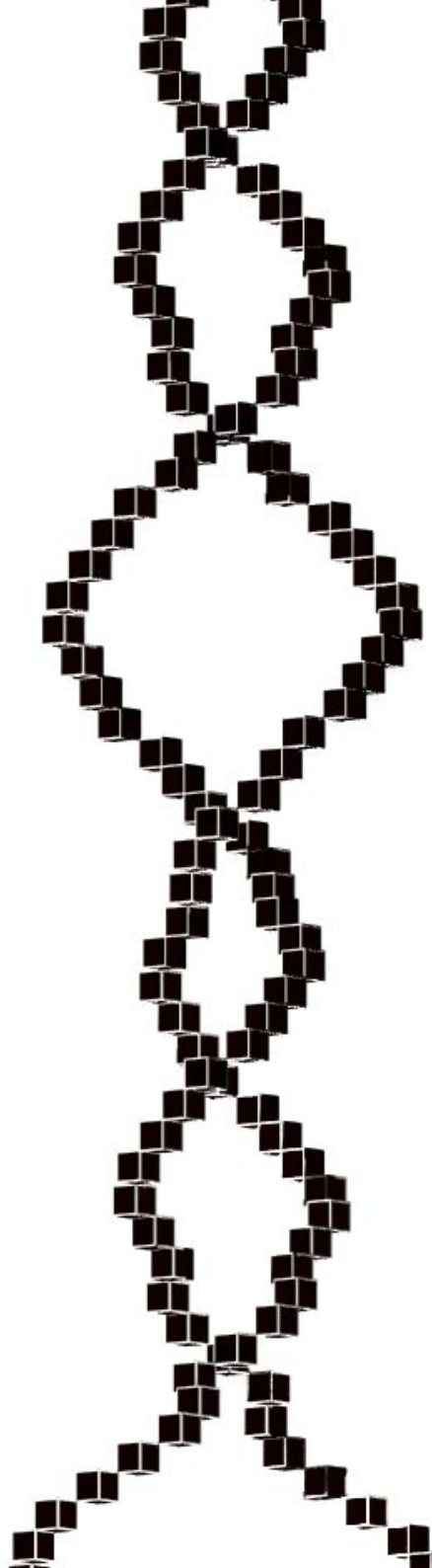


Gennaro Senatore

Morphogenesis of Spatial Configurations
MSc in Architecture, Computing & Design 2007/2009

Tutors:

Paul S. Coates
Christian Derix
Tim Ireland
Manos Zaroukas

Software engineer:

James Galasyn

Acknowledgments

It has been an intense and challenging learning experience which broadened my mind widely and overturned the way I conceived design. My thanks go to ***Paul S. Coates***, to whom this work owes its intellectual debts. His “open-source” character and his tireless willingness to keep a colourful debate with his students, encouraged me to undertake this adventurous endeavour. To ***Christian Derix***, whose teaching and help were fundamental to engage with the world of design - computing. To my tutors, colleagues, and friends ***Emmanouil Zaroukas*** and ***Tim Ireland***, who personally assisted me and whose tutoring and contribution have been essential for the development of this work. To ***James Galaysn***, without whom I would have never be able to conduct this experimental research. He provided me with a robust framework to work with and his patient advices made me able to extend it to fulfil my aims.

Abstract

The thesis discusses the possibility to build a design methodology based on the utilization of generative rule systems, evolutionary techniques and performance evaluation tools. This is conducted through the implementation of a computational system in which spatial configurations can be defined by a set of instructions framed in hierarchical data structures. These are graphically interpreted to create their geometrical expression which is evaluated according to design criteria such as structural stability and accessibility of its basic components. String rewriting Lindenmayer systems and Genetic Programming are combined to first create the definition of the problem and afterwards evolve possible answers to it. The aim is to integrate the generation of forms and the evaluation of their performances in order to design spatial configurations whose morphology emerges out of a process of formation rather than imposed by a fixed and predetermined representation. The adaptation of such rules system under the pressure of encoded design criteria resolves in the emergence of forms which become, hence, the expression of high level of abstractions.

Index

Intro

Since “Haring’s functionalist philosophy” (Steadman 2008, p.238) and Sullivan’s dictum “form follows function” (Sullivan 1896), formulated at the beginning of last century, many architects and engineers have used design methodologies seeking to generate form as an emergent property of what being designed, rather than imposing a priori its external appearance. Although these concepts were partially misinterpreted by modernist movements, leading to an architecture based on “spatially programmatic requirements” (Dollens 2006), they constitute the foundation of what has recently been defined as “organic architecture” (Steadman 2008).
Starting from Sullivan’s organic manipulations (Sullivan 1967), passing from Gaudi and Frei Otto’s form finding, Alvaro Alto’s biodynamism, Eisemann’s destructivist exercises, Stiny’s shape grammar, Hillier’s syntax of space, Lynn’s calculus based parametric tools, form started following forces rather than just functions. Forces are to be intended in the broadest meaning of the word, being the physical, social and economical environment modelled as a series of force fields. The force field can be abstracted and considered as the result of the interaction between the object being designed and its environment in a process of morphogenesis. Connectivity, materiality, the embodiment of parts-to-whole relationships and the structuring of their hierarchy, are at the base of the design methodology.
The advent of complexity theory and the availability of computational resources, has lead to the simulation of processes of formation based on evolutionary techniques, reaction diffusion mechanisms and developmental growth models . Form here comes out of a gestalt process that is much closer to biological development of an organism, where algorithms are able to “grow” and evolve design concepts (Steadman 2008).
Even beyond this, architects such as John Johansen envisages an architecture where the advances in nano-molecular engineering will allow for structures that literally build themselves, being self organising and completely responsive to their environment (Johansen 2003).
It is upon this theoretical background, the work of Paul Coates & coworkers at CECA (Coates et al. 1999), James Galasyn (Galasyn 2008) and Peter Bentley (Bentley 1999), that the thesis presented here builds its foundation. Coates & coworkers first pioneered and built the basis of the conceptual armour this work moved from. The vivid debate with prof. Coates at the Center for Evolutionary Computing in Architecture CECA (University of East London) generated my interest for the field of generative modelling whereby truly form-finding processes can be experimented.
Software engineer James Galasyn has been developing an experimental software platform for generative modelling which is based on concepts first experimented at CECA. His consultancy and advices gave me the capacity to extend his system to start this adventurous endeavour in the field of evolutionary design.
The conversation with computer scientist Peter Bentley and the reading of his work (Bentley 1999, Bentley 2002, Kumar and Bentley 2003) has broadened my understanding of generative design. It shed light on the limitations that belongs to the system that I’ve been developing and showed possible future ways to follow.
Evolutionary Design can be framed within the larger context of the field of Artificial Life. Artificial Life is not a unitary field of research, it takes contribution from Computer Sciences, Biology, Philosophy, Physics, Mathematics and Cognitive Sciences. At the very heart of this field is the attempt to synthesizes artificial systems with lifelike behaviour. The approach is mainly based on abstracting the logical principal of living organisms and the implementation of these “through synthesis on another medium”(Heudin 1995). Being the field of Biology the main source of inspiration, Evolutionary Design borrows from it, along with the Darwinian theory of evolution, also its terminology. For this reason, the value of a variable for a problem becomes a “gene”, while all the admissible values that the variable can take are the “alleles”. The way in which all the variables concerning a problem are connected is , hence, the “genome”.
The thesis discusses the possibility to build a design methodology based on the utilization of growth models, evolutionary techniques and performance based evaluation tools. A model that simulates the growth of an object can be thought of as a sequence of instructions that have to be executed to create it, starting from its basic components. The evolutionary process will explore the dominium of possible ways in which the components of the system can relate to each other. Performance based evaluation tools will simulate the presence of force fields acting on configurations under development, hence, triggering digital morphogenesis. In this way the system is made capable to develop its own structure, establishing the relations between its components.
I will start by giving a small introduction to Genetic Programming and to evolutionary techniques, speculating on the possibility for these systems to act in a creative manner during the exploration amongst possible design proposals.
I will then describe L-Systems, being the growth model adopted, and its combination with Genetic Programming. They cooperate towards the building of a system that does not require the description of any afore hand geometrical entity but rather the specification of the topology of what being designed.
It follows the description of a load propagation model (Galasyn 2008) that is able to assess the structural behaviour of the generated configurations, which are exposed to the action of a physical force field such as the gravity. In this way the system can rely on the evaluation of intensive quantities to navigate through the solution domain.
Finally, by reporting some of the experimental results obtained so far, I will highlight the capacity as well as the limitations of the system.

This work explores the utilization of Artificial Life techniques and performance evaluation tools within the architectural design process. It seeks to integrate the generation of forms and the evaluation of their performances in order to design spatial configurations, whose morphology is emergent rather than superimposed. It aims at defining “geometry” through “topology” where geometry means the set of features that describe the appearance of the artefact and topology is the set of relations between its components. Starting with an ill-defined or even absent definition of the form, the relations between its constituents are instead embedded in the system. These can be thought of as generative rules, whose variation can be put under pressure by means of evolutionary techniques, in order to explore a domain of possible solutions to a problem.
The design process is considered to be objective oriented, where the objectives are defined by performances that have to be evolved. The objective criteria mainly serves as feedback to understand the adaptation of an evolved configuration to changes in its topology. “Generating forms and having a simultaneous feedback on their behaviours, under real design constraints, can help to consider non conventional configurations of space” (Khristina Shea, cited in Kolarevic 2003).
This constitutes the foundation for a design methodology that allows for an efficient exploration of viable alternatives, while proposing solutions that are consistent with design constraints.

Past research works

Over the last two years, I have been mainly working on Artificial Life and Physical Form-Finding techniques aiming at building the base for a methodology whereby their integration with performance-based simulation tools can be achieved.
Fig1, fig2 and fig3 show some of the outcomes generated by an algorithm that I developed last year in May 2008. It mainly consists of a procedure to encode the “body plan” of the objects being designed, and a classic steady state Genetic Algorithm. The objects are generated out of key parameters, whose variation creates different geometric expression of their topology. By means of the Genetic Algorithm it is possible to vary these parameters and test the generated configurations against an environment. In this way the objects are evolved, generation over generation of attempts, reaching better performances, which also reflects the expression of new forms. The “environment” can be thought of as a set of design criteria that serves to evaluate what is generated in order to explore efficiently the solution domain. The solution domain is the universe of all possible combination of the “key parameters”.
We could say that the generated configurations, over generations of attempts, become expression and virtual representation of the design criteria (Delanda 2002).
For detailed explanation of the procedure visit http://wiki.uelceca.net/msc0809/published/GA+-+Solar+Gain and http://wiki.uelceca.net/msc0809/published/GA+-+Multi+Objective.

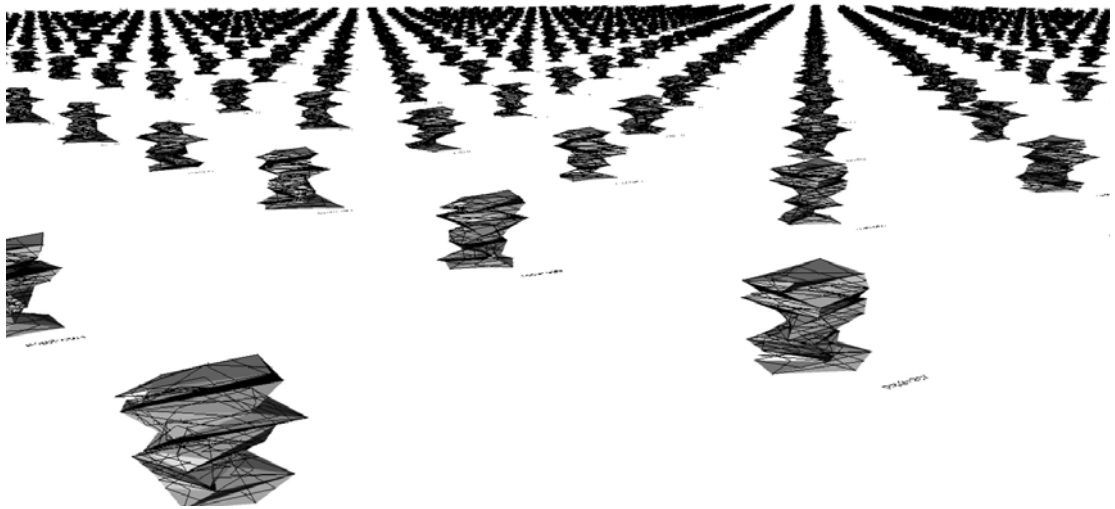


Fig 1 generation after generation

Although fig1, fig2 and fig3 show objects whose topology differs greatly from each other, they are generated by using the same mechanism, changing only the way to encode their body plan. It is worth highlighting the main shortcomings of this piece of work which constitute part of the theoretical impetus that has moved the current research:

• using Genetic Algorithm and an external encoding of the topology, it is not possible to evolve the body plan. Once the body plan is encoded, the system will explore possible expressions of the same topology but will never be able to change it, or to create a new one. This means that the user will never see something that he/she could not expect.

• with regard to performance-evaluation tools, it must be said that time and resources should be invested for the development of their light-weight or "low-resolution" version, in order to be used at the conceptual level within the architectural design process. A programmatic approach using vector algebra can be a substitute to Finite Element Method as well as Smooth Particles Hydrodynamics an alternative to heavy Computational Fluid Dynamic simulations.

