Author(s): Coates, Paul; Makris, Dimitrios.
Title: Genetic Programming and Spatial Morphogenesis
Year of publication: 1999
Citation: Coates, P. and Makris, D. (1999) 'Genetic Programming and Spatial Morphogenesis.' In: AISB Symposium on Creative Evolutionary Systems, Edinburgh College of Art and Division of Informatics (AISB'99), University of Edinburgh, March 1999. pp.105-114
Publisher link: http://www.aisb.org.uk/convention/index.shtml

Genetic Programming and Spatial Morphogenesis

Paul Coates, Dimitrios Makris
CECA
University of East London School of Architecture
Holbrook Centre
Stratford
London E15 3EA
p.s.coates@uel.ac.uk
mak1522r@uel.ac.uk

Abstract

This paper discusses the use of genetic programming (G.P.) for applications in the field of spatial composition. The G.P. was used to generate three-dimensional spatial forms from a set of geometrical structures .The approach uses genetic programming with a Genetic Library (G.Lib) .G.P. provides a way to genetically breed a computer program to solve a problem.G. Lib. enables genetic programming to define potentially useful subroutines dynamically during a run .

If you do not expect the unexpected, you will not find it.
Heraclitus

1. Introduction

This paper describes some experiments using Genetic Programming as a generative grammar of form. By a grammar we mean (after Stiny, Mitchell) a set of initial conditions, a lexicon of primitive objects, and a syntax of transformations on those objects. In these experiments the grammars are on the one hand a "personal grammar" and on the other a canonical architectural object Le Corbusiers simple concrete frame structure defined in his "DomIno House".

We know what the initial grammar produces, when the simplest sentence produces the most basic design. GP allows the parallel exploration of the design worlds defined by the initial axioms and productions.
Whether this is likely to be interesting depends entirely on the initial grammar. A badly chosen set of axioms and productions may lead to small design worlds.
With a well chosen grammar, leading to a large number of non trivial design worlds, the likelihood of finding a suitable candidate as the solution to a properly posed problem increases.

In recent papers(Coates and Broughton,1997) automated shape grammars have been reported, and the SEED project (Akin, 1997) has developed automated shape grammar algorithms using prolog like syntax, with goal driven logic in the form of rule sets of valid relationships.
In this paper we are turning this procedure around, providing no rules, but , by evolving productions, allow for the emergence of grammatical objects by selection.

Trere are two distinct ways of approaching the 'demands' and the 'singularities' of architectural design have been discussed widely eg. (Tzonis and White, 1994). The one is that in order "to decipher and explicitly encode the variety of combinations in a knowledge base is to solve a significant portion of the possible problems beforehand , and encode all the various characteristics with information of how to address each individually , while the program performs intelligently on the task, it does so in a purely static manner" (Angeline 1994).
The other is the "ability for the program to react dynamically as information about solving the task is obtained. This allows the method to opportunistically adapt to the idiosyncrasies of the current problem and its variants. If such a dynamism can be identified, then a successful program will not require explicit knowledge for handling the minute differences between problem instances and the knowledge associated with these differences can be removed" (Angeline 1994).

P.Angeline (1994) argues that: "...using Emergence Intelligence, allows the removal of explicit knowledge that is a natural consequence of the problem solving process interacting with the task environment .By allowing the task environment to be an integral component of the problem solving algorithm, all the natural constraints, include those too subtle for a knowledge engineer to extract, are available to the algorithm and emerge at appropriate moments while solvile solving problems".

Following Woodbury, (Woodbury, 1993) we consider as a design space all the machinery required to computationally search for architectural designs. That 'space' consists of a 'search space' and a 'search strategy'. The purpose of a search space is the description of all possible configurations that might be considered as solutions

to a design problem. The role of a 'search strategy' is on the one hand to establish a set of the decisions required in search, on the other hand is to act as a problem context in which these decisions applies.

None of the examples in this paper aim to be universal form constructors, in the sense of Bentley (1996) Coates & Jackson (1998), instead they take as their starting point a set of basic shapes and relationships which are themselves preselected by the designer as parts of a consistent grammar, the unfolding of which aims to explore its implications. The idea underlying these investigations is that architecture results from the multiplication of some simple relationship. The range of moves available when exploring these designs by hand are limited by the increasing complexity of defining anything more than short series of transformations. The use of a recursively defined generative grammar as defined here using GP, allows for recombination and embedding of morphological moves to any level of complexity required. Using a basic (seed) volumetric relation of solid to void for instance, gives rise to elaborations of this basic pattern, which exhibit a range of emergent forms which express the seed relationship.

The basic architectural design problem consists of the prediction-composition of the building structure from the primary determinants. Different design strategies contain different theories to approach that compositional problem. The problem is that although it is relatively easy to determine the putative structure of the problem , the determination of its possible formal structure is extremely difficult. Not only that, but the problem definition (brief, program,accommodation list etc.) does not imply a solution , but rather should be seen as a way of testing possible configurations.

