Representation and Use of Design Knowledge in Evolutionary Design

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.su.edu.au

This paper describes an approach to knowledge representation for an evolutionary design process. The concept of design schemas is introduced to provide the representational framework for design knowledge. Two kinds of design schemas, the design rule schema and the design gene schema, are proposed to formulate design knowledge and interpret the knowledge into genetic codes. A design problem which is used to exemplify this approach is that of a large office layout planning problem.

1. INTRODUCTION

The power of natural evolution has been proven in the real world through the history of life on Earth, despite what appears to be its simple mechanism. Much research has attempted to apply the evolutionary concept to design problem solving [3, 7, 8]. This has shown its robust search ability and has produced promising results. The advantages observed encourage the use of the evolutionary concept in design problem solving, especially for such fundamental computational questions as combinatorial explosion, non-linearity, knowledge dependency, non-continuity, etc. Issues derived from the research include how to formulate and represent design knowledge in the evolutionary design process.

An analogy between biological evolution and a general design mechanism involves two layers of representations: implicit and explicit [17]. The implicit representation is a design space and corresponds to the biological genotype. The explicit representation is the result of 'executing' a design space, and corresponds to the biological phenotype. This paper focuses on the representation of genotypes rather than phenotypes because design synthesis in the evolutionary design process occurs at the genotype level.

A fundamental axiom in cognitive psychology and computer application in design is that knowledge is organized as structured packages, commonly referred to as schemas [16]. 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 [12]. The representation methods which are to be considered fall in the category of design prototype schema, design rule schema, and design gene schema. The design prototype schema [2] is a conceptual framework providing a comprehensive representation and organisation of design information and knowledge. However, this will not be further elaborated in this paper. The design rule schema defines a class of design transformation rules and provides an elementary framework to embed knowledge of transformations through the causality relationships between concepts. The design gene schema represents the design information in a genetic code. The encoded information is manipulated by genetic operators. Such schemas are used in combination to represent all necessary design information and knowledge for the design process. The remainder of this paper provides a brief description of the evolutionary process in design before concentrating on the genetic representation of design knowledge. Finally, the genetic design paradigm is used to solve a layout design problem from the literature [10].

2. GENETIC EVOLUTIONARY PROCESS IN DESIGN

Natural evolution is a slow process, operating over a long period. Most organisms evolve by means of two primary processes: natural selection and sexual reproduction. Natural selection determines which members of a population survive to reproduce. Individuals being more highly suited to the environment have more chances to survive and therefore to produce their descendants, than the others. Over time the population of individuals becomes increasingly adapted to the environment in which they live.

A genetic evolution mechanism was introduced into the artificial world as a search algorithm [4, 6]. The algorithm is a simple and blind process which does not necessarily need any specific heuristic guidance. The process is very economical and the solutions can be more varied than those of classical search processes. The search space is not limited by restrictive assumptions concerning continuity, existence of derivatives, uni-modality, and other matters [4]. 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 multi-modal (many-peaked) search space, 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 getting close to the global optimal solution and the probability of finding a false peak is reduced over other methods.

One aim of a design process is to produce optimal or near optimal structures which satisfy the given requirements. Design usually proceeds in a step-wise and a cyclic manner. It is almost impossible to pick up the solution expected at once because of the complex design criteria involved. These aspects can be treated as an analog to the natural evolution process where the goal of natural evolution is to adapt individuals to the given environment gradually. When the evolutionary concept is applied to the design process, the advantages of the genetic search process are applicable to design problem solving. The simple algorithms reduce the need for sophisticated design formulations which may be the hardest part of solving a complicated design problem. The random but probabilistic search allows a diverse range of search. Working with a population of solutions helps the design process to escape from local optimal points easily, and provides a variety of alternative solutions for the designer to choose from.

3. GENETIC REPRESENTATION OF DESIGN KNOWLEDGE

The terminology of genetic representation is based on both natural genetics and computer science, in which the idea is rooted. The terms discussed below will provide a basis of the knowledge representation for the evolutionary design process. The separate representation of genotype and phenotype, or implicit and explicit, seems to provide both efficiency and opportunities in problem formulation and search.

3.1 Genetic Representation

In genetics, a set of genes composes a chromosome which is analogous to a string of symbols in an artificial genetic system. They take some values called alleles. A gene, in artificial terminology, can be considered as a particular character in a string or an instruction of a recipe. A locus is the position of a gene and is identified separately from the gene's function [4].