• the absence of any link with the material system. The embedding of a procedure that simulates propagation of loads, will open the way for introducing the evaluation of intensive quantities such as distribution of stresses or energy of deformation. These can be used as design criteria in order to explore the solution domain while, at the same time, developing a meaningful structural system.

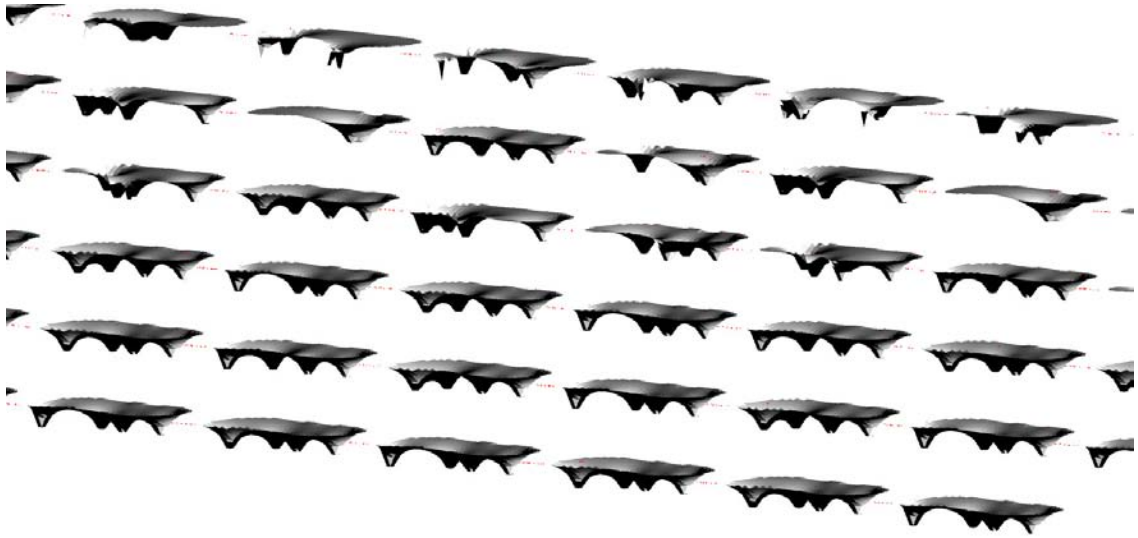


Fig 2 generation after generation

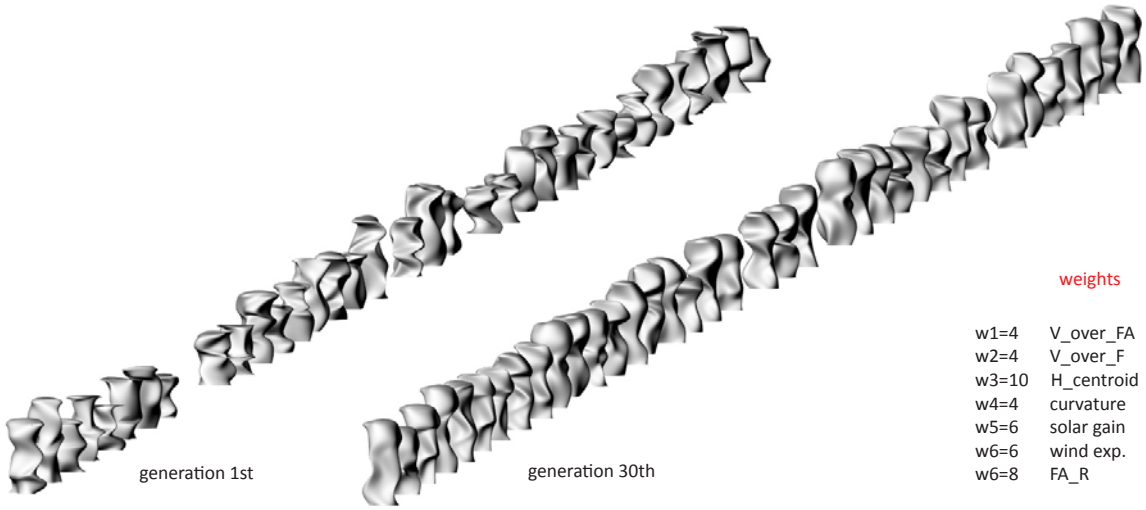


Fig 3 generation after generation

On Evolutionary Design

Evolutionary Algorithms started being developed in the early 60s in Computer Science. Those methods are all based and inspired by the analogy to natural evolution. Evolution is a good general purpose problem solver which means that it can be applied to a wide variety of design problems without the need to develop ad hoc specialised procedures (Bentley 1999). The main activity in Evolutionary Computation is "searching", having any general problem a search space or solution domain of possible solutions to it. We can think to encode a problem in a set of parameters whose values correspond to a solution, every legal set of parameters correspond to a point in the search space. What an evolutionary system mainly does is indeed to navigate this search space visiting different points, searching for good solutions. "Evolution" is not explicitly encoded in these methods but it just happens by using a simple mechanism. Working with population of individuals (possible solutions) rather than just one at time, these systems usually keep the good ones and recombined their genotypic representation in order to pass next generations their good "genes" . Keeping the analogy with biology, the word "gene" here is used instead that parameter of the problem, its value could indeed be called "allele". This simple mechanism is sophisticated enough to ensure the production of better performing individuals over generations. All Evolutionary Algorithm require a steer to drive the evolution to desired places of the solution domain. This is done by evaluating all the generated individuals and assigning them a score, which is usually called "fitness". It is, hence, possible to rank them according to their fitness favouring the emergence of better solutions during the evolutionary process. Due to the fact that population of individuals are evolved, these methods perform a searching that is massively parallel (Holland 1992). Even for a small set of parameters describing a problem, the combinatorial range can reach significant proportions. The time that is required, if one wants to try all possible combinations in a serial manner, would be too long, if not infinite, to be feasible. A parallel search, instead, is able to cut off the required time to a reasonable amount (Coates 2010).
The four main type of evolutionary algorithm are :

• evolutionary programming (EP), one of the first evolutionary algorithm created by Lawrence Fogel in 1963
• evolution strategies (ES), created by Ingo Rechenberg in 1973
• genetic algorithm (GA), created by John Holland in the early 70s and further developed by Goldberg
• genetic programming (GP), invented by John Koza in 1992, (subject of this work)

All EAs perform reproduction of the generated individuals either by cloning or using recombination, which could be also followed by small random mutations. In this way the new individuals can inherit the characteristics from their parents whilst ensuring a certain degree of variation. They all perform "selection" to determine which solution will have more probability to transmit next generations its "genes". Selection can be considered as the evolutionary pressure exerted by the virtual environment in which the individuals are placed.
Generally speaking the structure of an EA can be divided in five main parts:

• ***initialization***
• ***mapping***
• ***evaluation***
• ***selection***
• ***reproduction***
• ***termination***

Initialization consists in seeding the initial population with random values. In this way random solutions are generated which are ready to be evaluated. In some cases instead than using random initial solutions, nonrandom initial "embryos" can be used as a starting point. It is worth remembering that in the case of Genetic Programming (subject of this work) the solutions/individuals do not have fixed structure. This means that this technique allows not only for the evolution of the values of the parameters but also of their number and the type of relations that exist between them. This means that it allows the evolution of the structure itself rather than just the details (Bentley 1999). The initialization in this case requires the seeding of both the "structure" and the "values".
Mapping is a process that comes in when there is a neat distinction between the search space and the solution space. Mapping involves the decoding of the instructions enclosed in the genotype to produce the phenotype. Not all the EAs perform mapping because many of them do not make distinction between genotype and phenotype. Recalling the biological analogy, mapping can be associated to "embryogeny". Implementing "embryogeny" , although can be very complicated, allows highly complex phenotypes to be expressed by very compact genotypes. This reduces the dimension of the search space which increases the efficiency of the EA.
When mapping, genotype can be thought of as a set of instructions that define how the phenotype will be expressed. Further more, instead that defining directly the instructions, it is possible to implement the way these instructions are generated.
Evaluation is the way to recognise which solutions are better than others, to drive the evolution towards desired regions of the search space. Often the evaluation of the individuals is performed by dedicated software analysis which can require many hours for the examination of just one individual. How the results of the evaluation analysis are used and elaborated to assign a score to the individuals, it is specified by the fitness function. A fitness function can be single or multi-objective, static or dynamic. Often human interaction is used as fitness function itself or as part of it. Evaluation and fitness function represent the virtual environment which exerts certain forces on the individuals.
Selection is performed to preserve already known well performing solutions while discovering others good ones. The preservation of good traits necessitates a

method of selection that is able to discern to which extent information are good or not. The most common strategies to specify parents selection are :

• *tournament selection*, this strategy combines random selection and performance based evaluation. First there is a tournament of individuals that are selected randomly, amongst which the best performing is selected.

• *fitness ranking*, selection is not based on the actual distribution of fitness but rather on fixed range of values that determine the classification of the individuals. If the fitness value, for a configuration, lies in between the limits of a rank, it will be assigned the fixed score for that rank . This method ensures the tendency towards better configurations but does not allow the discernment of small similarities amongst the individuals in terms of fitness.

• *roulette selection*, this strategy provides a method of selection where the probability for an individual to be chosen is proportional to its fitness value. It gives a high probability to fitter individuals but also leaves a certain possibility to the less fit.

• *elitism*, in contrast to the previous strategy, elitism entails the copying of a certain number of best performing members into the new generation. In this way good qualities will not be lost through mutation and cross over (breeding). These members remain unaltered until better performing individuals have been found, acting as a sort of source material.

Along with parents selection, pressure can be exerted by other methods such as "*fertility*", "*replacement*" and "*death*" (Bentley 1999).
Fertility is the number of offspring that parent can have. Selection can be based also on this value rather than just on the fitness score. Most likely fitter individuals will also have a higher degree of fertility.
Replacement can be performed by comparing the fitness values of new individuals with the ones of older generations. If the offspring has a higher value, it will be replaced in the population, taking the place of its parent.
Death involves the elimination of a solution due to the non satisfaction of certain essential design criteria. It is also designed to prevent very fit individuals to persist too long and corrupt the evolution causing fast convergence.
Reproduction is the action of manipulating the selected individuals in order to create next generations. It is usually performed by recombining the genotypes of couple of parents, and mutating to a certain degree the newly obtained offspring's genotypes. It can be also done by mutating a copy of the selected parent. A good reproduction technique must be able to partially preserve useful traits of the parents while ensuring a reasonable degree of variation.
Termination can be based on different kind of considerations. The system can stop running because the satisfaction of a certain set of design criteria has been achieved or due to the reaching of limits in computational resources.

Another reason can be a very small rate of improvement in the fitness value, which means that convergence or undesired local optima has been reached.

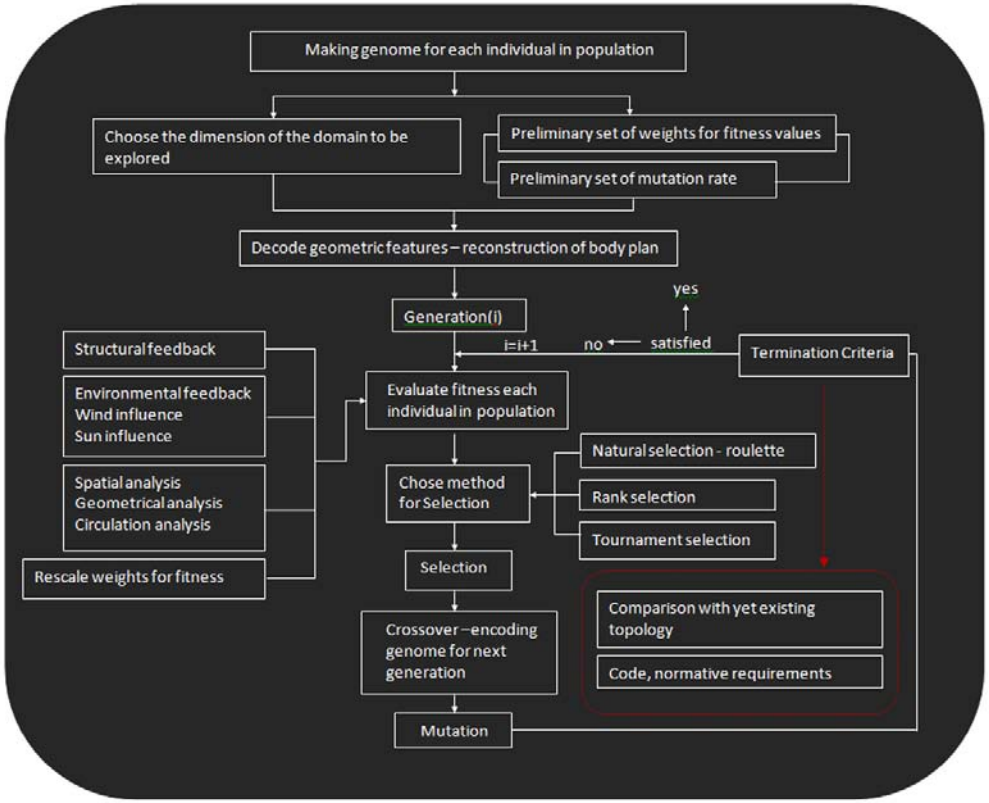


Fig 4 the general scheme for Genetic Algorithm

Genetic Algorithm, a brief description

Genetic Algorithm follows the structure outlined in the previous paragraph. There is usually a clear distinction between search space and solution space so that at each genotype correspond a phenotype. Genotype are initialised with random alleles, phenotype are produced and evaluated. After that, the individuals are ranked according to the type of selection technique and recombined to create next generation. This repeats until termination criteria are met. Usually the alleles of the genotype have the form of strings of binary numbers of a certain length, which can be decoded in order to represent any sort of information. Fig 5 shows how the cross over operation is performed between the genotype of two selected solutions. This operation randomly allocates genes from parents to offsprings by swapping selected parts of their genotypes. Fig 6 shows a series of generations where the configuration to be evolved is a circular surface who can represent a n abstract reprxesentation of a high rise building. In this case the alleles of the genotype are the coordinates of points that are used to describe those surfaces. The environment for these individuals is a series of fitness functions which test them against different design criteria. For instance, one of these criteria is the maximization of absorbed solar radiation during the winter or the minimization of exposure to wind in the peak period of the year. The last two generations in fig 6 show the coloured map of absorbed solar radiation and pressure caused by the wind, projected on the external surface of the individuals. There can be many other design criteria to be introduced, and many are the techniques available to combine them. Sophisticated procedures involve the use of Pareto Optimality. One of the simplest is to scale each value, using the size of the correspondent solution domain, in order for them to be expressed in terms of percentage. A set of “weights” is then introduced in order to have the possibility to bias the evolution to fit different scenario. When having more than one design criteria, Genetic Algorithms take the name of Multi Objective.