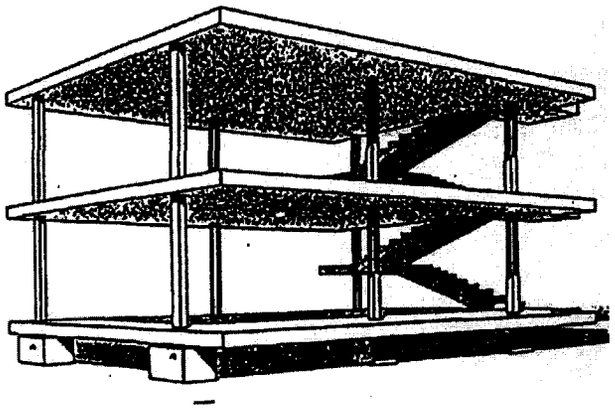

Figure 1:Le Corbusier's drawing of the Domino House

2. Grammars of the dom-ino house

Using GP of morphological functions to explore the space of grammars of form.

In earlier studies (Coates et. al (1995,1996, 1997), we have reported on the development of form using s-expressions of morphological functions. These earlier experiments simply started with an arbitrary set of functions - usually the boolean operations of union, intersection and difference, and the affine transformations (translate scale rotate) both in 'copy' and 'move' format. All these functions were used to transform the elements of the terminal set which were arbitrary blocks of various shapes. These experiments were done to establish the validity and utility of using the GP operations of crossover and mutation on these s-expressions to explore a design world of CAD operations.

The morphology that developed was initiated with a random selection of primitives and functions, and evolution was driven by artificial selection (Dawkins 1986) with the user selecting the parent phenotypes for crossover or mutation .

In the summer of 1998 the Baunetz architecture internet prize `http://www.baunetz.de/internet-preis/` announced a student competition based around a reinterpretation / deconstruction of Le Corbusiers 'Dom-Ino' house. One of the prize winners ("genetic bastards" Christoph Körner, Lars Krückeberg, Wolfram Put 1998) was a lively essay on the (as yet unrealised) possibilities of recombining the elements of the standard house. I decided that if it was possible to define the domino house as an s-expression in the terms of the GP system we had already developed, then the evolution of the phenotype would offer just such instructive insights into the overall architectural possibilities inherent in the canonical morphology offered by Le Corbusier as the starting point for building.

The domino house was originally conceived by Le Corbusier as a concrete frame structure with vertical columns and an active reinforced slab (no beams as such).

In these experiments, the initial object was a simplified version of the canonical drawing, expressed as the result of making different shaped blocks and copy and move functions to get them into the right position.

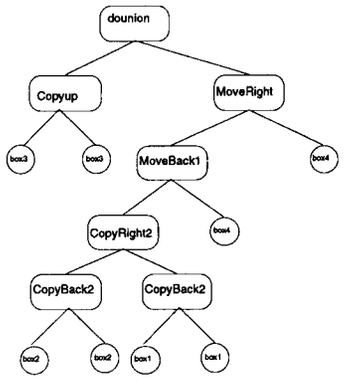

Figure 2: Domino additive function tree

2.1 Experiment 1

Domino by block move & copy - the additive model

```
(dounion    (copyup2 (box3) (box3))
     (movert1
          (moveback1
               (copyrt2
                    (copyback2 (box2)(box2))
                    (copyback2 (box1) (box1))
               )
          (box4) )
     (box4) )
)
```

This can be expressed in english as:

"Union the result of copying up box3 and box3 with the result of moving right the result of moving back everything copied right after copying back both box2 and box2 and box1 and box1".

The use of dangling *box4s* at the end of the diagram is to satisfy the requirement that each morphological function needs two arguments, box 4 is a tiny "dummy" object to plug into the expression to mend this gap.

Each phrase of this expression (copy up, move back etc) can be seen as a little "constructor" which corresponds to the transformations in a shape grammar. The blocks used

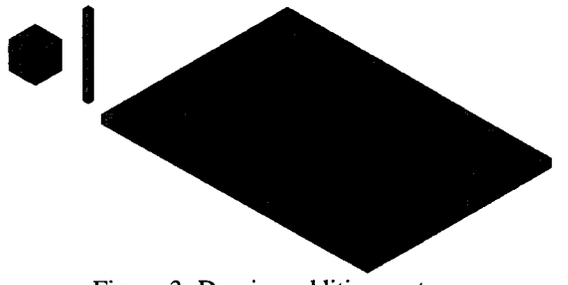

Figure 3: Domino additive parts

were defined as 3 objects plus the extra box4

```
(defun    box1 ()
(ssadd (mybox  '(-0.5 -0.5 -1) '(0.5 0.5 0) ))
     )
(defun       box2 ()
(ssadd (mybox  '(-0.1 -0.1 0) '(0.1 0.1 3) ))
     )
(defun       box3 ()
(ssadd (mybox  '(0 0 0) '(10 7 -0.3)  ))
     )
(defun       box4 ()
(ssadd (mybox  '(0 0 0) '(0.1 0.1 0.1)  ))
     )
```

