Architectural Science Review

ISSN: 0003-8628 (Print) 1758-9622 (Online) Journal homepage: https://www.tandfonline.com/loi/tasr20

A Genetic Search Approach to Space Layout Planning

Jun H. Jo & John S. Gero

To cite this article: Jun H. Jo & John S. Gero (1995) A Genetic Search Approach to Space Layout Planning, Architectural Science Review, 38:1, 37-46, DOI: 10.1080/00038628.1995.9696774

To link to this article: https://doi.org/10.1080/00038628.1995.9696774

Published online: 10 Oct 2011.

UDC: 72.011:518 Keywords: Computer-aided space planning;
Design computing; Layout planning; Space Layout.

Architectural Science Review
Volume 38, pp 37—46

A Genetic Search Approach to Space Layout Planning

Jun H. Jo* and John S. Gero*

Space layout planning is one of the most interesting, difficult and controversial area of computer-aided architectural design. Two major issues from the area are the topological assignment of space elements and the dimensioning of those space elements to meet certain criteria and constraints. Critical difficulties include the complexity of design information involved and the combinatorial explosion of alternatives. This paper introduces a design method which uses the evolutionary concept as a mechanism for controlling search over the huge design space. Schemas are introduced to formulate and interpret design information. The Pareto optimization concept is used in the solution of space layout planning as a multi-criteria optimization problem. An example is presented.

1. Introduction

Much research has been directed at the space layout planning problem over a long period (Buffa et al, 1964; Landsdown, 1972; Eastman, 1975; Liggett, 1980a). This research has shown the difficulties associated with the design information and the search through the huge design space generated during the design process. This paper presents an approach to these problems by utilising a search process whose concept is derived from natural genetics.

Genetic algorithms (GAs) have been introduced in the optimization problem solving area by Holland (1975) and Goldberg (1989) and have shown their usefulness through numerous applications. The advantages observed from these applications motivates the use of GAs in this design area. However, the introduction of the new concept from outside the design area to solve the space layout planning problem requires an appropriate knowledge representation.

The aim of this paper is to develop a new design method for space layout planning. Space layout planning is difficult to formulate and to solve algorithmically due to the multi-criteria nature of the problem and the combinatorial explosion of alternatives.

* Key Centre of Design Computing, Department of Architectural and Design Science, University of Sydney, NSW, 2006, Australia

2. Space Layout Planning

Space layout planning involves the range of such design tasks as shaping, arranging, and dimensioning space elements to satisfy a set of constraints whilst achieving one or more objectives. The constraints are specified either precisely or in a range. Some examples include floor layout planning (Auger, 1972; Landsdown, 1972; Eastman, 1975; Liggett, 1980a), site planning (Bijl and Shawcross, 1975; Flemming, 1985), allocation of furniture (Pfefferkorn, 1975), plant arrangements in a factory site (Buffa et al, 1964), electronic circuit design (Nilsson, 1969), and so on.

2.1 General Approaches

Two major problems are implicated in space layout planning and these are associated with the topology and geometry of the solution. The two distinct approaches address the two major problems: generative systems and optimization.

The generative approach often depends on shape grammars whose idea is based on linguistic grammar systems (Chomsky, 1957). Shape grammars are systems 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). Steadman (1973) proposes the exhaustive generation of topologically distinct rectangular dissections as a design tool (March and Steadman, 1971). Flemming (1978) applied the dissection

approach to space allocation. One major drawback of this approach is the combinatorial explosion problem. As the number of possible elements increases, the possible number of solutions increases exponentially. Table 1 shows an example of the exhaustive generation produced manually by Mitchell et al (1976). They applied the dissection rules for up to eight rectangles and showed the NP-complete nature of the approach. In general, heuristic solution techniques have been developed which incorporate a variety of schemes for limiting the set of solutions explored (Liggett, 1980b).

Table 1

Number of Topologically Distinct Rectangular Dissections in Relation to the Number of Component Rectangles Generated (Mitchell et al, 1976).

No. of component rectangles	2	3	4	5	6	7	8
No. of rectangular dissections	1	2	7	23	116	683	4866

Optimization techniques, including linear, nonlinear and dynamic programming have been applied to produce the dimensioning of floor plans. Mitchell et al (1976) used the linear programming technique to solve a floor dimensioning problem for a mobile home, where dimensions in one direction are fixed. They also presented a worked example of a nonlinear programming technique for dimensioning a rectangular apartment plan so as to optimize the construction cost. Gero (1977) has demonstrated dimensioning of floor plans by the use of dynamic programming in producing global optimal and near optimal solutions, so that various analyses of different solutions can be carried out. Since realistic design problems usually have multiple objectives to be satisfied, Balachandran and Gero (1987) used Pareto optimization techniques for dimensioning architectural floor plans under conflicting objects.

2.2 Issues in Space Layout Planning

Applications of generative algorithms to solve the space layout problem reveal several weaknesses. First, the process easily meets the combinatorial explosion problem. To avoid this, some sophisticated heuristics are required, however, they usually need computationally expensive tasks to be formulated. In general, faulty information can guide the process in a wrong direction. Second, these approaches have been mainly applied to the topological problem without dealing with the geometrical problem. However both problems in space layout planning have to be solved in parallel. Third, even for a simple design problem, rich design rules are required.

