An Exploration-Based Evolutionary Model of a Generative Design Process.

John S. Gero and Vladimir A. Kazakov
Key Centre of Design Computing,
Department of Architectural and Design Science,
The University of Sydney, NSW 2006 Australia.
e-mail:{john,kaz}@arch.su.edu.au

Abstract

An exploration-based generative model of design is proposed. Shape grammars are used as a framework for the knowledge representation within this model. The routine design problem over the state space generated using a fixed shape grammar and the creative design problem over the extended state space generated by using an extended shape grammar - the family of shape grammars which includes the original one - are formulated. Two evolutionary algorithms which produce superior designs and shape grammars are presented. In the first method exploration and search processes occur simultaneously, whilst in the second one they are explicitly separated.

1 Introduction

Design is that activity which is primarily concerned with changing the physical world. Design can be characterized, initially, as an intentional, purposeful activity concerned with developing suggestions for changing the physical world. This implies that design can be modelled as being goal-oriented, but unlike problem solving which is also a goal-oriented activity, the goals in designing are not always fixed and determining them is part of designing. Design is also a constrained activity, ie there are constraints which limit the designer's ability to achieve goals. Some of these constraints are based on the behaviour of the physical world, others are based on the designer's perceptions and interpretations of design situations, and others are implicit in the representations and processes utilized.

Design can be conceived of as a decision making activity. This implies that choices exist; the choice of structure design space, that is a set of decision variables and a model which describes valid designs; the choice of behaviour space (objectives) that describes the quality of design; and the choice of the particular design (point) within the particular structure design space. *Search* is used in finding an improved design (in terms of pre-defined objectives) within a given structure design space. The process of determining of the space within which to search and/or objectives to direct the search is called *exploration* (Gero, 1993). In addition to exploration and search design

[0]To appear in "Microcomputers in Civil Engineering"

involves learning where *learning* implies restructuring of the knowledge used. It is further suggested that exploration is an important idea in design creativity. *Creativity*, the ability to produce novel solutions which are qualitatively better than previous solutions, is a critical aspect of designing. Without such creativity designing only involves incremental improvement on existing designs. It is in developing support for designers to be more creative that one of the future directions of design research lies.

2 Search in Design

Search, as a computational process, is defined as that process which finds new points which are better that previous ones. It is assumed that they belong to the same space and that they are being compared using the same objectives. It can be any space - the state space of particular design style, or state space of all design styles which possess some distinguishing feature, or all possible designs or even the space of objectives. The nature of the search space or what kind of objectives is used is of no significance. The only conditions which need to be met are that the search space which contains all the points is defined a priori as well as the objectives used to compare these points.

How does search in structure design space contribute to design? In that aspect of designing when all the variables (at some level of granularity) have been produced and the relationships between them are known (that is, the structure design space and the objectives are defined) then search is a most appropriate process. Whilst search in structure design space does not provide an adequate paradigm for design as a whole, such situations do occur at the base of many design activities. Search does form the basis of all classes of design and is a comprehensive model of so-called "routine design". Thus, in a sense, search is a foundational, necessary but not sufficient, process for any computational model of design. What search in structure design space fails to deal with is the process of producing variables and determining relationships between them (that is, changing the structure design space). It does not address the process of producing new objectives (behaviour variables).

3 Exploration in Design

Exploration in design can be characterized as a process which creates new design state spaces or modifies existing design state spaces. New state spaces are rarely created de novo in design rather existing design state spaces are modified. The result of exploring a design state space is an altered state space.

Exploration maps onto the concept of creative design (Gero, 1993). Any of the spaces associated with behaviour (objective) or structure (decision variables and model) could be changed although, in general, in design it is the structure space that is changed.

A creative design process can be very complicated - the exploration in structure and behaviour spaces can proceed simultaneously or be separated from each other or they may produce feedback for each other. Similarly search and exploration may occur simultaneously or in sequential stages.

