THE GENERATION OF FORM USING AN EVOLUTIONARY APPROACH

M. A. ROSENMAN
Key Centre of Design Computing
Department of Architectural and Design Science
University of Sydney NSW 2006 Australia
Email: mike@arch.su.edu.au

1. Introduction

Design is a purposeful knowledge-based human activity whose aim is to create form which, when realized, satisfies the given intended purposes.[1] Design may be categorized as routine or non-routine with the latter further categorized as innovative or creative. The lesser the knowledge about existing relationships between the requirements and the form to satisfy those requirements, the more a design problem tends towards creative design. Thus, for non-routine design, a knowledge-lean methodology is necessary. Natural evolution has produced a large variety of forms well-suited to their environment suggesting that the use of an evolutionary approach could provide meaningful design solutions in a non-routine design environment.

This work investigates the possibilities of using an evolutionary approach based on a genotype which represents design grammar rules for instructions on locating appropriate building blocks. A decomposition/aggregation hierarchical organization of the design object is used to overcome combinatorial problems and to maximize parallelism in implementation.

2. An Evolutionary Model Of Design

2.1 Genetic Algorithms

Genetic algorithms (GAs) are a class of algorithms, based on the adaptive process of natural evolution, employing a general uniform knowledge-lean methodology.[2,3] While GAs have been used to solve mainly optimization, learning and control problems,[4,5] there has been very some research and applications in design.[6-13] and in the generation of creative Art forms.[14] The trend which has developed in GA applications is to encode the genotype as a string of, usually binary, characters representing a one-to-one correspondence to a property in the phenotype. This differs from nature where the genotype encodes instructions for locating building blocks of living form.

2.2 Evolution And Design

The evolutionary approach is basically a generate and test approach which corresponds well to the procedures for design synthesis and evaluation in the design process. The specific characteristics of the approach are:

- a large pool or population of members (e.g. design solutions);
- members are selected for 'survival' using a biased random selection mechanism based on their 'fitness', i.e. relation to a fitness function;
- new members are generated from the existing ones using evolutionary mechanisms as crossover, inversion, 'gene splicing' and mutation.

The advantages of the evolutionary approach are:

- diverse sections of the state space can be investigated thus enhancing the possibility of discovering a variety of potential solutions;
- a probabilistic selection method directs the random generative process to produce meaningful and satisfactory solutions.

2.2.1 Growth of Form

Whereas the process of generation of form in living systems involves the placement of different kinds of protein in particular locations, the process of generation of form in design involves the placement of units of different kinds of material in particular locations. We may describe an object, at a lower level of abstraction, as a composition of material units (cells or building blocks), where the type and scale of such units are chosen depending on the context and suitability for the level of abstraction required. An object can then be 'grown' by locating a required number of such units, one at a time in sequence. The form produced will depend on the form of the unit material and the location procedures, i.e. rules of growth, used, as in crystal growth.

The genotype for a homogeneous object is thus the sequence of coded instructions for selecting and locating material units, analogous to the DNA string in natural evolution. When this code is interpreted and executed, the phenotype, i.e. the object (or rather its representation), will be generated. A general model of form growth can be proposed as:

For given total units of material required
SELECT a unit of material, Mm
LOCATE unit of material, Mm relative to other units

A gene in such a model becomes (Ot, Mm, L(Mm)), where L(Mm) is the instruction for locating the unit of material Mm relative to the generated object at

each step, Ot. Initially Ot is a single unit. The genotype is a sequence of such genes. Where a homogenous object is considered, the material identification is constant and the gene is basically a sequence of location operators. Obviously, such a model is computationally infeasible in general and a more computationally feasible approach is required.

2.2.2 Elements, Components and Assemblies

An object may be simple or complex. A simple object is termed an *element* and by definition is homogeneous otherwise it can be decomposed into separate homogeneous elements An element is thus a composition of material units. A complex object is termed an *assembly* and is composed of objects termed *components*. Recursively, components may be assemblies or elements. A formal description is as follows:

O :-	A | E	*An object is an assembly or an element*
A :-	(C, R(C))	*An assembly is a set of components and* C :- { (C_i} a *set of relationships among the components*
C_i :-	O	*ith component is an object*
E :-	(M_m, R(M_m))	*An element is a set of material units of type*
M_m :-	{ (M_{mp}}	M_m *and a set of relationships among the material units*
M_{mi} :-	ith unit of material M_m	

2.2.3 Evaluation of Design Solutions

The evaluation of designs is carried out by interpreting the generated design solution, the phenotype, and determining its performance according to a set of behavioural requirements (formulated from the design requirements). The performance of the object may be determined using formulae or rules, etc. or by users exercising judgment in the case of qualitative behaviours. Such judgments are subjective and personal and the designer takes full responsibility for such evaluations. Such subjective evaluation coupled with user interaction is the approach taken in the generation of *Biomorphs* [15] and creative Art forms.[2]

Since an object exhibits more than one behaviour, the evaluation of the fitness of the object is a multiobjective problem and hence will involve evaluation using concepts similar to Pareto optimization.[9,16-17] Constraints can be implemented through the use of penalty functions.

