Author(s): Coates, Paul; Broughton, Terence; Jackson, Helen
Title: Exploring Three-dimensional design worlds using Lindenmeyer systems and Genetic Programming
Year of publication: 1999
Citation: Coates, P., Broughton, T., Jackson, H. (1999) 'Exploring Three-dimensional design worlds using Lindenmeyer Systems and Genetic Programming.' In: Bentley, P. (ed.) Evolutionary Design Using Computers, Morgan Kaufmann Publishers Inc.: San Francisco, California, pp. 323-341.
Publisher link: http://www.cs.ucl.ac.uk/staff/P.Bentley/evdes.html

Exploring 3D design worlds using Lindenmeyer systems and Genetic Programming

1.0 Architecture, aesthetics and the utilitarian tradition

Since the end of the last century it has commonly been seen as decadent tosimply apply aesthetics to the structure of a building to make it beautiful (with the exception of the deliberately ironic, although irony itself would have been thought decadent by the stern moralists of the modern movement).

Architects such as Louis Sullivan, Mies Van der Rohe, Le Corbusier and so on used the example of engineering to help to explain the relationship between form and function. Based on the simplistic assumption that engineers do not design form, but that it emerges from the correct solution to mechanical realities (cf. the Eiffel tower, Brunel's bridges and the "dom-ino" concrete frame) the modern movement declared such objects as *pure* and *right* . The functionalist tradition has suffered many blows in the last 50 years, partly because it was always an oversimplification, and partly because technology has now reached a point where the constraints of structure have almost vanished, with form becoming the precursor of function rather than it's determinant, ie. anything is possible (cf. Utzon's Sydney Opera House, The new Bilbao gallery etc.)

The study of evolutionary algorithms allows us to get back to a more rigorous analysis of the basic determinants of form, where the global form of an object not only should not but actually cannot be predetermined on an aesthetic whim. Thus with genetic algorithms we have an opportunity to experiment with the true determinants of form in a way that the pioneers of the modern movement would have relished - an aesthetic of pure function whose outcome is totally embedded in the problem to be solved.

Contents

Why Lsystems & GP ?

This chapter explains recent experiments at the University of East London School of Architecture in form generation using Lindenmeyer systems and genetic programming. In order to explain the choice of L systems and the use of the Isospatial array, one must make a distinction between the use of genetic algorithms to optimise parameters to a given geometry, and the use of evolutionary algorithms to generate form .

In the case of the optimising approach (used for the past 20 years by engineers) the macro decisions about form,

and the fundamental design decisions are already taken. For instance in the well documented procedures for developing the hull of a boat, while the algorithm can develop a set of appropriate spline curves to make up the shape, using random perturbations of spline curves, and a model of hydraulic drag on the developed surface, it would not be possible for such an algorithm to generate a multi hulled solution (catamaran) , since the initial problem description has specified a solution space that only includes single hulled morphologies.

The problem with such attempts is that nothing more than fine tuning can take place. The aim of the current research is to try as far as possible to avoid over specifying of the problem domain, in the hope that the evolutionary algorithm will be able to search over a wider area of possible solutions. In order to do this we have to find a neutral description of 3D form, which is capable of embodying the widest possible range of objects, and a methodology of form generation which is responsive to evolution and not predefined by a particular technology.

The isospatial grid

The fact that the Cartesian coordinate system is so universally adopted in mathematics, software design and the design professions generally has lead to an over reliance on orthogonal geometry, and a tendency to assume that cubic tessellations are the *natural* way to represent 3D form. The Cartesian grid with its three planes and 3 major axes per point , while capable of representing any arbitrary object, is nevertheless fundamentally aligned to the idea that the three dimensional objects are just projections of the two dimensional plane, which in CAD terms are usually defined as two and a half dimensional. Things which are not projections of the plane are much more difficult to model than simple orthogonal objects.

The orthogonal grid also has built in to it a lack of homogeneity (and a bias in favour of orthogonality) because the orthogonal distances between points are not the same as the diagonal differences. In the Cartesian grid, the distance between a point and its 27 neighbours (plus or minus one cell away in all directions) varies between the unit distance along the 3 axes, the distance to edge joining cells, and the distance to vertex joining cells. In a cellular automata for instance, where neighbour counting is the basis of state change rules, it is necessary to build in weighting factors to overcome the problem of the three different distances between the neighbours of a point (face,edge and vertex).

The isospatial grid on the other hand , is defined as a point and its 12 neighbours as defined by a dodecahedron. This repeats across 3d space, just as the orthogonal grid does, but without the three different point to point distances of the orthogonal grid. Carter Bays and John Fraser (Bays 87,88) (Frazer 95) have both used this grid in cellular automata, in place of the cubic grid, and triangular tiling rather than square tiling has been adopted for 2d cas as well.
This grid , with its 12 same distance neighbours from any point would seem to represent a more "neutral" method of representing spatial objects, and in particular offers generative algorithms a simpler and more robust set of relationships between particles of the system. With 6 axes and 4 planes of symmetry it is a superset of the Cartesian grid, and can accommodate orthogonal relationships, but does not presume them.

In a project where the aim is to work towards the evolution of form with the minimum of preconceived notions of what constitutes it, this geometry seems more appropriate than many, hopefully leading to novel objects where the form could truly be said to emerge from the process of fitness testing, rather than being an artifact of the method of representation.

Lsystems as a generative method

Various computer-generated models of morphogenesis have been used to help understand the emergence of

complex forms in living organisms since Turing proposed the reaction-diffusion process in 1952. Diffusion limited aggregation models have been used to simulate crystal formation in super-saturated solutions and J. Kaandorp in "Fractal Modeling:Growth and form in Biology " (Kaandorp 94) uses an aggregation model to investigate the growth of corals. L-systems were developed by Astrid Lindenmayer in 1968 to model the morphology of organisms using string re-writing techniques (Lindenmeyer 88). These techniques have been applied in a variety of studies to the production of abstract models of biological forms as an aid to interspecial comparison and classification see (deBoer, Fracchia and Prusinkiewicz 92) .

As well as settling on a representational method, any evolutionary design algorithm needs a generating engine whose parameters are subject to evolutionary pressure. As mentioned above, one of the factors which can militate against an open ended process is the separation of the phenotype development process from the genotype description. That is to say, the genotype only describes parameters to the phenotype construction process, whose fundamental operation remains the same. This must of course always be the case , but in general it would seem to be better to make the genotype be as complete a description of the phenotype as possible, so that as many aspects of its behaviour are under evolutionary pressure as possible.

Genetic Programming as an apt approach

The development of Genetic Programming provides a clue as to how this could be achieved. GP was developed as part of the attempt to use the genotype as - literally- a code system for evolving software where the genotype maps very literally onto the phenotype. The original attempts to use crossover on the code expressed as an undifferentiated list resulted in 100% failure, as random chunks of code recombined always resulted in a crash of the phenotype.

By defining the code as a tree structure of s-expressions, and only allowing crossover / replication and mutation to take place on the sub tress, sexual reproduction could take place and the resulting code trees would always be syntactically correct - so no crashes.

This use of s expressions to represent recursively defined algorithms can be paralleled in the productions found in branching Lindenmeyer Systems. L-systems have been used to demonstrate a wide range of recursively defined objects since being originally defined as a formalism for describing botanical systems. They work using turtle geometry , using polar rather than Cartesian coordinates, and the production system itself forms a very compact description of a wide variety of forms. The productions themselves are s-expressions, and can be manipulated using GP just like the programming examples of Koza et. al.(Koza 92)

Using L-systems in the isospatial array combines two ideas both concerned with minimising the number of "givens" for a form evolving system.

1) It provides as far as possible a neutral description of space :
2) It reduces the number of arbitrary mappings between the genotype and the phenotype:

2.0 L-Systems Experiments

An L-system model starts with an initial axiom and one or more production rules. Axioms and production rules consist of symbol strings whereby individual symbols in the axiom are replaced by a string of symbols designated by the production rule. This process of character recognition and string substitution is carried out iteratively with each successive iteration producing a symbol string of greater complexity.

The resultant symbol string is interpreted as a series of drawing instructions which produce an abstract

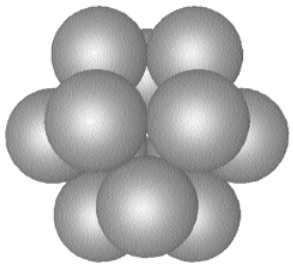
representation of the desired organism. The success of the L-system model lies in the self similarity of cell structure that many biological systems exhibit which is mirrored in the grammar based string re-writing process.

The L-system method of modeling developmental processes has been the subject of considerable research. Our approach is to use the L-system biological model in conjunction with an evolutionary algorithm and is an area which has seen relatively little investigation, recent work is reported in (Jacob 96)(Horling)

A standard Genetic algorithm and the associated Genetic Programming strategy are models which utilise the processes involved in the evolution of biological organisms

Visualisation of the growth model was carried out in Autocad, a 3-D modelling application using the Autolisp programming language.
Due to the constraints imposed by using Autocad we kept to the tri-axial Cartesian co-ordinate system rather than the alternative , more 'elegant' four axial system preferred by Frazer.

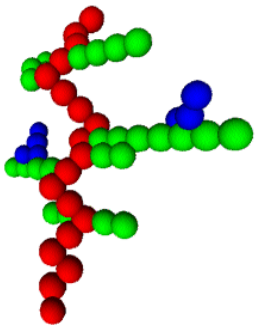
The twelve neighbours of any given point-'apoint'- are given by the autolisp function 'getneighbours':
Figure 2.1 the 12 balls of the isospatial array



```
(defun getneighbours ( apoint / neighbours offsets p np )
(setq neighbours '()
offsets(list '(1 1 0)
'( 1 -1 0)
'( -1 1 0)
'(-1 -1 0)
'(1 0 1)
'(0 1 1)
'(-1 0 1)
'(0 -1 1)
'(1 0 -1)
'(0 1 -1)
'(-1 0 -1)
'(0 -1 -1)))
(foreach p offsets
(setq np (mapcar '+ a point p)
neighbours(cons np neighbours))
)
```

The method of representing form is through the insertion of spheres drawn and saved in separate drawing files. Different coloured spheres represent different levels of recursive branching. The geometry of the 3-D space into which the spheres are inserted is an iso-spatial grid where the co-ordinates of the grid are the vertices of close packed cuboctahedrons as used by J. Frazer in "Data Structures for Rule-based and Genetic Design" . Spheres can only be inserted at these vertices and each sphere has 12 equally spaced neighbours.

Figure 2.2 branching tree structure



A variation on this geometry is the one used by Carter Bays' in "Patterns for Simple Cellular Automata in a Universe of Dense-Packed Spheres" which has a four axial system where 12 neighbouring points are the vertices of a dodecahedron.(Bays 87)

There are three primary genetic operators used in this experiment, initialisation, crossover and mutation. The first stage of the program is initialisation where a function is called to produce an initial gene pool. The genes are a randomly generated series of nested brackets containing three types of symbols;

Initialisation

An instruction to insert a sphere;
this is represented by the symbol **F** (for "forward")
A positional variable which refers to where to place the next ball; these are given in the symbology as **POS1** .. **POSn** for each of the n possible headings from any sphere. Because L-systems use Turtle graphics (polar as opposed to cartesian coordinates) the POS function should be thought of as a "turn" instruction, where the actual direction taken depends on the current heading. Early experiments used 6 possible turns, later ones reduced this to 4.
A bracket
an open one **(** indicates a branching point,
a closed one **)** is an instruction to return to a position on a lower 'limb' where it last branched.