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. Changes including mutation and crossover occur in the genotype. The fitness of an individual is derived from the phenotype which exposes itself to the environment. In nature, a phenotype is the visible expression of an organism. It is a decoded genotype, therefore, tangible and confronts the environment. The mapping from the genotype to the phenotype allows the realization of the structure of an individual solution and the behaviours can be extracted from it for evaluation of the structure's fitness. The genetic search process works with a population of genotype strings. The expression of the gene, genotype, phenotype and a population of the t-th generation are described below:

$$g_i = \{\text{locus, allele}\}$$

$$G_i = \{\Sigma g_i\}$$

$$P = m(G)$$

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

where,	g_i	a gene.
	G_i	a genotype string.
	m	mapping or interpretation operator.
	P	a phenotype,
	p(t)	population of the t-th generation.

3.2 Design Rule Schema for Knowledge Formulation

Rules have been shown to be useful especially for organizing generative knowledge. Rules specify allowable moves providing a derivation tree or a search space. A rewrite rule [14], which is a typical generative design rule, consists of two parts: a left-hand side (LHS) and a right-hand side (RHS). The LHS specifies the target design elements on the structure space and the RHS replaces the LHS, where LHS is normally specified by the designer. A rewrite rule is applied to a set of design elements which matches the LHS of the rule, and then replaces it by the specified RHS. The general expression of a rewrite rule is shown as below:

r_i : if **LHS**, then **RHS** or **LHS** □ **RHS**

where,	r_i	design rule label,
	LHS	left hand side,
	RHS	right hand side.

An attempt to manipulate components of a rule is made by constructing a schema of the design rule. A design rule schema [7] is defined as a class of design transformations and a design rule becomes an instance of a specific design rule schema (S_r). Then together with the design elements involved, a rule schema provides a design space for the design process. A design rule schema in this approach does not consider RHS of a rule, instead, deals with a design transformation operator (□) which indicates a design transformation action. A design rule is applied to a set of design elements which match the LHS of the rule. Then the design element set (LHS) is transformed by the design transformation operator and the result (RHS), which is not in the design rule, appears in the phenotype. The general expression of a design rule schema is shown as below:

S_r = {LHS, ▯}

where,	S_r	design rule schema
	▯	design transformation operator

Figure 1 shows examples of simple design rules. In the design rule schema of Figure 1(a), the design transformation operation (▯) introduces a new design element, “triangle,” and its location about the LHS, “on the top side of”. The RHS which is the result of the design rule application, is a combined shape with a square and a triangle and appears in the phenotypic design description. The design rule schema can be generalized as "if there exists a design element (E_x), then transform it by introducing a new design element (E_n) and a transformation action (▯) ," and the instantiated design rule, in Figure 1(a), is shown as below:

S_r = {E_x, (E_n, ▯)}
r_i = {□, (△ , ▯)},

where,	▯	transformation action
	E_n	new design element
	E_x	existing design element

In the above case, the shapes are distinct enough to be identified therefore labels or markers are not necessary. If the design elements include labels for any functional or identity purpose, the design rule schema can include the labels with their design elements, Figure 1(b).

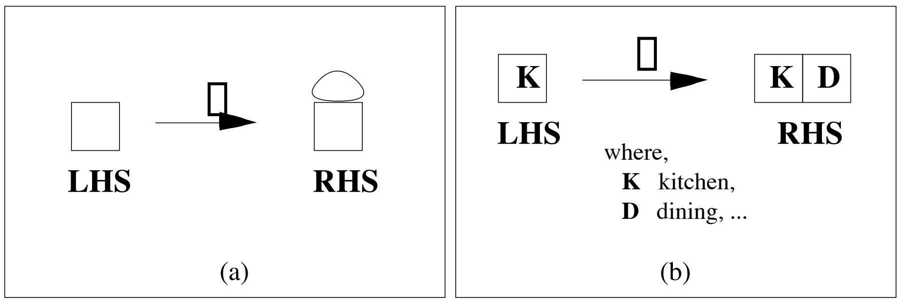


Figure 1. Design rules consisting of LHS, ▯and RHS: (a) without labels and (b)with labels.

The design rule schema of Figure 1(b) follows the same format as the previous example except for the labels. Then the design rule schema can be described as "if there exists a design element (E_x) with its label (L_x), then transform it by introducing a new design element (E_n) with its label (L_n) and an action (▯)."

S_r = {(E_x, L_x), ((E_n, L_n), ▯)}
r_i = {(□, K), ((□, D), ▯)}

where, L_x label of existing element(E_x),

L_n label of new element(E_n).

