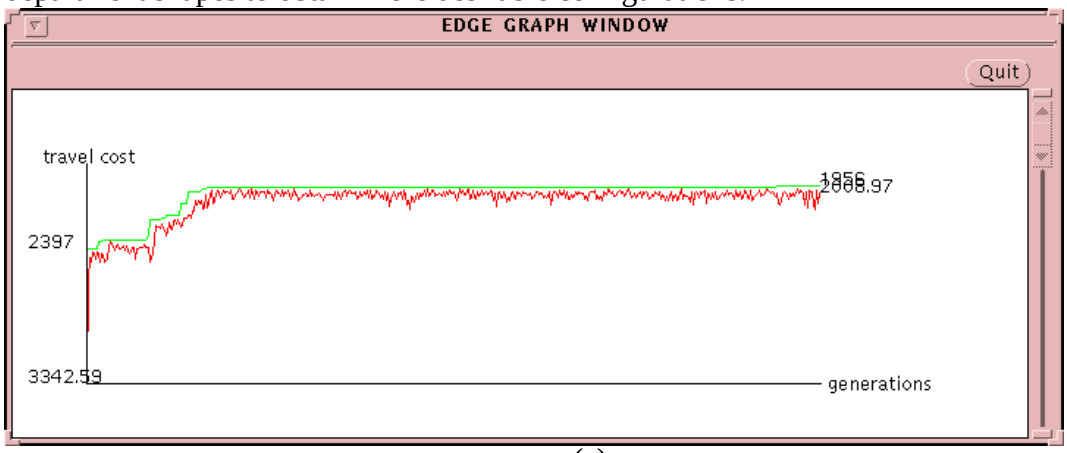
The evolutionary design process was executed with two different sets of initial populations. One commences with a randomly generated population and the other commences with Liggett's final solution, Figure 6 (b), in order to see if the EDGE system can improve this solution. The population is made up of 100 individual genotypes and the number of generations, which specifies the number of iterations of the genetic search process, was set at 500 for both.

The EDGE system uses the same evaluation method as does Liggett. However, the behaviour value of Liggett's solution is not the same as that in her paper (Liggett, 1985) because the "split penalty" interaction value is not specified in the literature. Therefore, the solutions of the EDGE system are compared with the fitness value of her final solution which is evaluated by the EDGE system. The graphs in Figure 8 plot the changes of the maximum and average values of the actual behaviour, i.e., minimising travel cost of activities. The vertical axis of each behaviour graph represents the inverted value of each behaviour where the upward direction corresponds to the improvement of behaviour. The horizontal axis represents the change of generations. The upper line on each graph traces the best behaviour of each population while the lower line represents the average behaviour of each population. The graph shows that the value of the best and average behaviours improve as the generations increase.

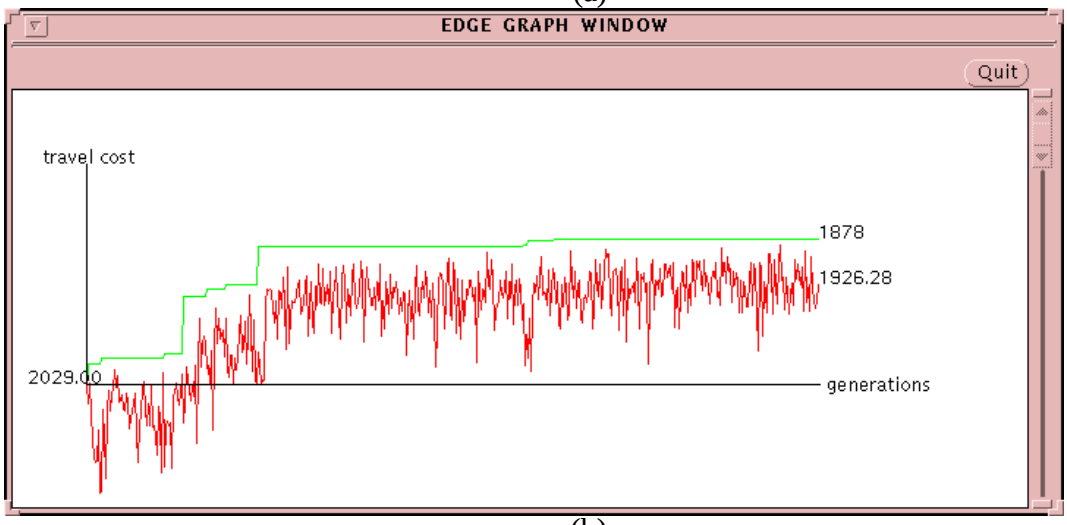
In Figure 8(a), the graph traces the best and average behaviours of the solutions over 500 generations where the initial population was seeded randomly. The behaviour of the best solution produced by this method is 1956 compared to Liggett's solution of 2029, a 3.6% improvement. The best behaviour produced by the process commencing with Liggett's final solution gives 1878 after 500 generations, Figure 8(b), a 7.4% improvement. The results from both initial populations show increased performances compared to Liggett's and verify the capability of the EDGE system. The rapid improvement of the behaviours exhibited in Figure 8(a) is significant at the beginning stage since the behaviours of the initial solutions are poor. In Figure 8(b), the average behaviours do not exceed the initial ones at the beginning stage because the initial solutions are already very good.