These symbol strings which make up the gene pool are the production rules of the L-system.
Production rule

The next stage of the program is to iteratively apply a production rule to the initial axiom - in this experiment a simple 'f symbol. It only requires 3 or 4 iterations to generate a symbol string of considerable length as every instance of 'f is substituted by the production rule which itself is made up largely of 'f's.

F ->

(F(F POS1(F F POS2)F)F) ->

(F(F POS1(F F POS2)F)F)((F(F POS1(F F POS2)F)F) POS1
(F(F POS1(F F POS2)F)F)(F(F POS1(F F POS2)F)F)POS2)
(F(F POS1(F F POS2)F)F)(F(F POS1(F F POS2)F)F))

The re-written symbol string is passed to a function for interpretation of the symbols, ie its genetic code, and the realisation of the artificial organism in the drawing database. Every symbol in the string is evaluated and the corresponding function is called which carries the instruction for either;

(1) changing the positional variable or
(2) inserting a sphere at a point on the grid.

Before a sphere is inserted clash detection is carried out and if a collision is imminent the insertion command is ignored and the next symbol evaluated.

Genetic programming operations on productions

The L-system productions can be seen as nested bracketed s-expressions, and operations carried out on the nodes, just like Koza's examples, but with the functions F and POS1..n rather than the arethmetic functions of classic GP.

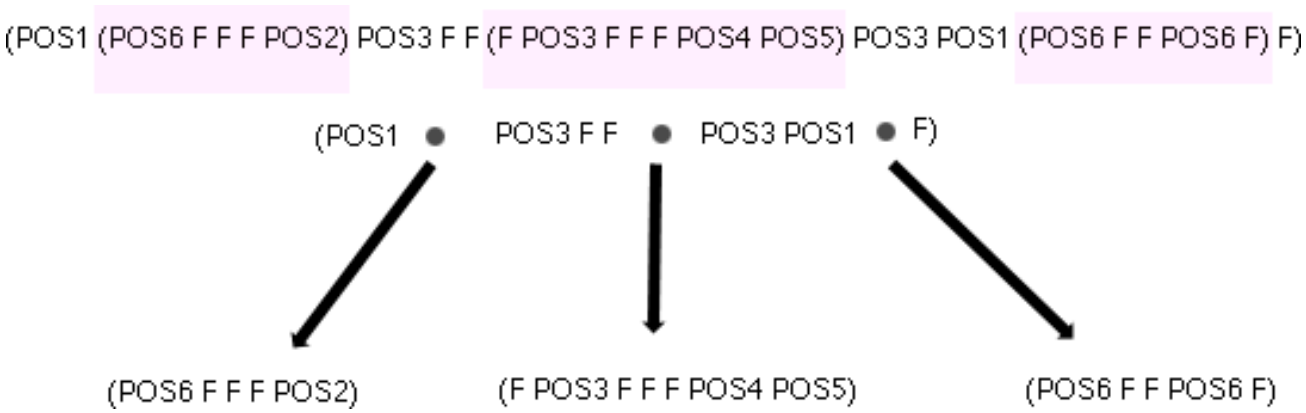


Figure 2.3 Productions as trees of S-expressions

In the example above, the three shaded components can be seen as subtrees from the main rule, and can be isolated as candidates for mutation or crossover. A fully developed tree is shown below

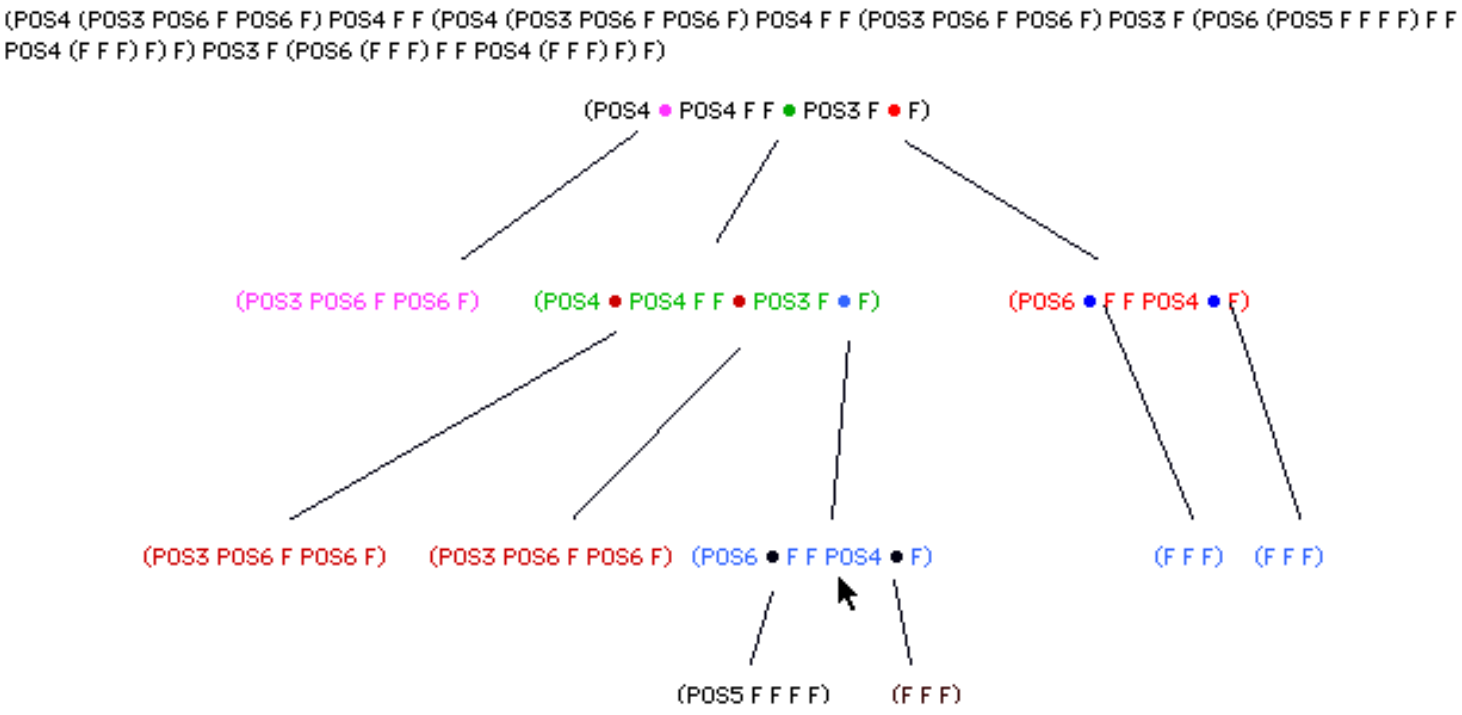


Figure 2.4 Example of a genotype generated by the L-system, drawn As a tree of functions

Mutation

The mutation genetic operator works by counting the levels of recursion or nested brackets in an individual's genetic code and selects, at random, a self-contained balanced chunk of code to be replaced by a similar, randomly generated, chunk. The chunk of code selected always starts with a branching bracket symbol and ends with closed bracket. The symbol string is not of fixed length so mutated organisms can have varying lengths of genetic material.

Crossover

Crossover works by choosing two organisms, selecting suitably balanced sections of code and swapping them. As in mutation the amount of genetic material an individual has can change, producing differences in generations ranging from slight to radical.

Initial experiments were carried out to test the crossover function. In the figures below, the left hand and centre configurations are two objects generated from a random production rule, the right hand object is generated from the resulting rule after crossover of the other two rule sets.

Figure 2.5 example of phenotypes demonstrating crossover

Objective function
After choosing a datastructure and method of representation and defining the type of genetic operators to be used, the third element of the algorithm is defining the objective function or 'fitness' function.

3.0 Evolutionary Experiments

Artificial Selection
Our immediate goal was to develop a model which would respond to the user's selection of the characteristics of two individuals which would survive and combine and become accentuated over successive generations - the 'eyeball' test as used in Dawkins' Biomorph program(Dawkins 90). Fig 3.1 show screen shots of two experiments.
The mechanism used here is to display in three dimensions the members of the isospatial grid occupied by balls. The breeder can view the 9 individuals rendered in isometric, and make an eyeball judgement as to how appropriate each candidate is to their preconcieved goal shape.;For each generation 9 phenotypes are displayed

and the user chooses two parents from which to breed the next generation. These two candidates' production rules (genetic material) are them crossed over, 9 times to ,create the 9 genotypes for the next generation, which are then developed into the 9 phenotypes for display.
IN the example below the user was selecting for length and regular branching, which rapidly evolved to the collection of pipe cleaner like objects in the lower right panel

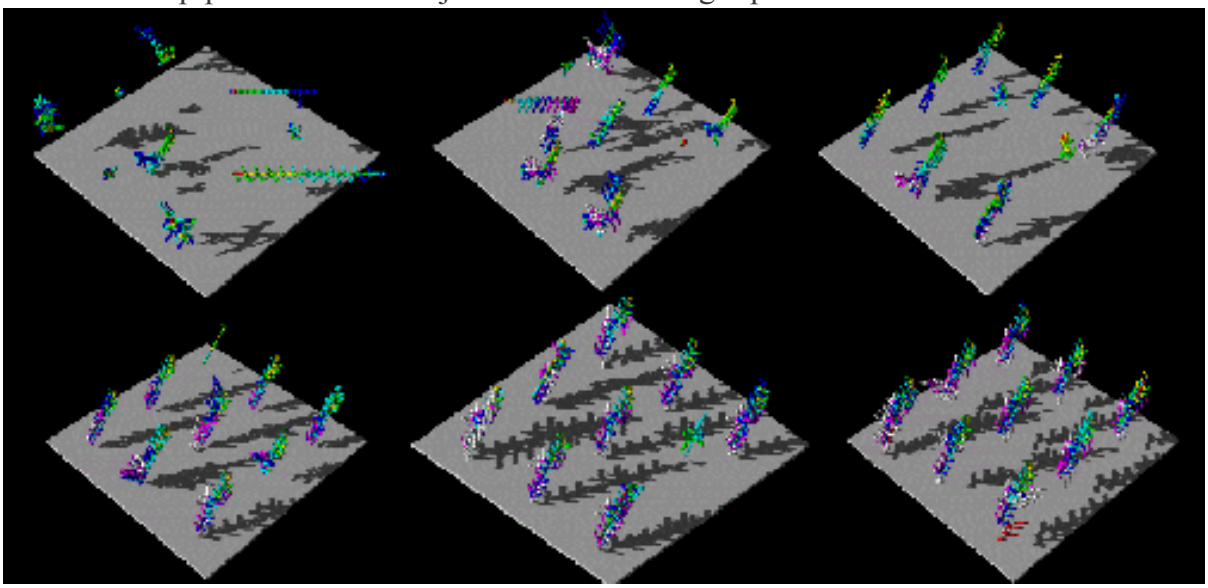


Figure 3.1 6 generations of artificial selection, starting random population at top left, final evolved population bottom right

Such systems as this are already in use in the two dimensional world (Kais power tools for Photoshop) and allow the user to search through a potentially infinite set of forms .

Natural Selection

Simple one goal Natural selection was used in the following 5 experiments. The L-systems were developed in an environment with a stream of particles moving across the area where the phenotypes were constructed. Each phenotype was bombarded with particles, which were tracked across the universe and the number of hits counted.