The use of optimization techniques to solve the layout problem requires a complete formulation of the optimization problem such as objective function(s), constraint equations and bounds on the design variables. This task can have a potential source of error and, moreover, it can be beyond the expertise of many designers as they do not usually think in terms of mathematical models. Furthermore, in many practical design situations, the designer may want to introduce changes to the design and reformulate the problem during the design process but conventional optimization systems do not provide sufficient flexibility to allow the designer to modify the design easily.

3. Genetic Evolutionary Process

Holland (1975) and Goldberg (1989) constructed search algorithms by applying the genetic evolution mechanism into an artificial world, making it possible to search a far greater range of potential solutions to a problem than do the conventional search methods.

A genetic algorithm, as a computational model, has mainly been used to solve optimization problems and its use has shown many advantages. The algorithm is a simple,and blind process which does not need any specific heuristic guidance. It just repeats simple routines. The process is very economical but the solutions are varied without any predetermined prejudices. The search space is not limited by restrictive assumptions concerning continuity, existence of derivatives, unimodality, and other matters (Goldberg, 1989). These characteristics make the genetic search process applicable in a broad range of domains. Genetic search also offers robust procedures that can exploit massively parallel architectures. Particularly in multimodal (many-peaked) search spaces, point-to-point search methods have the danger of locating on the local optima. By contrast, the genetic search method climbs many peaks in parallel, thus there is safety in numbers in finding the global optimal solution and the probability of finding a false peak is reduced over other methods. Simplicity of operation and power of effect are two of the main attractions of the genetic approach (Goldberg, 1989).

As in the case of a knowledge-based design model, the representation and the process of genetic evolution in a model can be considered separately. This consideration helps not only our understanding 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 and provides a basis of the knowledge representation for the evolutionary design process.

In genetics, a set of *genes* composes a chromosome which is analogous to a string of symbols in an artificial genetic system. They take values called alleles. A gene 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* consists of a finite set of genes and combines the separate information to constitute an entire individual structure. The information embedded in a genotype can be a set of instructions or procedures. The meaning of a genotype is identified by decoding the genotypic information into the phenotypic information explicitly.

$$G = \{\Sigma g_i\}$$

where,

g_i gene i,

G genotype

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:

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

where

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

In nature, a *phenotype* is the outward, visible expression of the hereditary constitution of an organism (Miller and Keane, 1987). It is a decoded genotype, therefore, tangible and confronts the environment. The mapping from the genotype to the phenotype allows realization of the structure of an individual solution and therefore the behaviours can be extracted from it for evaluation of the structure's fitness. The fitness is defined as the performance of an individual structure against its environment.

$$P = m(G)$$

where

P phenotype,

m mapping or interpretation operator.

The separate representation of genotypes and phenotypes provides both efficiency and opportunities in problem formulation and search.

3.2 Genetic Search Process

The basic idea of the genetic search process is founded on natural adaptive systems. The process comprises a set of individuals or a population and a set of biologically-inspired operators over the population. Search is guided by random choice and probability instead of the well structured and deterministic information of other formulations. It also reduces any incorrect biases leading the process to inappropriate directions which are caused by the wrong problem formulation with faulty or insufficient information, and together with its population, makes the search occur over a wide range. 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).

The process is continuous and cyclic. In each cycle the current population is transformed into a new population in parallel and therefore a new generation of possible solutions for a given problem is produced. The successive populations are generated through an iterative process. The search process continues until the termination condition is met. The mechanism is computationally simple yet powerful in its search for improvement. The genetic operations are composed of an initialization which is the provisional stage for the main operations at the beginning stage, and three iterative operations: evaluation, selection and recombination.

The genetic search process starts with an *initialization*. A population of genotype strings are 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 in this process. Thereafter the evaluation-selection-recombination loop starts running iteratively until certain parameters are satisfied.

In the *evaluation* process, 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 the phenotypes in the beginning of this process so that their fitness values can be derived.

The *selection* process is carried out based on each individual's fitness which is measured in the evaluation process. The goals and evaluation devices form a simulated environment in which individuals either 'live' or 'die' (Woodbury, 1993). The probabilistic method allows the more highly fit individuals to have more chances to be selected than the lower 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 to be 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 with low fitnesses which are discarded at each generation so that the total population remains the same size. The selected solutions, or a new population, are then sent to the mating pool and 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.

Recombination operators manipulate the selected individuals in order to produce a new population carrying different features from those of the former population. The children produced by the genetic operation then originate the next population. The recombination operators comprise crossover and mutation.