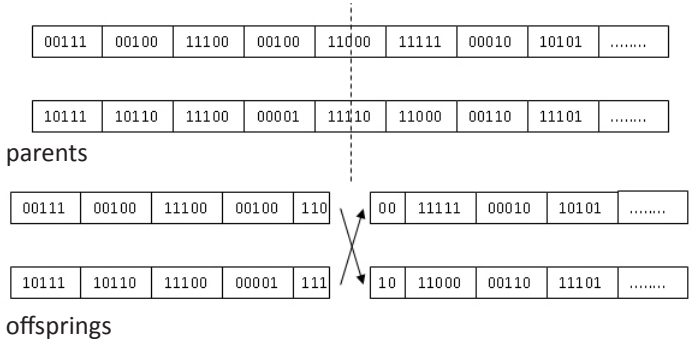


Fig 5 cross over

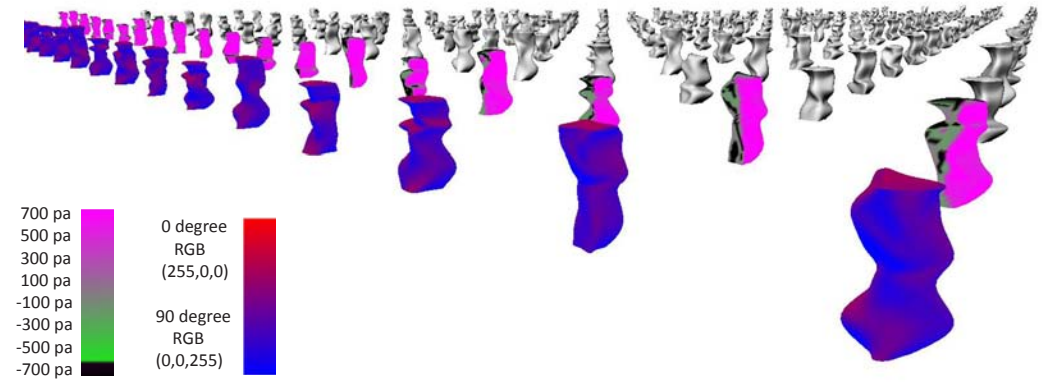
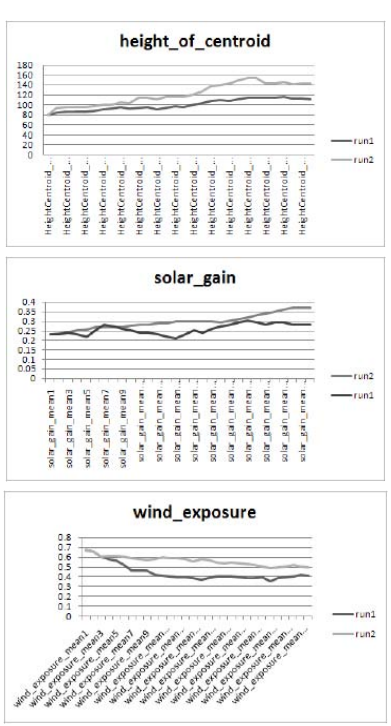
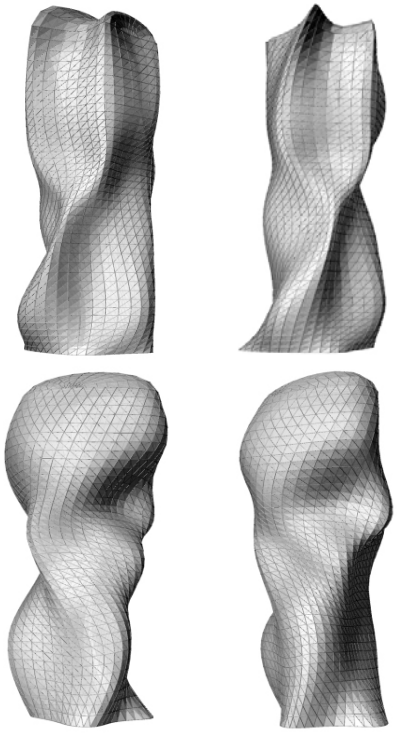


Fig 6 solar gain and wind exposure





run1 weights

w1=2 V_over_FA
w2=2 V_over_F
w3=10 H_centroid
w4=1 curvature
w5=2 solar gain
w6=10 wind exp.
w7=6 FA_R

run2 weights

w1=3 V_over_FA
w2=5 V_over_F
w3=10 H_centroid
w4=5 curvature
w5=8 solar gain
w6=4 wind exp.
w7=6 FA_R

Fig 7 different outcomes by varying the inflluence (weights) of the design criteria

Introduction to Genetic Programming

Genetic Programming can be considered as a generalization of Genetic Algorithm. The main difference is that the genotype of the individuals are represented as a tree data structure which replaces the sequential string of binary numbers used in Genetic Algorithm. The data structure mainly contains the rules that define the developmental process of the configurations. In this way there is not direct encoding of the body plan and the system can develop autonomously its own hierarchy. When cross over is performed for two selected individuals, one of the node is swapped with another node from another individual,hence, replacing an entire branch. Fig 1 shows the expression for a balanced tree data structure. This tree has a depth equals to 3 which indicates the number of times (in this case only two siblings for each branch but there can be more) it branches out. A node can be either an “operand” or an “operator”. An operand is the argument for the function performed by the operator. At the very end of the tree there can be only operands (also called terminals), which in this case are just numbers. The nodes that lie on the lower level branches, can be instead both operands and operators. Respecting the hierarchy of the tree, one can unfold from its “leaves” to its root expressing the phenotype that is encoded in it. In this particular case, the operands are just math functions such as addition or subtraction which take two arguments each. Shifting from numbers to geometrical entities and from math function to procedures that perform action on the terminals, it is possible to build a system that creates spatial configurations.

One of main limit of GA was the fixed size length to which the genotype was constrained to be. This entails the impossibility for the mapping procedure (the developmental process), which is the set of rules that operate on the genome for producing the phenotype, to evolve. The hierarchical and variable structure of genotype in Genetic Programming, naturally leads to the emergence of new (from the initial random set of solutions) organised structures which can increase their complexity over generations.

However, it must be said that different techniques have been developed to allow variable-length chromosomes also in Genetic Algorithm (Bentley 1999).

As for GAs, the representation of the environment is again the key factor. In contrast to GA, the solution domain is not only constituted by the variables that are encoded in the genome but also by the domain of possible functions (operand) that can operate on the variables.

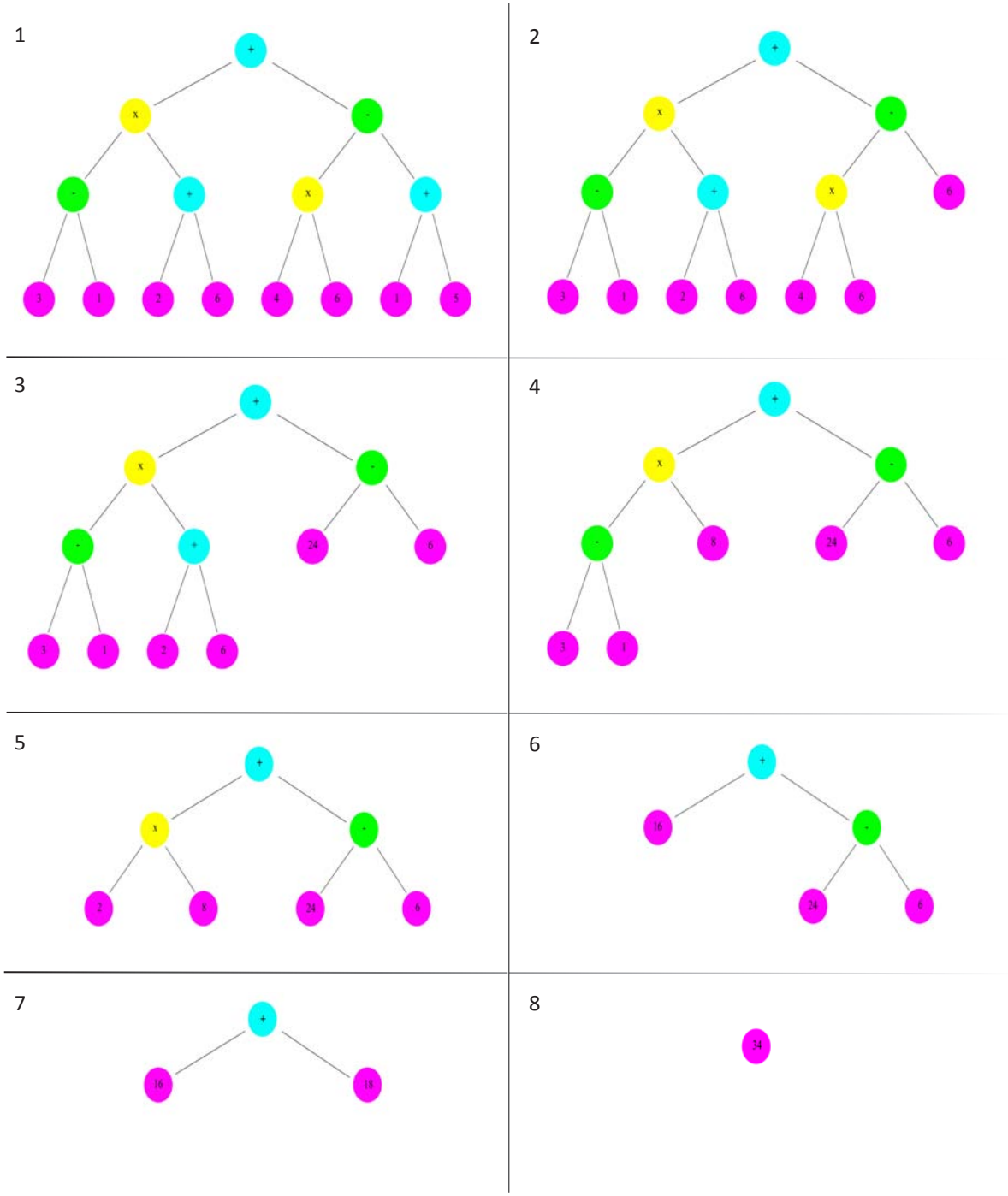


Fig 8 representation of the genotype as a tree data structure

Computational Embryogenesis | three types

The development of an animal or a plant from zygote to its birth and, consequently, the emergence of its form, are mainly due to the generation of ordered spatial patterns of cell activities (Bard 1990 cited in Kumar and Bentley). Cells become specialised for a particular activity through process of *cellular differentiation*, the formation of patterns of cell activities causing *morphogenesis*. "The way in time and space these processes occur is called **embryogeny"** (Kumar and Bentley). Evolutionary computation has developed forms of embryogeny which borrow the key concepts from embryology.
Mapping is the procedure whereby the genotype is decoded into the phenotype. There can be no mapping at all, which is the case when the alleles of the genotype are directly transformed in values for the phenotype (there is not distinction between the search space and the solution space) or very sophisticated techniques can mimic biological "embryogeny". Genotype and phenotype should effectively enumerate the search space and the solution space, in order to identify coherent regions in the solution domain. As a rule of thumb, small changes in the search space should produce small changes in the solution space (Bentley 1999). In other words offspring should resemble their parents for the evolution to rely on inheritance. If the genetic operators cause too disruption and the search space is too discontinuous, evolution deteriorate into random searching.
Advanced form of mappings, also called "embryogenesis", are able to grow complex phenotypes starting from simple genotypes. Advanced forms of embryogenies use also recursion, in order to promote the emergence of features such as symmetry, segmentation, and adaptation. Embryogenesis can be classified in three main types:

- ***external***
- ***explicit***
- ***implicit***

External embryogenesis happen when evolutionary systems use fixed and non-evolvable mapping procedures to decode the genes into the parameters of the solution space. This is mostly the case of evolutionary optimization systems. Fig 9 shows an example of this type of mapping. Genes are decoded in numerical values that represent the coordinates of points in 3D space. These points are used to make sections which are successively lofted to make a cylindrical surface.
This type of mapping will remain static and unchanged over generations, not allowing new topologies to emerge.
Explicit embryogenesis is constituted by a series of instructions which are organised as a data structure. These instructions are interlinked and organised according to a certain hierarchy. Phenotypes are grown by following these instructions, which can contain any sort of data, recursive procedures, conditional statements

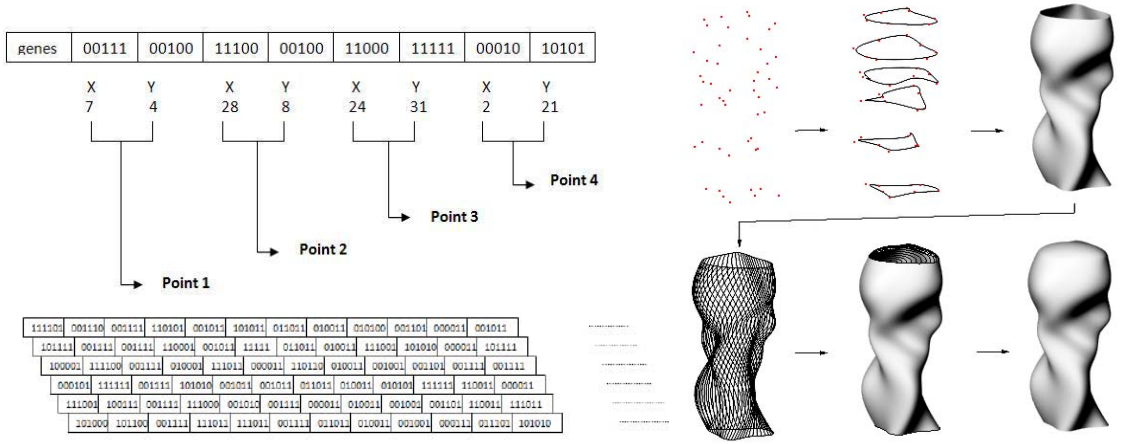


Fig 9 external, non-evolved embryogeny

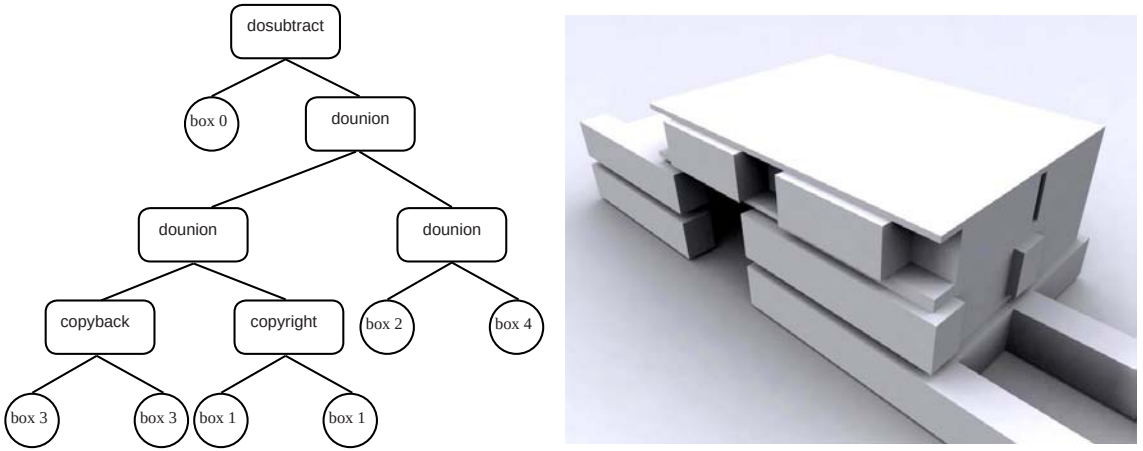


Fig 10 explicit, evolved embryogeny

and even entire subroutines. Explicit embryogeny is usually performed using Genetic Programming as generative engine because of the inherent tree data structure nature of the genotypic representation. Being the structure of the genotype under evolution, embryogeny develops autonomously, without the need to be designed by hand. This allows the emergence of adaptive mapping from genotype to phenotype as a series of interlinked instructions which has a certain hierarchy. Fig 11 shows an example of tree data structure genotype used in one of the experiment carried out by Coates (Coates 2010). Each node can be an instruction such as "boolean add" or "boolean subtract" which takes some arguments, in this case 3D geometrical entities like boxes. Starting from the leaves of the tree, the command are executed respecting the hierarchy of the data structure. In this way it is possible to map the genotype to obtain 3D spatial configurations such as the one shown in fig 11, right side. Evaluating and recombining these type of structure gives, over generations, the possibility to evolve different topologies of spatial configurations.

Although this type of mapping allows the emergence of an evolved hierarchy in the genotype of the individuals, complex procedures such as recursion or iterations need to be defined explicitly (Bentley 1999). This means that with explicit embryogeny and Genetic Programming, it is possible to evolve the connections between the instructions but no instruction can emerge autonomously. A set of functions (the instructions) and terminals (the arguments of the functions) is defined a priori. Through evolution, a hierarchical structure that builds upon these instructions will emerge. By changing the relations between its constituents and even changing the dimensionality of the solution domain, this type of evolutionary system explores multiple topologies in parallel.

Implicit embryogeny does not follow the step by step specification of the growth process through instructions, it uses a series of interacting rules whose activation is not predetermined but highly parallel and adaptive. The embryogeny of a natural organism is characterised instead by a many-to-one relation between the genome enclosed in each cell of the organism to its phenotypic manifestation. "Many-to-one genotype-to-phenotype mapping has been identified by numerous researchers as an important property for evolvability "(Kumar and Bentley). Borrowing the concept from natural embryology Peter Bentley and coworkers (Kumar and Benley) have developed a form of implicit embryogeny which uses an advanced variable length genotype GA as evolutionary machine. Each cell, which assumes the form of a 2D or 3D geometrical entities in a CAD environment, has its own genome constituted by a variable number of rules. Each rules is itself made of a "precondition" section and an "action" section. The precondition section stores information that regard the neighbourhood of the cell such as which neighbours position is filled or empty. For a rule in a genotype of a cell to be activated, a number of preconditions need to be matched by the surrounding neighbourhood of cells.

When a rule is fired, it causes growth in a specific direction that match the requirement of the precondition. This system acts like the chemicals that activate or suppress genes in a cell's chromosome, triggering pattern of cellular growth (Kumar and Bentley). This system is very promising because different from external embryogenesis, where parallel processing, conditional iteration, subroutines must be introduced manually and predetermined, it inherently incorporates them. Using implicit embryogeny, the emergence of solutions of increasing complexity does not necessarily require an equivalent increase of complexity in the genotype (Kumar and Bentley). Further more, with this approach, both the phenotype and the mapping process are dependent on the genotype, which depends itself on the phenotype (fig 11). In so doing, the mapping process can evolve adaptively during the development of the phenotype. With explicit embryogenies the genotype is no longer dependent on the state of the phenotype, which is the reason why the mapping process evolves without being adaptive during the development of the phenotype.

To summarise, external embryogeny results in a not evolvable mapping while explicit embryogeny allows the mapping process to evolve. The mapping process in implicit embryogenesis is, instead, evolvable and adaptive during the development of the phenotype. On the other hand, achieving evolvability with implicit mapping is even more difficult than for explicit embryogeny probably due to the discontinuity it causes in the search space (Kumar and Bentley). Recent experiments have shown that "despite having this desirable many-to-one genotype-to phenotype relationship, the system still does not perform as well as desired. This implies that a many to-one genotype-to-phenotype mapping, on its own, is not enough to ensure evolvability" (Kumar and Bentley).

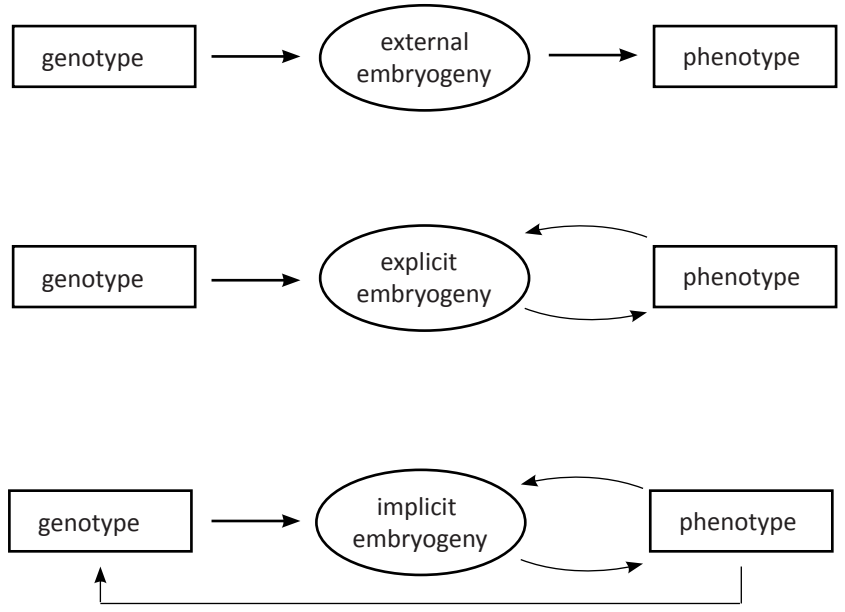


Fig 11 external, explicit, implicit embryogeny

Creative evolutionary systems

Creative evolutionary systems is an expression that wants to highlight the capacity for evolutionary systems to create novelty, exploring the search space in a "creative" manner. One of the most important requirement for this to happen, is to rely on a *component-based* representation of the genotype (BentleyP., Corne D. 2002). The representation of the genotype is the way it enumerates the search space, or simply how a solution is represented at the search space level. With a fixed-length parametrisation, evolution is only able to find the combination of parameters that feature a global optimum for the analysed scenario. It is clear that with a fixed number of parameters, and a predetermined way to link them, the evolutionary system cannot explore more than what is given to it. If the representation is instead based on components, which can be considered a set of building blocks from which the solution or the phenotype is constructed, the evolutionary system will "explore" the search space. Being the solutions, this time, not defined by parameters, but by the components, the aim of the evolutionary system becomes to find the way to build the solution by changing the relations between the components. It does not evolve only the details, but the whole structure that characterize the phenotype. With a component-based representation of the genotype (an explicit or implicit embryogeny), the dimensionality of the search space can be altered, the structure of the solution can be redefined allowing for a full exploration of the solution domain. When referring to architectural application of "creative evolutionary systems", if is often used the expression "*body plan*" instead of genotype representation, due to the fact that the system evolves, most of the times, 2D or 3D geometric entities. With a component-based representation the "*body plan*" it is not defined a priori, being the definition of its structure one of the result of the evolution.

The objective function is the way in which the designer embeds knowledge in the system. Creative evolutionary systems usually do not have a static one criteria objective function but rather multiple and dynamic to explore alternative search spaces. Multi objective environments have been developed for evolutionary system to rely on a set of evaluation criteria, exploring the search space from different perspective. Fitness functions can also be based on Co-evolutionary approach where the success of solutions in a population depends upon the success of solutions in another population. This approach comprises the individuation of two or a set of sub-systems which are epistatically linked and whose evolution depends on the co-evolution of all the other sub-system/"specie" (Jackson, H. 2002). In this way a solution will be subjected not only to the pressure of the "abiotic environment" but also to the influence of the "biotic environment".