2.2 Experiment 2

The domino house can also be constructed by using boolean (constructive solid geometry) operations.

```
(dosubtract (box0)
      (dounion
            (dounion (copyback (box3) (box3)) (copyrt (box1)(box1)) )
            (dounion (box2) (box4))
      )
)
```

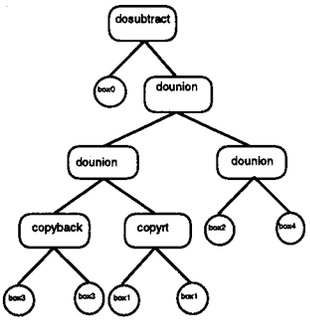

Figure 4: Domino subtractive function tree

In this case the blocks are designed to carve out the domino morphology from one big block by boolean subtraction. The english version of the above expression is : *"subtract the union of on the one hand the union of copyingback box3 & box3 and copyingright box1 and box1, and on the other hand the union of box2 and box4, from box 0."*

There are five blocks , block 0 the one from which the object is carved (shaded pale grey in figure 5), and blocks 1 to 4 (+ plus copies) which are copied, unioned and subtracted from block 0.

```
(defun    box0 ()
(ssadd (mybox  '(0 0 0) '(10 7 3) ))
)
(defun    box1 ()
(ssadd (mybox  '(-0.5 -0.5 0.3) '(1.9 7.5 2.7) ))
```

```
)
(defun       box2 ()
(ssadd (mybox   '(2.1 -0.5 0.3) '(7.9 7.5 2.7) ))
)
(defun       box3 ()
(ssadd (mybox   '(-0.5 -0.5 0.3) '(10.5 1.3 2.7)  ))
)
(defun       box4 ()
(ssadd (mybox '( -0.5 1.5  0.3)'( 10.5  5.5 2.7)  ))
```

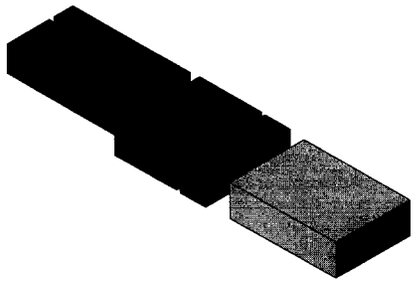

Figure 5: Domino subtractive parts

```
)
```

2. 3. Experiments with the additive model

The GP system was set up with a lexicon consisting of :

The Function Set of 13 copy and move functions made up of the 5 in the original expression plus their symmetrical equivalents, and the 3 Boolean shape operators, one of which was used in the canonical model. The use of extra functions was to provide the GP system to explore a wider range of designs using the Mutate operation. The domino functions are in **bold**

copyUP1
copyUP2
copyDN1
copyRT1
copyback1
copyback2
copyRT2
moveRT1
moveBack1
moveBacK2
moveUP2
moveDN1
moveRT2

dounion
dosub
doint

And The Terminal Set of the 4 cuboids
Box1
Box2
Box3
Box4

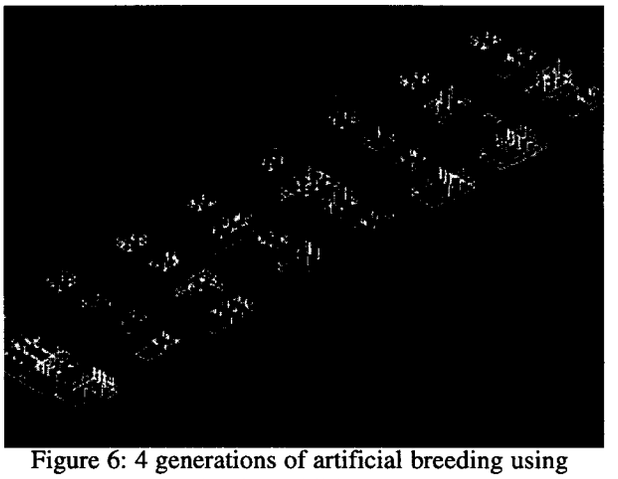

Figure 6: 4 generations of artificial breeding using domino additive set

For the initial experiments only crossover was used, thus restricting the evolution/breeding to shuffling the canonical function set. Figure 6 shows the first four generations using crossover only on two individuals per generation. The initial population are set manually to the domino expression. the second generation (next row towards the viewer) is the result of the evaluation of a further 8 expressions consisting of randomly shuffled subtrees of the domino gene. In this first generation , since all individuals in population 1 are identical genotypically, it makes no difference which two individuals are chosen. In later generations selection is made by eye

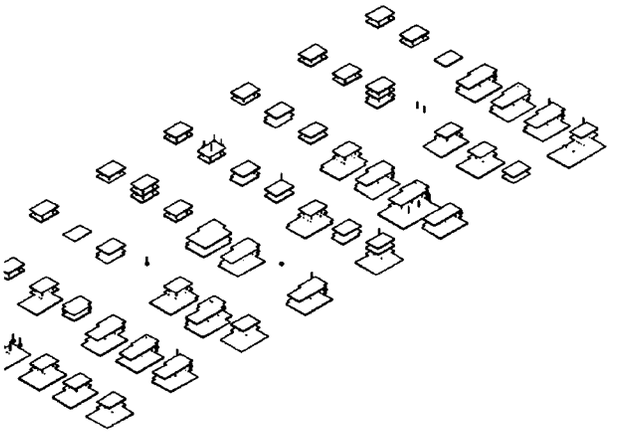

Figure 7: Domino additive crossover

on the basis of subjective judgements about the appearance of the individuals.
Using only crossover for 7 generations (figure 7) of the additive model, and selecting for overall height and 'sensibleness' by eye, the successive populations slowly increase in variety, but because the lexicon is restricted to the original set, the range of outcomes is often limited.