The windtunnel experiment

Here the fitness function was to reward those individuals which avoided being hit by the "wind particles". The outcome after 20 generations is a flat sheet of balls where only those on the leading edge are hit, the rest sheltering behind them

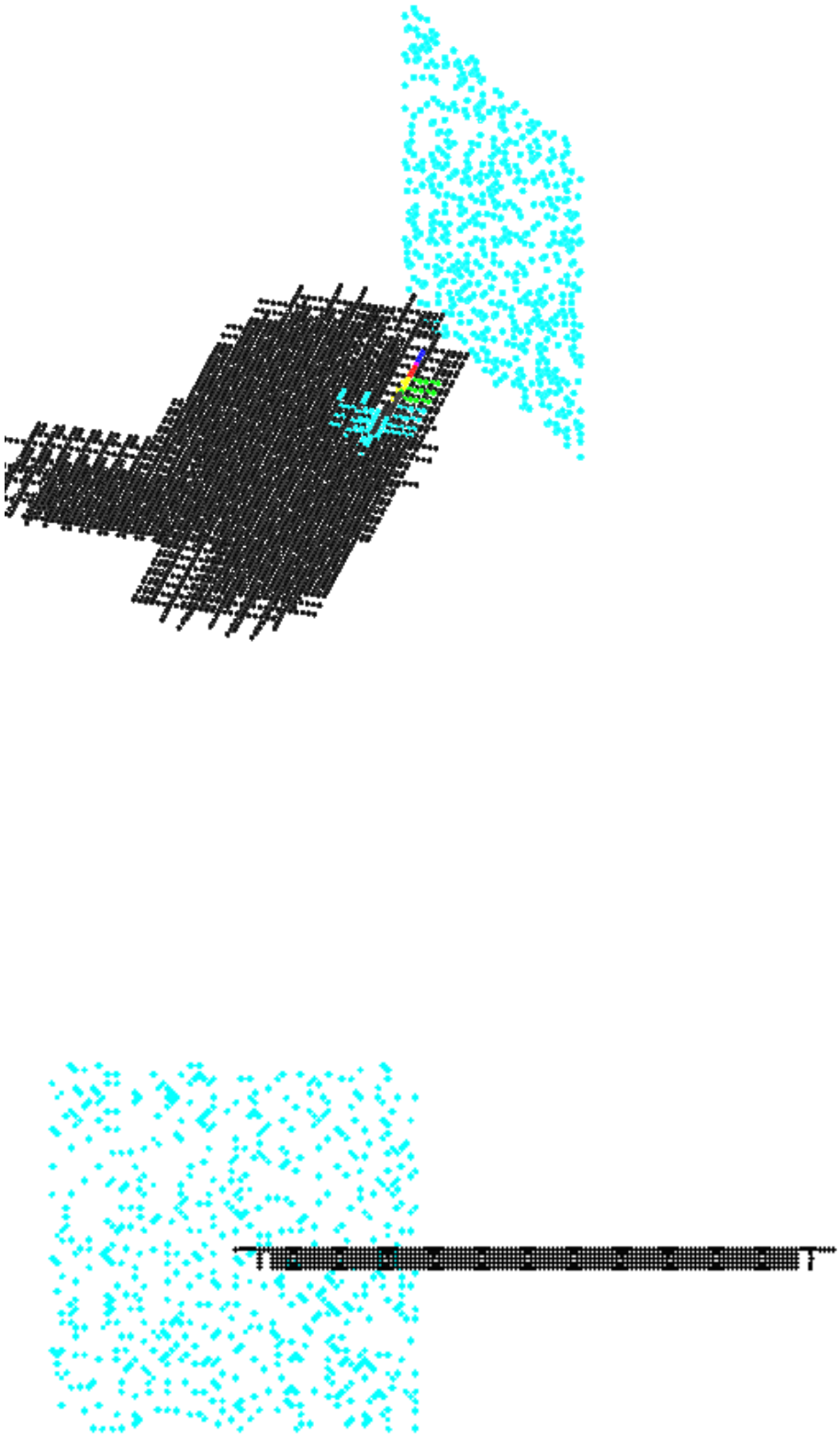


Figure 3.2 Windtunnel experiment - phenotype evolved as a flat sheet (forground) particle screen shown in cyan

Fitness Function = number_of_balls - number_of_hits * hit_cost

The flytrap experiment

Was the exact opposite, with individuals being rewarded for capturing the most particles.Early experiments were disappointing , with large negative scores and very little growth, often with the best individuals being reduced to a single ball . This turned out to be because the default cost of constructing an object was too high, and made it impossible to grow a big enough configuration that could take enough hits to promote development. Once the ratio between the cost of a ball and the cost of a hit was altered to be more like 1:20, positive scores and useful shapes emerged.

This parrallels Dawkins work with spiders webs (Dawkins 96) , where a fly is seen as food, and web construction is a cost to the fly. Our early experiments made it impossible to survive, since a fly was only worth one ball, the developing phenotypes ran out of energy before being able to catch any flys. With a ratio of 1:20 or thereabouts, each hit can justify more building, and evolution can take place. The configuration evolved in both cases into a flat sheet, oriented in the case of the ball catcher towards the particle origin. In the figure the evolved individual is on the right, the particle sheet on the left. Particles travel diagonally across the environment.

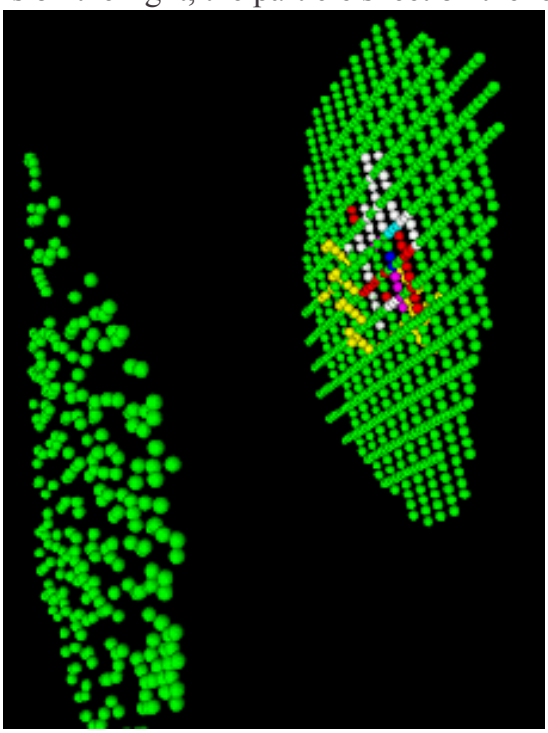


Figure 3.3 flytrap with particle screen in forground

The example shown here used hit cost of 20 and ballcost of 0.2
Further flytrap experiments were undertaken using Macintosh Common Lisp (MCL) with the ball data exported to microstation. Longer runs were attempted, usually 40 or 80 generations, with similar results. The chart shows the gradual evolution of the fitness over time.

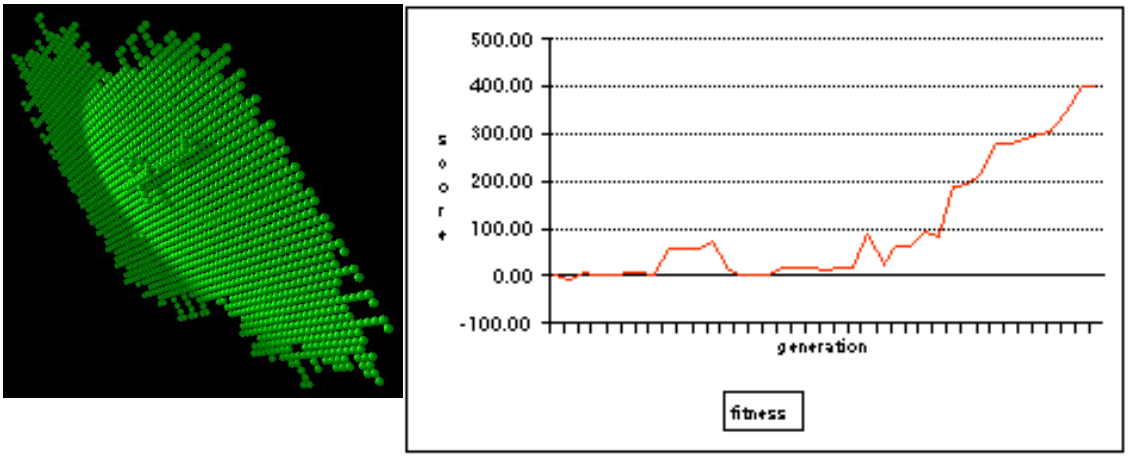


Figure 3.4 flytrap after 40 generations

Fitness function number_of_hits * hit_cost - number_of_balls * balls cost .

Suntrap

The flytrap results were extended with experiments to explore the effect of evolving configurations with vertical particles. These individuals were mainly horizontal configurations, but unlike the windtunnel experiments, there are many spikes growing up from the surface. This can be explained by noting that a ball can catch a particle whatever Z height it is at, the only condition being that it should not overshadow balls below it. A more open lattice evolves, rather than the flat widtunnel shape.

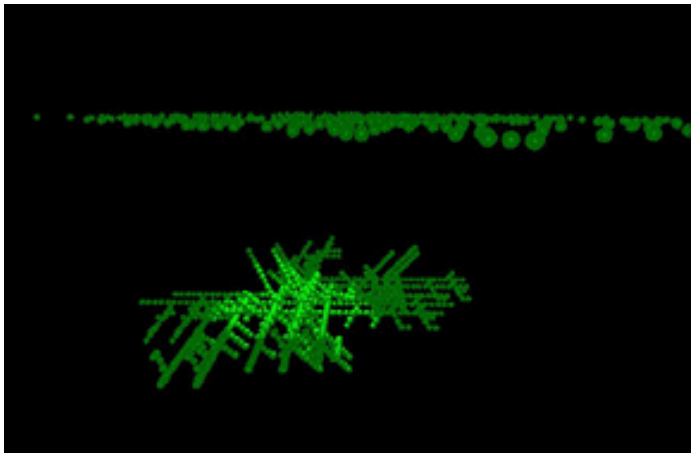


Figure 3.5 shows the evolved lattice, with the particle screen shown above.

Sun and Flys

Using both horizontal and vertical particle screens leads to the evolution of individuals with combinations of both types of configuration seen singly in the flytrap and suntrap examples. A central vertical sheet is surrounded

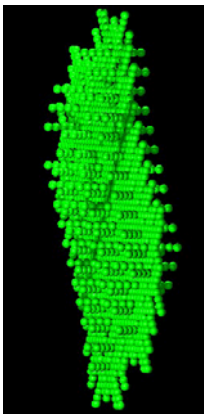
by horizontal extensions. 