2.2.4 Design Through Hierarchical Decomposition / Aggregation

Simon[18] points out, that even though organisms are very complex, it is only possible for them to evolve if their structure is organised hierarchically. The above formulation allows for the generation of objects through the recursive generation

of its components until a level is reached where the generation becomes one of generating an element. Such an approach assumes a formulation of the decomposition structure of an object.

There are basically two approaches. The first is a top-down approach, used by Cramer,[19] Dasgupta and McGregor[20] and also in Genetic Programming.[21,22] The second is a bottom-up multi-level approach where, at each level, a component is generated from a combination of components from the level immediately below. At each level, an initial population is generated and then evolved over a number of generations until a satisfactory population of objects at that level is obtained. Members of that population are then selected as suitable components for generating the initial population at the next level. The process is repeated for all levels, Figure 1.

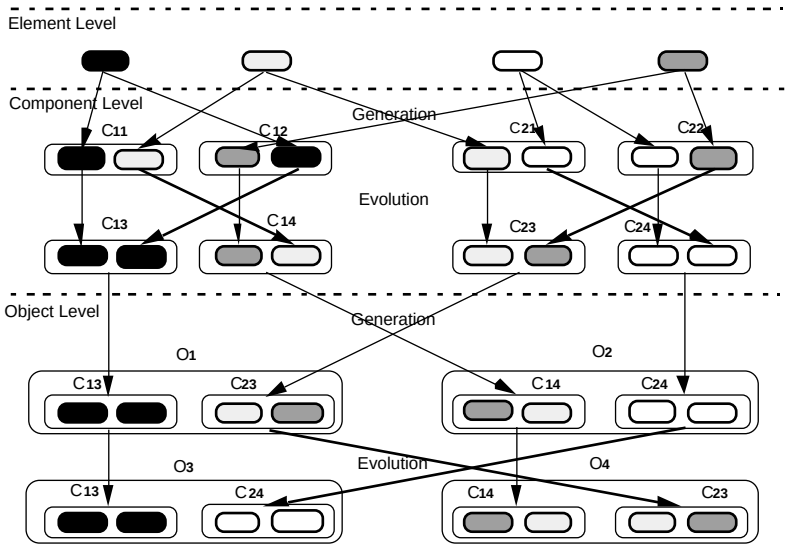


Figure 1 . Multi-Level Combination and Propagation

The advantages of a hierarchical approach are that only those factors relevant to the design of that component are considered and factors relevant to the relationships between components are treated at their assembly level. Instead of one long genotype consisting of a large number of low-level genes, the genotype is composed of a set of chromosomes relevant to their particular level. In addition to reducing the combinatorial problem substantially, parallelism is supported since all the different chromosomes (components) at a particular level can be generated in parallel. If the set of possible alternatives of component types is sufficiently large and varied, then many different combinations of members of different such sets are

possible, at the next level, with a good chance of satisfying the criteria and constraints at that level. Only when no such possible combination satisfies such criteria is there a need for some generation of new alternatives at the lower level.

In a flat model of form generation, a genotype will consist of a string of a very large number of basic genes. In a hierarchical model, there are a number of component chromosomes, at different levels, consisting of much shorter strings of genes which are the chromosomes at the next lower level. All in all, the total number of basic genes will be the same in the flat and hierarchical models.

2.3 Design Grammars

In order to generate a design solution a generative method, such as a design grammar, is required. A design grammar deals with a vocabulary of design elements and transformation on these elements and hence defines a design space.[13,23,24] While design grammars provide a syntactic generative capability, they lack the evaluative mechanisms for directing the generation towards meaningful solutions.

2.3.1 Recipes and Blueprints - Genotypes and Phenotypes

The aim of the design process, in an evolutionary approach, is the attainment of a set of instructions, a genotype (recipe), that when executed, yields a design description of a product, a phenotype(blueprint), whose interpreted behaviours satisfy a set of required behaviours, the fitness function. In this approach, a grammar rule is a gene, the plan (sequence of rules) is the genotype and the design solution is the phenotype.

The advantage of the use of recipes rather than blueprints as the genetic information is in the simplicity of the information since the generative rules are fewer in number than solution parameters and generally less complex. Moreover, small changes in such rules or their combinations can lead to large and unexpected changes in the design solutions, a desirable property for creative design.[13]

2.3.2 The Evolution of New Rules and Plans

There are basically two approaches in the generation of genotypes of design grammar rules, analogous to the Michigan and Pitt approaches in classifier systems.[5,25] The first approach, as taken by Gero et al.,[7] attempts to 'learn' new grammar rules. The second approach, which is taken here, is based on the premise that the grammar rules are fundamental operators, which cannot be decomposed or recomposed, that the particular grammar contains all required rules and that the aim of the design process is to find satisfactory sequences of such rules.