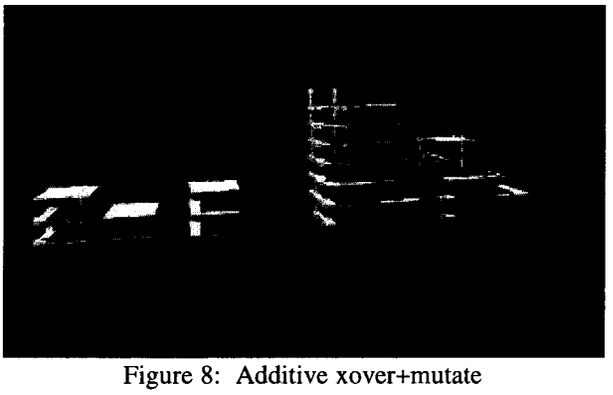

Figure 8: Additive xover+mutate

Using a judicious mixture of crossover (to explore a closely related set of promising outcomes) and mutate (to force the inclusion of functions from the wider lexicon)it is possible to drive the system towards more extreme outcomes. Figure 8 shows members of the last (12th) generation where selection had been again for height and usefulness.

FIgure 9: Additive mutate

Using mutate only leads to rather uncontrollable outcomes, and rarely resulted in acceptable objects emerging. The forest of columns and floating slabs was the best that was achieved (figure 9)

2.4 Experiments with the subtractive model

The Function Set of 13 copy and move functions made up of the 5 in the original expression plus their symmetrical equivalents, and the 3 Boolean shape operators, one of which was used in the canonical model. The use of extra functions was to provide the GP system to explore a wider range of designs using the Mutate operation. The domino functions are in **bold**

copyback
copyforward
copyRT
copylT
copyUP2
moveUP2
moveRT
moveBack
movelT
moveforward
DoUnion
Dosubtract
Dointersect
And the Terminal Set of the 5 cuboids
Box0
Box1
Box2

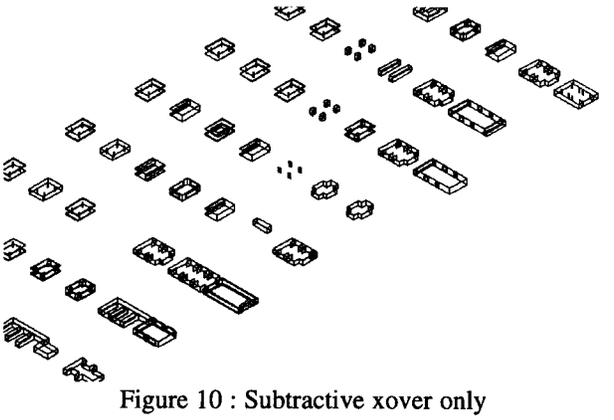

Figure 10 : Subtractive xover only

Box3
Box4

2.4.1.Crossover only with the subtractive model.

In this method of creating the domino geometry, the base set of functions is very small, and as a consequence the design space is restricted to single height arrangements. Where the cutting blocks move beyond the reach of the original 'positive' block 0 they survive and form agglomerations of large volumes.
With Mutation also used a wider range of options is available. The illustrations show (figure 11) examples of

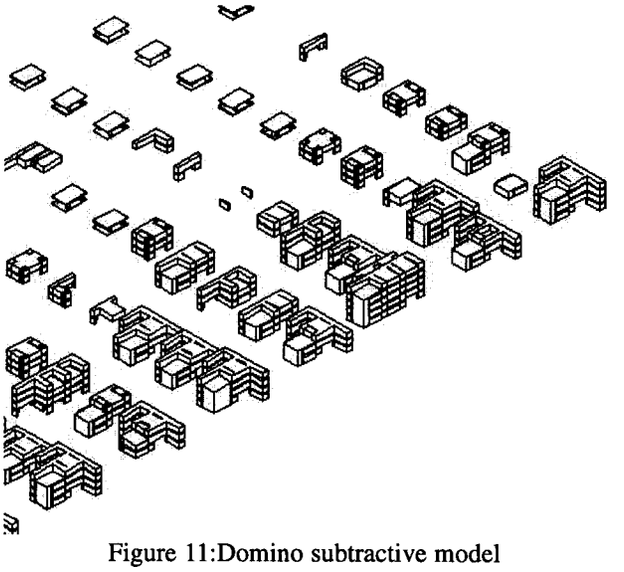

Figure 11:Domino subtractive model

12 generations showing the evolution of multi storey units with structural bays.