Figure 10 shows the object was generated using a low ball cost, and with (in this case) a much higher success in catching horizontal balls than vertical ones

A mainly vertical sheet with a regular array of spikes has evolved. By increasing the ball cost, and weighting the vertical hits at twice the value of horizontal ones, a range of more balanced objects was produced

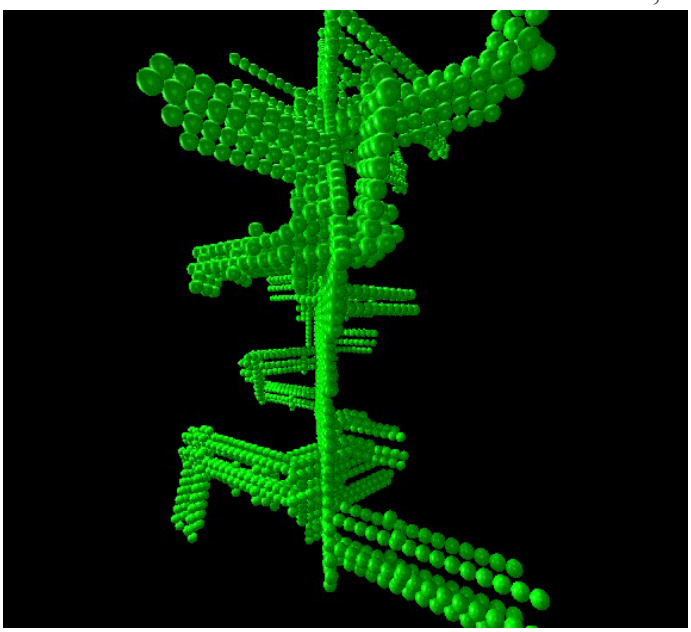


Figure 3.7 Sun and flytrap perspective view from the front

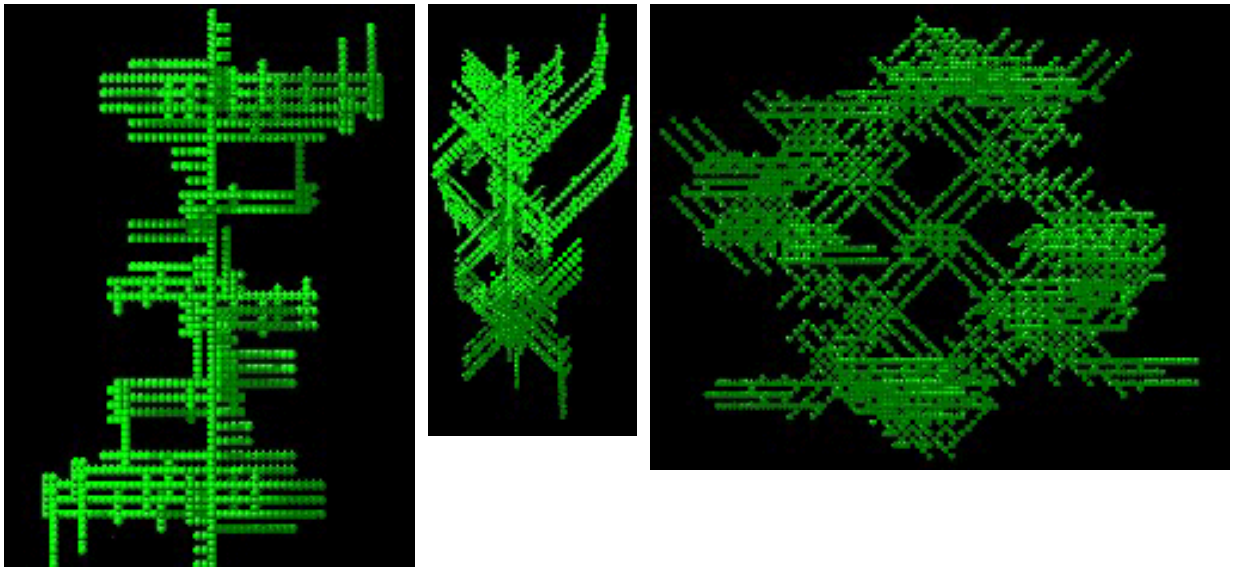


Figure 3.8-3.10 front, top and right views of same object

4.0 Space and Enclosure

After these initial abstract experiments the remainder are concerned with evolving shapes that might be useful as configurations of space. Our use of the term space, and the methods of analysis derive from the Space Syntax research at University College London (Hillier & Leaman, 76)(Hillier & Hanson 89) particularly calculation of accessibility indeces of the graph of spaces. Space in buildings in our definition, must be permeable (you must be able to get to all the spaces) and has to be distinguished from the outside by some kind of enclosure. In the turtle experiments (described below) we also needed to dfine a point of connection between the internal space and the outside world, in order to calculate the syntactical measures mean depth and accessibility (RA) with the outside

Enclosure

An early attempt to evolve for a single characteristic useful within the sphere of architectural design led to experimentation with the definition of enclosure. An architectural form is the enclosure of a system of space that is intended to serve some purpose. The internal space can be defined as that part of the universe which cannot be seen from the edges of the universe due to the position of an opaque building form (figure 4.1).

In terms of configurations of solid matter a good enclosure form is one with a high volume enclosed compared to the volume of the configuration. This criterion is the fitness function for a simple single-goal genetic program. An individual's score is the ratio internal volume: wall volume expressed as a percentage. The minimum score is 0%, but there is no specified maximum. The GP operates equally well in two or three dimensions, but results are quicker to obtain and easier to read as diagrams in two dimensions.

Figure 4.2 shows the highest scoring configurations from a evolutionary run using this GP. It can been seen that the simple fitness function produces recognisable space enclosures which can begin to be viewed architecturally.

Figure 2.2.1

Figure 4.1 The internal space is that part of the universe invisible from outside.

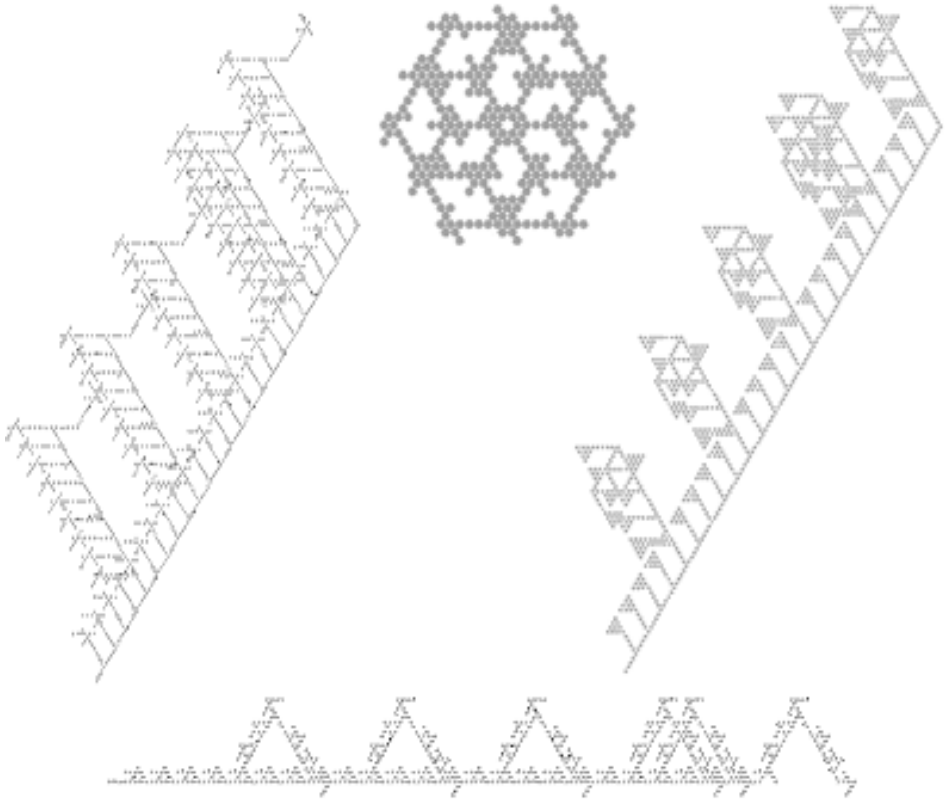


Figure 4.2 High scoring individuals using the space enclosure criterion.

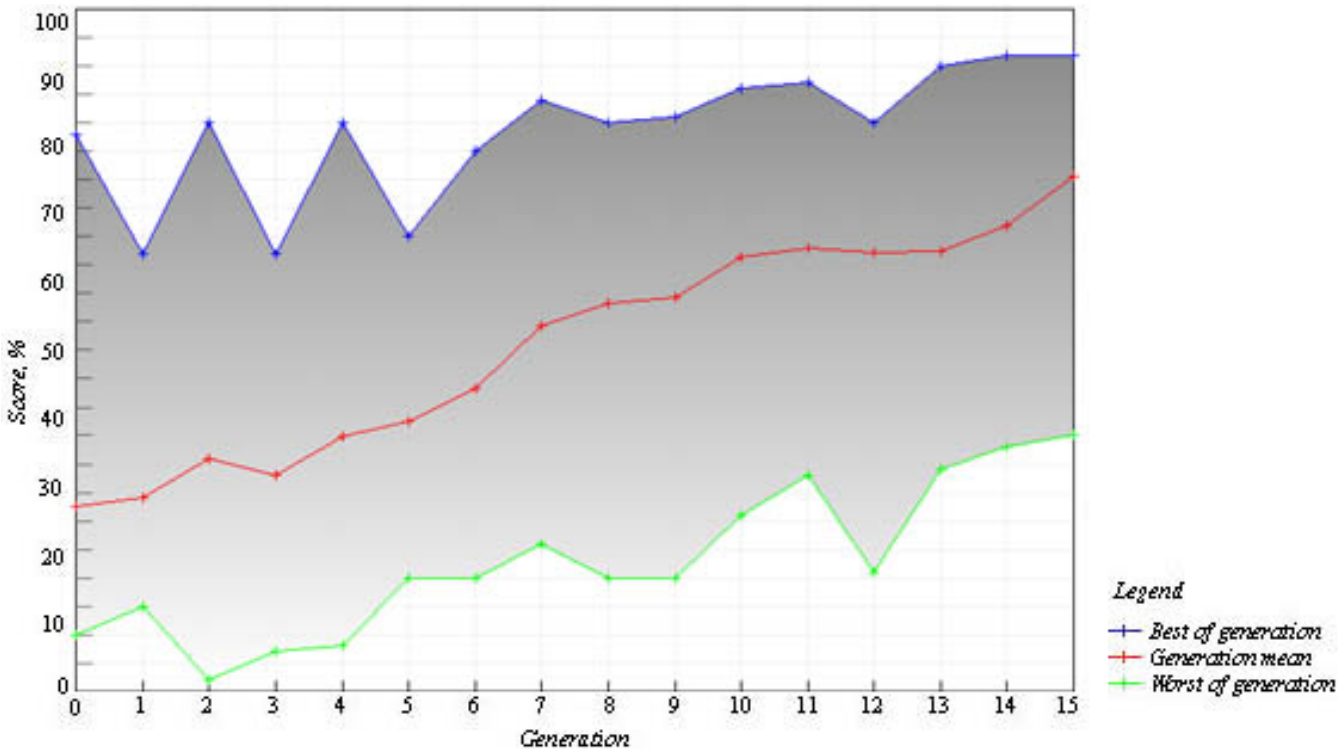


Figure 4.3 Graph to show minimum, maximum and mean scores each generation during a GP run of single-goal evolution towards efficient space-enclosing configurations. There are 16 two dimensional isospatial individuals in each generation.

The mapping from genotype to phenotype in all the models so far has been as simple as possible In the next set of experiments two new types of ball are introduced such that:
F > place a wall ball
F1 > place a permeable ball
F2 > place a space ball

As well as this, the evolving configurations are visited by a walking turtle, moving around the grid within the space balls, to build up a network diagram of the spaces

5. Turtles and Connectivity

Investigation of the configuration of an enclosure is carried out in a series of tests. Given the enclosure in figure 5.1, it can be seen that there are two types of building blocks making up the form. The the green blocks symbolise permeable wall, the blue solid. Turtles can move through green blocks.
FIG

FIGure 5.1

Figure 5.1 A random two-dimensional enclosure.

The first test identifies internal spaces, making up a list of grid positions which are defined as being inside the enclosure.

Figure 5.2 The internal space inside the above enclosure.

In order to analyse individual rooms, a floodfill algorithm is used on the inside spaces. This allows the number and size of separate spaces to be assessed. The example space configuration (see figure 5.3) has eleven rooms with a total area of 121 blocks, giving a mean area of eleven blocks. The largest space has an area of 48 blocks.

Figure 5.3 The internal space showing separate spaces in differing colours.