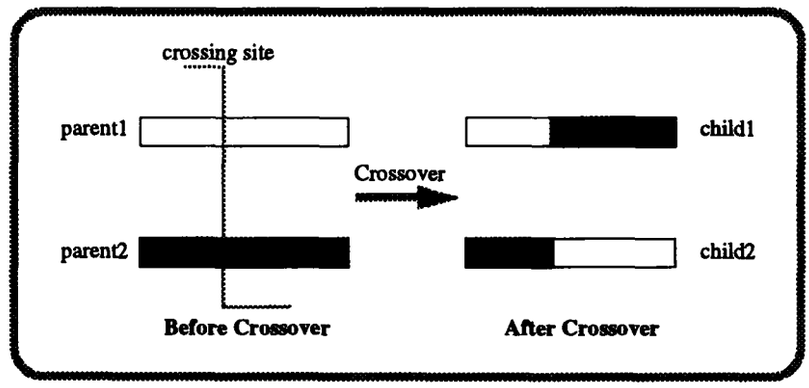


***Figure 1** A schematic of simple crossover shows the alignment of two parent strings and their two children generated by the partial exchange of information.*

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, Figure 1. The crossing site can be chosen at random and specifies the location where the information is swapped in the parent genotypes.

Mutation is an alteration of the value in a random position in the genotype string. The example below presents the mutation operation being applied to the second bit (underlined) of a genotype string.

1010 —> 1110

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.

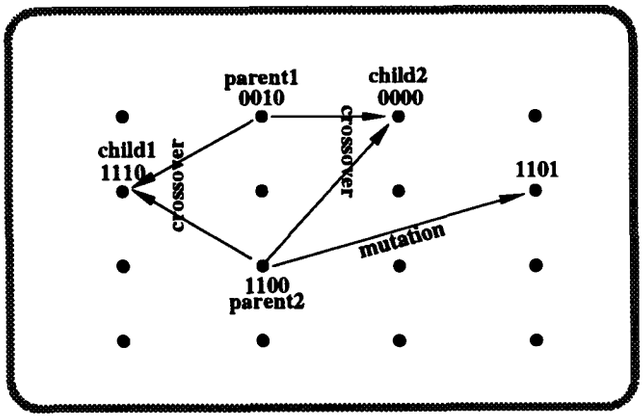


***Figure 2** Crossover and mutation operations in the search space.*

4. Edge: Evolutionary Design Process

Evolutionary Design based on Genetic Evolution (EDGE) (Jo, 1993) is a system which integrates the genetic evolution concept in a design process. Because this system includes two different approaches, communication between them is necessary and therefore a method of interpretation between design knowledge and genetic representation is necessary. The Pareto optimality concept (Pareto, 1896) is employed to support the evaluation task under multiple criteria.

4.1 Knowledge Formulation and Interpretation

The concept of a schema provides an efficient framework in order to organise and represent design information and knowledge. Three kinds of design schemas are considered for the EDGE system: design prototype schema, design rule schema, and design gene schema (Jo, 1993). The design prototype schema (Gero, 1990) is used as a conceptual framework for general design information and knowledge. Thus, initial design elements, which will be transformed during the design process, and their related design knowledge are retrieved from a design prototype which is an instance of the schema. The design rule schema is used to embed and formulate the retrieved design elements and knowledge. The formulated design information is then translated into the design gene schema so that the instances of the design gene schema, or design genes, can be manipulated by the genetic search process.

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 discrete design elements, which are retrieved from a design prototype, into a homogeneous type of structured design rule and therefore provides a basis for the design transformations. A set of design rules is instantiated from the design rule schema and acts like a set of instructions for constructing a building. A design rule schema includes a target situation (LHS), and a transformation operator (τ). The result (RHS) of the rule application appears on the phenotypic structure and is not included in the schema. The general form of a design rule schema is described as below:

$S_r = \{LHS, \tau\}$

where

LHS left hand side

S_r design rule schema

τ transformation operator

4.1.2 Design Gene Schema for Knowledge Interpretation

Once the design elements are retrieved from a design prototype and formulated based on a design rule schema, in order to be utilised by the genetic engine, the design information needs to be translated into the genetic language. While keeping the original semantics of the design rule schema, a design gene schema is constructed by restructuring the design rule schema based on the following principle: if a component of a design rule schema needs to be transformed by the genetic operation then the component is active and translated into the design gene schema, otherwise it is inactive and is not included in the design gene schema. Design genes are generated in the form of binary numbers or symbols, by instantiating the design gene schema. Here the design gene schema is the restructured design rule schema and design genes are genotypic representation of design rules. This alternative representation of design information makes the design

transformation easy and rich by using simple genetic operations. 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 described as below. This *interpretation knowledge* (K_i) is specified by the user and provides information required for the interpretation between design representation and genetic representation. The information includes the design rule schema, the design gene schema, the components involved and their possible values, and the initial/terminal rules.

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

where

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

In Figure 3, for example, there are four design rules instantiated from a design rule schema. Several components on each design rule always have the equivalent values, such as same square shapes and a mark, and do not need to be transformed because they do not identify their rules. Therefore the active component in the example is only the transformation action and this becomes the component of the design gene schema. The so called 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..

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:

$$S_r = \{LHS, \tau\}$$
$$= \{E_x, (E_n, a)\}$$

where

E_x existing design element,
E_n new design element
a transformation action and possible values are: {➡ ⬇ ⬅ ⬆ }

The instantiated design rules are:

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

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

$S_g = \{a\}$

4.1.3 Representation of Genotypic Information

Design genes are instances of a design gene schema and carry a set of design elements interpreted into the language of the genetic search system. 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 00010110 which is composed of four design genes. Symbolic, binary and semantic representations of the design genes are described below and their phenotypes are shown in the figure.