Two other examples (figures 12 & 13) were bred with a more domestic scale in mind, with configurations that made useful interior space and/or Pilotis.

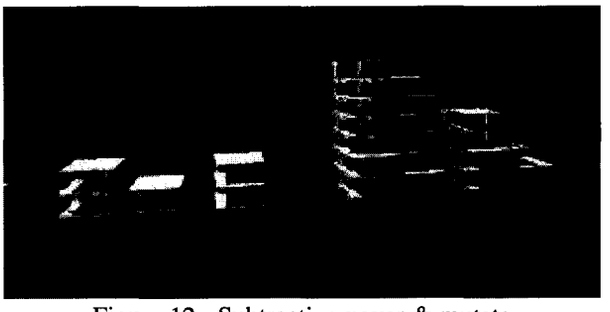

Figure 12 : Subtractive xover & mutate

Figure 13: Subtractive xover & mutate

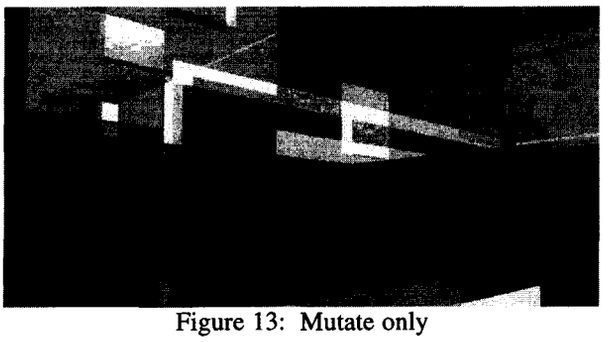

Figure 13: Mutate only

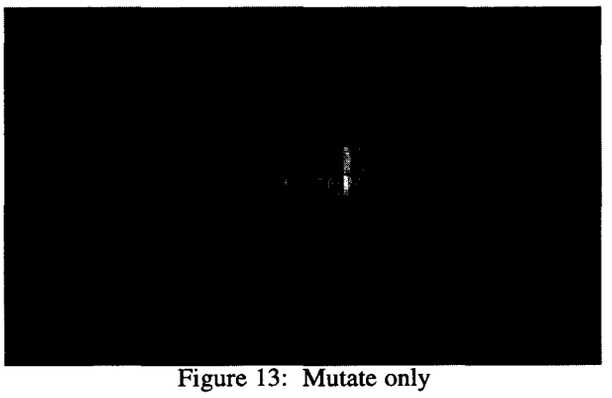

Figure 13: Mutate only

With mutate only the results become too complex to control, but by rendering the combined blocks transparently it is possible to observe that the spatial logic embodied in the syntax results in internal walls and columns, single and double height spaces. (figures 14 & 15)

3. Personal Shape Grammar

The second set of experiments studied evolving three-dimensional forms that might be useful as compositions of space. This use of the term space and the method of analysis derive from the ideas that the manner in which the volumes of solids-voids are composed chanells the flow of space .(Wright,1941 ; Zevi, 1974).

The more open ended experiments are also based on a similar pair of conceptually different grammars, a structural additive grammar and a void solid one. The additive grammar was defined to produce assemblies with a vaguely Japanese quality, reflecting the proportions and relationships found in traditional domestic construction. The void-solid idea was pursued further to simple natural selection based on global form and location. The void-solid grammar is based on the idea of the primary architectonic relation being a partial overlap between similarly sized but slightly offset cuboids. This provides the seed of an idea about the overall form of a particular arrangement, which can be observed in figures 21 & 22.

The operations of copy and union provide ways of agglomerating spatial units into large volumes and mass-

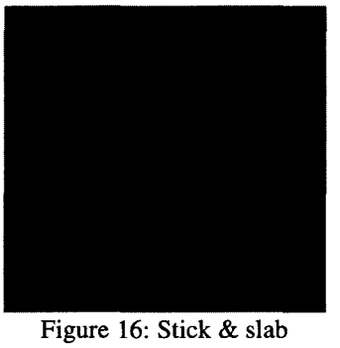

Figure 16: Stick & slab

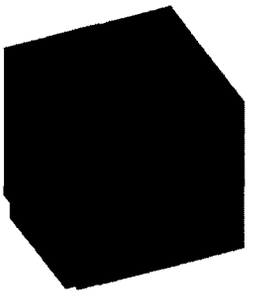

Figure 17:Solid & void space

es, or alternatively scattered pavilions. In each case the resultant forms reflect at different scales and recombinations the seed configuration.

The functions in both cases consist of:

1) The full set of boolean functions Union, Subtract and difference

2) Translate and Copy in the 6 directions aligned to the faces of a cube

3) Rotate in all three axes positively and negatively.