Along with a completely automated evaluation process, there is also the possibility of adding human interaction, the role of the human user being to judge solutions that are presented to him. This judgement can be even the only evaluation criteria for certain application or, part of a set of fitness functions, reaching a kind of collaborative environment between the machine and the user. The main advantage to introduce this form of collaboration, is to overcome problems due to the

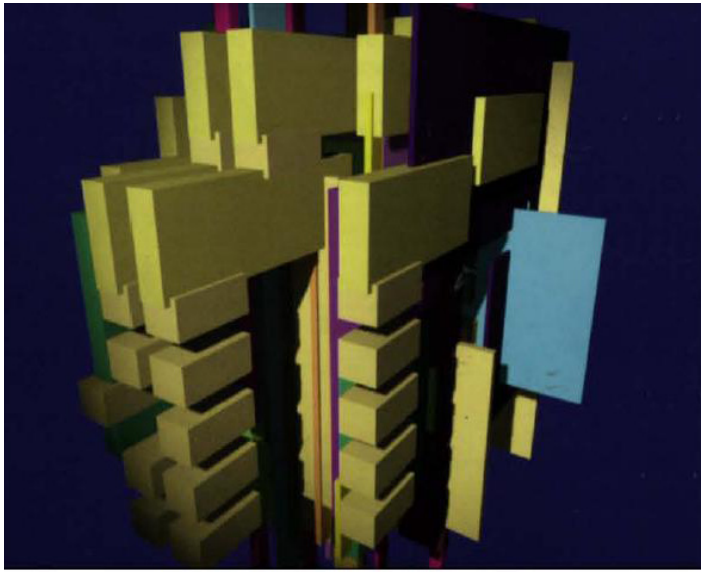


Fig 12 John Frazer - Multi State Three Dimensional Cellular Automata

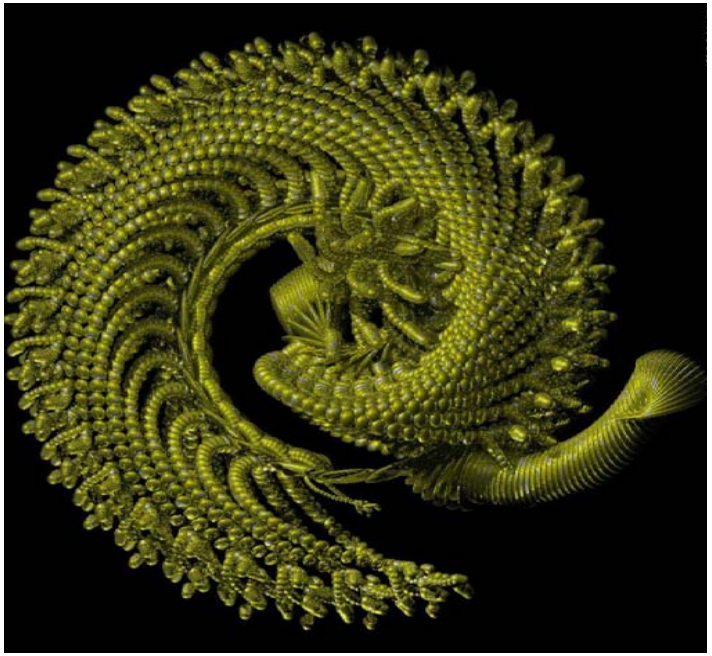


Fig 13 William Latham - Mutator

impossibility to define a clear and objective fitness function. This is often the case when the evolutionary system try to evolve on the base of aesthetic criteria. The other advantage is the possibility to help the system in case a local optimum occurs. On the other hand, the rate at which a human can evaluate a solution is very slow being solutions evaluated one at time. This can be just impossible for many applications that require the use of very large populations and to run for many generations. In this case, it is better to present periodically to the user only a small set of solutions.

It should be clear that, with an embryogeny based on components and a “collaborative” fitness function, the main goal of the evolutionary system is not to find a global optimum. Exploring the search space to find “interesting solutions” which still obey to design constraints and respect evaluation criteria, is what they focus on.

What gives to these systems the ability to explore the search space in a “creative manner”, it also causes the main problem in terms of efficiency and consistency.

Embryogeny based on components is the major contributor to the discontinuity within the search space. Genotypes with very different structures can be mapped onto phenotype that share close value of fitness. This causes the identification of very different genotypes as part of the same region in the search space, even if they are not. This effect is often worsened by poorly designed fitness function which are not able to discern structural differences between phenotypes.

The nature of embryogeny based on components brings inherently about a certain degree of *epistasis. Epistasis* is the degree of dependency between the genes in a genome (Bentley 1999). When having epistasis the phenotypic effect of a gene (during the mapping process) is related to the alleles of other genes.

With explicit or implicit embryogeny, genotype are constituted by a chain of interlinked instructions. The effect on one of these, depends on the effect of the instruction that precedes or succeeds it. The problem when having a high degree of epistasis is that every part of the design will be epistatically linked to any other part , making any evolution hardly reachable. A small changes in a part of the design would end in a series of changes in all the other parts, making very difficult any improvement.

To summarize, embryogeny based on components, fitness functions that encourage the exploration of the search space and human interaction, can result in a evolutionary system acting creatively.

Creativity in this context refers to the ability of such evolutionary system to be innovative and efficient in their search. They are capable to elaborate solutions that, generation over generation of attempts, perform better and whose topology was not predetermined or enclosed in the genotypic representation.

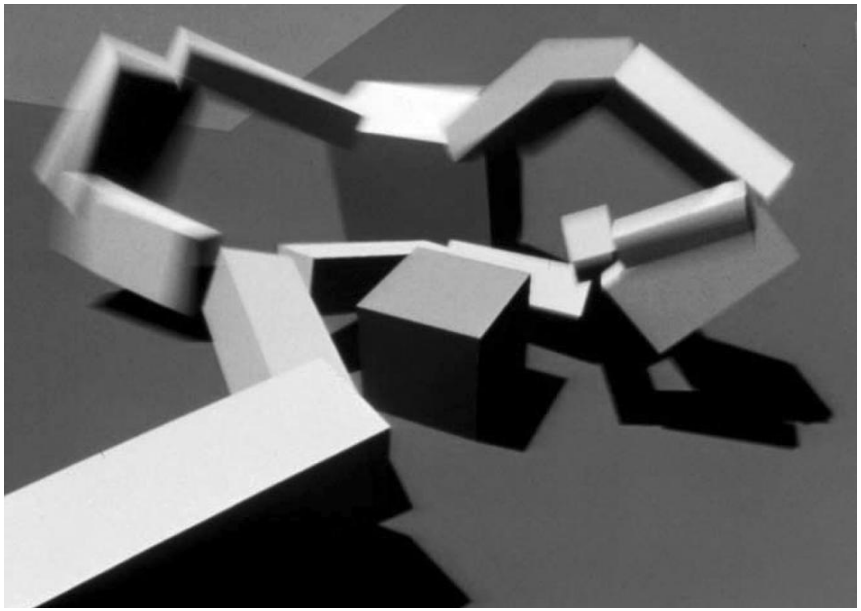


Fig 14 Karl Sims - two creatures competing for food source

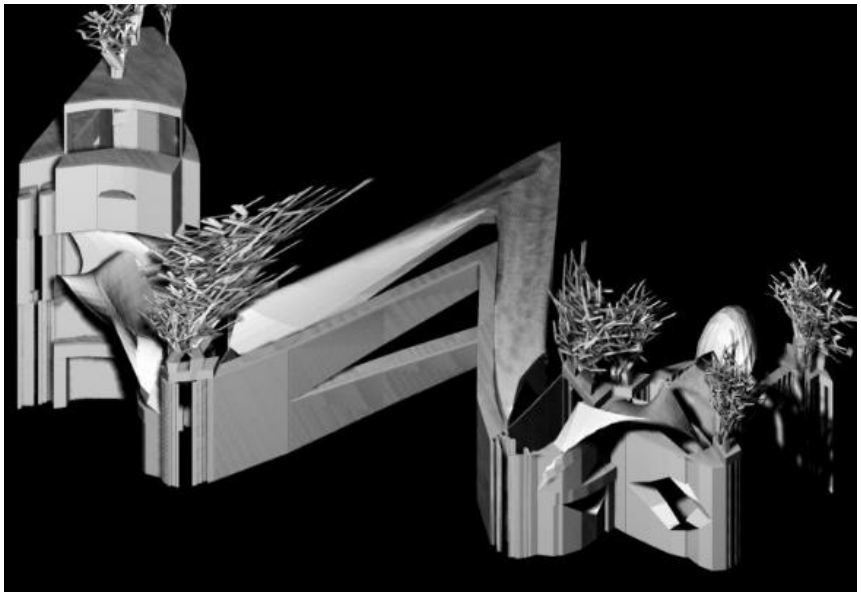


Fig 15 Celestino Soddu - Shangai chinese garden walls generated with Argenia

This chapter starts by presenting examples of past works upon which this research builds its foundations. It moves to a detailed explanation of the structure of the system which is mainly based on the combination of a growth model and an evolutionary technique. It follows the description of the objective function, which is the set of design criteria used to navigate through the search space.

Precedents

In his paper "E*xploring 3D design worlds using Lindenmeyer systems and Genetic Programming*", Paul Coates explores the potential of an evolutionary system based on L-Systems and Genetic Programming, where L-Systems are used as a growth model (Coates et al. 1999). Their production rules, which are at the base of the string rewriting mechanism, can be seen as sort of "master genes". The spatial representation can be achieved by interpreting with turtle graphics the symbols in the L-System string genotypes.
A certain symbol in the string is an instruction to draw a building block in a particular position of the 3D environment, being the position determined by precedents symbols in the string and by the position of already existing "blocks". When I refer to building blocks I mean the type of elements, spheres, cubes, octahedrons etc. that will compose the evolved configuration. The biological analogy would bring us to consider a building block as a cell in an organism, being the process by which all the blocks are put together the growth of the organism.
Coates chooses not to rely on a 3D Cartesian Space but rather to represent the structure's form using the geometry of iso-spatial dense-packed spheres, the iso-spatial grid first used by Frazer (Frazer 1995). The co-ordinates of the grid are the vertices of close packed octahedrons. Elements (the building blocks of the structure) can only be inserted at these vertices and each sphere has 12 equally spaced neighbours. The main reason why using the iso-spatial grid is because it offers a natural way to represent 3D forms without lack of homogeneity (Coates et al. 1999). In 3D Cartesian systems the distance between a point and its 26 neighbours is contingent upon the position of the neighbour, whether it lies on one of the orthogonal axis or on a diagonal one. "The aim is to work towards the evolution of form with the minimum of preconceived notions" (Coates et al. 1999), by means of the generations of computer programs that go through a process based on Darwinian theory of evolution. The system was developed in Autocad using AutoLisp, an Autocad version of Lisp, which was used to implement many Artificial Life techniques including Genetic Programming (Koza 1992). Fig 1 (left side) shows one of the outcomes when the generated configurations are rewarded for capturing the most particles possible. Particles are just others spheres that, by moving in the 3D environment can hit the evolving structures. In this case they successfully evolve, increasing their exposed surface area as much as they can, in order to trap the most flies (the particles) possible. Fig1 (right side) shows instead an evolved configuration when the objective is to maximise the exposure to sun

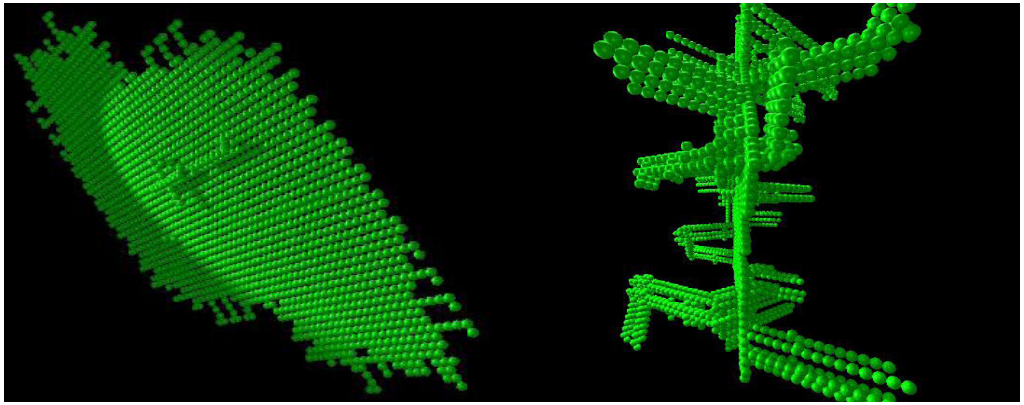


Fig 1 Coates | flytrap and suntrap

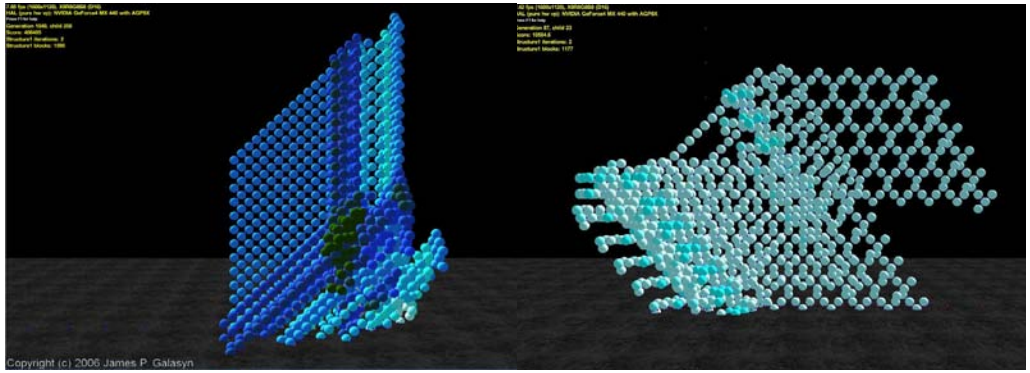


Fig 2 Galasyn | high surface area and airborne particles trap

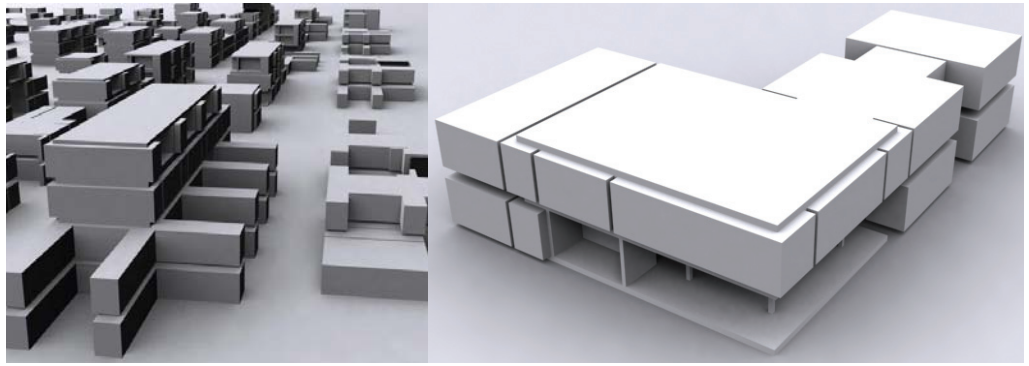


Fig 3 Coates | Domino House

.without creating overshadowing. Fig 2 shows the experiments carried out by James Galasyn who, following the route first pioneered by Coates, developed an analogous evolutionary system based on L-System and Genetic Programming in 2006. The way in which the emerging configurations are evaluated, in Galasyn's system, mainly relies on a "load propagation model" that he developed (Galasyn 2008) . This procedure is able to simulate the propagation of loads through the structure of the generated configurations in order to assess whether they are stable or not. The model is able to recognise whether a failure occurs, for instance due to an appendix of the body that cantilevers too much, according to a set of abstract material properties that the user can specify. The structural stability is indeed the first criteria requirement that has to be respected, a design constraint, on top of which other design criteria can be added.