Assume that the formulation of the design problem does not depend on time or some external processes. This stationary problem corresponds to a picture of a "closed" world. In this case the search in a high level space can be used to mimic exploration when viewed from a low level one. For instance, if there exists a search

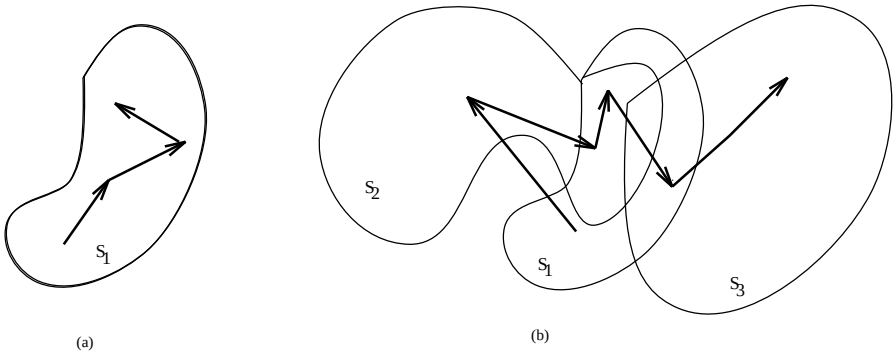

Figure 1: (a) search in a single state space S_1; (b) search in an extended state space - a union of state spaces $S_{ext} = S_1 \cup S_2 \cup S_3$.

algorithm which operates in a space that is a union of a number of different state spaces (that is, in some subspace of the space of all structure state spaces) then search in the union of state spaces looks like an exploration process to an observer in any one of the constituent state spaces, Fig. 1. Moving from one constituent state space to another is, effectively, converting one formulation of a design problem into another. A computational model of exploration which is based on such a meta-search is presented in the paper. Similarly, if there is a search method which operates in the union of a number of behaviour spaces it can also be interpreted as exploration.

If the world is open or more strictly if the structure and/or behaviour spaces are time or some external process dependent due, for instance, to human intervention, then we get a different mathematical problem.

In this paper just one of the possible formulations of a creative design problem is considered. We do this by formulating the problem as exploration in an extended structure design state space under fixed behaviour. The extended state space is defined as a space which contains the original one. In the simplest case it may be just the union of the potential structure design state spaces. Evolution in behaviour space as well as nonstationary evolution in an "open world" model are left beyond our consideration.

The remainder of this paper commences by formalizing (routine) design using shape grammars as the knowledge representation formalism. It then goes on to develop an extended formulation of design where creative design is modeled within a closed world as a form of exploration which, here maps, onto meta-search. An example is presented and the results derived from it are discussed.

4 Shape Grammars and Design

Shape grammars (Stiny, 1980) have been used to generate design for many systems (Stiny and Mitchell, 1978; Stiny and Mitchell, 1980; Koning and Eizenberg, 1981; Fleming, 1987; Knight, 1986) . They consist of a finite set of possible initial shapes and another finite set of shape transformation rules which are applied recursively. A particular design within the given grammar is represented by some choice of the initial shape and some finite sequence of transformation rules. As soon as the shape grammar is defined the state space of all designs which can be generated using it is also implicitly defined. The design problem can be formulated as an optimization problem if it has a

quantifiable objective function and a set of design constraints. Within the framework of a shape grammar it can be formalized as the following optimization problem with discrete time and integer control

$$F(x(N)) \to \min, \tag{1}$$

subject to constraints

$$x(i+1) = f(u(i), x(i)), \quad u(i) \in V, \quad i = 0, \ldots, N-1, \quad x(0) \in X^0, \tag{2}$$

and

$$g(x(N)) \geq 0. \tag{3}$$

Where x is a state vector that uniquely characterizes the particular shape, F is a vector of objectives, the control u belongs to a finite set of m different shape transformation rules V, function $f(u, x)$ represents these rules, X^0 is a set of initial shapes and functions g represents the constraints on the design. The shape grammar G is defined as

$$G \equiv \{X^0, V, f(u, x)\}. \tag{4}$$

It is fixed if sets X^0, $V \equiv [1, \ldots, m]$ are given and transformations $f(u, x)$ are fixed. We denote the space of all shapes that can be generated using the grammar G as S.

The solution to the problem (1)-(4) $\{x(0), u(0), \ldots, u(N-1)\}$ can be found using any optimization method (Bryson and Ho, 1969). For instance the simulated annealing method has been formulated in shape grammar's terms (Cagan and Mitchell, 1992). Similarly the genetic algorithm technique has been applied to it successfully (Gero et al., 1994).