2.3.3 A General Model for An Evolutionary Approach to Design

The general model of design using an evolutionary approach may be stated as follows:

for all levels in the object hierarchy
  for all components at that level
    GENERATE initial population of members by synthesizing lower level units through random application of rules
    EVOLVE population until satisfactory

3. A House Design Example - Space Generation

The above concepts can be exemplified through the generation of 2-D plans for single-storey houses. Previous work demonstrated that a single-level approach was not able to converge towards satisfactory solutions mainly due to the interactions of the various factors of the fitness function required for the various elements.[9,26]

3.1 A House Spatial Hierarchy

A house can be considered to be composed of a number of zones, such as living zone, entertainment zone, bed zone, utility zone, etc. Each zone is composed of a number of rooms (or spaces), such as living room, dining room, bedroom, hall, bathroom, etc. Different houses are composed of different zones where each zone may be composed of different rooms. Each room is composed of a number of space units. Generally, in a design such as a house, the space unit will be constant. The scale (level of abstraction) of the space unit depends on the precision required in differences between various possible room sizes. The smaller the unit, the longer the genotype for a given size of room but the greater the shape alternatives.

3.2 Generation - The Design Grammar

In the above formulation, the generation of spaces, basically comes down to locating spatial component units for that level. At the room level, the component unit is a fundamental unit of space. At the zone level, the component unit is a room and at the house level the component unit is a zone.

The design grammar used here is based on the method for constructing polygonal shapes represented as closed loops of edge vectors.[27] The grammar is based on a single fundamental rule which states that any two polygons, Pi and Pj, may be joined through the conjunction of negative edge vectors, V_1 and V_2, (equal in magnitude and opposite in direction). The conjoining of these vectors results in an internal edge and a new polygon, Pk. The edge conjoining rule, R1 is:

Vm + Vn = Vmn = 0 ..... **R1**

Rule R1 is commutative and applies for all values of vector direction. This rule ensure that new cells are always added at the perimeter of the new resultant shape. This rule is shown diagrammatically in Figure 2.

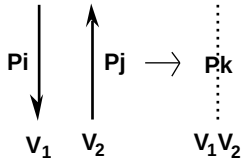


Figure 2. Edge Vector Rule for the Construction of 2-D Shapes

The fundamental conjoining rule can be specialized for different types of geometries. Orthogonal geometries are based on the following four vectors of unit length: W = (1, 90), N = (1, 0), E = (1, 270), S= (1, 180).

so that Rule 1 becomes:

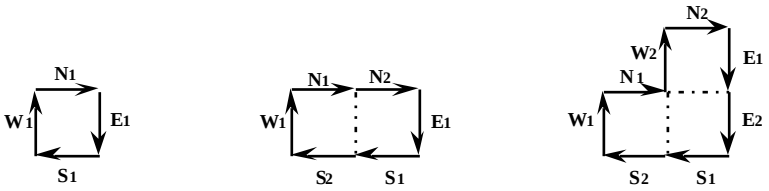
N + S = NS = 0 ..... **R1a**
E + W = EW = 0 ..... **R1b**

These two (sub)rules allow for the generation of all polyminoes. Orthogonal geometries will be used in this example without loss of generality. Other (sub)rules may be formed for other geometries.[27]

3.2.1 Genotype and Phenotype

A polygon is described by its sequence of edge vectors. A suffix is used to identify individual edges of the same vector type. Thus the square cell of Figure 3 is described as (W1, N1, E1, S1). The sequence of edge vectors for a shape is the phenotype providing the description of that shape's structure. The genotype for any generated polymino is the sequence of the two subshapes (polyminoes) used and the two edges joined. An example of the generation of a trimino is shown in Figure 3. Figure 3 shows a basic unit or cell, P1, which provides a starting point for the generation of polyminoes. Each generated shape is accompanied by its genotype and phenotype. The generation of these polyminoes occurs from a random selection of edges in the first shape conjoined with a random selection from equal and opposite edges in the second shape. At each step in the generation, the phenotype is reinterpreted to generate a new edge vector description and the conjoining (sub)rules applied. The genotype for the generated trimino is given as

(P2, P1, N2|S1). This can be expanded as ((P1, P1, E1|W1), P1, N2|S1). When the same units are used for generation, the unit can be omitted and the genotype represented as the sequence of edge vector conjoinings. That is P3(g) = (E1|W1, N2|S1).



P1(p) = (W1,N1,E1,S1)

P2(g) = (P1,P1,E1|W1)
P2(p) = (W1,N1,N2,E1,S1,S2)

P3(g) = (P2,P1,N2|S1)
P3(p) = (W1,N1,W2,N2,E1,E2,S1,S2)

Figure 3. Generation of a Trimino

The length of the genotype (and phenotype) depends on the size of the polymino to be generated, that is on the area of the polymino. This corresponds to required room sizes.

Once a population of different rooms is generated for each room type in a given zone, the zone can be generated through the conjoining of rooms in a progressive fashion. Because of the cell-type structure of the polygons, the conjoining may occur at any appropriate pair of cell edges. Therefore, a large number of possible zone forms can be generated from two rooms. An example of some possibilities arising from the conjoining of two polyminoes is given in Figure 4.