As an example of the way these functions work, the object shown in Figure 18 was constructed by the s-expression above (figure 19) . On inspection it can be seen that the resultant form is the result of two Boolean

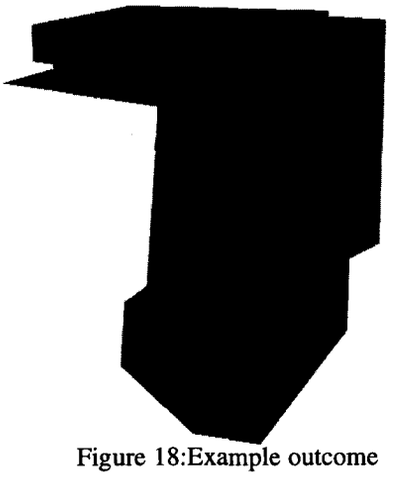

Figure 18:Example outcome

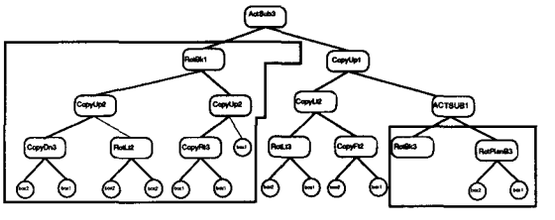

Figure 19: Function tree for figure 18 object

subtraction operations (Actsub3) on the accumulated geometry described by the dependent subtrees - shown with boxes round them in the diagram.

3.1 Artificial selection

Initial experiments were run using both the stick and slab (figure 20) and the solid void (figures 21 & 22) to test the usefulness of the primitive objects and function set.

It is possible to observe in these outcomes the characteristic morphology derived from the primitive objects (in both cases the function set remained the same) with the solid-void dyad generating species of enclosure qualitatively different from the stick / slab version.