The final results from EDGE have 13 solution types, Table 8. Seven types are occupied by 92 solutions and have the best behaviour of 1878. Each gene in the genotypes represents the digital identification code of each activity (refer Table 7). These final results present promising solutions and provide several alternatives. The human designer can choose a

solution among them during any intermediate stage or at the final stage and may alter the department shapes to obtain more desirable configurations.



(a)



(b)

Figure 8 The changes of the maximum and average values of the travel cost behaviour for (a) when the initial population is seeded randomly, and (b) when Liggett's final solution is the initial population.

Types	number of solutions	Genotypes	Behaviour
1	20	9 12 14 17 13 11 15 5 7 6 10 16 0 1 2 8 3 18 4	1878
2	15	9 12 14 18 13 11 15 5 7 6 10 17 0 1 2 8 3 16 4	1878
3	37	9 12 14 17 13 11 15 5 7 6 10 16 2 0 1 8 3 4 18	1878
4	4	9 12 14 17 13 11 15 5 7 6 10 16 0 1 2 8 3 4 18	1878
5	4	9 12 14 17 13 11 15 5 7 6 10 16 0 1 8 2 3 4 18	1878
6	11	9 12 14 17 13 11 15 5 7 6 16 10 0 1 2 8 3 4 18	1878
7	1	9 12 14 17 13 11 15 5 7 6 10 18 2 0 1 8 3 4 16	1878
8	1	9 12 14 17 13 11 15 5 7 6 10 18 8 0 1 3 2 4 16	1915
9	3	9 12 13 14 16 11 15 5 7 6 10 18 0 1 2 8 3 4 17	1933
10	1	9 12 14 17 13 11 15 5 6 16 10 18 2 0 3 7 8 1 4	2801
11	1	9 12 14 17 13 11 15 2 3 4 10 7 0 1 16 5 8 6 18	3003
12	1	9 4 14 17 5 11 12 13 7 6 15 10 2 0 1 8 3 16 18	3006
13	1	9 12 14 18 13 11 15 7 8 6 10 16 0 1 3 17 2 4 5	3328

Total : 100

Table 8 Genotypes and their behaviours evolved after 500 generations by the EDGE system.

When the 7 best genotypes in the table are examined it can be seen that there is a common schema amongst them. If the empty modules including 16, 17 and 18 are ignored, the genotypes share the genotype schema, 9 12 14 13 11 15 5 7 6 10 * * * * 3 4, where * can be 0, 1, 2 or 8, or the departments 0210, 0211, 0220 and 0550. These genes are relatively small in size and therefore affect the behaviour of a solution less. The genes appear on the first floor because this floor is the only one with the external design element which is 'access.'

Figure 9 presents the details of two solutions obtained through the evolutionary design process. Figure 9(a) is a solution evolved using a randomly seeded initial generation and Figure 9(b) is a solution using Liggett's solution as the initial generation. Three highlighted modules in each solution represent the empty modules and may be regarded as a rest room or a hall. The final solutions from these two beginnings are very different from each other. This implies that the initial solutions, which were scattered over the design space at the beginning, do not keep the same location in the design space during the process but move freely toward near optimal positions as generations proceed.