The other family of experiments carried out by Coates at CECA (UEL) were based on Genetic Programming and generative grammar of forms where grammar, following Stiny and Mitchel (Stiny, Mitchel 1978), is considered to be "a lexicon of primitive objects, and a syntax of transformations on those objects" (Coates et al. 1999). GP allows the exploration of search spaces defined by an initial set of axioms and production rules. Eventually, not just automated shape grammars but also the emergence of grammatical rules and their mapped grammatical objects will manifest. Fig 3 shows one the outcome that is possible to achieve when having as "objects" (the terminal set) some rectangular boxes of different size. Rules are simple instructions such as copy or move one the box along a particular direction, boolean subtract or add one box to another etc. Rules and objects are again organised in a hierarchical tree data structures. The mapping, evaluation, selection, recombination and mutation of population of these, will eventually produce spatial configurations that respond to specified criteria.

Coevolutionary approach to architecture

When talking about biological systems, coevolution refers to the adaptation of an organism triggered by the adaptation of another one. Each system exerts selective pressures on the other system affecting each others' evolution. Coevolution can manifest in a "host-parasite" fashion or as "mutualism". A mutualistic coevolutionary system between two species produces life-cycle interaction which leads, over generation, to co-adaptation (Thompson 1994). In layman terms, each species benefits from the evolution of the other species. This symbiotic partnership can produce highly complex behavioural pattern and physical characteristics which are observable in many biological system such as wasp and the plant of fig, hummingbirds and ornithophilous flowers, legumes and bacteria etc. Coevolution is a one to one interaction but often is the case when many species evolve in response to the evolution of other species, "diffuse coevolution" (Thompson 1994). Although in coevolution both systems are themselves under the pressure of the abiotic environment, is the biotic environment that exerts the highest pressure resulting in evolutionary adaptation. As far as architecture goes, coevolution can be seen as an alternative method to mono and multi objective optimisation. Instead

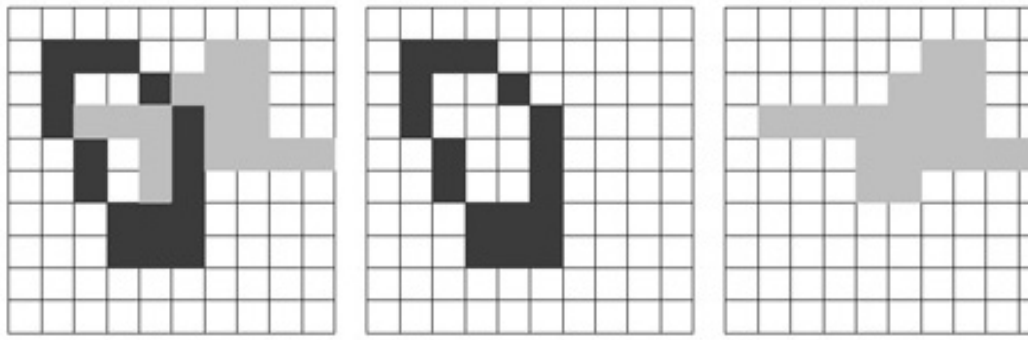


Fig 4 Coates - "space" |"enclosure" 2D

Fig 5 Coates - "space" |"enclosure" 3D

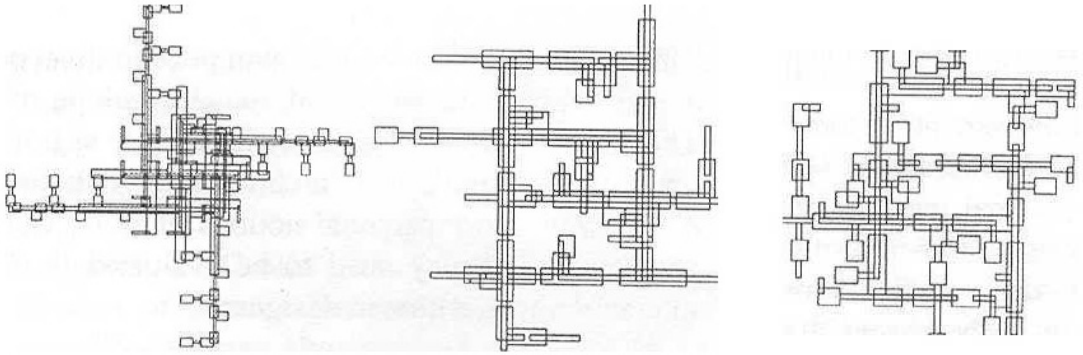


Fig 6 Helen Jackson - "space" |"enclosure" 2D (minimization of "I" value)

than having only an architectural system to evolve, we can add information regarding regarding the relations that occur between its sub-systems. Instead than evaluating a spatial configuration from different perspectives, it is possible to describe it as the expression of multiple systems. This entails the identification of two or more different parts which have their own identity, but whose behaviours and patterns are interlinked (Coates et al. 1999).
Once these systems have been identified, coevolution acts as the most prominent way to exert pressure leading to a symbiotic process where the fitness of the systems evolves through co-adaptation (Coates et al. 1999).
Coates identified as two closely, yet separated, species "enclosures" and "space". "Enclosure" (black squares in fig 4, black spheres in fig 5) represents wall systems while "space" (grey squares in fig 4, grey spheres in fig 5) represents the actual space used for programmatic reasons. "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" (Coates et al. 1999).
"In terms of configurations of solid matter a good enclosure form is one with a high enclosed volume compared to the volume of the configuration" (Coates et al. 1999). Same applied for the "space" species. The volume ratio of the two species represents, hence, the fitness criteria for this single-goal coevolution.
In her paper "*A Symbiotic Coevolutionary Approach to Architecture*" Helen Jackson (Jackson 2002), after Coates, shows the development of the "enclosure"| "space" symbiotic system based on three main design criteria:

• the minimisation of the I value (Hillier 1996, p.283 cited in Jackson 2002) which is an index useful for the recognition of well-integrated spatial layout (the lower, the more integrated is the space)

• the circulatory route within a building in order to 'link spaces in a way that enables their configurational requirement' (Coates et al. 1999)

• aesthetic of the created envelope of the generated forms (phenotype of the individuals)

Following this line, Galasyn further developed his system to achieve the growth and the evolution of his "evolved structures" in a co-evolution fashion. The body of the structure can be built by multiple parts which are independently grown subjected to structural evaluation. At the same time, they are also scored to promote their union in order for them to support each other, in this way triggering structural co-evolution. Although some results have been produced the co-evolution approach still need to undergo major developments.

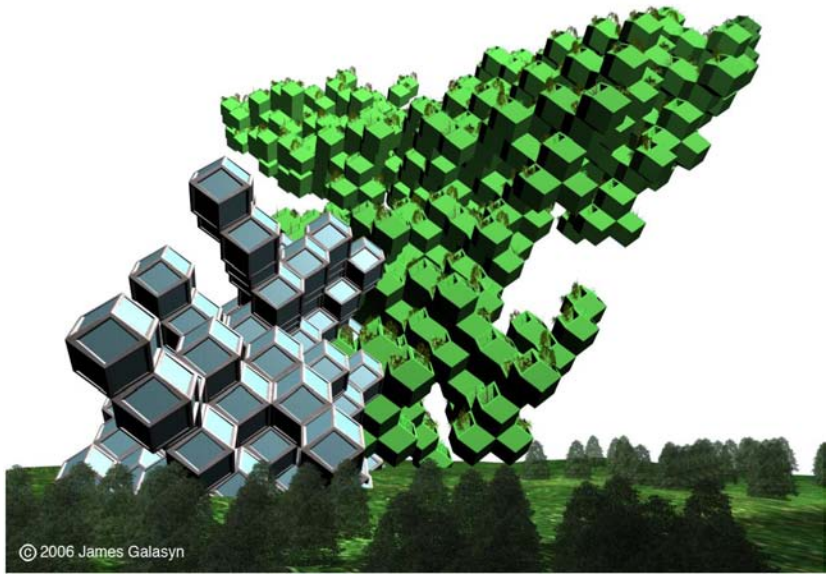


Fig 7 Galasyn | Coevolved stable elevated structure

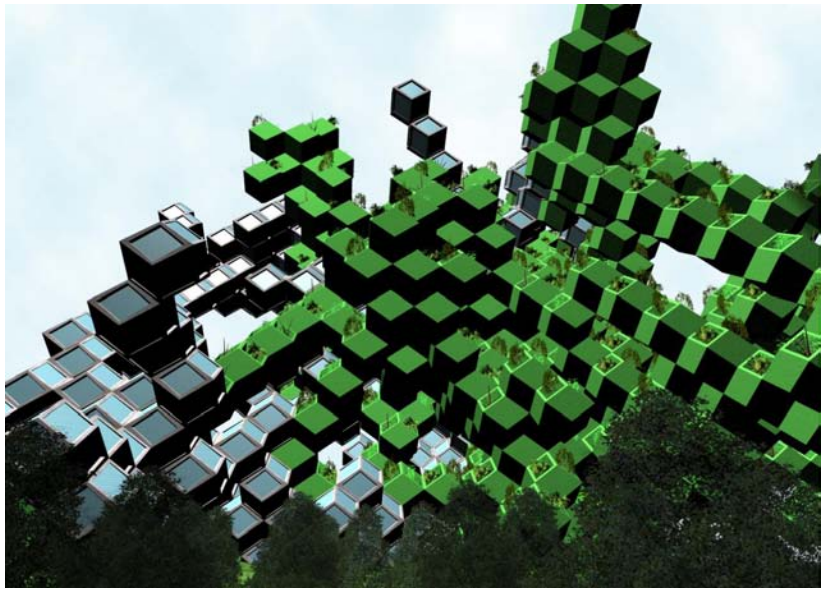


Fig 8 Galasyn | Coevolved high surface area structure

String rewriting techniques : L-Systems

An the evolutionary system needs to have a “mapping” procedure interpreting the information enclosed in the genotype representation to produce the phenotypic representation. To avoid to search only in a limited region of the solution domain, we need to embed into the system a model of morphogenesis which is able to allow the emergence of complex entities under evolution.
Different model that can simulate biological growth have been developed since Turing introduced the reaction - diffusion model in 1952 (Turing 1952). L-Systems are based on the concept of rewriting which consists in the replacement of characters in a string of symbols with other string of symbols (Prusinkiewicz, Lindenmayer 1990). The string used to start the rewriting procedure is called axiom, while the rules that specify which character of the axiom has to be replaced with other set of symbols, are called production rules. After the first substitution, the newly formed string (originally the axiom) is scanned. The characters that match the pattern designated by the production rules are replaced again. This goes on until the specified number of iterations is reached, usually creating string of considerable length even when this number of iterations is set to be small.
To run a string rewriting system we need to think at a vocabulary of admissible symbols each associated to a particular meaning (Prusinkiewicz, Lindenmayer 1990). With turtle interpretation of strings, first introduced by Prusinkiewicz, we use the analogy of turtles to represent a point moving in a 2D or 3D environment. A symbol “F” (meaning forward) tells the turtle to move forward for a certain length and other symbols such as “+” and “-” tells the turtle to steer right or left according to the verse of the chosen world coordinate system. Fig 9 gives an example of what kind of pattern can be created if two successive position are connected with a line.
In this particular case the symbols “+” and “-” tells the turtle to rotate 90 degree around the z axis. This can be easily extended in 3D when we think at three pairs of symbols each of which tells the turtle to rotate around one of the coordinate system axis (fig 9). Fig 10 presents a family of classic Koch curves which are created by applying very simple productions rules and with a very small number of iterations. Starting with the axiom F-F-F-F, which would be represented by a square, each F is iteratively replaced by the production rule(for instance F —> F-F+F+FF-F-F+F) (Prusinkiewicz, Lindenmayer 1990). These simple example fully demonstrates the power and the beauty of L-Systems, being them able to produce highly complex patterns starting with very simple rules.
L-Systems have been developed to a great sophistication in order to model different phenomena of morphogenesis such as the differentiation of cellular layers. As far as our project goes, I will deploy the basic form of rewriting system, above explained, using stochastically generated production rules and having the structure of the strings organised in a hierarchical manner (stochastic and bracketed L-Systems).