$S_g = \{a\}$

Possible design genes :

Binary rep.	Symbolic rep	Semantics
00	➡	put a cell on the right side
01	⬇	put a cell on the bottom
10	⬅	put a cell on the left side
11	⬆	put a cell on the top

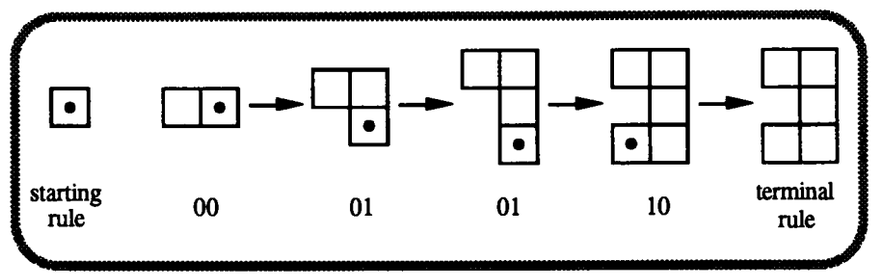


Figure 4 An example of a genotype, 00010110, and its phenotype. The interpretation knowledge provides necessary information including the starting and terminal rules.

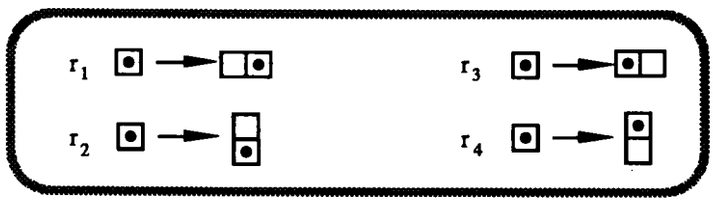


Figure 3: Simple design rules with a marker. Only the transformation action among the components of each design rule identifies its rule and becomes a component of the design gene schema.

When genotypes are decoded in the design space, the interpretation knowledge guides the process by providing information such as a design rule schema, a design genotype schema, a starting rule, a terminal rule, inactive elements, active elements with their binary codes, information of transformation actions, etc. The terminal rule removes the marker from the currently active element and terminates the whole process.

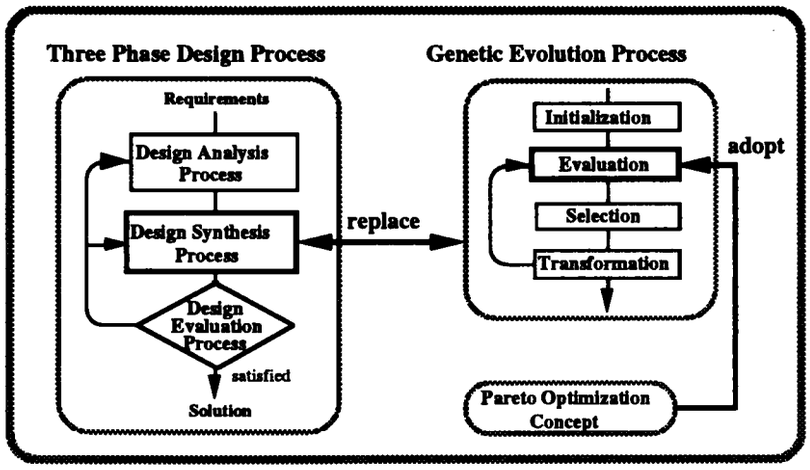


Figure 5: The integration of two different models : the design analysis-synthesis-evaluation process model, and the genetic evolution model. The Pareto optimality concept is used in the evaluation task of the genetic evaluation process

4.2 Evolutionary Design Process Model

The evolutionary design process model is an integration of the design analysis-synthesis-evaluation process model and the genetic evolution model as shown in Figure 5. The analysis-synthesis-evaluation paradigm is one of the most widely accepted design process paradigms for viewing design, and is employed as a meta-level process to control the whole process of the design model. The main reason for adopting this paradigm in the model, is that the tasks of the design process are well articulated here and therefore, can be modified or integrated with any particular tasks for the specific purpose. Within the synthesis stage of the design process model, this approach employs the genetic search mechanism to improve the search performance. The Pareto optimality concept is used in the genetic evolution model to evaluate solutions efficiently under multiple criteria.

The evolutionary design process model is composed of three main processes: design analysis process, genetic search process and design evaluation process. The basic structure of the process maintains the analysis-synthesis-evaluation process except that the synthesis process is replaced by the genetic search process.

4.2.1. Design Analysis Process

The design analysis process prepares a set of design elements which will be transformed by the genetic search process. This process receives a set of requirements from the user and sends a set of genotypically formulated design elements and their related behaviours to the next process. The entire task of the process is executed in three consecutive steps: retrieval of an appropriate design prototype from the design prototype base, retrieval of appropriate design elements abductively and their related behaviours from the design prototype, and formulation and interpretation of the retrieved design elements based on the design rule schema and the design gene schema.