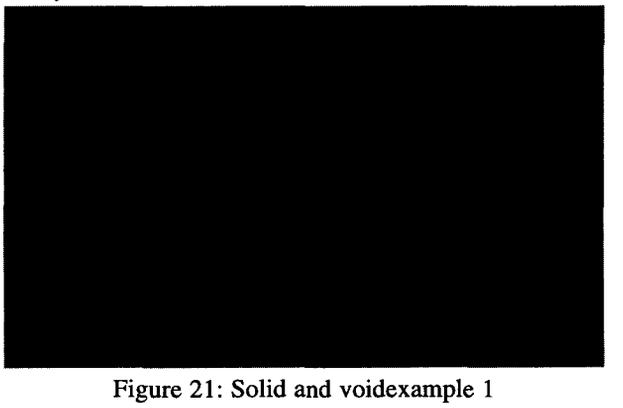

Figure 21: Solid and voidexample 1

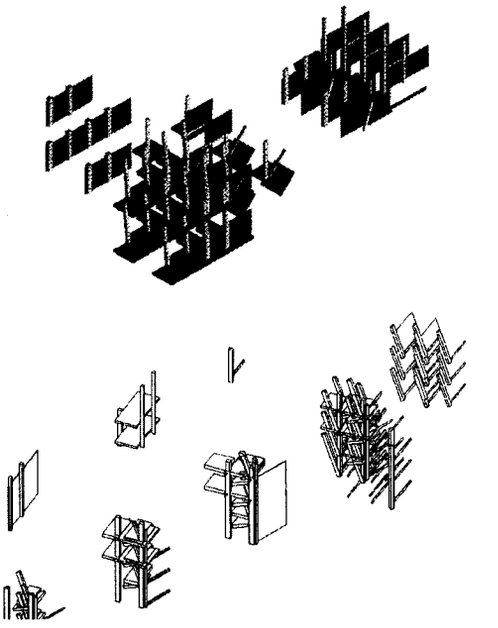

Figure 20: Slab & stick outcome examples

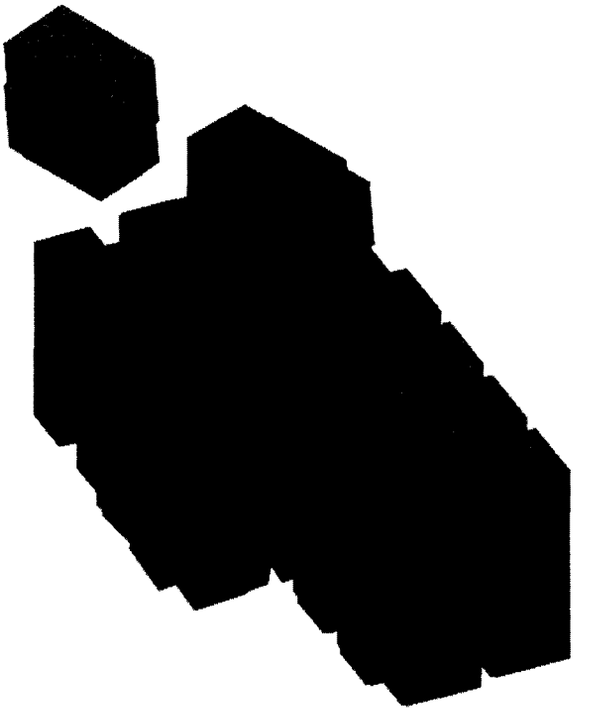

Figure 22: Solid and void example 2

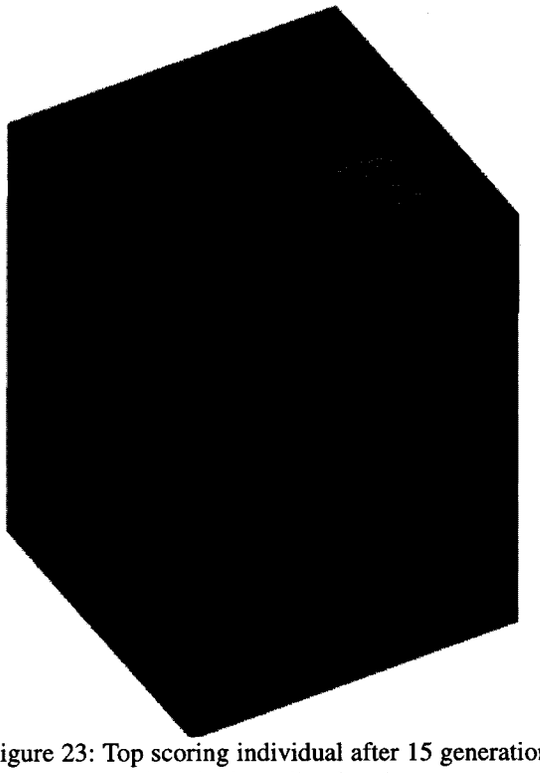

Figure 23: Top scoring individual after 15 generations showing site bounding box in grey

3.2 Natural selection

The idea of using natural selection was not to arrive at 'optimal' designs, because of the open ended nature of the problem, any fitness function can only be a partial assessment of the morphology; but to remove completely useless individuals from the gene pool prior to breeding by eye. A range of objective functions were tested, the simplest being the site bounding box.

At the same time as conducting these experiments we also implemented the Genetic Library (Angeline,1994) idea, where small subtrees of successful parents are compressed and saved as a library of new functions. the individual above (figure 23) contains two such functions (nos 4 and 17) within the function tree shown above (figure 24)
The likelihood of identical subtrees being embedded at different nodes in the same genotype (function tree) leads to fractal like self similarity at different levels of unfolding. This recursive embedding of morphological motifs at different scales or with different elements is a common property of architectural objects, responding to similarly nested social/ structural organisation of the brief.

4. Discussion

Our first results lend support to the hypothesis that an evolutionary approach is well-suited to the difficult

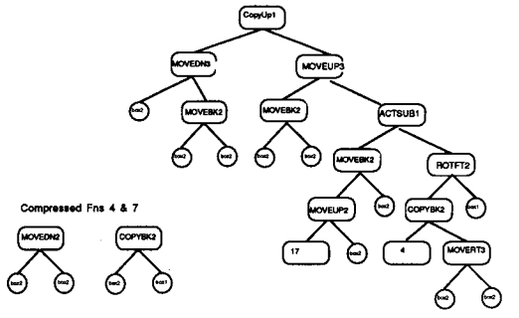

Figure 24: Top scoring individual after 15 generations - the function tree showing compressed functions.

problem of generating compositions of space .We want to propose an evolutionary system open to the inner logic that a potential user-designer might have for the creation of space. Evaluating the performance of our genetic programming system in a systematic way is not easy, because it is more of an illustration of an approach to designing "design environments", than an attempt to solve a set of particular design problems.

4.1 The problem of analysing complex 3D models.

The fitness function required for such outcomes will depend very much on the design model behind the grammar. In the solid void model, the fitness would depend on a configurational analysis of the global outcome within the same spatial types, and between different types. Global measures such as those suggested in (Coates & Jackson ,1998), where accessibility indices and size and shape quotients are calculated.

A complete examination of the implications of GP in architectural design would necessarily reflect the inevitably complex and dynamic character of architecture, and draw some lines towards new methodologies to model brief and space. To investigate the potential of that new design environment is to understand the internal rules governing the relationships between forms and forms, forms and demands, forms and ideas. The new understandings will inevitably influence architectural design in ways that have yet to be realised by architects. The G.P.S. can be as creative as the human designer who defines it can; it is also obvious that it can not provide any guarantee of creativity. We believe that the really implementation would be an advanced cooperation between the computational power and human creativity.

4.3 Conclusions

Architectural design 'problems' need a careful study before we try to apply any example from the genetic pro-

gramming paradigm. That happens because the nature of these so-called problems involves evolving cooperative behaviour and /or a changing 'landscape' and of course we can not apply just a function optimiser, no matter how powerful is its performance.
It is accepted, (A Tzonis, 1994) that that process is based on ill-defined problems, and it is not a routine process at all. It is in fact its inexactitude that drives the progress in architectural thinking. Because of this, these processes are not well understood, and therefore can not simulated by any simple algorithmic approach.

There are two main reasons that inspired ,from the beggining of this work, our intentions to avoid defining any problem-specific functions :
First, we wanted to introduce a learning process with little human intervention, so did not imply any prior knowledge to the representation of possible solutions.
Second, inspired by other genetic programming examples we wanted to find out how it would be possible to allow the 'learning process' to discover the more specialised or complicated group of commands to evolve specific architectural design 'demands'.