5 Learning in Design

We can interpret the shape grammar G as existing (available) knowledge about the design space. The search for a solution to problem (1) - (4) occurs in the predefined sub-space S of the space of all possible shapes. It can be interpreted as a model of a routine design process (Gero, 1993). If we want to simulate non-routine or creative design we should come up with a model which utilizes the additional knowledge which is obtained during the search process. The effect of this is that the grammar itself could be changed based upon the intermediate results of the search. Thus, there is a restructuring of the knowledge used which corresponds to learning a new shape grammar. This newly learned grammar has properties which produce improved designs in terms of the defined objectives. An important aspect of creative design in this case involves this learning of new grammars.

6 The Extended Design Problem

Let us write down formally a model of creative design in a "closed" world as an extension of the routine design problem. Assume there exists some parameterization of the shape grammars

$$G(\phi) = \{X^0_\phi, V_\phi, f_\phi(u, x)\} \tag{5}$$

where ϕ is some functional parameter. Then the following extended problem can be stated

$$F(x(N)) \to \min, \tag{6}$$

subject to constraints

$$x(i+1) = f_\phi(u(i), x(i)), \quad u(i) \in V_\phi, \quad i = 0, \ldots, N-1, \quad x(0) \in X_\phi^0, \tag{7}$$

and (3). The control here is $\{\phi, x_0, u(0), \ldots, u(N-1)\}$. The search space $\hat{S} \equiv \cup_{\forall\phi} S_\phi$ is the union of all feasible grammars.

We can use any computational method to solve the extended problem (3), (5) and (6). The iterative method to solve it has the following structure :

Algorithm A

(0). Set counter of iteration $k = 0$. Take some $\{\phi^0, x(0)^0, u(0)^0, \ldots, u(N-1)^0\}$.

(1) Calculate $F(x(N)^k)$. Set $k = k+1$. Calculate new $\{\phi^{k+1}, u(0)^{k+1}, \ldots, u(N-1)^{k+1}\}$ based on the value of $F(x(N)^k)$.

(2) Repeat step (1) until the stop conditions are met.

The formulas which are used to calculate the new $\{\phi^{k+1}, u(0)^{k+1}, \ldots, u(N-1)^{k+1}\}$ in step (1) of the algorithm depend in the method chosen to solve the problem (3), (5) and (6). A set of l points $\{\phi_i^k, u(0)_i^k, \ldots, u(N-1)_i^k\}$ $i = 1, l$ instead of just one point can be used in every iteration.

The trial points $\{\phi_i^k, x(0)_i^k, u(0)_i^k, \ldots, u(N-1)_i^k\}$ $i = 1, l$ on k-th iteration during a search will be located in different (in general) sets of subspaces $S_{\phi_i^k}$ and these subspaces will have different populations. Presumably the search process will use the information obtained about the current spaces $S_{\phi_i^k}$ to select a new set of grammars $S_{\phi_i^{k+1}}$ on the $k+1$ iteration. Two processes - search for optimal design x^* and a search for optimal grammar $G(\phi^*)$ - occur simultaneously. In terms of (Gero, 1993) the proposed algorithm consists of a mixture of the search and exploration processes which can not be separated explicitly.

In order to illustrate how these algorithms work we assume that parameter ϕ can only take the finite set of values $\phi = \{\phi_1, \phi_2, \ldots, \phi_p\}$. The subsets S_{ϕ_i} can partially overlap each other.

The search process for Algorithm A is shown schematically in Fig. 2 where more than one grammar is searched in parallel. Assuming that the number of trial points is constant for each iteration then the distribution of population on $S_{\phi_{\alpha_i}}$ changes automatically as the algorithm proceeds. Some of the sets $S_{\phi_{\alpha_i}}$ can lose population and drop out of the current set of $S_{\phi_{\alpha_i}}$ and others may gather population and join it.

7 Separated Extended Design Problem

Let us define

$$F(\phi) = F(x^*(N)),$$

where x^* is a solution of the problem (1)- (3) for a given grammar $G(\phi)$. Then the problem (3), (5) and (6) can be rewritten in the following equivalent form

$$F(\phi) \to \min_\phi \tag{8}$$

$$F(\phi) = argmin_{(x,u)\in G(\phi)} F(x(N))$$

as two subproblems: the high-level problem (7) of finding an optimal grammar $G(\phi^*)$ where the control is ϕ and the low-level problem of finding the optimal design within the fixed grammar $G(\phi)$ where control is $\{x(0), u(0), \ldots, u(N-1)\}$.