Working with labels sometimes encounters the case that LHS does not match with the specified labels on the existing design elements. Using a marker in design rules is often helpful to identify the design elements to which the next rule will be applied. Figure 2 shows examples of design rules using markers. The meaning of the design rule of Figure 2(a) is "if there exists a design element (E_x) with a marker(•), then transform it by introducing a new design element (E_n) and an action (▯)."

$$S_r = \{(E_x, \bullet), (E_n, ▯)\}$$
$$r_i = \{(\square, \bullet), (\square, ▯)\}$$

where, • marker,

If necessary, the design rules can include complex rule components such as distinct shapes, labels, marker and various actions. The design rule schema of Figure 2(b) includes a new design element (E_n) with its label (L_n), topological and connectional relationships (▯) between the existing and new design elements.

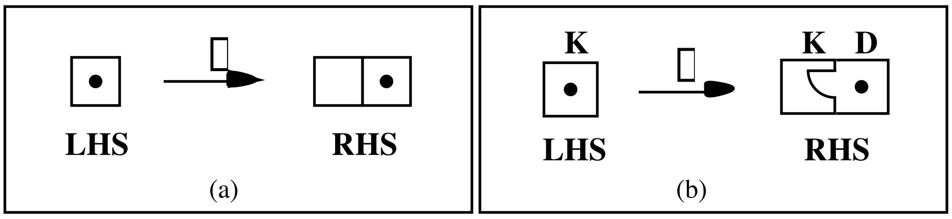


Figure 2. Simple rules with markers: (a) without labels and (b)with labels.

Design rules using the design rule schema are different in several aspects from rewrite rules [7]:

1. The application of rewrite rules does not need knowledge about how LHS is transformed to make the result (RHS). The use of design rule schema, however, includes the consideration of the semantics of the rule instead of just the syntactic result, and this makes the transformation process more rational.

2. The approach with rewrite rules needs all necessary rules to be specified by the designer before the design transformation process is started. Whereas the approach employing the design rule schema can start the process with even only one design rule schema. During the process, all possible design rules are generated by instantiating the schema with possible design elements. Therefore the approach with the design rule schema seems to be simpler and more dynamic than that with rewrite rules. As the number of design elements involved increases, the difference between the two approaches becomes larger. To describe even a simple design, thousands of design elements and their relations may be involved and the use of rewrite rules is likely to produce the combinatorial explosion problem.

3. Because the design rule schema and the design elements are independent of each other, both are easily modified. When new design elements are introduced during the process, the design rule schema easily accommodates them and produces new design rules. However, the approach with the rewrite rule is to

reconstruct a new set of rewrite rules by the designer. Therefore the use of the design rule schema seems to be more flexible.

4. Insisting on fixed rules may restrict the variety of the solutions produced. The use of a design rule schema, however, allows search over a wide range of the structure space by instantiating all possible rules. For example, if there are 8 design elements and 8 actions involved, then 224 design rules can be generated by the instantiation of the design rule schema, which is a difficult task manually. As the number of design elements increases, a larger number of possible design rules can be generated from the combination of the elements in the design rule schema.

3.3 Design Gene Schema for Knowledge Interpretation

A hybrid design system constructed by combining two heterogeneous processes, a model of natural evolution and a design process, requires a new strategy to allow communication between the two processes. A design gene schema [7] is built to provide a framework to represent design knowledge in a genetic language. While keeping the original semantics of the design rule schema, a design gene schema is constructed from a design rule schema based on the following principle: if a component of a design rule schema needs to be transformed by a genetic operation then the component is entered in the design gene schema otherwise it is inactive and does not appear in the design gene schema. The design elements and transformation actions, which are components of the design gene schema, get their own loci and alleles in the genetic code. Design genes are usually in the form of binary numbers or symbols. Because the design gene schema is a restructured design rule schema, design genes are genotypic representation of design rules. The design gene schema allows for consistent information maintenance during the genetic transformation process, and for the transformed solutions to be translated into the design world correctly. This alternative representation of design information makes the design transformation easy and rich by using simple genetic operations.

The interpretation knowledge (K_i) 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.

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

where,	S_g	design gene schema
	K_i	interpretation knowledge
	$\square_s$	schema transformation operator

For example, there are four simple topological design rules instantiated from a design rule schema as shown in Figure 3. Notice that several elements in each design rule always have equivalent values, such as the same square shapes and a marker, and do not need to be transformed. Thus the active component in the example is only the topological 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 decoded to the design rules.