Once the individual rooms have been identified (i.e. there are eleven separate lists of grid positions), it is necessary to find out how the rooms are connected. A turtle begins a random walk in each of the rooms and is allowed to walk through grid spaces which are either internal or permeable (green enclosure blocks)., but not through solid wall.

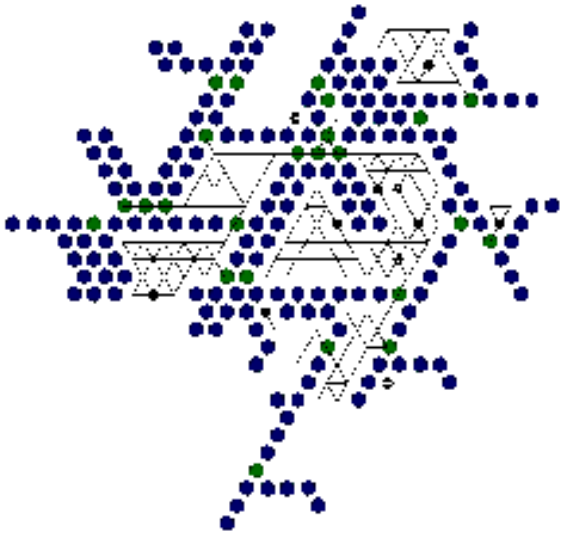


Figure 5.1 Random turtle walks around the enclosure.

Each turtle builds up a list of the rooms she has visited, showing which spaces are directly connected (so, if a turtle's list runs [1 2 1 3] space one is connected to both spaces two and three whereas a list [1 2 3 2] signifies that space one is connected to space two which in turn is connected to space three, see figure 5.5).

Figure 5.6 shows the j-graph produced for the example enclosure. A percentage connectivity score is given by the ratio of rooms connected in the j-graph to the total number of rooms. In this case the connectivity is 5/11*100, or 45%. This can be used as a fitness function to encourage evolution towards configurations where increasing numbers of spaces are connected.

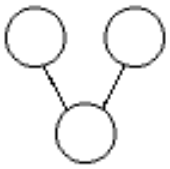


Figure 5.5(a) J-graph representing turtle walk [1 2 1 3].

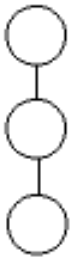


Figure 5.5(b) J-graph representing turtle walk [1 2 3 2].

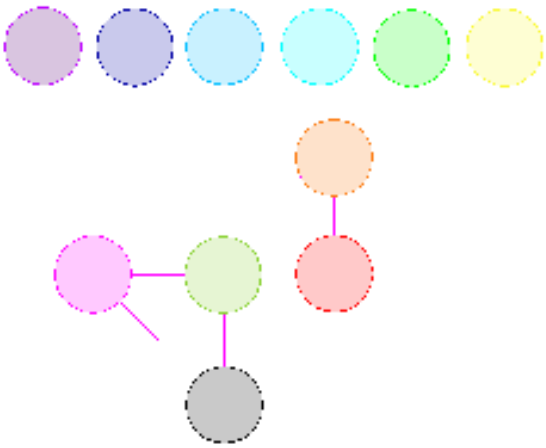


Figure 5.6 J-graph of example enclosure, giving a connectivity score of 45%. .JPG

6.0 Co-Evolution of space and enclosure

Symbiotic Co-Evolution

The next set of experiments moved away from simple one category fitness tests and used a symbiotic coevolutionary technique. Any two linked systems within a building could be considered as two mutualistic species.

Introduction

Basic single-goal single-species evolution is already an established process, demonstrated here in simple terms. The development of techniques for dual-species and multi-goal evolution from this basis is explored.

Once a genetic programming technique is established the success of the evolutionary process depends solely on the selection of fitness functions. Any specific design problem requires the identification of two interlinked systems and analysis of the relationship between the two. Testing based on project-specific criteria can then be developed and carried out.

(Back to the top)

Symbiosis

Symbiosis in the natural world is now recognised as a powerful force for evolutionary change. The full understanding of symbiotic relationships is a relatively recent development in biology, but its importance is now clearly recognised. Albert Schneider, an early pioneer of mutualism, stated that: "Theoretically, there is no limit to the degree of specialisation and perfection that this form of symbiosis may attain. In fact, mutualistic symbiosis implies that there is a higher specialisation and greater fitness to enter into the struggle for existence." (Schneider, 1897).

The evolution of two species towards mutual benefit produces lifecycles linked at all levels. The interaction involved can be very simple, as in the relationship between legumes and bacteria:

> "In leguminous plants there is an association between the roots and a Bacterium ... These bacteria have the ability to fix nitrogen ... Some of the nitrogenous substances are passed on to and utilised by the leguminous plant, which thus has its supply of nitrogen augmented. At the same time the bacteria obtain

> their carbohydrates from the host plant, and so we have an intimate association of two plants from which both members derive benefit. This appears to be a true symbiosis." (Lowson, 1962)

Alternatively, species can evolve extremely complicated behavioural patterns and physical characteristics in order to benefit from a partner, as seen in the fig: fig wasp symbiosis:

> "Figs and wasps occupy the high ground of evolutionary achievement: a spectacular pinnacle of Mount Improbable. Their relationship is almost ludicrously tortuous and subtle. It cries out for interpretation in the language of deliberate, conscious, Machiavellian calculation. Yet it is achieved in the complete absence of any kind of deliberation, without brain power or intelligence of any kind ... It is all the product of an unconscious Darwinian fine-tuning, whose intricate perfection we should not believe if it were not before our eyes." (Dawkins, 1996)

The evolutionary impetus given by a mutualistic co-evolutionary system allows both species to gain advantage through the evolution of characteristics which benefit the partnership.

Single-Goal Coevolution

Symbiosis within architectural genetic programming requires the identification of two related systems. In this case the basic pairing is the association of enclosure and space. This combination provides a way of assessing a GP symbiotic methodology via a simple example.

The genotypes of both species - empty space and solid enclosure - are L-system production rules, giving phenotypes that consist of a list of grid positions. Testing of the phenotypes is carried out on pairs of individuals, one of each species. Crossover of successful individuals is also carried out between pairs: the space individuals crossing over at the same time as their partner enclosure individuals cross. The nature of symbiosis only allows individual fitness to be discussed in relation to a particular partner.

A good enclosure is defined as one which surrounds a maximum of its space individual, while a good space is an individual with a maximum of itself inside the enclosure. Each individual can therefore be given a percentage score; the overall symbiotic score is the mean of these two percentages (figure 6.1).

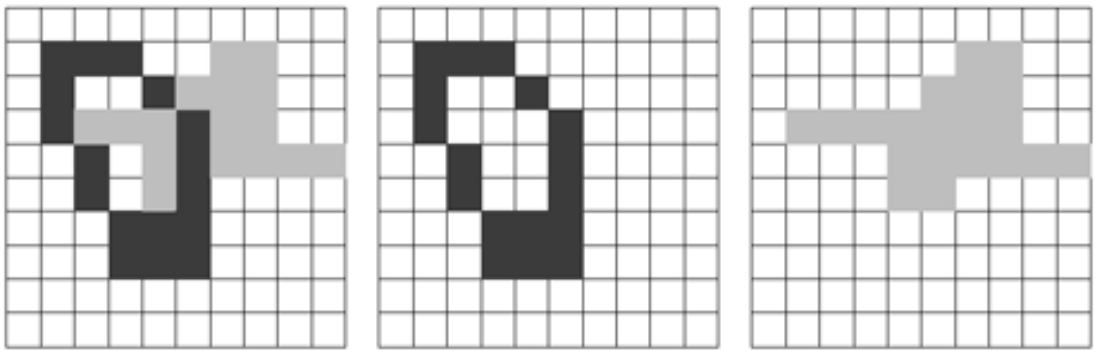


Figure 6.1 Fitness calculation. *Of the nine grid spaces inside the black enclosure individual, five are filled by the grey space configuration. This gives the enclosure a score of 5/9 or 55.6%. The space individual has five of its twenty grid spaces inside the enclosure, giving it a score of 5/20 (25%). The overall score is therefore 40.3%.*

Using this fitness criterion a GP has been developed to evolve symbiotically linked pairs of space and enclosure individuals.

Running the GP in two dimensions using an orthogonal grid led to the best-of-run individuals shown in figure 6.2 (scoring 70%). Each generation contained 32 pairs of individuals and the run lasted for 32 generations. Figure 6.3 shows the progression of scores during the run. The symbiotic process is successfully evolving populations with increasing average fitness.

Figure 6.2 Best-of run pair of individuals (enclosure individual shown in black, space in grey).

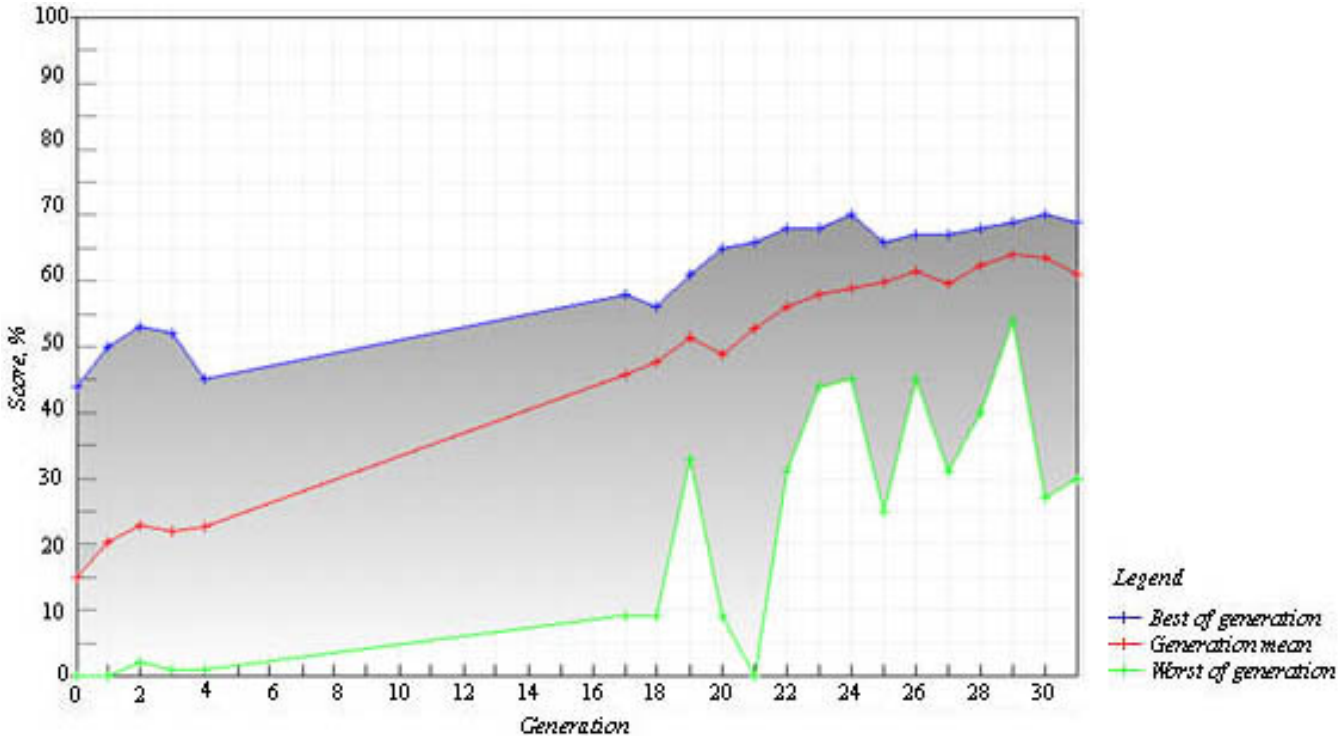


Figure 6.3 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic coevolution between enclosure and space. There are 32 pairs of two dimensional orthogonal individuals in each generation.