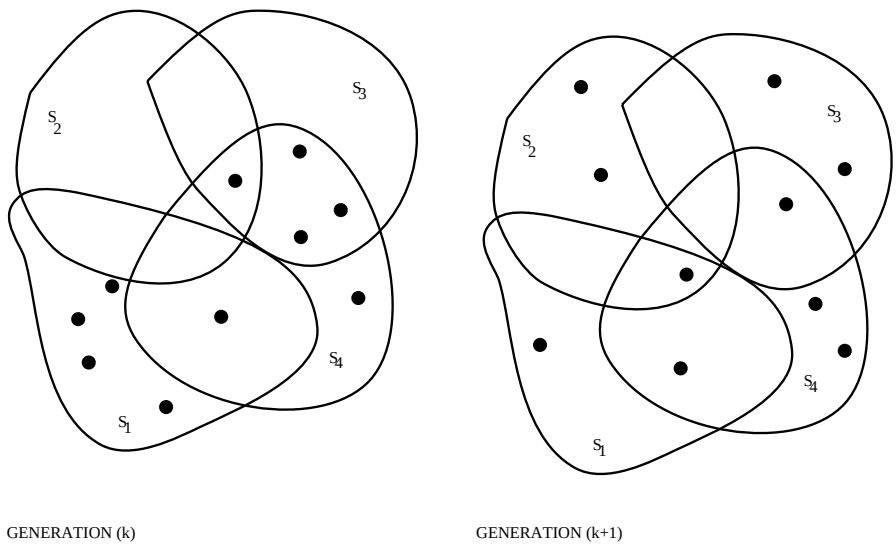

Figure 2: The search process for Algorithm A. The total population of all current sets, S_i, is constant in each generation.

If we separate the extended problem into the high-level subproblem (8) and a low-level one (1),(2), (3) and (4) then the solution process has the following structure :

Algorithm B
(0) Take some ϕ^0. Set counter of iteration $k = 0$.
(1) Find the solution $x^*(\phi^k)$ of the low-level problem (1) - (4) for given grammar $G(\phi^k)$. Calculate $F(\phi^k) = F(x^*(\phi^k))$
(2) Set $k = k + 1$. Calculate ϕ^{k+1} based on the value of $F(\phi^k)$.
(3) Repeat steps (1) - (2) until stop conditions are met.

Again if we use different methods to solve the high-level problem of finding the optimal grammar then the formulas for computation of ϕ^{k+1} are different. We can use not one but a set of l_1 trial points in every iteration $\{\phi_1^k, \ldots, \phi_{l_1}^k\}$.

Step (2) of Algorithm B contains some iteration process which produces the solution of the design problem (1) - (4) within given current grammar $G(\phi^k)$.

In terms of (Gero, 1993) this algorithm can be interpreted as a repetition of the routine search and exploration steps based on the search's results.

Both the low and high level sub-problems can be solved using any standard method (for instance (Cagan and Mitchell, 1992) or (Gero et al., 1994)). Similar to traditional implementations (Bertsekas, 1982) we can execute a few iterations to find an approximate solution to the low-level problem (1) in Algorithm B.

Algorithm B has a fixed population for each of the current set of $S_{\phi_i^k}$. When step (1) of the algorithm, the search for a solution of the low-level problem (1) - (4), occurs each of the trial points moves inside the same set $S_{\phi_i^k}$ where it was at the beginning of step (1), Fig. 3 Then at step (2) the set of current grammars is replaced with another one with the same constant populations, Fig. 4, and the search on this new set proceeds .

One reason for introducing Algorithm B is to reduce the effective size of the search