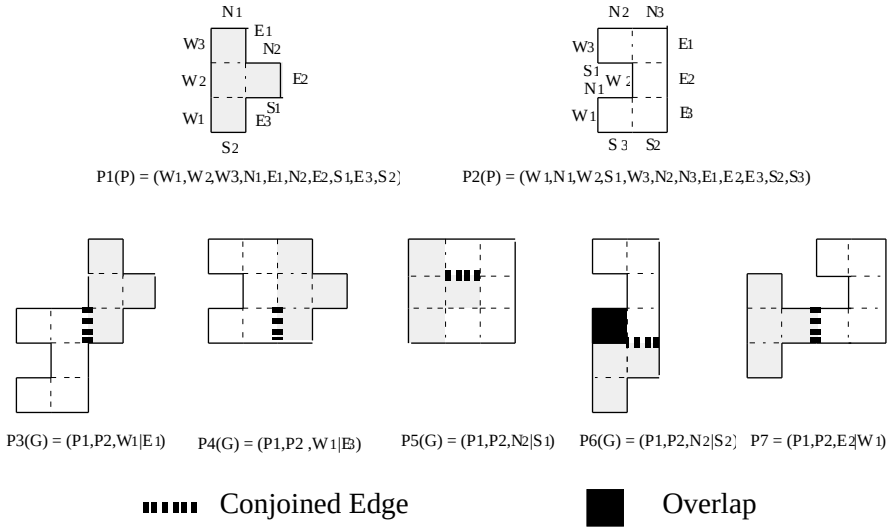


Figure 4. Some Examples of Conjoining Two Polyminoes

The two polyminoes, P1 and P2, represent instances of two different room types and the polyminoes resulting from the joining of the two rooms represent instances of a particular zone type. When one pair of edges are conjoined other edges may also be conjoined, e.g. P4, P5 and P6. In the case of overlap, as in P6, the resultant shape is discarded.

The same process used for generating zones is used to generate houses. The joining of different instances of different zone types generates different instances of houses.

3.2.2 Order of Selection

At the zone and house level, the order of selection of the units to be joined may influence the solution and its performance. However, there are problems with choosing a random order of room selections for every zone instance generation for the same population as crossover may lose some room types and include more than one of the same type. Thus for any population, the same order of room types must be kept. A given order may be chosen randomly or from an algorithm based on the number (and/or strength) of interconnections required in an interconnection matrix.

3.3 The Evolution Of House Designs

The above grammar can be used to generate initial populations for each level in the spatial hierarchy. Each such initial population is then evolved, as necessary, so that solutions are ‘adapted’ to design requirements.

3.3.1 The Evaluation Criteria - Fitness Functions

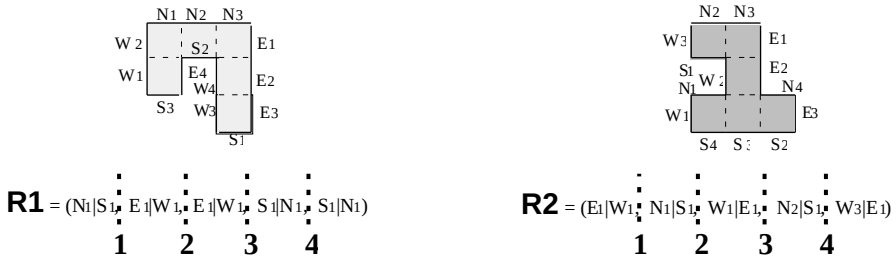
At each level, different fitness functions apply according to the requirements for that level. While the requirements for designs of houses involve many factors, many of which cannot be quantified or adequately formulated in a fitness function, some simple factors will be used initially to test the feasibility of the approach. For this example, the fitness function for rooms will consist of minimizing the perimeter to area ratio and the number of angles. This requirement tends to produce useful compact forms. For zones, the fitness function will consist of minimizing a sum of adjacency requirements between rooms reflecting functional requirements. At the house level, the fitness function will consist of minimizing a sum of adjacency requirements between rooms in one zone and rooms in other zones. This has the tendency to select those arrangement of zones where adjacency interrelations are required between rooms of different zones. In addition to these quantitative assessments, qualitative assessments will be made subjectively and interactively by a user/designer.

Although the above criteria have been described in terms of optimizing functions, the aim is not to produce global optimum solutions but rather to direct the evolutionary process to produce populations of good solutions either as components for higher levels or at the final level itself. So that, even though the global optimum solution for the shape of a room using the above criteria, may be known, this may not be the optimum solution at the zone and house levels. By selecting other non-optimal but good solutions, according to the given criteria, good unexpected results may be achieved for the overall design.