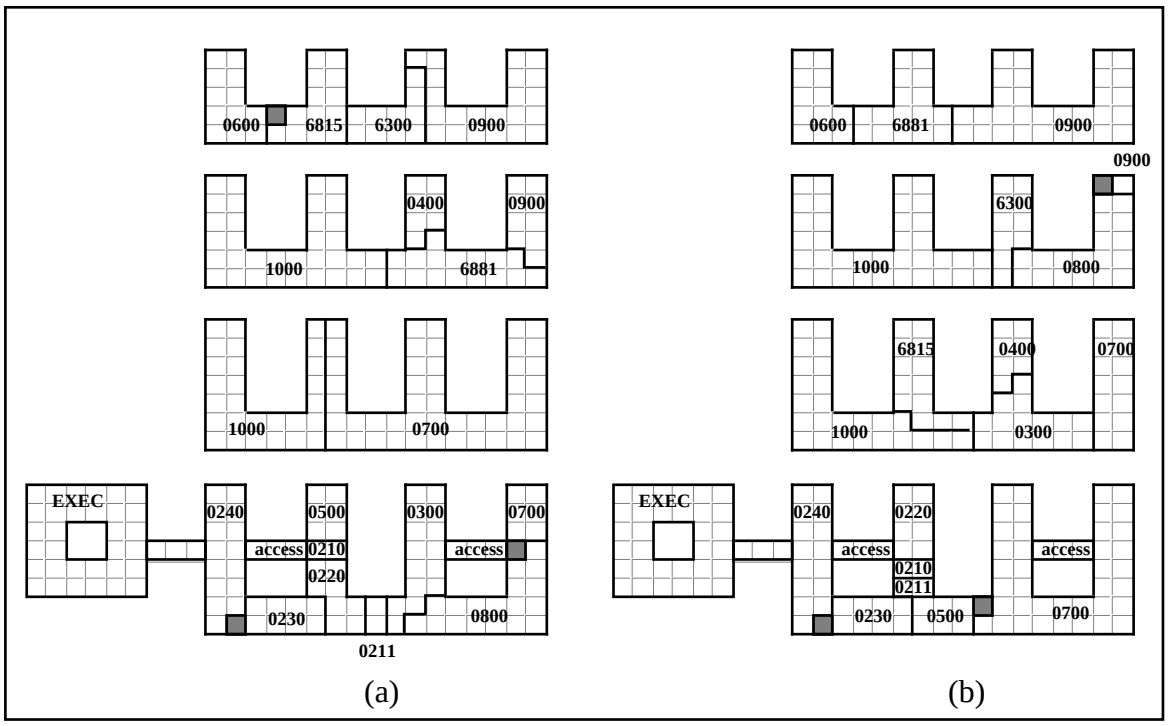


Figure 9. Transformed final solutions after 500 generations where (a) is one of the solutions evolved from the randomly seeded initial generation, and (b) is a solution evolved from using Liggett's solution as the initial population.

Liggett uses a probability scheme to decide the next unit to assign. Both the constructive and improvement strategies need evaluations at every stage to select or switch the promising activity-location pair yielding an improved value. Therefore the evaluation costs are very

high in producing a solution[2]. The improvement is heavily dependent on the initial solution and therefore the search is implemented around a local optimum which is close to the initial solution. This indicates that the boundary of the search is narrow. On the other hand, the EDGE system does not require any special heuristics to generate and evolve solutions and evaluates a whole solution instead of its component details. This allows the process to be much more economical computationally. While the initial population can affect the final solutions, it is not as significant as in Liggett's process. The EDGE system searches through a large area of search space towards the global optimum. Table 9 shows the comparison between processes in Liggett's Space Layout System and the EDGE system.

	Liggett's Process	**EDGE Process**
generation	constructive method with evaluation	random generation
improvement	pair-wise method	evolutionary method
search	multi stage assignment	single stage assignment
	step-by-step on a tree-like search within a feasible space	jumping and hopping over the entire search space

Table 9 Comparison between processes in Liggett's Space Layout System and the EDGE system.

6. DISCUSSION

An evolutionary design method was introduced to solve a certain class of design problems. Space layout planning was considered as a typical and complex design problem. General approaches and their limitations when applied to the space layout problem were discussed. Some advantages of the genetic evolutionary process, which may be useful in design, were investigated. These include simple but powerful operations, two levels of representation, and search in a population in the design space. An evolutionary design process model was constructed to apply these concepts.

The evolutionary design approach and its efficient representation mechanism show ways of overcoming some of the limitations of conventional design approaches, especially those specified on Section 2.2. The difficulty of the formulation for complex design problems is eased by the introduction of design schemas. The use of Pareto optimization technique helps the evaluation task under multicriteria performances, removing biases on each criterion. Search with a fixed number in the population resolves the combinatorial explosion problem. Search within a population, its random generation, the use of probabilistic selection, and the

[2] For every possible exchange, each selection needs n(n-1) evaluations with n units.

simple but powerful genetic operations allow search to escape from most local optima traps and to find solutions likely to be closer to the global optimum.

Two issues arising from constructing this design model, which combines both heterogeneous areas of natural genetics and design, are how to represent the design knowledge and the usefulness of the new design model. For the representation of design knowledge in the model two kinds of design schemas were introduced: the design rule schema and the design gene schema. The design rule schema is used in the design formulation, whereas the design gene schema is used in the transformation of the design knowledge to the knowledge manipulable by the genetic search engine. As an example the model was implemented for an office layout problem, where the problem configurations and the evaluation criteria were drawn from the literature. The results show the usefulness of this design process model.