It can be seen from figure 6.2 that a perfect solution of the space-enclosure criteria has not been produced. The enclosure individual is highly scoring, being almost entirely filled by space, but the space individual is poor. It is possible that a longer run of coevolution could produce a better solution. Figures 6.4 and 6.5 illustrate a further two dimensional run, this time based on isospatial geometry and continued for 64 generations.

The best-of-run individuals again score 70%, with the space individual scoring badly. It can be seen from the graph (figure 6.4) that average fitness steadily increased throughout the run, but a much longer evolutionary time

will be required before fitness approaches 100%.

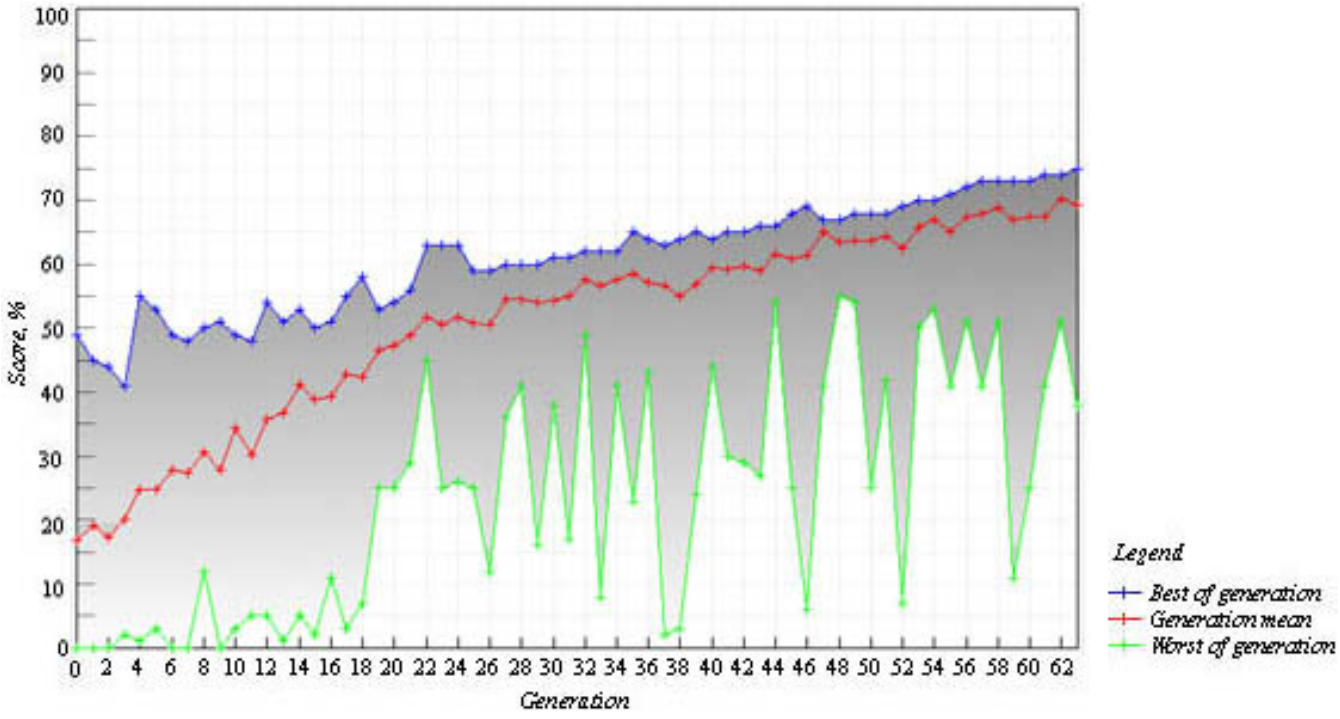


Figure 6.4 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic coevolution between enclosure and space. There are 32 pairs of two dimensional isospatial individuals in each generation.

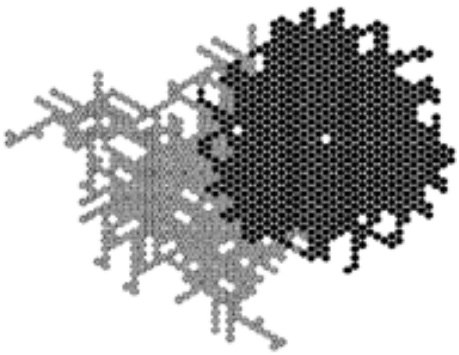


Figure 6.5 Best-of run pair of individuals (enclosure individual shown in black, space in grey).

The enclosure individual, while fitting the criterion given by containing part of its space partner within its small enclosures, is a very inefficient way of confining space. While this might resolve itself on longer runs even using these simple criteria, it would be helpful to insert a requirement for efficient enclosures.

7. Multi-Goal Coevolution

Basic symbiosis criteria are used, with additional criteria used to calculate the fitness of the enclosure individual: a requirement for efficient enclosure and two criteria intended to reduce "bloat".

Bloat

One major problem in the evolutionary runs reported so far is that of "bloat". This is the uncontrollable increase in size of production rule which can occur, leading to ever larger individuals. In the enclosure experiments (section 4), bloat considerably slowed the evolutionary process. In these experiments criteria to penalise very large configurations are added to the fitness calculation.

(Back to the top)

Room numbers

In an architectural context it is likely that a specific project brief will have an ideal room requirement. If the number of rooms is specified at the beginning of a run, huge layouts with far too many enclosed spaces will have a low fitness.

If used as a fitness score, the test result is given as a percentage between 0% and 100%, where a score of 100% implies that the configuration has exactly the correct number of rooms and 0% implies that there are either zero or twice the ideal number of spaces.

Alternatively, this criterion could be used as a multiplication factor between 0 and 1 which adjusts other scores. This may encourage the evolution to focus on other, more desirable, fitness criteria.

Site boundary

Given a context for a built development, site constraints control the building size. If the site boundaries are defined, no building blocks can be placed outside those boundaries. In order to obtain a fitness score, a count is made of the number of times the production rule causes an attempt to place a ball outside the boundary. The fitness results from the percentage of the number of building blocks inside the boundary which would have been positioned outside the boundary. (Score range 0% - 100%, or multiplication factor between 0 and 1).

If more than one criterion is to be applied to evolution there must be a way of combining the individual scores into an overall fitness. The method used may have a large effect on the evolutionary path. There are two major alternatives.

Combined average

The overall enclosure fitness is the mean of the individual scores. This depends on the ranges of possible scores being equal. In the current symbiosis the first enclosure score, relative to the space individual, is a percentage between 0% and 100%. The second fitness, measuring the efficiency of the enclosure, is also a positive percentage. However, the criterion as discussed in section 2.2 has no maximum value. It is proposed that a score of 500, which gives a highly efficient configuration, is considered a maximum and the fitness given as a percentage of this optimum result. (A score of above 500 will be given the fitness 100%).

In the following runs, the ideal number of rooms is specified as 20 and the site is 21.0m by 9m.

Bloat testing as percentage score

The final score of a configuration is given by the mean of the enclosure individual and space individual's scores, where the enclosure individual's score is the mean of the four individual criterion (relation to space individual, enclosure efficiency, room numbers and site constraints).

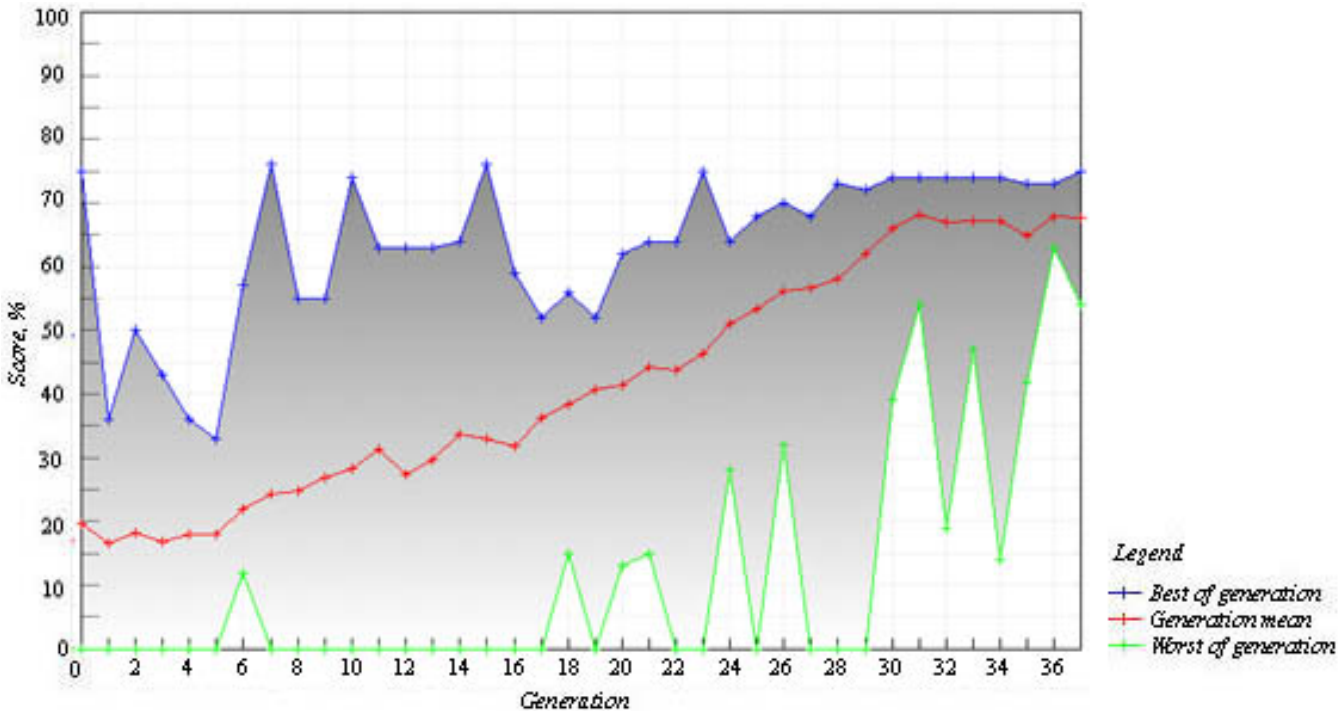


Figure 7.1 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic multi-goal coevolution between enclosure and space, using a combined average score. There are 32 pairs of three dimensional isospatial individuals in each generation.

Figure7.1 shows the scores over a run of 37 generations. The best-of-run enclosure individual, scoring 73% in combination with its space individual, is shown in figure 7.2. The score breakdown shows that the enclosure individual scores well for room and site requirements, but badly in terms of efficiency. This could be problematic if the "bloat" testing criteria are considered to be less important than the other criteria: a pair of individuals can gain a high score purely on the basis of good room count and size.