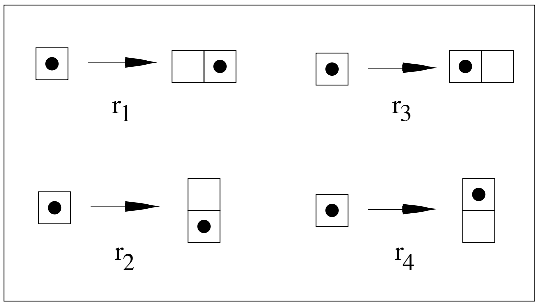


Figure 3. Simple topological design rules with a marker.

S_r = {LHS, ▯}
= {E_x, (E_n, ▯)}

The instantiated design rules are:
r_1 = {□, (□, ▯)}, r_2 = {□, (□, ▯)}, r_3 = {□, (□, ▯)}, r_4 = {□, (□, ↑)}.

There is only one active component to be transformed: transformation action. Therefore, S_g = {▯} and the instantiated design genes are: g_1 = {▯ }, g_2 = {▯}, g_3 = {▯ }, g_4 = {↑}.

The interpretation knowledge, K_i, includes information about:

1. S_r, and S_g
2. Active components of the design rule schema and their possible values: ▯ , ▯, ▯ , ↑
3. Inactive components which do not appear in any design gene: an existing and a new design element (□) and a marker (•).
4. Starting and terminal rules.

A design produced by the genetic search process can be varied according to any of the following:

(i) what the components of a design gene (or a design rule) are,
(ii) how a design rule schema is interpreted into a design gene schema,
(iii) how the transformed design genes are translated into the phenotype.

Item (i) indicates which design elements are involved in the design process and this decides the size of the design space. Item (ii) concerns which, among the elements involved, are active and therefore are to be transformed. This decides the variables of the design. Item (iii) is about the decoding stage and decides how to manage the conflicting rules or illegal solutions.

3.4 Design Genotype

A design genotype is a collection of a set of design genes. It combines separate design information into the complete genetic representation of an entire individual structure. When a linguistic analogy is used, the genotype can be regarded as syntax due to its explicit existence in a search process whereas the phenotype is considered as semantics representing the meaning. The interpretation knowledge then provides information for the interpretation between the syntax and semantics. A genotype can be represented in binary, digital or codes. Advantages and disadvantages of the use of the different codes were investigated by Jo [7]. The implementation in this paper adopts the binary code representation because:

1. The system using binary codes requires a process interpreting solutions to digital or text. However the system with binary codes is more problem independent than others because, for a new design problem, it needs to change only the module concerning the interpretation while other approaches may need to modify the program itself.

2. The binary representation usually takes more digits to represent a design gene and this provides more possible crosspoints for crossover than the other representation methods.

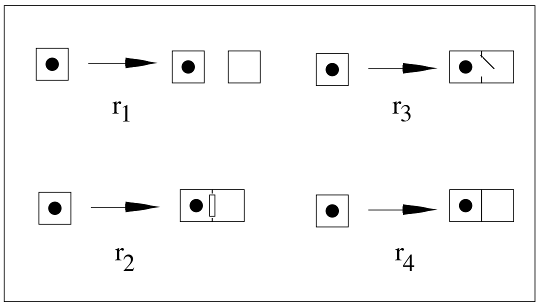
3. The mutation operation which flips a bit from 0 to 1 or vice versa, is very simple and convenient in the binary representation. Whereas the operation in other representations needs permutational choices which is a more expensive task.

4. Computation with the binary representation is very fast and economical since it operates with a lower-level representation.

From the previous example in Figure 3, the design gene schema carries only one component which is the topological transformation action. Then there are four possible design genes with their specific values to be instantiated from the schema and they can be represented in a binary representation as 00, 01, 10 and 11. If four other connectional design rules are involved, Figure 4, the formulated design rule schema and its interpreted design gene schema have following structure.

$$S_r = \{m, (L_n, \square_c, \square_t)\}$$
$$S_g = \{L_n, \square_c, \square_t\}$$

where,	m	a marker,
	L_n	a label of a new space element,
	$\square_c$	connectional transformation action,
	$\square_t$	topological transformation action.



where,	r_1	separation relationship,
	r_2	door or accessibility relationship,
	r_3	window or transparency relationship,
	r_4	wall or isolation relationship.

Figure 4. Design rules with four connectional transformation actions.

For example, if a set of design rules is {..., {marker, (living_room, window, west_of)}, {marker, (dining_room, door, west_of)}, {marker, (kitchen, door, south_of)}, {marker, (bedroom1, wall, east_of)}...} then the resulting structure will be as shown in Figure 5.