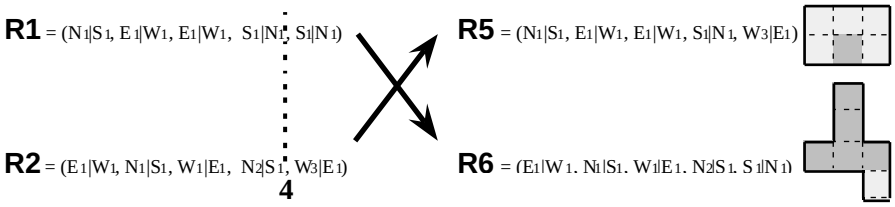
Other factors are required in more realistic design contexts. For example, a house needs to meet site requirements, both in terms of size and orientation for view or climate. Such factors can be formulated as constraints at various levels. These constraints can be handled explicitly as survival factors or as penalty functions. That is, a solution which does not meet a constraint is eliminated or, alternatively, a penalty can be added to the fitness of the solution according to the degree of violation. It is argued that with a sufficiently large population of room and zone alternatives such constraints can be met. If not, then redesign, i.e. new zone and/or room forms, must be produced.

3.3.2 Propagation - Crossover

Simple crossover is used for the production of 'child' members during the evolution process. Looking first at the room level to see the effect of such a crossover process, crossover can occur at any of the four sites as shown in Figure 5(a) with two results as shown in Figure 5(b). Since we are always dealing with cells of the same space unit, the cell identification in the genotype representation has been omitted for simplicity.



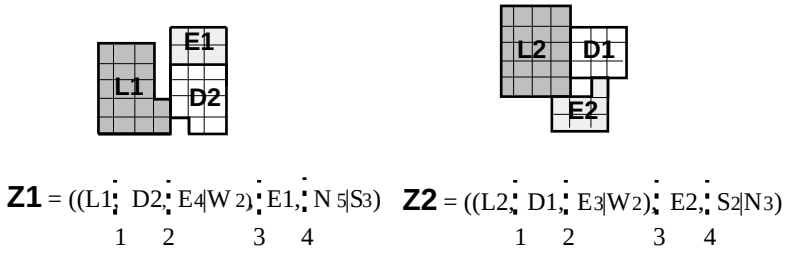
(a)



(b)

Figure 5. Crossover at Room Level; (a) initial rooms R1 and R2 generated from unit square cell U1, (b) crossover at site 4

At the zone level, crossover occurs as shown in Figure 6. Two initial instances of living zones, Z1 and Z2 are shown in Figure 6(a). Each zone has one instance of each of living room, dining room and entrance. Figure 6(b) shows crossover for one of the four possible sites. A similar process is followed at the house level.



(a)

SITE 2 - CROSSOVER

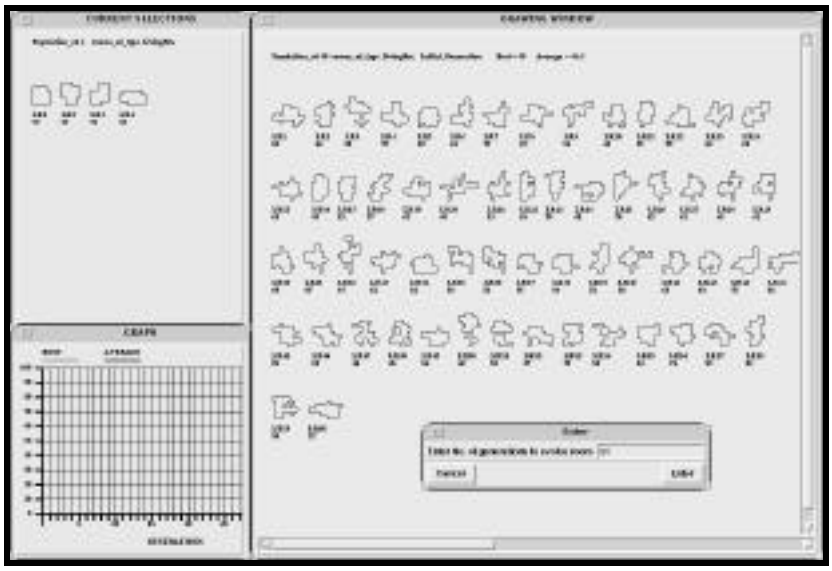
(b)

Figure 6. Examples of Zone Crossover; (a) rooms and initial zones, Z1 and Z2, (b) crossover at Site 2

4. Implementation And Results

A computer program written in C++ and Tcl-Tk under the Sun Solaris environment is under implementation. Currently, only the simple criteria described previously have been implemented. Each evolution run, for all levels, tends to converge fairly quickly to some dominant solution. Rather than use a mutation operator to break out of such convergence , it was found that a more efficient strategy was to generate multiple runs with different initial randomly generated populations. This produces a variety of gene pools thus covering a more diverse area of the possible design space. Moreover, such runs can be generated in parallel. Users can nominate the population size, number of generations for each run and select rooms, zones and houses from any generation in any run as suitable for final room, zone or house populations. These selections are made as solutions appear which find favour in users, based perhaps on factors not included in the fitness function. Such selections may therefore not be optimal according to the given fitness function.

Results are shown in the following figures, Figures 7 to 10 for room, zone and house solutions.