The design rule mechanism is introduced to embed and formulate the retrieved design elements. Design rules are instantiated by the combination of all possible design elements based on the design rule schema. For example, if a design rule schema is "if <a space element>, then put <a space element> in <a direction>," eight space elements and four directions can produce 112 design rules through the design rule instantiation. As the number of design elements grows, the utility of the design rule schema is more critical. The formulated design information by the design rule schema, however, is the phenotypic information and needs to be translated to the genotypic information in order to be manipulated by the genetic engine. The design gene schema provides a framework for the translation.

4.2.2 Genetic Search Process

The genetic search process is simple and acts like a black-box. In outline, the process starts with a randomly seeded initial population of genotypes. Thereafter the evaluation-recombination-selection loop starts running iteratively until certain parameters are satisfied. The initial population is evaluated against the given design environment and according to their performances or fitness, individuals are selected once or more than once, or not at all. The Pareto optimization concept is used to convert the performances to their Pareto group values for efficient selection under multiple criteria. Then the selected solutions are sent to the mating pool for the genetic operations. The genetic operation combines two individuals of a population to allow their information to be exchanged by the crossover operator, or flips

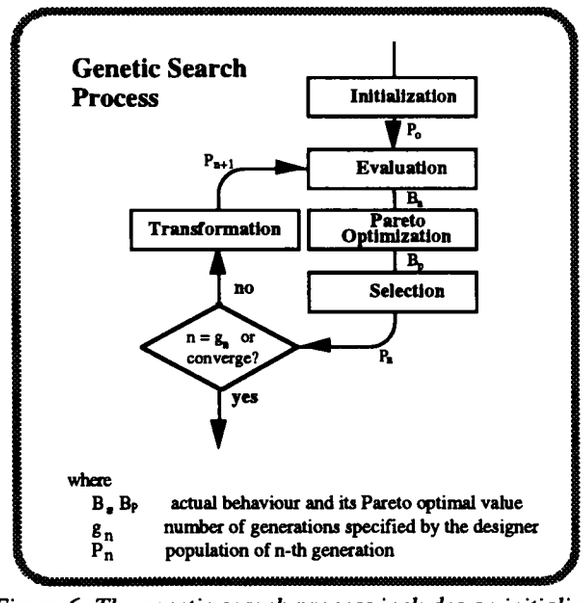


Figure 6: The genetic search process includes an initialization and three iterative operations: evaluation, selection and transformation.

bits of individuals to change their information by the mutation operator. The transformed population is evaluated and a new population is selected from the old one for the next generation, and thereafter successive populations are generated through an iterative process. This search process continues until the termination condition is met, Figure 6.

The *initialization* task starts with instantiating an empty genotype with random seeds and therefore generates a population of genotype strings. The empty genotype is constructed based on the design gene schema in the design analysis process. Values for a number of parameters affecting the process are assigned in this process. The parameters include population size, crossover and mutation rates, convergence criteria, etc. The initial design space does not need to be complete or have the potential solution represented explicitly, and this is one of the merits of the system (Maher and Kundu, 1993).

The *evaluation* task is to measure the quality of a solution by comparing the actual behaviour of the generated solution with the expected behaviour specified by the functional requirement. The choice of different evaluation functions allows the design process to focus on a specific design space and changing the value or a range of values of expected behaviour allows the solutions to converge to a specific location in the design space. For example, the proportion constraint on a rectangular shape introduces the geometric design space and the expected behaviour value leads the solutions into long or short rectangular shapes. Constraints can work in combination. The evaluation operation in the genetic search process consists of three serial tasks: decoding of the genotypic information into the phenotypes, deriving actual behaviours from the phenotypes, and the comparison between the actual and expected behaviours. Since the design problem is likely to be a multiple criteria one, the operation applies the Pareto optimality concept. Instead of the actual behaviours, the Pareto group values are then the basis of the selection operation.

The decoding task can be done in the reverse direction of the design formulation process. As an example, if one of the binary solutions is "00010110" and each design gene takes two bits in a genotype, the process splits the string into a number of segments, each having two bits: 00, 01, 01 and 10. According to the design gene schema and the interpretation knowledge, the segments are translated into the corresponding design information.

For multi-criteria optimization problems, there has been much work using weighted sums and penalty functions to turn the problems into single-attribute problems. The weighted method, however, can lead the selection in a wrong direction due to the incorrect assignment of weights. This danger can be reduced by incorporating the concept of Pareto domination in the evaluation operator. The selection operation then operates on vector values which are based on a partial order of the solutions' performance rather than their scalar values. An individual with a higher vector value in the order than that of another is preferred. This precludes the premature convergence of solutions, and maintains the diversity of the solutions, Figure 7.