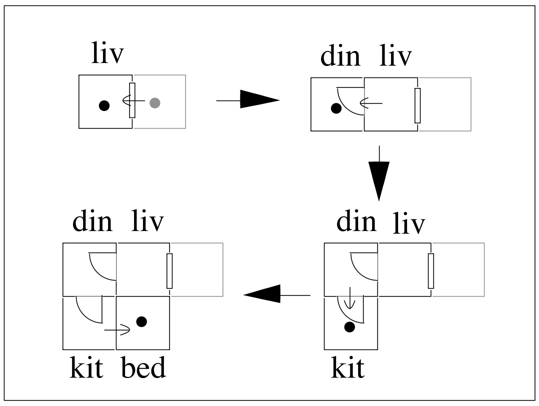


Figure 5. Application of a set of design rules.

When the four rooms involved in the above example are represented in the binary codes as living_room(00), dining_room(01), kitchen(10) and bedroom1(11), the design genotype of the above example can be represented as {...,(00, 10, 10)}, (01, 10, 10)}, (10, 01, 01)}, (11, 11, 00)}...}, or ...0010100110101001011111100... The design genes of this genotype do not include the marker, the inactive component. When the number of rooms involved in the design process is more than four, the number of binary digits, which represents a design gene, can be increased as required.

The design knowledge, including a set of design elements and transformation actions, is prepared as binary codes for the genetic search process through a formulation stage by the design rule schema and an interpretation stage by the design gene schema, Figure 6.

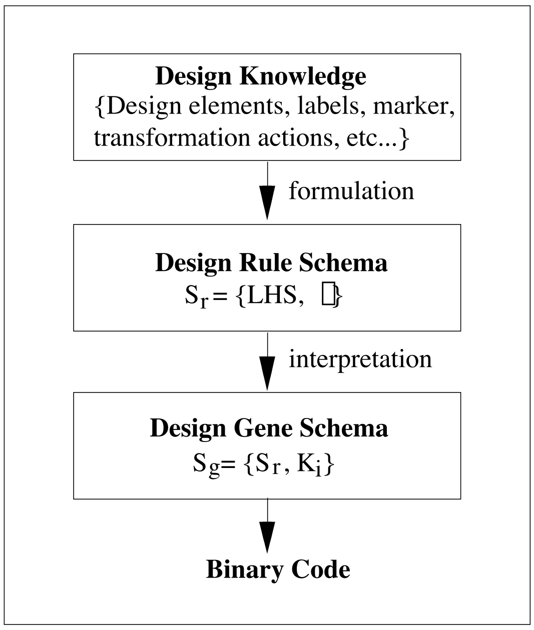


Figure 6. Knowledge formulation and interpretation.

4. EDGE: Evolutionary Design using Genetic Evolution

The evolutionary design system, EDGE (Evolutionary Design using Genetic Evolution), was constructed based on the evolutionary process [7]. Due to the domain independent nature of the genetic search mechanism, the system can be applied to solve various design problems with appropriate formulations and interpretations according to the domain. EDGE was developed in a Unix environment using the C programming language.

In outline, the process starts with a randomly seeded initial population of genotypes. The initial population is evaluated against the given environment and according to their fitness relative to others, individuals are selected once or more than once, or not at all. Then the selected solutions are sent into the mating pool for the genetic transformation operations. The genetic operation randomly combines two individuals of a population to allow their information to be exchanged by the crossover operator, or flips 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. At the beginning of the evaluation stage, the transformed solutions are in binary codes and need to be decoded to a proper level of expression so that behaviours of the solutions are recognized. The Pareto optimization concept is used to convert the evaluated behaviours to their Pareto group values for efficient selection under multiple criteria. This search process continues until the termination conditions are met, Figure 7.