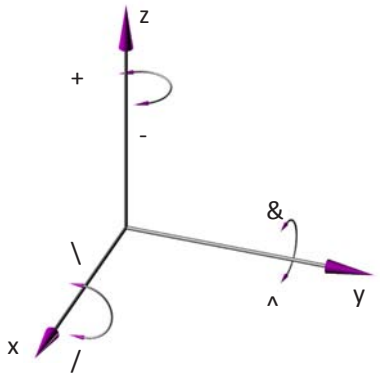
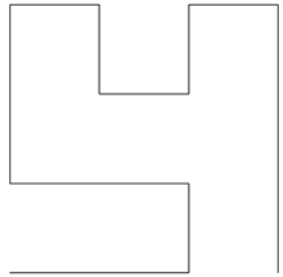
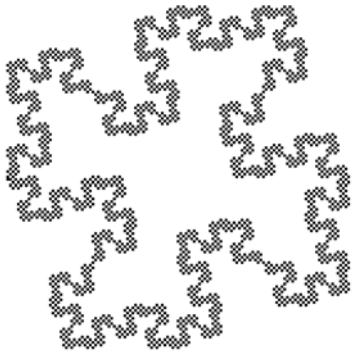


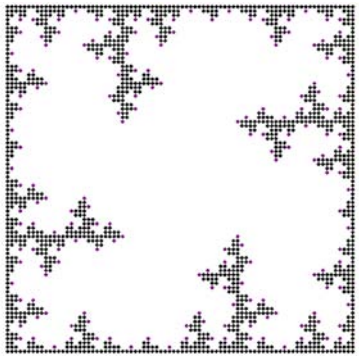
Fig 9 graphic interpretation of turtles



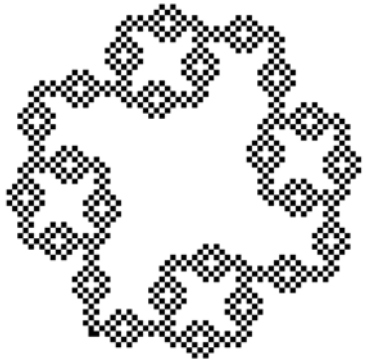
FF-F-FF+FF+F-F-F+F+FFF



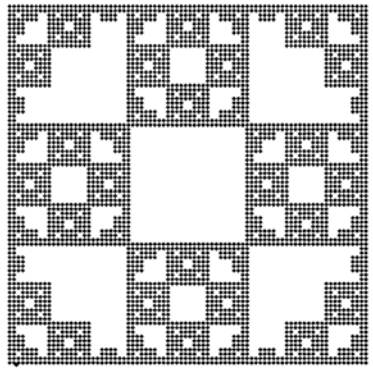
n=4, d=90
F-F-F-F F —> F-F+F+FF-F-F+F



n=4, d=90
F-F-F-F F —> FF-F--F-F



n=4, d=90
F-F-F-F F —> FF-F-F-F-F-F+F



n=4, d=90
F-F-F-F F —> FF-F-F-F-FF

Fig 10 family of Koch curves

At the very heart of the evolutionary system lies the combined action of L-Systems and Genetic Programming which constitute the generative engine of the system. In order to represent the genotype, a string of symbols is first generated using string rewriting systems. Axiom and production rules are randomly generated. The symbols chosen to generate the strings, following Prusinkiewicz in "T*he Algorithmic Beauty of Plants*" (Prusinkiewicz, Lindenmayer 1990), are :

• *F*, is interpreted as "move forward", keeping the previous orientation, and placing a cube in the reached position. This cube constitutes one of the "building blocks" whose arrangement in cubical package makes up the structure of the individuals. I choose to explore the space using a 3D orthogonal grid rather than the more elegant Iso-Spatial one. The reasons for this choice is first to not over-complicate the process. The Iso-Spatial grid requires with use of rhombic dodecahedron as building blocks to achieve fully connectedness in the individuals. The other reason is to test the an element different from the already experimented spheres and rhombic dodecahedron by Coates and Galasyn.

• *+*, yaw left by 45 degree around the z axis (fig 9). The value of the angle is imposed by the cubical neighbourhood which surrounds each building block

• -, yaw right by 45 degree around z axis

• *&*, pitch down by 45 degree around y axis

• ^, pitch up by 45 degree around y axis

• *£*, roll left by 45 degree around x axis

• *$*, roll right by 45 degree around x axis

• %, turn by 180 degree around z axis

The interpretation of strings formed by these symbols turns to generate not more than a series of blocks piling on top of each other. To give the individuals the possibility to develop appendices or to branch off, we need to represent the genotype in a hierarchical manner, parsing the strings into tree data structure. This is already offered by "bracketed" L-Systems which use the additional symbols "*[*" and "*]*" to create branches. Fig 13 shows an example of a bracketed string expression which is rendered by connecting the position reached by the turtle with lines. Every time a *[* (open branch character) is encountered a new branch is created which can develop further more if other *[* are met. When a *]* (close branch character) is met, the turtle goes back along the path, remembering in which position it was at the time the current branch was open (Prusinkiewicz, Lindenmayer 1990). This system

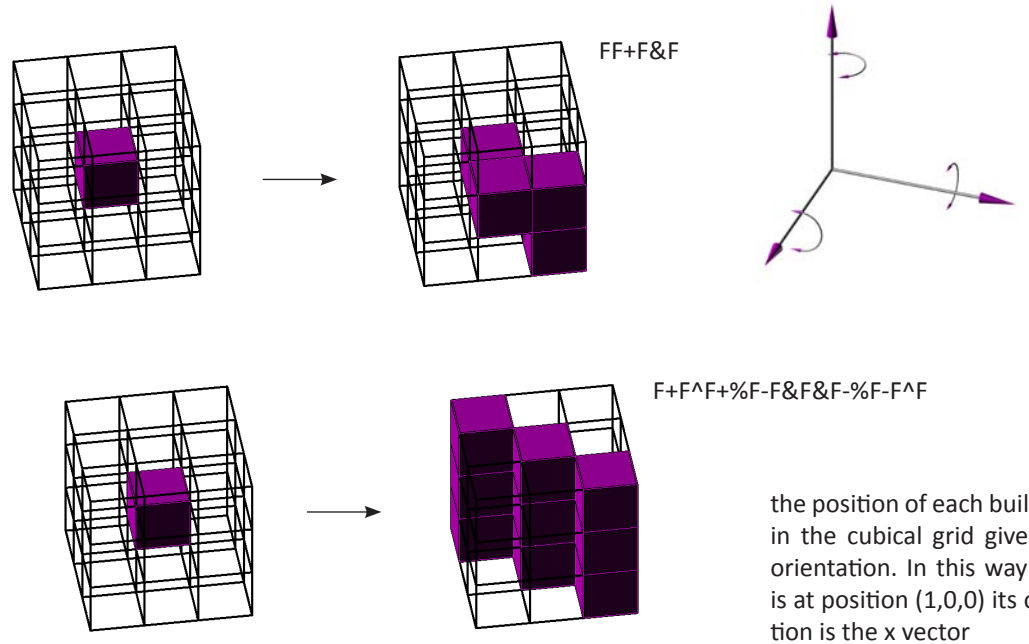


Fig 11

the position of each building block in the cubical grid gives also the orientation. In this way if a block is at position (1,0,0) its own direction is the x vector

placing the building blocks

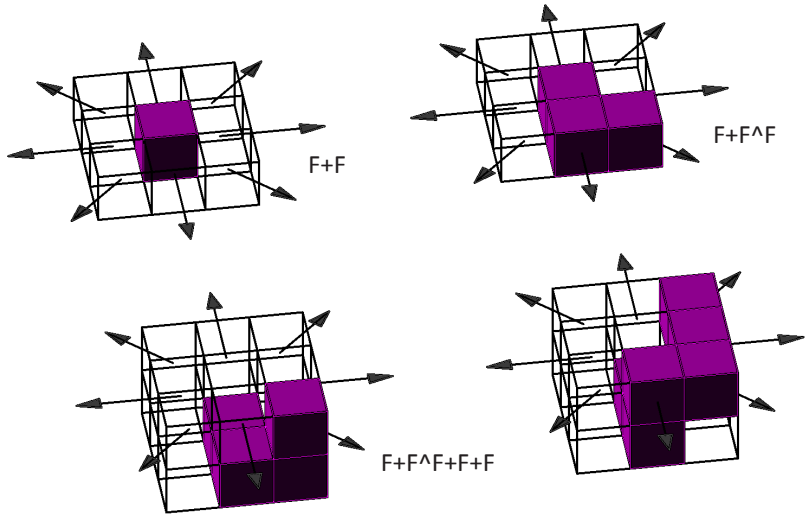


Fig 12

the vector owned by each position, determines next position when applying one of the possible turns. In this way there is no confusion with the admissible positions and there is no need to use any matrix of transformation

turtles' neighbourhood

has been used to model numerous species of plants (Prusinkiewicz, Lindenmayer 1990).
As far as architectural design goes, this lead us to intend the organization of space and system of spaces as trees, or more generally, as a branching structures. As already outlined by Coates (Coates et al. 1999) there are different reasons why this can be useful in architectural context:

• the structure that are generated are inherently consistent in terms of connectedness. Every new building block grows starting from an already existing part of the structure so that there can be no gaps.

• the connectedness of the configurations is useful to design spatial systems because all their parts are reachable through viable paths.

• the recursion logic of the string rewriting system offers the possibility for the generated configurations to feature symmetry and self-similarity without this being embedded in the initial representation. This leads towards geometrical arrangement of spaces as expression of process rather than representation.

Fig 14 shows one of the spatial configurations that are possible to obtain when evolving the initial randomly generated genotypic representations. Although it is not obvious to look at such a structure as a tree, it is very close to them. They start from an initial position which is the root of the tree and develop by piling, blocking and branching under the pressure exerted by their environment.
As far as Genetic Programming goes, the bracketed form of the string expression s favours their parsing in tree data structures, which are the working genotypes of the individuals. A tree data structure can be literally thought as a tree with a root, branches and leaves. A joint between two branches can be thought as a node which contains information. A node it is just a container, where is possible to store any sort of data such as the character symbols of the string expressions, fig 14.
Every part of the string, which is enclosed by two "branching characters" (*[,]*) is one of the node of the tree. If the nodes belong to the same parent, which means that they have the same depth level in the tree, they are called siblings. The symbol to indicate when two nodes have the same level is "|". Each node is usually both parent of deeper level nodes and child of lower level ones. The nodes that do not have any child are called terminals or leaves of the tree. Tree data structures have a very broad range of applications. To give an idea, it is sufficient to think at the system of folders that we build on our personal computer usually starting from c:\. This is represented as a tree data structure, c:\ being the root, the folders being the nodes.
Differently from the computational environment in which Coates developed his system (AutoLisp) (Coates et al. 1999), C # does not offer the "EVAL" command. By using this command is possible to evaluate bracketed string expressions, getting both their parsing and their interpretation for free. Although this has to be manually implemented in C #, the advantages of using this computational environment

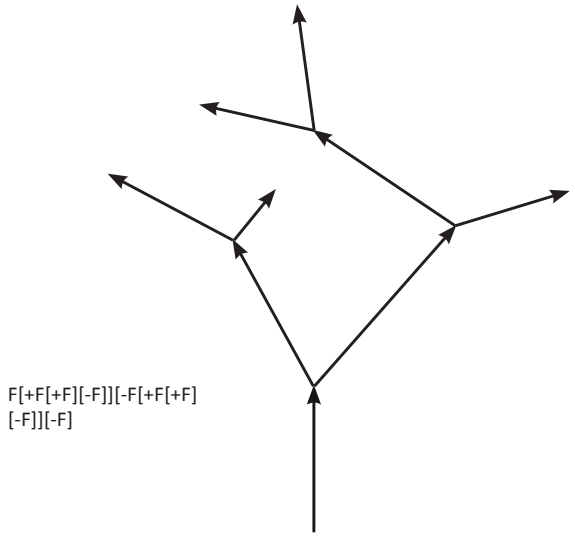
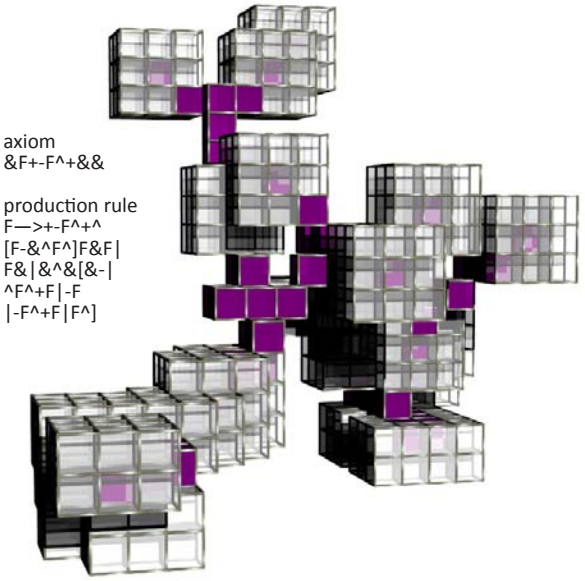


Fig 13

bracketed string expression



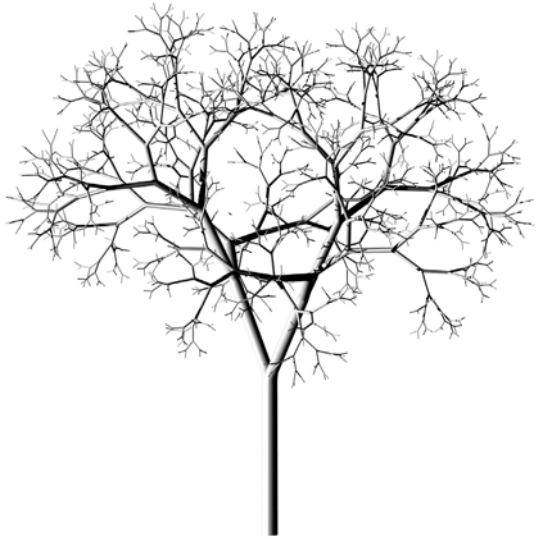


Fig 14

spatial systems as tree branching structures

go far beyond this little extra work.
To parse a bracketed string expression in to a tree data structure we need to read the string, recognise every time an open or close or sibling branch character is met and the help of an external “collection” called “Stack”. A Stack is very close to an array but it has the special ability to retrieve the last data inserted in the collection (last in, first out). We can divide the parsing procedure in 4 main parts;

• *[* , a open branch character is met ; a new branch is about to be created which means that we have to assign to the current node of the tree all the information red so far. After that, this node is “pushed” into the Stack collection in order to be stored and remember which height of the tree this branch has been opened at. A new node is also created and assigned as current node.