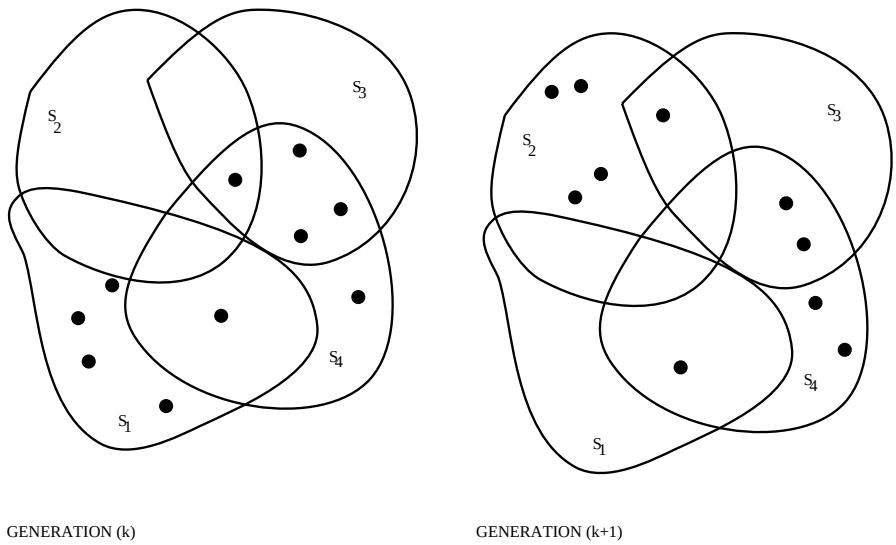

Figure 3: The search process at step (1) of Algorithm B. The trial points are moved within the same set of current sets, S_i. k_1 here is a an iteration counter of the solution of the low-level design problem within fixed set of grammars, G_i.

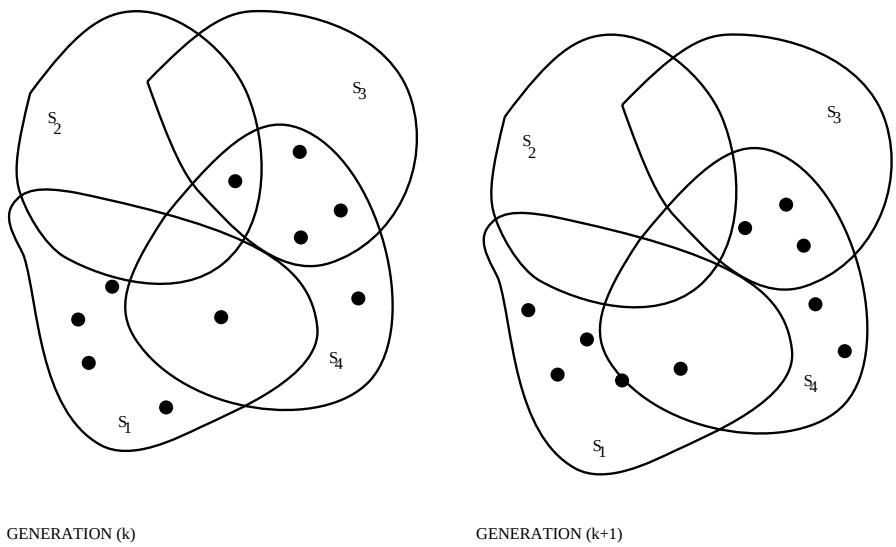

Figure 4: The search process at step (2) of Algorithm B. The number of current grammars, G_i, for all generations is constant. The populations of every of the sets, S_i, is constant also.

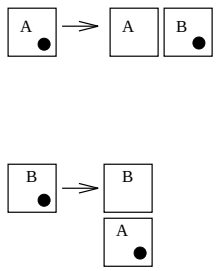

Figure 5: Two of the transformation rules used in the example.

space. Intuitively it seems natural that in many cases this reduces the computational cost and makes the search of extended state space more efficient. It seems also to be closer than Algorithm A to the intuitive picture of a design process where a designer tends to investigate the current state space prior to attempting to explore further.

8 Example

As an example we take the problem of beam section design (Gero, 1993). A beam is a component of an engineering structure and it transmits load by bending. In addition to its bending behaviour its stiffness behaviour is of interest and when the beam is exposed its total exposed area is also of interest. The stiffness can be directly related to the moment of inertia of the beam's cross-section and the total area can also be directly related to the perimeter of its cross-section when the beam has a uniform cross-section along its length. Thus, geometric properties of the beam cross-section can be used to calculate the behaviours associated with moment of inertia and perimeter. Let us use a shape grammar which generates beam cross-sections by producing cells in a grid. Joining the elementary cells in this grid of cells after the shape grammar has been executed produces the beam cross-section. Here the state vector $x^k_{i,j}$ $k = 1, 2$ $i, j = \{0, \ldots, N\}$. $x^1_{i,j} = \{0, \ldots, W_{max}\}$ is the weight of an elementary square cell at grid point (i, j) ; W_{max} is the maximum weight of the elementary cell and 0 represents unfilled cell positions. The second component $x^2_{i,j} = \{0, 1\}$ represents the so-called marker of the elementary cell. The transformation rules can only be applied to it if $x^2_{i,j} = 1$. The shape grammar G_{ϕ_i} is a set of transformation rules of the following form:

$$f_{n_1,n_2,,k_1,k_2}(u, x) : if \quad \left\{ \begin{array}{ll} x^1_{i,j} & = n_1, \\ x^2_{i,j} & = 1, \end{array} \right\} \longrightarrow \left\{ \begin{array}{ll} x^1_{i+k_1,j+k_2} & = n_2, \\ x^2_{i,j} & = 0, \\ x^2_{i+k_1,j+k_2} & = 1. \end{array} \right.$$