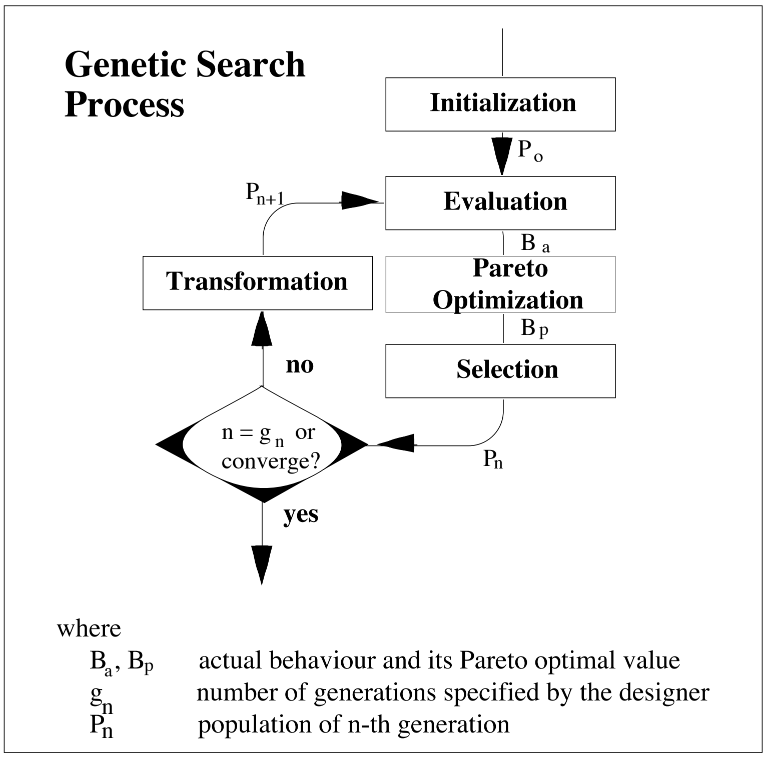


Figure 7. The genetic search process includes an initialization and three iterative operations: evaluation, selection and transformation.

5. EXAMPLE

The EDGE system is used to solve an office layout problem which was attempted by Liggett [10]. Through this example the ideas in this paper are applied.

Space layout planning is the process of allocating a set of space elements according to certain design criteria and usually results in topological and/or geometrical relationships between elements. Two major approaches are the generative method using grammars [9, 13, 15] and optimization techniques [1, 11]. Generally, a set of discrete but interdependent space elements makes the formulation of the problem difficult. 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 guaranteed algorithms for this problem. In the evaluation stage, the multiple criteria for the problem require expensive computations.

The EDGE system seems to be an appropriate tool to address some of the difficulties of space layout problems. The representation with a design rule schema and a design gene schema provides an efficient and well organized framework for knowledge formulation. The use of the genetic search mechanism can ease the combinatorial explosion problem. The Pareto optimization concept can relieve the evaluation process from both the complicated mathematical formulation of fitness function and the bias arising from multiple optimization problems.

5.1 Problem Specifications for the EDGE System

The given problem is a topological and geometrical assignment of office departments in a specified four-level terraced building. The given design elements for the problem include 21 departments, where 5 of them are prearranged. The locations to be assigned by the departments are segmented into 19 zones and 3 of them are taken by the prearranged departments, Figure 8. The required areas of the departments and zones are defined in terms of a number of equal-sized modules. Because the total area of the building has 3 more modules than those of the departments, the three modules will be considered as three separate departments. Therefore the problem is an assignment of 19 departments over the 16 zones. For complete details of the problem specification, see Jo [7].

5.2 Evaluation Criteria

The evaluation criteria 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. Each criterion is provided in the matrix by the user. The travel time matrix specifies distance between locations. The travel cost between a pair of activities is gained by multiplying the distance between their locations by this factor. 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 interaction matrix is based on subjective judgements of the client. The client rates adjacency needs on an ordinal scale (3 – most important, to 0 – not important). In order to neutralize 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. The problem can be stated as:

$$\min (\Sigma\, a_{ij} + \Sigma\Sigma\, q_{i_1 i_2}\, c_{j_1 j_2})$$

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

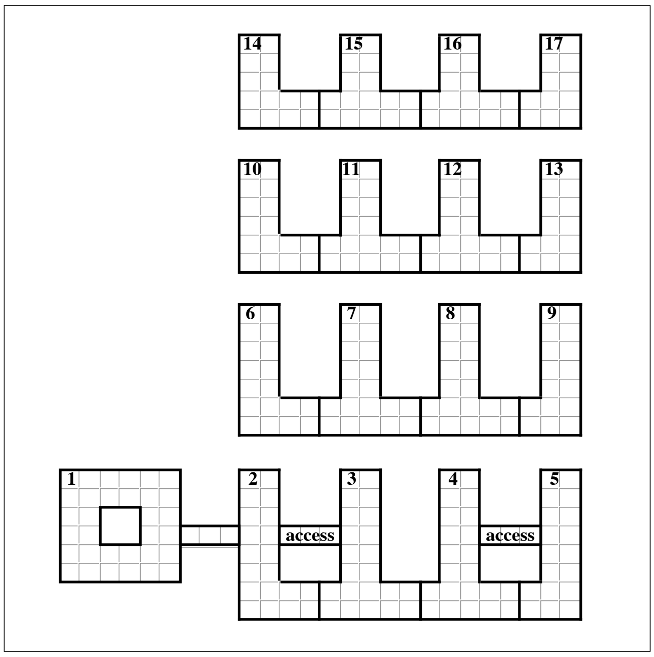


Figure 8. The data defining zones for the office layout problem.