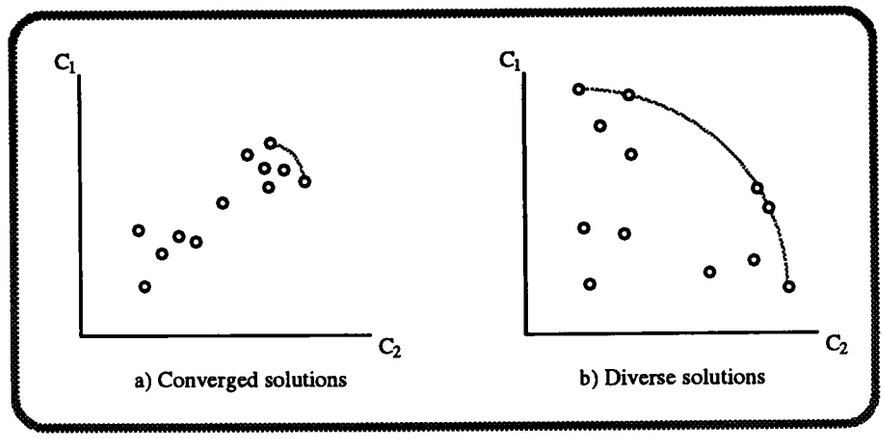


Figure 7: The use of Pareto optimization concept prevents premature convergence, and keeps the solutions more diverse over the design space.

The *selection* operation is a process in which strings are copied according to their performances, or the Pareto optimal values. The solution with a higher value has a higher probability of contributing one or more offspring in the next generation. Inferior solutions, which may be located outside of the feasible area, also have a chance to be selected and this makes the range of the search wider. The selected strings are then used for reproduction. The new population is selected from the transformed, or child population, and the previous, or their parent population. The selection from the populations of two generations gives good solutions more chances to be selected. The newly selected population then replaces the parent population and the next child population will be generated from the parent population through the transformation operation.

$$p(t) = \text{select}\ (\ p(t\text{-}1) + p(t)'\ )$$

where

p(t)' newly generated population or child population,

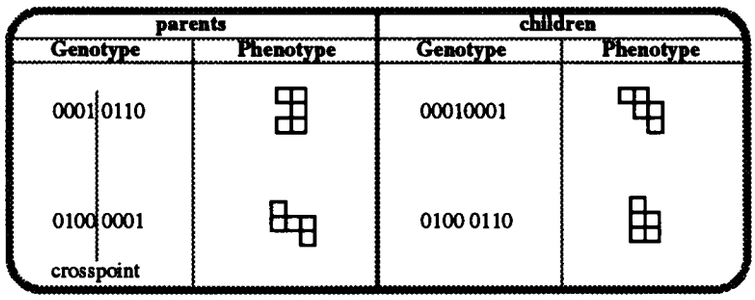
p(t-1) previous population or parent population,

p(t) newly selected population.

Another strategy for the selection operation is to keep the best individuals. This prevents good information from being lost by the stochastic mechanism. In every generation, the individuals having Pareto optimal performances are chosen first and the remaining positions of the population are filled with others by the normal selection method.

There are two kinds of genetic transformation operators considered for the operation: crossover and mutation. The operators manipulate the genotype which is the alternative design information.

Crossover proceeds in two steps (Goldberg, 1989). First, members of the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes crossing over. Two new strings are then created by swapping all characters. For example, if a randomly selected pair of individuals are 00010110 and 01000001 then the possible children from the couple can be 00010001 and 01000111



where

00	→	or 'put a cell on the right side'
01	↓	or 'put a cell on the bottom'
10	←	or 'put a cell on the left side'
11	↑	or 'put a cell on the top'.

Figure 8 An example of the crossover operation. Two alternatively represented structures exchange their design information and create new two child structures.

when the random crosspoint is in the middle of a genotype, as shown in Figure 8.

Mutation can introduce solutions which crossover cannot produce at all. There are two main benefits from the mutation operation. First, even though selection and crossover effectively search and recombine extant notions, some groups of genes affecting high fitness value are likely to remain intact defining regions in the design space and to be propagated to future generations. This leads the process to become overzealous and lose some potentially useful genetic material and therefore the solutions develop into a uniform population. The mutation operator of the model is an insurance policy and protects against premature loss of important notions (Goldberg, 1989). Second, the mutation operator can introduce unexpected solutions which are not defined by the given constraints. The simple operation only flips one or several bits then moves the current design onto some new places which may be somewhere in the current design space or the outside the current design space expanding the design space. This may introduce new design variables (Gero and Maher, 1991).

An example of a mutation operation in design is illustrated in Figure 9. An individual genotype, 00010110, is transformed to 01010110 by mutating the second bit of the string from 0 to 1. This transformation produces a different phenotype.

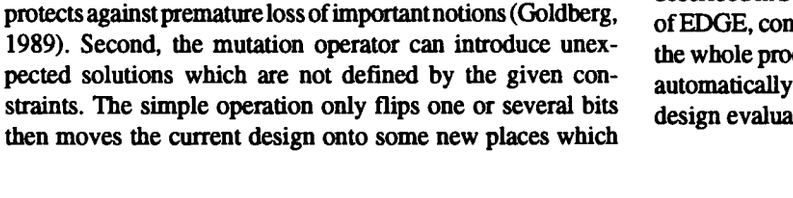


Figure 9: An example of the mutation operation

4.2.3 Design evaluation process

As the generations continue, the genetic search process makes solutions converge onto a certain point which is directed by the evaluation functions. The threshold of convergence is decided by the designer a priori. When the convergence occurs or the specified number of generations has been executed, the system asks the designer to judge the next step. If the solution satisfies the requirements then the process terminates successfully, otherwise, the designer can guide the process back to the design analysis process in order to retrieve other design elements or evaluation functions from the current design prototype, to choose another design prototype, or to restructure the design rule schema or the design gene schema. The designer can also guide the process back to the initialization stage in the genetic search process to change the parameter values or random seeds, or to the genetic search loop to continue transformation without any change. The changed information reconstructs the current search space or provides a new search space to be searched by the process.