Where $n_1, n_2 = \{0, \ldots, W_{max}\}$, $k_1, k_2 = \{0, 1\}$ are given. The choice of n_1, n_2, k_1, k_2 defines the rule. Two rules are shown in Fig. 5. We assume that the initial shape $x(0)$ is one of the feasible elementary cells with a marker. The particular grammar G_ϕ consists of a finite set of transformation rules $G_\phi \equiv \{f_{n^1_1,n^1_2,k^1_1,k^1_2}, f_{n^2_1,n^2_2,k^2_1,k^2_1}, \ldots, f_{n^p_1,n^p_2,k^p_1,k^p_2}\}$. Parameter ϕ here is a vector of variable length $p : \phi = \{(n^1_1, n^1_2, k^1_1, k^1_2), \ldots, (n^p_1, n^p_2, k^p_1, k^p_2)\}$
.

The objective vector function is 2-dimensional :

$$F = \{-moment \quad of \quad inertia; \quad +perimeter\} \longrightarrow \min$$

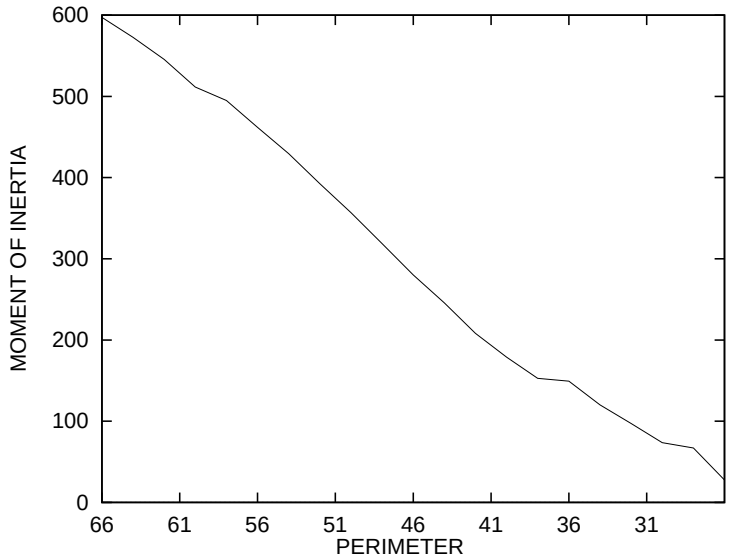

Figure 6: The Pareto-set.

We used the genetic algorithm to solve the extended problem (3), (5) and (6) as well as both subproblems in the Algorithm B. In order to handle the multiobjective optimization we use a simple Pareto-based ranking procedure. We take some positive number $U > 0$ and set counter $k = 1$. Then we find all points in the current population which belong to its Pareto-set and assign the fitness value for all of them equal $\frac{U}{k}$. Set $k = k + 1$. Then we consider the rest of the population without the points just found, single out the Pareto-set for this reduced population and set the fitness values for the members of this set equal $\frac{U}{k}$. Set $k = k+1$. The cycle continues until all the points are ranked. We assume that the maximal weight of the elementary cell is 7 . The number of all feasible rules equals $7 * 7 * 4 = 196$. The Pareto-set at the last generation for the problem is shown in Fig. 6. We measure the convergence of the sequence of the current grammars $\{G(\phi^k)\}$ to the optimal grammar $G(\phi^*)$ indirectly by measuring the convergence of the total number of rules used to generate the population at each generation and the number of those which are used from previous generations, Fig. 7. By plotting the density distribution of the rules at various generations,1 Fig. 8, one can see how the distribution of rules tends to concentrate in a small part of rule space and how the number of rules used converges.

The result of applying this model of exploration to this example is the production of a set of optimal grammars for the generation of beam cross-sections. (The optimal grammar is defined as a grammar, which can be used to generate an optimal design). The entire computational process can be treated as an evolution of the initial grammar or as a selection of a specialised grammar from all possible grammars. In either case the behaviour of the entire system can be seen as one of producing the optimal formulation for a specified design problem (where formulation means the selection of appropriate design variables and the determination of their relationship in order to completely describe a formalisable design problem).