5.3 Liggett's Approach

The two typical strategies in Liggett's system include constructive placement and pair-wise improvement. The constructive placement algorithm was originally developed by Graves and Whinston [5] and 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.

5.4 Design Formulation

A department is defined as a set of activity modules. 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 2 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.

5.4.1 Preparing Design Elements using the Design Rule Schema

A solution satisfying the given requirement 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:

$$S_r = \{LHS, ▯\}$$
$$= \{marker, new\_activity\}$$

Then the semantics of this design rule schema is “if a module has a marker, assign a module 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:

$$r_i = \{\bullet, (\square, l, ▯)\}$$

where,		
	•	marker,
	l	the label of an activity module,
	□	an activity module of a department,
	▯	assign a new module on the location specified by K_i.

5.4.2 Encoding the Design Elements using the Design Gene Schema

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 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 are kept in the interpretation knowledge (K_i)." Among the components of the design rule schema, the new activity (L_n) is manipulated by the genetic search process and therefore is the only component included in the design gene schema.

The total number of these design rules, or design genes, is the same as the total area of the space plan excluding the area for fixed activities. 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 duplicates. To prevent an activity from being used more than once, a reordering function is necessary to make all activities of a genotype distinctive.

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 [7]. The activities need five bits per byte to represent their elements, Table 1. The codes between 10011 and 11111 are unoccupied by any activities. These codes will be distributed to their corresponding activities when they are decoded.

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 1. 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. The length of a genotype string is calculated by multiplying the number of design genes in the genotype, number of components of a design gene and number of bits in a byte. In this example, the length of a genotype is 95 bits, where a genotype is composed of 19 design genes, each gene contains 1 component and each component has 5 bits. When the transformed genotypes are decoded, in the beginning of the evaluation stage, each gene is separated from the genotype and is translated into its corresponding design rule by referring to Table 1. Then, following the order in the genotype string, the design rules are mapped to produce a phenotype.

5.5 Results

The evolutionary design process starts with a population of genotypes where 5% of them are the same as a Liggett's final result and the remainder of the solutions are randomly seeded. This is to save time and to see if the process can produce better solutions than Liggett's final 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. The EDGE system uses the same evaluation criterion as does Liggett, minimizing travel cost of activities. However, the behaviour value of the Liggett's solution is not the same as that in her paper [10] 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.

After the 500 generations, it is observed that a few types of solutions dominate the population. The best behaviour produced by the process gives 1862, compared to Liggett's solution of 2029, a 8.2% improvement. The results show the superior performances of the EDGE system verifying its capability. The best behaviour is shared by 10 types of solutions and provides alternatives to the user. Figure 9 presents the details of solutions produced by Liggett's approach, Figure 9(a), and by the EDGE system, Figure 9(b). Three highlighted modules in each solution represent the empty modules and may be regarded as a rest room or a hall. However, both solutions have a similarity near the left 'access' on the first floor. This phenomenon may be caused by the strong interactions between the 'access' and some departments. Since the movement of modules within a zone does not affect the behaviour of a solution, some modules of solutions can be exchanged with others for the user's taste while keeping the behaviour same.