Experiments in this paper focus towards discovering such specialised groups of function commands (specific sub-trees) by starting the search space with some simple and general functions , from which the program defines the more specific groups.

They do not progress toward a predetermined state , but are always evolving within a loose envelope of constraints .

4.2 Some ideas for future work

We have presented an application of evolutionary computing to architectural morphogenesis. Our experiments based on the use of Genetic Programming confirm the idea that such an approach could be applied successfully in the real framework of architectural design .The preliminary results are encouraging.
In order to better understand the process of architectural morphogenesis and to create a more explicit and accurate techniques for the aid of the architectural-spatial design using a genetic programming system like the one we initially proposed, the following issues need to be resolved:
Do the specific function and terminal sets facilitate or hinder the ways of solving an architectural design problem? Of course there are many, (and with different degrees of complexity) function and terminal sets that possibly would suffice for a particular design problem.
The improvement of the effectiveness of the fitness function is a very important factor. The fitness function sets need to be explored in terms of representations of the brief, a large subject which will provide work for many generations of researchers!

A modelling of the way the designer interacts with the program with the aid of a user-interface is one of our next steps. The genetic library will be developed using a user interface to select from a set of much used emergent functions-subtrees. Using a phenotypical representation of each subtree, genetic manipulation can be achieved by selecting likely candidates for inclusion by mutation.

We hope that this approach towards computing in architectural design will teach us to observe in a certain way so as to discover new procedures and methodologies of architectural morphology - morphogenesis. As the model becomes more complicated, the search may become more difficult, and we may consider further modifications based on our knowledge about the application domain. Genetic programming may be very suitable for composing simulations and thus for understanding spatial composition. By exploring this process we may be able to learn more not only about the spatial structures, but also about the compositional process in architectural design.

4.4 Acknowledgments

This work is part of a Masters Thesis to be submitted to the School of Architecture at the University of East London, in September 1999. The work was supported by a grant, from the Greek National Scholarship Foundation.

References

Akin, O., Aygen, Z., Chang, T.W., Chien, S.F., Choi, B., Donia, M., Fenves, S., Flemming, U., Garrett, J., Gomez, N., Kiliccote, H., Rivard, H., Sen, R., Snyder, J., Tsai, W.J., Woodbury,R. and Zhang,Y. 1997. "A Software Environment to Support Early Phases in Building Design." *International Journal of Design Computing*. 1997, Vol. 1, http://www.arch.usyd.edu.au/kcdc/journal/.

P. J. Angeline and K. E. Kinnear, Jr. (eds) . Advances in Genetic Programming 2. MIT Press, Cambridge, MA,1996

P. J. Angeline. Genetic Programming and Emergent Intelligence. *In Advances Genetic Programming, K. Kinnear (ed.)*, pp. 75-96. MIT Press, Cambridge, MA,1994

T. F. Back, Hoffmeister and H. P. Schwefel. A survey of evolution strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms*,

R.K.Belew and L. B. Booker (eds),2-9: Morgan Kaufmann Publishers,San Mateo CA,1991

P. Bentley and Wakefield. The evolution of solid objects using genetic algorithms. *Modern heuristic search methods* ,Chapter 12 . John Wiley & sons.

T.Broughton, A.Tan, ., P.S. Coates, . The Use of Genetic Programming in Exploring 3d Design Worlds. *CAAD Futures 1997*, R. Junge (ed), Kluwer Academic Publishers ,1997

R. Dawkins .The blind watchmaker .Penguin Books, London ,1986

L. J.Fogel,, A. J. Owens and M. J. Walsh. Artificial Intelligence Through Simulated Evolution , John Wiley, New York , 1966

D. B. Fogel. Evolving Artificial Intelligence. Doctoral Dissertation, University of California at San Diego,1992

D. Goldberg. Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley Publishing Company Inc. Reading, MA,1989

J. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI,1975

C J. Jones. Essays In Design,, John Wiley and Sons,New York ,1984

K. E. Kinnear Jr. (ed.) . Advances in Genetic Programming. MIT Press, Cambridge, MA. 1994

J. R Koza. Genetic Programming: On the programming of computers by means of natural selection. Harvard University Press,Cambridge MA,1992

J. R Koza. Genetic Programming II. Automatic Discovery of Reusable Programs.Harvard University Press,Cambridge MA,1994

J, P. Mitchell. The logic of architecture. Harvard University Press,Cambridge MA, 1990

G. Stiny, Introduction to shape and shape grammars. *Environment and Planning* B7:343-351,1980

A. Tzonis and I . White. Automated Based Creative Design. Elsevier Science, 1994.

R.F. Woodbury. A genetic approach to creative design . *Modelling Creativity and Knowledge-based Creative Design.* Lawrence Erelbaum, 1993

F. L.Wright.On architecture. *Selected Writings*.Grosset & Dunlup,New York, 1941

B. Zevi. Architecture as space. Horizon Press,New York ,1974