The design process may give sub-optimal solutions instead of a complete and optimal solution. For example, a design solution may demonstrate good characteristics except in a small part. In this case, the designer may fix it very simply without further computation. It seems to be appropriate that the designer judges the termination of the design process and finalizes the process with some modifications.

5. Results

The EDGE system is an implementation of the concepts described in Sections 3 and 4. As an example of the application of EDGE, consider the layout of an architectural floor plan. In the whole process, the genetic search process is implemented automatically while the design analysis process and the final design evaluation process is carried out manually.

5.1 Design environment

For the example, it is assumed that the design elements, which are retrieved through the design analysis process, include 8 spaces and 4 topological relationships. The space elements include living room, sleeping room1, sleeping room2, dining room, kitchen, bath room, laundry and storage. The topological relationships include east_of, west_of,

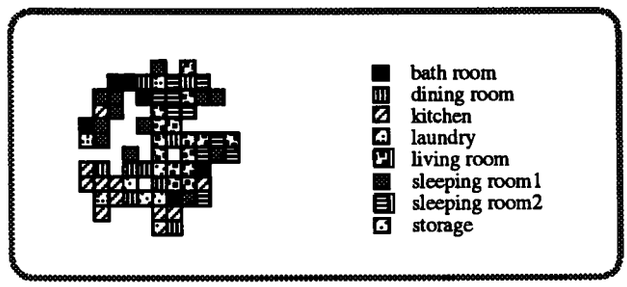


Figure 10 An initial solution generated by random seeding.

south_of and north_of. The population of solutions is made up of 88 individual genotypes. The number of design genes in a genotype is 70 and specifies the number of pixels comprising a 70m^2 house plan. Figure 10 shows a randomly seeded an initial individual where the 70 pixels are randomly scattered. The solutions can be mapped as widely scattered points over the design space and will start moving to the better points as the number of generations proceeds.

Three kinds of evaluation functions are employed: reducing the length of the perimeter of each room, reducing travel cost between rooms and producing an appropriate area for each room. The first criterion guides the rooms of a solution to have an economical shapes such as a rectangular or square ones. The second criterion is to locate space elements, such as the kitchen and the dining room, close to each other. Any constraints related to shape are not considered in order to give more freedom to the evolution of the solutions. The calculation is applied to every pixel using travel costs. The third criterion is concerned with the geometric aspect of the solutions. It prevents a room from being too small or big size, or missing. A table provides the penalty system according to the difference between actual and required room areas.

5.2 Solutions

Operations of the EDGE system, based on the given design environment, show that the noisy initial solutions are evolved and transformed to the solutions which are economically shaped, topologically reasonable and proper sized. Figure 11 presents transformed solutions of the initial population after 100 generations. The noisy pixels may need more generations to converge or can be refined manually.

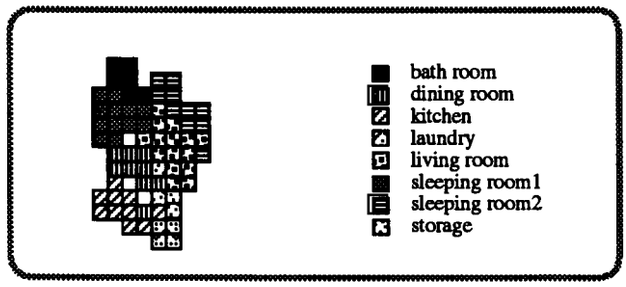


Figure 11 Phenotype of a solution after 100 generations of operations.

In the previous section, it was mentioned that the solutions can be modified by the designer. Figure 12 shows how a phenotype solution is modified and becomes a house plan. Some noisy pixels are moved or exchanged with others to make the shape of the plan simpler. The blank space inside or around the house plan can be regarded as a courtyard, a hall or a corridor. The transformed results can be different depending on the designer and/or the given design situations. These results can give some initial ideas to designers so that they can be used as the beginnings of a practical space layout design.

6. Discussion

The aim of this paper was to develop a new method for space layout planning. To achieve it, a genetic search mechanism was introduced. The advantages of the genetic evolutionary concept were addressed. The genetic search process replaced

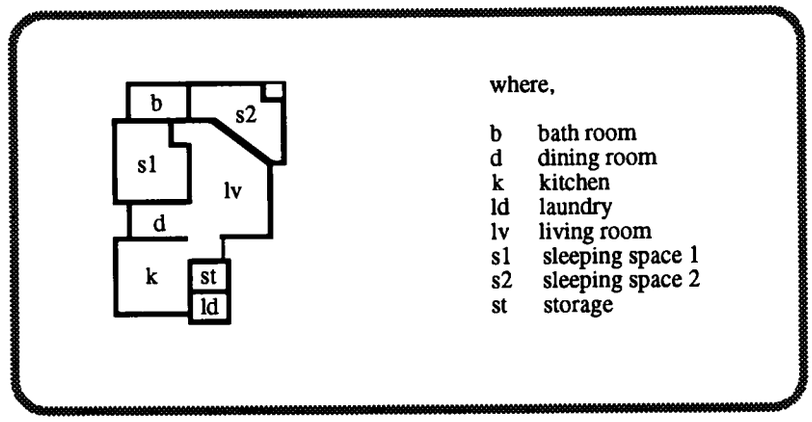


Figure 12: A modified space floor plan of Figure 11. The designer can choose and modify a promising solution for practical use.

the synthesis part of the analysis-synthesis-evaluation process model of design and constructs a new design process model. A system, called EDGE, was implemented. The major contributions of this paper are as follows.