Figure 8: Density distribution of the rules. The density of the rule i in generation k is defined as a ratio of the number of applications of this rule to the total number of rule applications for generation k.

9 Discussion

A number of interesting additional results emerge from the example. An examination of Figs. 7 and 8 shows that although there are 196 possible rules and that they are uniformly used in the seed generation, there is a shift in the pattern of their usage as the generations proceed with a preference of a small subset of the possible rules being increasingly used. This indicates the a primacy of certain rules over others in the production of the final solutions. This has ramifications for knowledge-based design systems which utilise generative grammars. The knowledge level can be recast as one of finding the optimal grammar.

This model of exploration can be considered as a "weak" model as it maps directly onto meta-search. It requires a knowledge of the parameterization of the variables used. However, it does form the basis of a stronger model of exploration where the state spaces cannot be defined a priori but where each new state is dependent on what has done before. This can occur when we have an "open" world such as when the human designer forms part of the fitness evaluation or where the fitness or the grammars are dependent on some other external variable.

The extended problem (3), (6) and (7) allows us to take into account the evolution of the shape grammars but it does not consider the possible dependence of the objective on the grammar or possible nonstationarity of the objective and grammar itself (that is, they depend on some external variable "time" t). If we assume these dependencies exist, then problem (8) takes the form

$$F = F(x(N), \phi(t), t) \rightarrow \max_{x \in G(\phi(t),t)} \tag{9}$$

and we get a richer model of exploration.

In these cases it may be that the design problem is less one of finding an optimal design and more one of exploring possible state spaces for interesting designs. Here, the use of an evolutionary model provides us with a useful computational engine.

10 Acknowledgments

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

References

Bertsekas, D. (1982). *Constrained Optimization and Lagrange Multiplier Method*, Academic Press, New York.

Bryson, A. and Ho, Y. (1969). *Applied Optimal Control*, Blaisdell, Mass.

Cagan, J. and Mitchell, W. (1992). Optimally directed shape generation by shape annealing, *Environment & Planning B.* **2**: 11–19.

Fleming, U. (1987). More than the sum of parts: the grammar of queen anne houses, *Environment & Planning B.* **14**: 323–350.

Gero, J. (1993). Towards a model of exploration in computer-aided design, *in* J. S. Gero and F. Sudweeks (eds), *Formal Design Methods for CAD (preprints)*, IFIP - University of Sydney, pp. 271 – 291.

Gero, J., Louis, S. and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **8**(2): 83–94.

Knight, T. (1986). Transformations of meander motif on greek geometric pottery, *Design Computing* **1**: 29–67.

Koning, H. and Eizenberg, J. (1981). The language of the prairie: Frank lloyd wright's prairie houses, *Environment & Planning B.* **8**: 295–323.

Stiny, G. (1980). Introduction to shape and shape grammars, *Environment & Planning B.* **7**: 343–351.

Stiny, G. and Mitchell, W. (1978). The palladian grammar, *Environment & Planning B.* **5**: 5–18.

Stiny, G. and Mitchell, W. (1980). The grammar of paradise: on the generation of mughul gardens, *Environment & Planning B.* **7**: 209–226.

Figure captions

Fig. 1 (a) search in a single state space S_1; (b) search in an extended state space - a union of state spaces $S_{ext} = S_1 \cup S_2 \cup S_3$.

Fig.2 The search process for Algorithm A. The current set of S_i are denoted by hatching. The total population of all current sets S_i is constant in each generation.

Fig.3 The search process at step (1) of Algorithm B. The trial points are moved within the same set of current sets, S_i. k_1 here is a an iteration counter of the solution of the low-level design problem within fixed set of grammars, G_i.

Fig.4 The search process at step (2) of the Algorithm B. The number of current grammars, G_i, for all generations is constant. The populations of every of the sets, S_i, is constant also.

Fig. 5 Two of the transformation rules used in the example.

Fig. 6 Pareto-set.

Fig. 7 Density distribution of the rules. The density of the rule i in generation k is defined as a ratio of the number of applications of this rule to the total number of rule applications for generation k.

Fig. 8 The dependence of the total number of rules used and the number of these which have been used previously on generation number.