On the basis of the advantages of genetic evolutionary design process and the results of the implementation, it is concluded that the coupling of an evolutionary search technique with a design process can produce very good results, especially for large-scale problems which are at present computationally difficult.

Acknowledgments

This work has been supported in part by a grant from the Australian Research Council to John S. Gero. Computational support was provided by the Key Centre of Design Computing, University of Sydney. The authors wish to thank the anonymous referees who helped make this a better paper.

7. REFERENCES

Akin, O., Dave, B. and Pithavadian, S. (1992). Heuristic generation of layouts (HeGel): based on a paradigm for problem structuring, *Environment and Planning B* **19**: 33-59.

Balachandran, M. and Gero, J.S. (1987). Dimensioning of architectural floor plans under conflicting objectives, *Environment and Planning B* **14**: 29-37.

Buffa, E.S., Armour, G.S. and Vollman, T.E. (1964). Allocating facilities with CRAFT, *Harvard Business Review* **42**(2): 136-140.

Chomsky, N. (1957). *Syntactic Structures*, Mouton, The Hague.

Dawkins, R. (1987) *The Blind Watchmaker*, Norton, New York.

Eastman, C.M. (1975). The scope of computer-aided building design, *in* C.M. Eastman, (ed.), *Spatial Synthesis in Computer-Aided Building Design*, Applied Science, London, pp. 1–18.

Gero, J.S. (1977). Note on "Synthesis and optimization of small rectangular floor plans" of Mitchell, Steadman, and Liggett, *Environment and Planning B* **4**:81-88.

Gero, J.S (1978). Computer aided dimensioning of architectural plans, *CAD78*, IPC Press, Guilford, pp.482-493.

Gero, J.S., Louis, S.J. and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement, *AIEDAM* **8**(2):83-94.

Gilleard, J. (1978). LAYOUT--hierarchical computer model for the production of architectural floor plans, *Environment and Planning B* **5**(2): 233-241.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts.

Grason, J. (1971). An approach to computerized space planning using graph theory, *Proceedings of the Design Automation Workshop*, Association for Computing Machinery, New York.

Graves, G.W. and Whinston, A. (1970). An algorithm for the quadratic assignment problem, *Management Science*, **17**(3): 453-471.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.
Holland, J.H. (1992). Genetic algorithms, *Scientific American*, pp.66-72.
Jo, J.H. (1993). *A Computational Design Process Model using a Genetic Evolution Approach*, Ph.D. Thesis, Department of Architectural and Design Science, University of Sydney.
Jo, J.H. and Gero, J.S. (1994). A genetic search approach to space layout planning, *Architectural Science Review*, **38**:37-46.
Koning, H. and Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses, *Environment and Planning B* **8**: 295-323.
Koza, J. (1992) *Genetic Programming*, MIT Press, Cambridge.
Korf, R.E. (1977). A shape independent theory of space allocation, *Environment and Planning B* **4**: 37-50.
Lewis, H.R. and Papadimitriou, D.H. (1978). The efficiency of algorithms, *Scientific American*, **240**(5): 96-109.
Liggett, R.S. (1980). The quadratic assignment problem: an analysis of applications and solution strategies, *Environment and Planning B* **7**: 141-162.
Liggett, R.S. (1985). Optimal spatial arrangement as a quadratic assignment problem, *in* J.S. Gero, (ed.), *Design Optimization*, Academic Press, New York, pp. 1-40.
Liggett, R.S. and Mitchell, W.J. (1981). Optimal space planning in practice, *Computer-Aided Design*, **13**(5): 277-288.
Maher, M.L. and Kundu, S. (1994). Adaptive design using genetic algorithms, *in* J.S. Gero and E. Tyugu (eds), *Formal Design Methods for CAD*, North-Holland, Amsterdam, pp. 246–262.
Miller, W.R. (1971). Computer-aided space planning, an introduction, *DMG Newsletter* **5**: 6-18.
Mitchell, W.J., Steadman, J.P. and Liggett, R.S. (1976). Synthesis and optimization of small rectangular floor plans, *Environment and Planning B* **3**: 37-70.
Radford, A.D. and Gero, J.S. (1988). *Design by Optimization in Architecture, Building and Construction*, Van Nostrand Reinhold, New York.
Rumelhart, D.E. (1980). Schemata: the building blocks of cognition, *in* R.J. Spiro, B.C. Bruce and W.F. Brewer (eds), *Theoretical Issues in Reading Comprehension*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 33-58.
Stiny, G. and Mitchell, W.J. (1978). The Palladian grammar, *Environment and Planning B* **5**: 5-18.
Yoon, K.B. and Coyne, R.D. (1992). Reasoning about spatial constraints, *Environment and Planning B* **19**: 243-266.