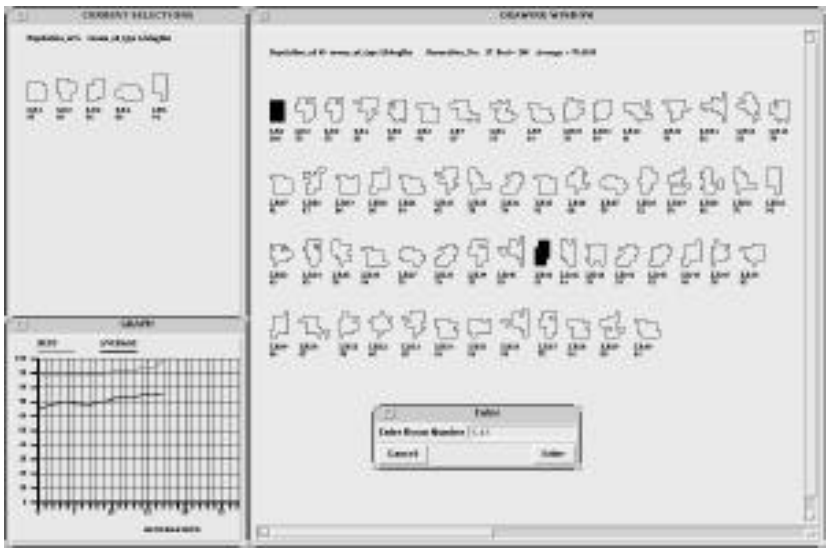
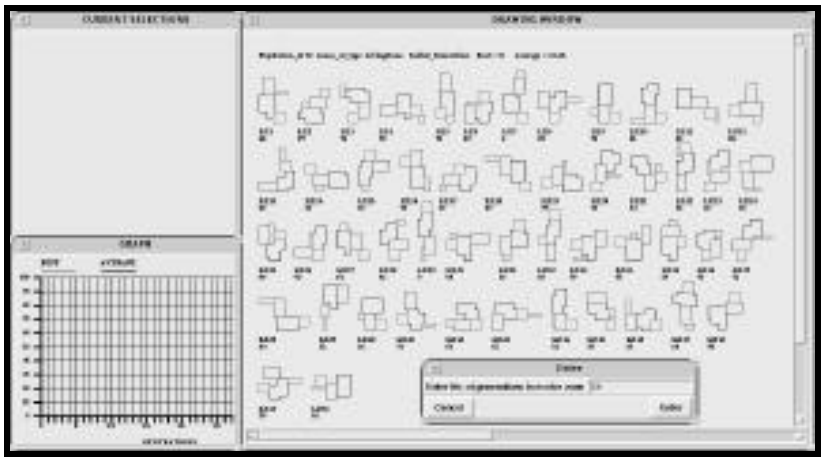
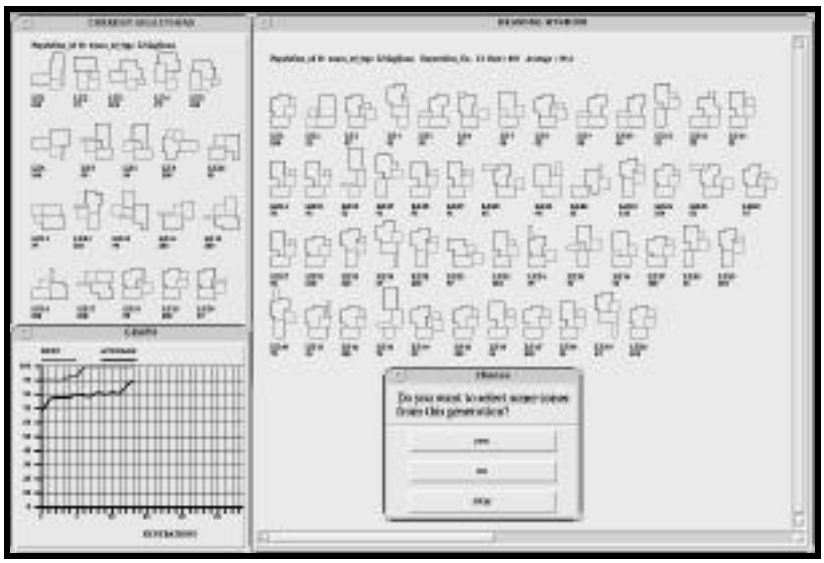


Figure 7. Results of Living Room Generation; (a) initial generation, (b) 17th generation



(a)



(b)

Figure 8. Results of Living Zone Generation; (a) initial Living Zone population, (b) evolved population