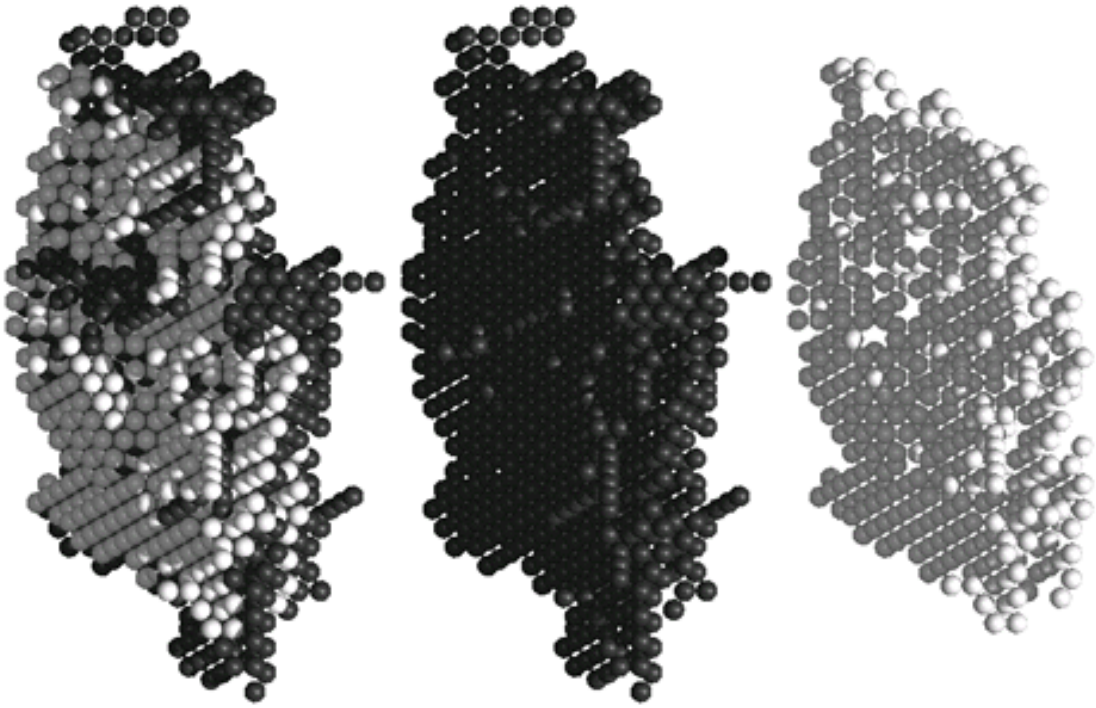


Figure 7.2 Isometric view of best-of-run enclosure individual (enclosure individual shown in black,its internal space in grey). Overall score is 75%; the enclosure individual scores 95% on room count, 73% for compliance with site constraints, but only has 10% efficiency and 53% enclosure of space individual.

Bloat testing as a multiplication factor.

The score for the enclosure individual is calculated as the mean of the two main criteria (relation to space individual and enclosure efficiency) multiplied by the two bloat-control factors. This means that an enclosure individual cannot score well without having good symbiotic and efficiency scores - the “bloat” tests only modify other scores.

The configuration shown in figure 7.3 has a high scoring symbiotic relationship, although the enclosure scores badly on its volume efficiency.

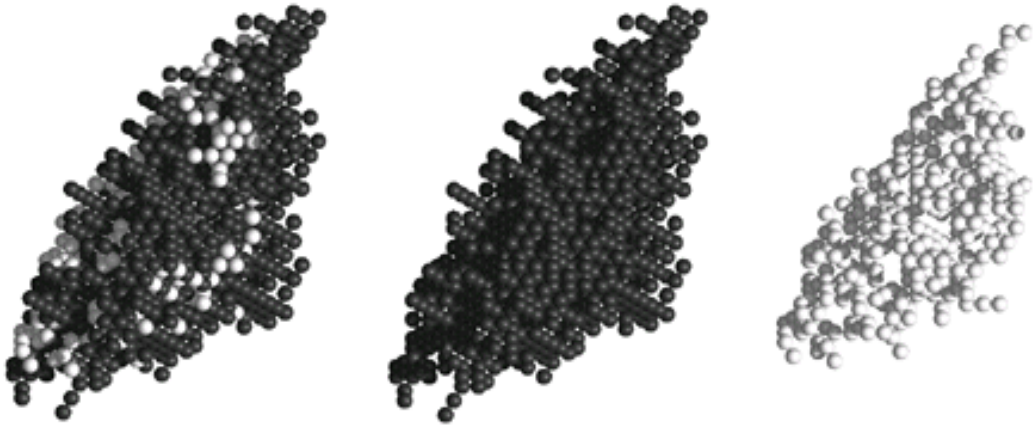


Figure 7.3 Isometric view of best-of-run enclosure individual (enclosure individual shown in black, its internal space in grey). Overall score is 73%; the enclosure individual scores 93% symbiotically, but only 6% for efficiency of the

enclosure. The multiplication factors are 0.95 and 1.00.

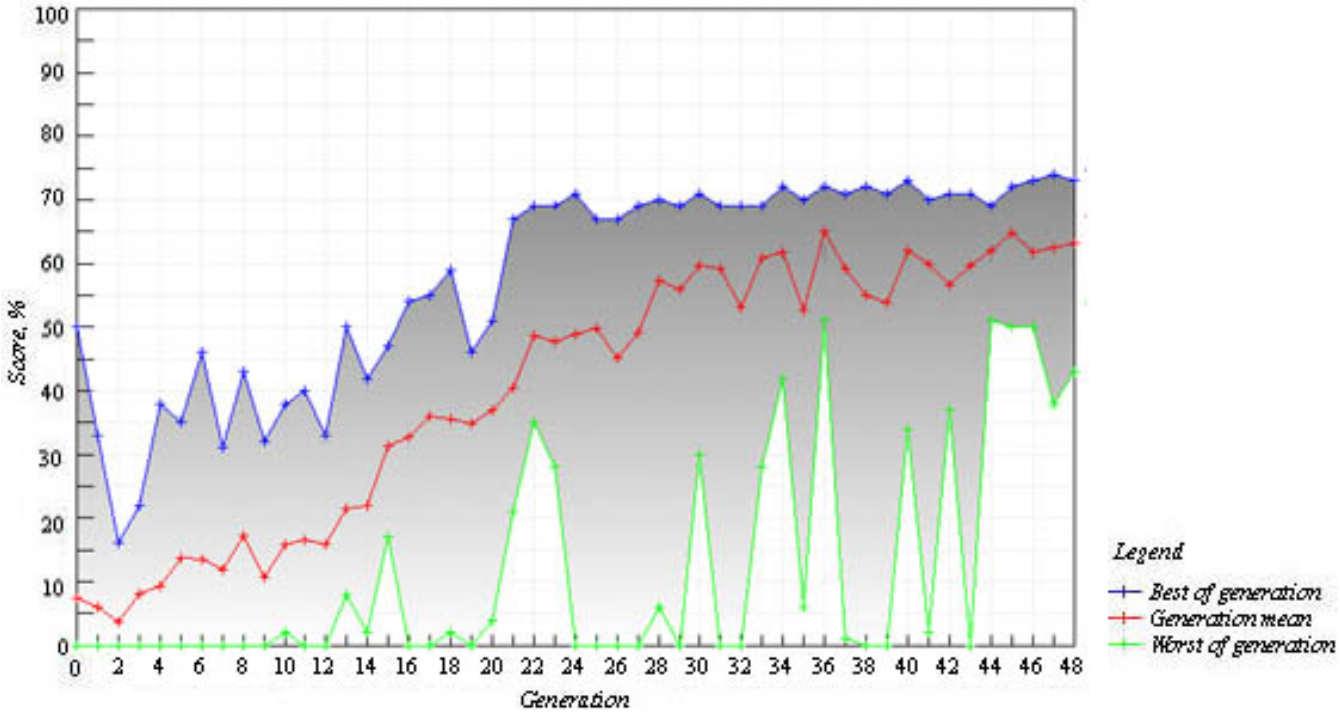


Figure 7.4 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic multi-goal coevolution between enclosure and space, using a combined average score. There are 16 pairs of three dimensional isospatial individuals in each generation.

Maximisation

The potential problem inherent in a "combined average" approach to dual- and multi-goal fitness calculation is that genotypes which score highly in one of the criterion but has poor performance in others could be lost (not bred from). This could result in the permanent loss from the gene pool of potentially useful genes.

Alternatively, in the previous two experiments it was seen that best-of-run individuals were achieving respectable results despite having very low scores from the enclosure efficiency test. This evolution is not truly multi-goal, as it is focusing on one of the criterion to the exclusion of others. Once a run has trapped itself into this focus it hits the "local optima" problem: very high scores will never be achieved but, other than as the result of lucky mutation, a change of evolutionary direction is only achievable through a temporary decrease in scores.

The two criteria which counter "bloat" are required in all configurations in conjunction with one or both of the other two criteria. One technique is to use the mean value when the symbiosis and efficiency scores are similar (within 20% of each other) but to use the higher score of these two in combination with the "bloat" criteria if there is a large difference. The two scores are assumed to have equivalent importance.

Bloat testing as percentage score

The final score of a configuration is given by the mean of the enclosure individual and space individual's scores, where the enclosure individual's score is either the mean of the four individual criterion (where scores for relation to space individual and enclosure efficiency are similar) or the mean of the "bloat" criteria and the

higher of the two main criterion.

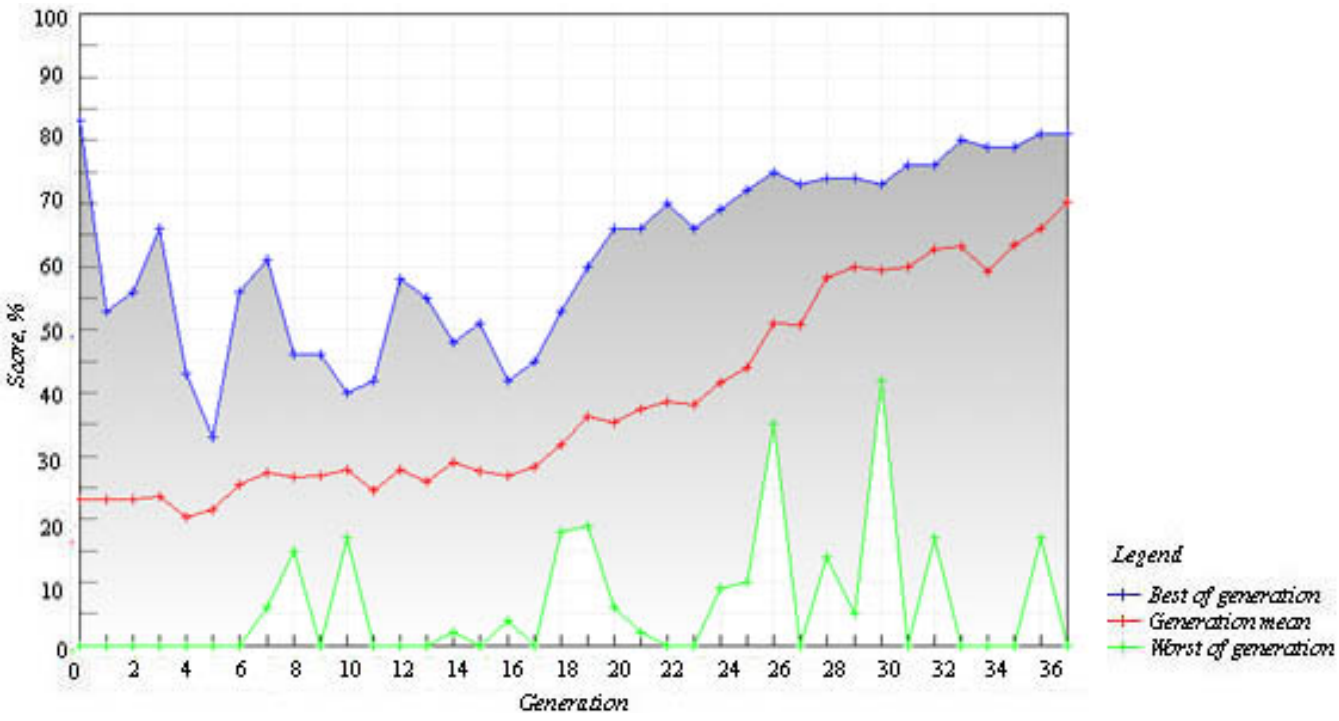


Figure 7.5 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic multi-goal coevolution between enclosure and space, using a maximised score. There are 32 pairs of three dimensional isospatial individuals in each generation.