The genetic search process was introduced to control the combinatorial explosion of the space layout planning efficiently. The random search and the selection based on controlled probability allows the process to search over the broad range of design space. The fixed size of population prevents the combinatorial problem from occurring while searching the design space massively and in parallel.

The concept of the design rule schema and the design gene schema made the formulation and interpretation of design information easy and efficient. Instead of preparing all design rules required for the design process, the design rule schema simply generates any number of design rules through instantiation of possible design elements. The design gene schema is formed by modifying the design rule schema. Design genes are generated by instantiating the design gene schema and compose a genotype string. A structure is then transformed to genetic codes and the transformed structure is decoded to be evaluated. Interpretation knowledge supports the translation tasks by supplying knowledge relevant to the interpretation.

The Pareto optimization concept was employed in the evaluation operation to handle the multi-criteria problem. This technique transfers an individual's actual behaviour to its vector value on which the selection operation can be performed.

The implementation manipulated the topological and geometrical problems of space layout planning in parallel. Each room was decomposed to its unit area and was manipulated in the form of a rule which includes topological relationships. This approach is hard to achieve using other methods because of the large size of the design space generated.

Promising solutions were found and, for practical purpose, modifications of the results are made by the designer. A genotype in the current stage includes a set of design genes which is produced by the homogeneous design gene schema. The design process would be more powerful if a genotype contains different kinds of design gene schemas so that each design gene has a different role depending on its locus in the genotype. For example in the house design, each design gene in a genotype can represent roof, wall, floor or window. This may also allow different abstraction levels of design information, including behaviours and functions, to be embedded in a genotype and manipulated.

References

Auger, B. (1972). Computer program for generating layout from blockspacing criteria, *Proc. PTRC Seminar on Housing Estate Layout*, London.

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

Bijl, A. and Shawcross, G. (1975). Housing site layout system, *Computer Aided Design*, **7**(1): 2-10.

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.

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.

Flemming, U.A. (1978). Wall representation of rectangular dissections and their use in automated space allocations, *Environment and Planning B* **5**: 215-232.

Flemming, U.A. (1985). A generative expert system for the design of building layouts, *Research Paper*, Department of Architecture, Carnegie Mellon University, Pittsburgh.

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. (1990). Design prototypes: a knowledge representation schema for design, *AI Magazine* **11**(4): 27-36.

Gero, J.S and Maher, M.L. (1991). Mutation and analogy to support creativity in computer aided design, *in* G. Schmitt (ed.), *CAAD Future '91*, ETH, Zurich, pp. 241-249.

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

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.

Koning, H. and Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses, *Environment and Planning B* **8**: 295-323.

Landsdown, R.J. (1972). *Design of Office Blocks in Computer Program Descriptions for Architects*, Department of Environment, Directorate of Research Requirements, London.

Liggett, R.S. (1980a). A partitioning approach to large floor plan layout problems, *CAD80*, IPC Science and Technology Press, Guildford, Surry, pp.705-714.

Liggett, R.S. (1980b). The quadratic assignment problem: an analysis of applications and solution strategies, *Environment and Planning B*, **7**: 141-162.

Maher, M.L. and Kundu, S. (1993). Adaptive design using genetic algorithms, *in* J.S. Gero and F. Sudweek (eds), *Preprints Formal Design Methods for CAD*, Key Centre of Design Computing, University of Sydney, pp.211-228.

March, L. and Steadman, J.P. (1971). *The Geometry of Environment*, RIBA Publications, London.

Miller, B.F. and Keane, C.B. (1987). *Encyclopedia and Dictionary of Medicine, Nursing, and Allied Health*, Saunders, USA.

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.

Nilsson, N.J. (1969). Searching problem-solving and game-playing trees for minimal cost solutions, *in* A.J. Morrell (ed.), *Information Processing 68 (Vol. 2)*, Amsterdam, North-Holland, pp. 1556-1562.

Pareto, V. (1896). *Cours D'Economie Politique*, Vols I and II, Rouge, Lausanne.

Pfefferkorn, C.E. (1975). A heuristic problem solving design system for equipment or furniture layouts, *Communications of the ACM* **18**(5): 286-297.

Steadman, J.P. (1973). Graph theoretic representation of architectural arrangement, *Architectural Research and Teaching* **2**: 161-172.

Stiny, G. and Mitchell, W.J. (1978). The Palladian grammar, *Environment and Planning B* **5**: 5-18.

Woodbury, R.F. (1993), Design genes, *in* Gero, J.S. and Maher, M.L. (eds), *Modelling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 211-232.