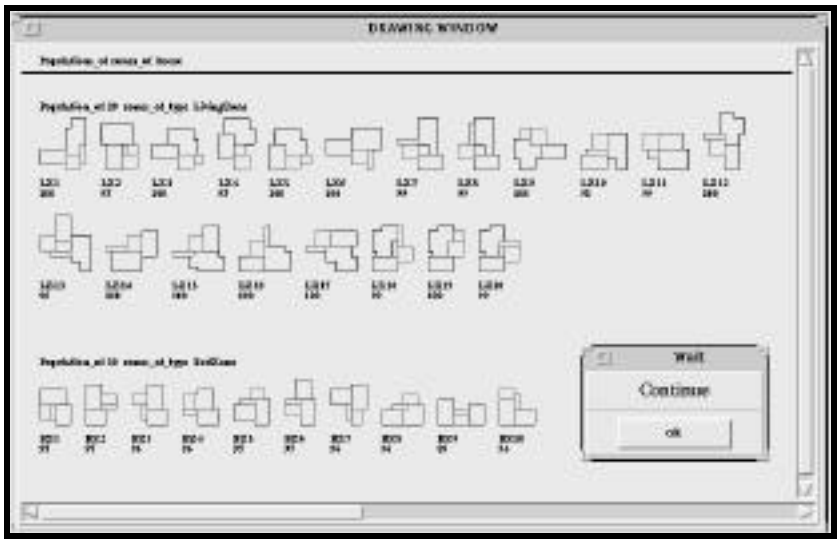


Figure 9. Results of Bed and Living Zones Generation

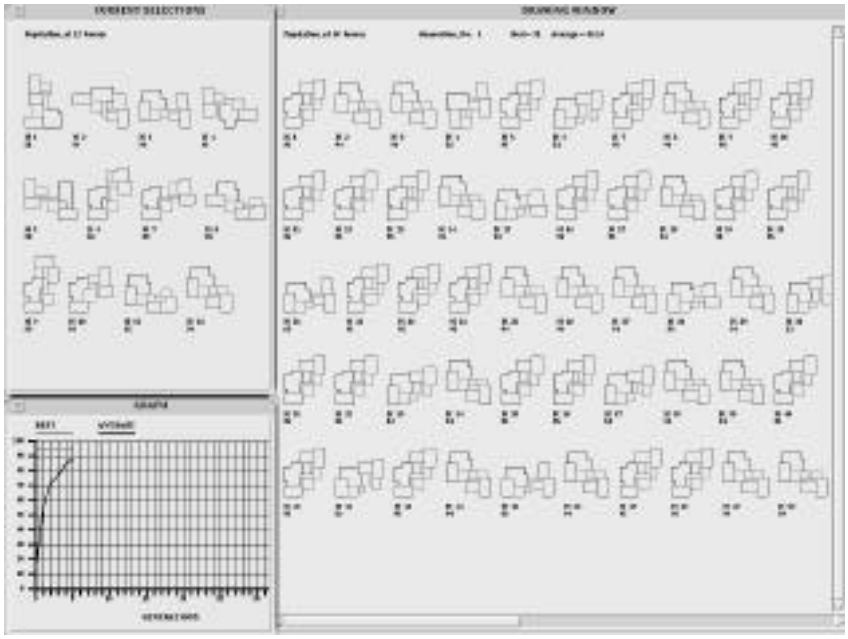


Figure 10. Results of House Generation.

Figure 7 shows the results of the 8th run of the generation of Living Rooms. Figure 7(a) shows 4 shapes selected from 7 previous runs together with an initial random population of 60. Figure 7(b) shows the 17th generation of the evolution of this population. A fifth shape was selected at the 14th generation and two more shapes are being selected. The upper line in the graph shows the evolution of the best solution while the lower line shows the evolution of the population average. All in all for this example, a total of 10 shapes were selected from 13 runs. The maximum number of generations before convergence for a run was 50 and the minimum 19 with an average of 32.

Other rooms were generated in a similar way. The room areas generated were: (a) Living Zone: Living Room 24; Dining Room 15; Kitchen 9; Entrance 4; (b) Bedroom Zone: Master Bedroom 15; Bedroom 12; Bathroom 6; Hall 3. Figure 8 shows the results of the Living Zone generation. Figure 8(a) shows an initial population of 50 Living Zones at run 1, randomly generated by selecting rooms from the final selections for the Living Room, Dining Room, Kitchen and Entrance. Figure 8(b) shows the 13th generation of the final run, run 7. Twenty Living Zones have been selected by the user.

Figure 9 shows the set of Bedroom and Living Zones selected. Figure 10 shows the final set of houses generated in this example. All in all 7 runs were carried out for a total of 12 suitable house plans.

The total area of this house type is 88 sq units, corresponding to a genotype of length 87 in a single genotype. The example of Jo,[26] showed that no satisfactory convergence was obtained with a single genotype of this length. The size of the population and the number of generations and runs required to generate a satisfactory number of members depends on the genotype length. The longest genotype was of length 23, for the Living Room. The addition of more zones and rooms presents no problem for the hierarchical approach used here although it would present extra combinatorial problems for the single genotype approach.

5. Summary

This work has presented concepts for a general evolutionary approach to the generation of design solutions based on the growth of cells in a hierarchical organization. The main advantage of a hierarchical approach is that genotypes are shorter and the fitness functions relates only to the requirements for that component(element or assembly). In addition to reducing combinatorial problems, parallelism is supported. The critical element becomes the genotype of the largest component genotype rather than the total length of a single-level genotype.