The best-of-run enclosure individual shown in figure 7.6 scores 81% in combination with its space individual (figure 7.5 shows the score development). However, the evolution is still not functioning equally for all criterion. Three out of the four individual scores are high, but the efficiency of the enclosure is still very low.

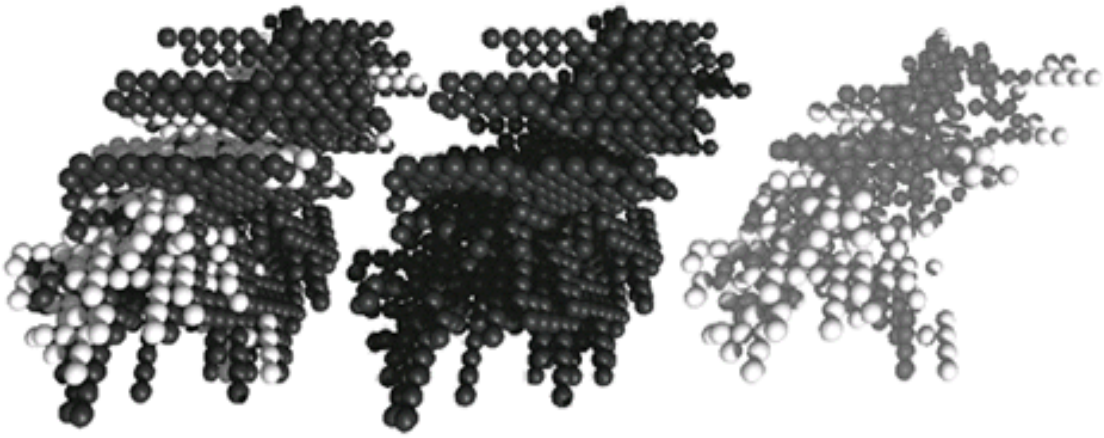


Figure 7.6 Isometric view of best-of-run enclosure individual (enclosure individual shown in black,its internal space in grey). Overall score is 81%; the enclosure individual scores 90% on room count, 99% for compliance with site constraints and 68% enclosure of space individual, but still only has 7% efficiency.

Bloat testing as a multiplication factor.

The score for the enclosure individual is calculated as the either the mean of the two main criteria (if the scores are close) multiplied by the two "bloat-control" factors or the higher of the two main scores multiplied by the "bloat" factors.

As can be seen in figures7.7 and 7.8, the first experimental run using this method became fixed at a local optimum score of 50%.

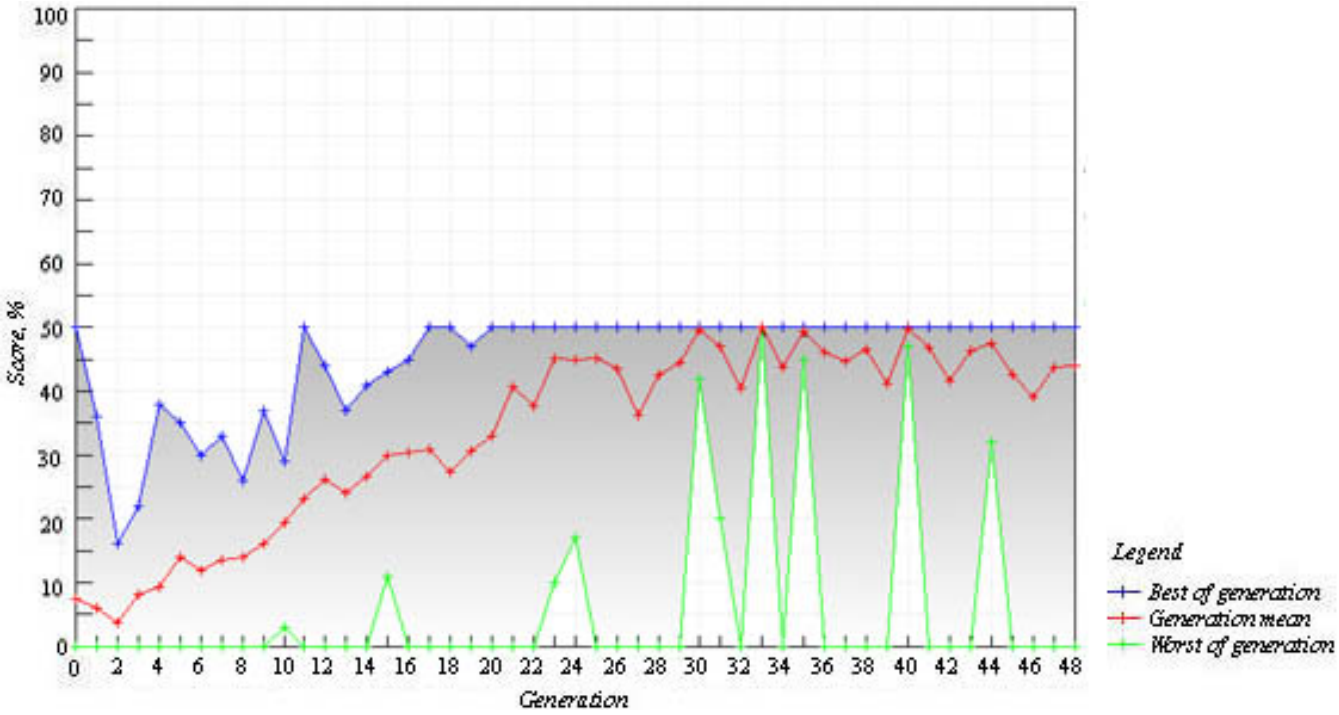


Figure7.7 Graph to show minimum, maximum and mean scores each generation during a GP run of symbiotic multi-goal coevolution between enclosure and space, using a maximised score. There are 16 pairs of three dimensional isospatial individuals in each generation.

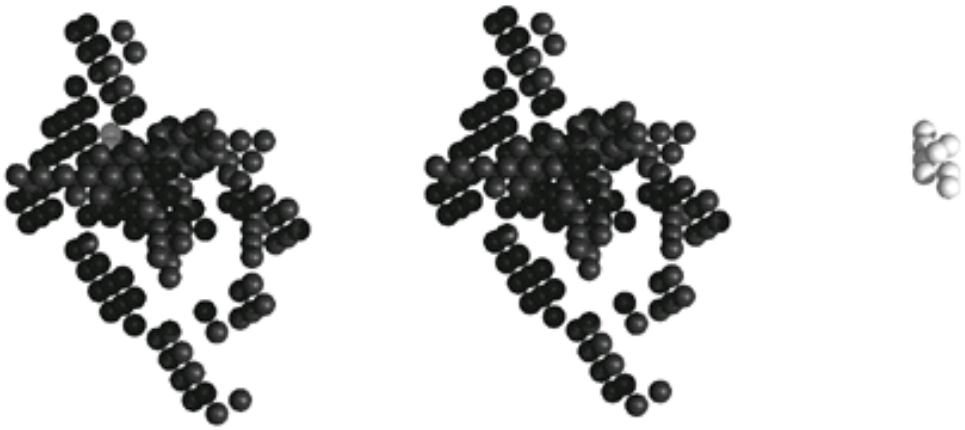


Figure 7.8 Isometric view of best-of-run enclosure individual (enclosure individual shown in black, its internal space in grey). Overall score is 50%; the enclosure individual scores 100% symbiotically, but only 5% for efficiency of the enclosure. The multiplication factors are 0.05 and 1.00.

Conclusions

It has been shown that single goal evolution and coevolution methods can be used effectively to evolve configurations meeting set criteria. However, multi-goal coevolution is more difficult to achieve.

Weighting of scores to suit their importance and attainability appears to be critical. In the experiments shown, it would seem the criteria defining number of rooms and site area are "easily" met, as is the criterion testing symbiotic relationship. These three criteria can be used together effectively. If the two "bloat" criteria are considered of secondary importance, using their scores as multiplication factors only prevents a score from being achieved without depending on the symbiosis result.

The test for volume efficiency of enclosure is more problematic. The runs shown have all displayed increasing average fitness despite low efficiency scores. Each run thus limits itself to a local optimum score, having failed to breed from any genes which could produce efficient enclosures. It seems that this criterion is "harder" to meet than the other three. A weighting system where, for example, the efficiency fitness was responsible for two thirds rather than one half of the overall score might "encourage" evolution towards more efficient enclosures. The exact figures used in such a weighting process can only be the result of (more) trial and error.

Acknowledgements

The original GP approach was developed by **Paul Coates** in collaboration with **Amy Tan**, The L-Systems algorithms were developed by **Terry Broughton** & Paul Coates, and the Space/ Enclosure work is currently under development by **Helen Jackson**. All students on the Msc Computing & Design programme at UEL

References

Bays C.(1988): Classification of semi-totalistic cellular automata in 3-D, *Complex Systems 2*

Bays C (1987).Patterns for Simple Cellular Automata in a Universe of Dense-Packed Spheres, *Complex Systems 2*

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

DeBoer,Fracchia,Prusinkiewicz (1992), "Analysis and Simulation of the Development of Cellular Layers" Artificial Life II ed. C Langton

Dawkins, Richard (1972). *The Selfish Gene*. Oxford University Press.

Dawkins ,R Evolving Evolvability in Langton C. G. (ed.) (1990).*Artificial Life II. Proceedings of the workshop on Artificial Life*. Santa Fe. Feb Addison-Wesley.

Dawkins, Richard (1991). *The Blind Watchmaker*. London: Penguin.

Dawkins, Richard, 1996, *Climbing Mount Improbable*, Viking. Frazer, John (1995). *An evolutionary architecture*. Architectectural Association,London:

Horling B *Implementation of a context-sensitive Lindenmayer-system modeler* Dept Engineering and Computer Science Trinity College Hartford USA

Jacob, C.,(1996)Evolving Evolution Programs:Genetic Programs and L-Systems. *Proceedings of the first annual conference on genetic programming*, Stanford USA MIT Press pp 107-115.

Jo, H. Jo & Gero, John (1995) Representation and use of Design Knowledge in evolutionary design. *CAAD Futures '95*. Singapore.

Kaandorp J.(1994)*Fractal Modeling: Growth and Form in Biology*, Springer Verlag,

Langton C. G. (ed.) (1990).Artificial Life II. *Proceedings of the workshop on Artificial Life.* Santa Fe. Feb Addison-Wesley.

Koza, John R. (1992). *Genetic Programming, on the programming of computers by means of natural selection*. Cambridge, Massachusetts: MIT Press.

Langton C. G. (Ed)*Artificial Life I II & III* Addison-Wesley Publishing Company

Lindenmayer A. and Prusinkiewicz P.*The Algorithmic Beauty of Plants*, Springer Verlag, 1988

Lowson, J.M., 1962, *Textbook of Botany*, University Tutorial Press (revised by Howath, W. O., and Warne L. G. G.). Rowe, Peter G. (1987). Design Thinking. Cambridge, Massachusetts: MIT Press.

Sapp, Jan, 1994, *Evolution by Association, A History of Symbiosis*, Oxford University Press. Todd, Stephen & Latham, William (1992). Evolutionary Art and Computers. London : Academic Press Ltd.

Walker, Miles (1993). *Digital evolutions*. M.Sc Thesis, University of East London.UK