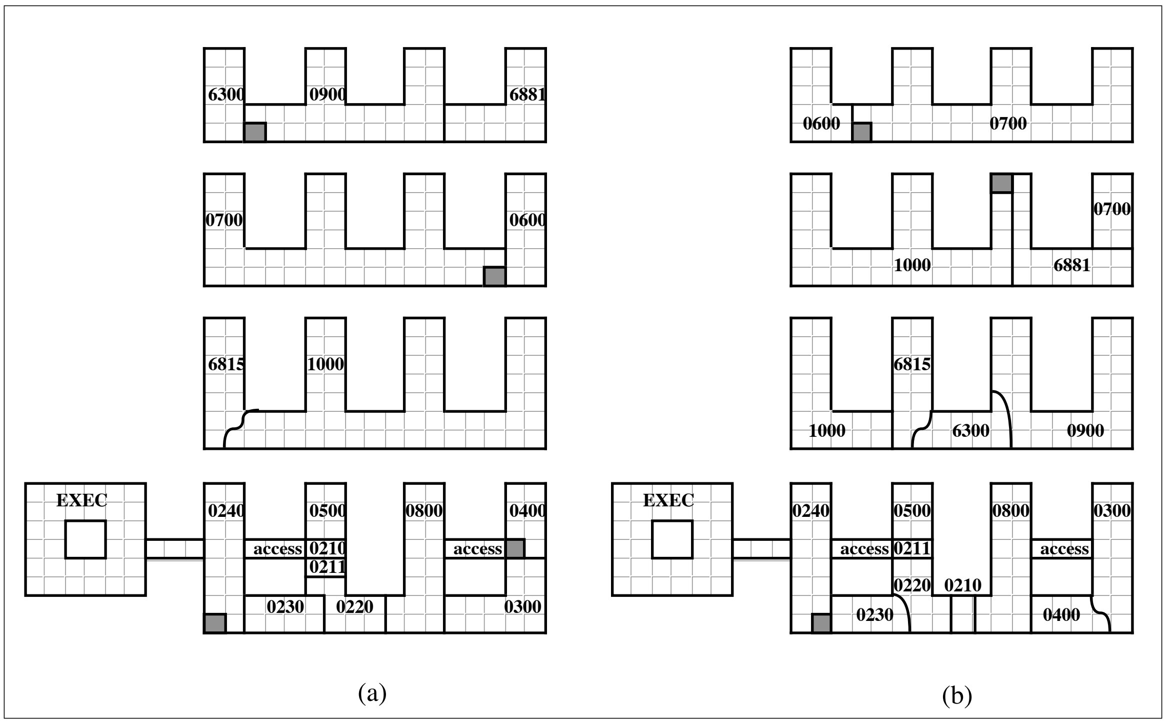


Figure 9. (a) The solution produced by Liggett's approach, and (b) a solution produced by the EDGE system after 500 generations.

6. DISCUSSION

This paper described the use of an evolutionary system in a design process. The characteristics of the concept were presented and their utility were investigated through an implementation. Two design schemas were introduced for the representation of design knowledge in an evolutionary design process. The design rule schema provides an efficient way of formulating design knowledge and the design gene schema is used to interpret the formulated design knowledge into the vocabulary of the evolutionary design process. As an example the EDGE system was constructed and applied to the office layout problem. The problem configurations and the evaluation criteria for the problem were drawn from the literature. The results show the usefulness of this form of knowledge representation and its efficacy in this design process model. This paper, however, did not deal with either the detailed structure of the schemas which improve the efficiency of the design process, or how to accommodate various other types of design knowledge. Further research on these subjects is necessary.

ACKNOWLEDGEMENTS

This work is supported in part by a grant from the Australian Research Council.

REFERENCES

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

[2] Gero, J.S., Design prototypes: a knowledge representation schema for design, *AI Magazine* **11**(4), (1990); 27-36.

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

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

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

[6] Holland, J.H., *Adaptation in Natural and Artificial Systems*, (Ann Arbor: University of Michigan Press, 1975).

[7] Jo, J.H., *A Computational Design Process Model using a Genetic Evolution Approach*, Ph.D. Thesis, Department of Architectural and Design Science, (Sydney: University of Sydney, 1993).

[8] Jo, J.H. and Gero, J.S., Space layout planning using an evolutionary approach, *Working paper*, (Sydney: Key Centre of Design Computing, 1995).

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

[10] Liggett, R.S., Optimal spatial arrangement as a quadratic assignment problem, *in* J.S. Gero, (ed.), *Design Optimization*, (New York: Academic Press, 1985); pp. 1-40.

[11] Mitchell, W.J., Steadman, J.P. and Liggett, R.S., Synthesis and optimization of small rectangular floor plans, *Environment and Planning B* **3**, (1976); 37-70.

[12] Rumelhart, D.E., Schemata: the building blocks of cognition, *in* R.J. Spiro, B.C. Bruce and W.F. Brewer (eds), *Theoretical Issues in Reading Comprehension*, (New Jersey: Lawrence Erlbaum, 1980); pp. 33-58.

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

[14] Stiny, G. and Gips, J., *Algorithmic Aesthetics*, (University of California Press Berkeley, 1978).

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

[16] Tham, K.W., *A Model of Routine Design Using Design Prototypes*, Ph.D. Thesis, Department of Architectural and Design Science, (Sydney: University of Sydney, 1991).

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