While the various fitness functions at the different levels of the component hierarchy have involved optimization criteria, the goal of the approach is not optimization of these factors per se but rather their use as a driving force in the generation of satisfactory form.

As an alternative to mutation in the evolution of populations, the use of multiple runs with new randomly generated initial populations was used. This also has the advantage of allowing parallel processing.

While the example presented is based on the generation of 2-D plans through the synthesis of a fundamental 2-D space unit, the approach can be generalized to the synthesis of any material cells. Although the example was based on orthogonal geometry, the method for growth is general for any polygonal geometry and may be extended to polyhedral geometry.

Further work will involve the inclusion of more realistic criteria and constraints as well as investigating the need for the recursive generation of new lower level components when no satisfactory assembly can be generated. Efficiency issues need to be investigated with respect to the type of crossover and selection mechanism.

Acknowledgments
This work is partially supported by the Australian Research Council.

References

1. Rosenman, M. A. and Gero, J. S. (1994). The what, the how, and the why in design, *Applied Artificial Intelligence*, **8**(2):199-218.
2. Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.
3. Beasley, D., Bull, D. R. and Martin R. R. (1993). An overview of genetic algorithms: Part 1, fundamentals, *University Computing*, **15**(2):58-69.
4. Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Reading, Mass.
5. Grefenstette, J. J. and Baker, J. E. (1989). How genetic algorithms work; a critical look at implicit parallelism, *in* J. D. Schaffer (ed.), *Proc. of the Third Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp.20-27.
6. Louis, S. and Rawlings, G. J. (1991). Designer genetic algorithms: genetic algorithms in structure design, *in* R. K. Belew and L. B. Booker (eds), *Proc. Fourth Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, pp.53-60.
7. Woodbury, R. F. (1993). A genetic approach to creative design, *in* J. S. Gero and M. L. Maher (eds), *Modeling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum, Hillsdale, NJ, pp.211-232.
8. Gero, J. S., Louis, S. J. and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement, *AIEDAM*, **8**(3):83-94.
9. Bentley, P. J. and Wakefield, J. P. (1995). The table: an illustration of evolutionary design using genetic algorithms, *1st IEE/IEEEConference on Genetic Algorithms in Engineering Systems: Innovations and Applications (Galesin '95)*, pp.412-418.
10. Jo, J. H. and Gero, J. S. (1995). A genetic search approach to space layout planning,*Architectural Science Review*, **38**(1):37-46.
11. Michalewicz, Z., Dasgupta, D., Le Riche, R. G. and Schoenaur, M. (1996). Evolutionary algorithms for constrained engineering problems, *Computers and Industrial Engineering Journal, Special Issue on Genetic Algorithms and Industrial Engineering*, **30**(2):(to appear).
12. Schnier, T. and Gero, J. S. (1995). Learning representations for evolutionary computation, *in* X. Yao (ed.), *AI'95 Eighth Australian Joint Conference on Artificial Intelligence*, World Scientific, Singapore, pp.387-394.
13. Rosenman, M. A. (1996). A growth model for form generation using a hierarchical evolutionary approach, *Microcomputers in Civil engineering, Special Issue on Evolutionary Systems in Design*, **11**:161-172.
14. Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers*, Academic Press, London.
15. Dawkins, R. (1986). *The Blind Watchmaker*, Penguin Books.
16. Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization, *Proc. of the First IEEE Conf. on*

Evolutionary Computation (ICEC '94), Vol1, IEEE World Congress on Computational Intelligence, Piscataway, NJ: IEEE Service Center, pp.82-87.
17. Osyczka, A. and Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm, *Structural Optimization*, **10**(2):94-99.
18. Simon, H.A. (1969). *The Sciences of the Artificial*, MIT Press, Cambridge, Mass.
19. Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs, *Proc. of an Int. Conf. on Genetic Algorithms and their Application*, pp.183-187.
20. Dasgupta, D. and McGregor, D. R. (1993). sGA: A structured genetic algorithm, *Technical Report No. IKBS-11-93*, Dept. of Computer Science, University of Strathclyde, UK.
21. Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass.
22. Rosca, J.P. and Ballard, D.H. (1994). Hierarchical self-organization in genetic programming, *Proc. of the Eleventh Int. Conf. on Machine Learning*, Morgan-Kaufmann, San Mateo, CA, pp.252-258.
23. Stiny, G. and Mitchell, W. (1978). The Palladian Grammar, *Environment and Planning B*, **5**:5-18.
24. Stiny, G. (1980). Introduction to shape and shape grammars, *Environment and Planning B*, **7:**343-351.
25. Wilson, S. W. and Goldberg, D. E. (1989). A critical review of classifier systems, *in* J. D. Schaffer (ed.), *Proc. of the Third Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp.244-255.
26. Jo, J. H. (1993). *A Computational Design Process Model using a Genetic Evolution Approach*, PhD Thesis, Department of Architecural and Design Science, University of Sydney, (unpublished).
27. Rosenman, M. A. (1995). An edge vector representation for the construction of 2-dimensional shapes, *Environment and Planning B:Planning and Design*, **22**:191-212.