• *]* , a close branch character is met; we need to pop the last node inserted out of the Stack collection and check if the next character in the string is of type *[* or *]*. Their occurrence means that there are other siblings on this level of the tree and, therefore, the branch is not yet completed. If this is the case, a new node is created and assigned as current. In opposite case, the branch can be considered completed an no other node are needed.

• | , a sibling character is met; a new node is created that stores the information red so far, the node is then appended as part of the current branch

This operation is also called “Deserialization” and this is how the genotype of our individuals are represented. The opposite procedure “Serialization” takes a tree data structure and turns it into a bracketed string expression. This operation is needed every time a new production rule is created in order to assess whether there is a malformed string (for instance due to a missing bracket). In this way integrality of data is ensured because the genotype becomes inherently correct.
To summarize, string bracketed expressions are generated using L-Systems which are afterwards red using 3D turtle graphic interpretation. The string is parsed in to a tree data structure which translates the information encoded in the string into a sort of hierarchical library of data. The tree data structure genotypes are afterwards used to perform genetic operations such as cross over and mutations. Fig 15 shows how one of this “object” can be represented as a tree of nodes each storing a certain amount of information which are linked in a hierarchical manner.

axiom —> F
production rules
F —> [F+F%F]
+ —> [-F+]
$ —> [F^-F]
^ —> [$F+]

n iteration = 10
String expression
[[F[-F+]F%F|-[F[F-|-F+]F%F|-F+]F[-F+[FF[++|-F|&&|+--]|%--|+F]F%F]%][F[-F+[FFF[%%|F-|++]|--|%F%]F%F]

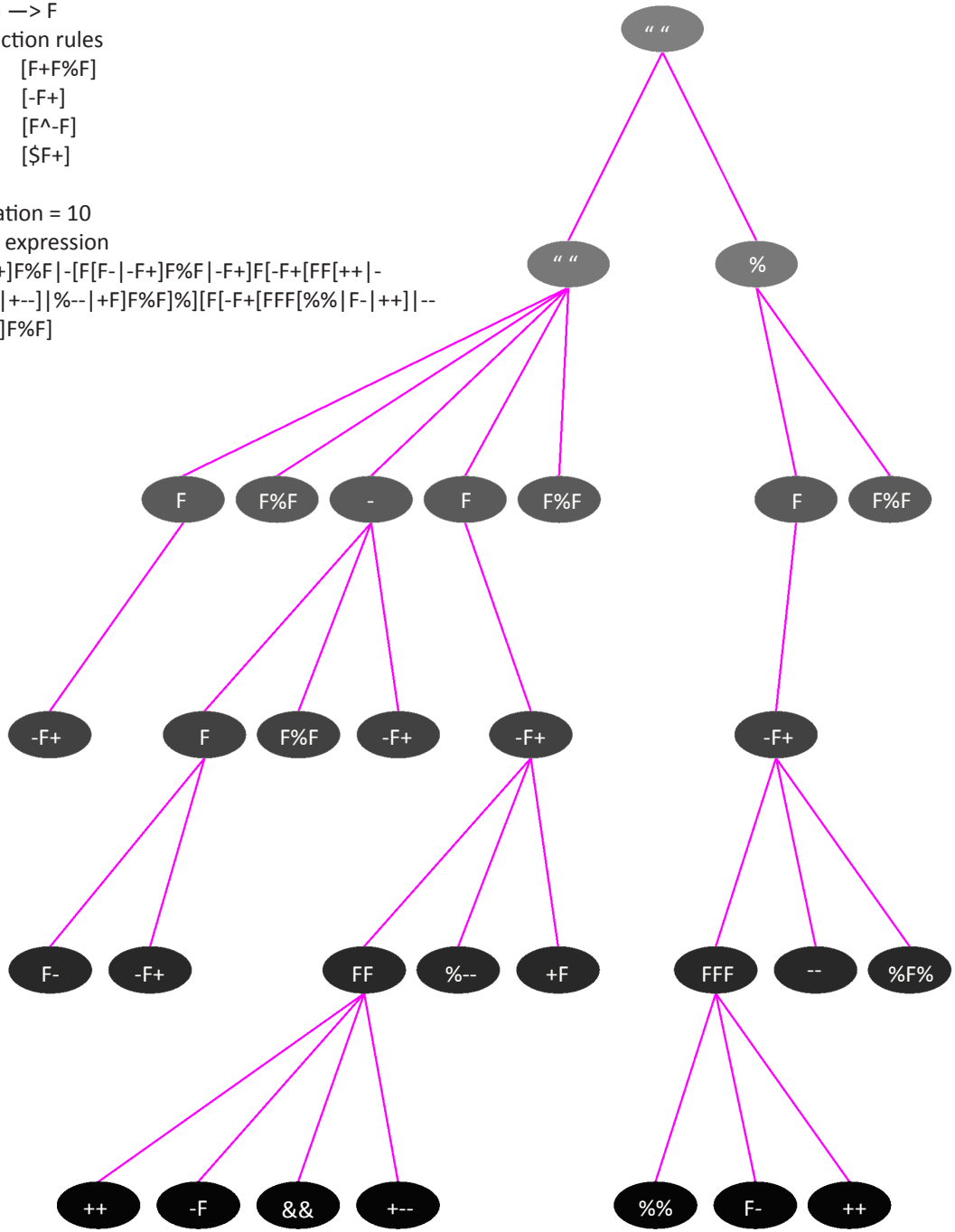


Fig 15 parsing the string expression into a tree data structure

The hierarchy of the structure

When interpreting the bracketed string expressions, a first pattern, which is the direct map of the genotype, is created. Following the instructions encoded in the string, “building blocks” are placed in the 3D environment generating what can be thought as a “skeleton”, the internal structure of the individual.
A building block is not only a geometrical entity, but can be thought as a unit of data such as the position of the centroid of the cube, its exposed surface area or its volume and so on so forward. Every time a “building block” is created, it is also stored in a special kind of array which in C # is called “Dictionary”. A “Dictionary” can store a certain number of elements and assign each of them an identification number. In this way, for each “value” there is a “key” which is its own ID. This allows to efficiently retrieve information related to each “building block” and avoid to have confusion or replication of objects. For instance, when interpreting the string expression, even if its hierarchical structure is correct, there can be blocks that share the same position. This is not only a waste of computational resources but also a major problem when propagating the load force vectors into the structure to assess its structural stability.
The Dictionary offers a very simple way to tackle this by simply assigning as the ID of the blocks, their centroid position. Every time a block is about to be created, a quick check in the BlocksDictionary evaluates if the ID (the position of the centroid) is already present. If this is the case, no block is added to it or created in the 3D environment. This procedure gives full control of the structure during its development and creates the possibility to add other two layers to it.
The body of the individuals that is generated every time a genotype is mapped, can be thought as divided in three interlinked systems whose bounding make the whole structure. The structure generated only by interpreting the string expression looks like the one shown in fig 16, left side. I call these type of blocks the “***internalBlocks***”. Each block is also designed to be aware of its immediate neighbourhood, for instance by recognising where they are located within the whole structure, and whether there are not taken position where would be possible to add other blocks. This method is used to create the “***boundaryBlocks***” of the structure, which are the layers shown in fig 16 right side. These type of blocks can be thought as the foundation for the structure. With the same method, the blocks are also able to detect the number of neighbours that surround them and to generate, whether is possible, an additional layer of cells , the “***envelopeBlocks***”. This constitutes an external envelope of the structure as shown in fig 17, left side. Although thought as separated, these systems are strongly interlinked by the series of force/reactions that each block exert on its neighbours, independently by the system which belongs to.
I would quite obviously think, in terms of architectural design, at the internalBlocks like both the skeleton and the network of paths whereby is possible to reach every position in the body, the boundaryBlocks as the foundation and the envelopeBlocks as the available “space”.

Fig 16 internal blocks | boundary blocks

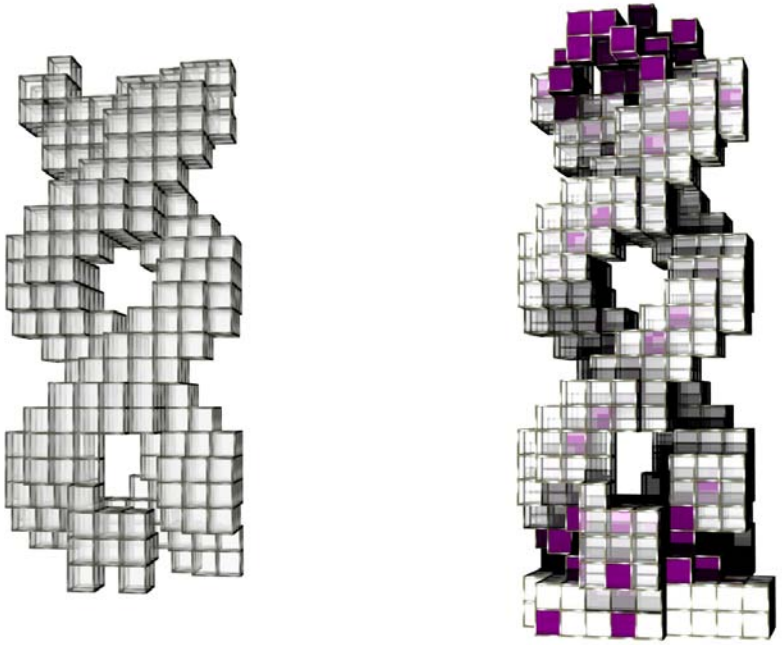


Fig 17 envelope blocks | the three systems together

The load propagation model

One of the biggest challenge when developing an evolutionary design system is the implementation of perfomance-based evaluation tools that can judge the individuals effectively. Even more difficult is to implement tools that are able to work with intensive quantities such as distribution of stresses in a body under load, or the induced pressure caused by the impact of wind on the external surface of the body. Most of the times designers rely on the methods offered by available software, based on finite element method or partial differential equations. These techniques are very expensive in terms of computational recourses and time requirement which is crucial when the number of solutions to evaluate reaches the order of hundreds of thousands (which is the working case rather than the exception).

Recently developed approaches rely, instead, on vector fields whose elements hold local information that can propagate through the body under examination. In general, these models are less accurate compared with Finite Element Methods but they are definitely faster.

James Galasyn developed a model for load propagation in 2D or 3D discrete medium which is mainly based on vector algebra and programmatic implementation (Galasyn 2008).

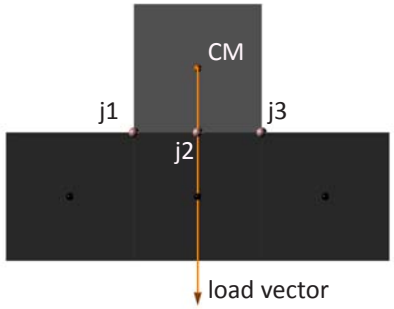
Although the assumptions on which the model is built up have to be further elaborated, it represents a good first approximation method that can be plugged in the evolutionary system to perform structural evaluation.

The load propagation model allows to achieve structural evaluation of each individual which can be used for different purposes. As a design constraints the structural failure of a solution does not let its genotype to be transmitted to next generation. The structural constraints is usually always kept as first rough selection of search space. Eliminating all possible body plans that are not structurally stable constitutes a way to size down the dimension of the solution domain .

As a design criteria the load propagation model can be used to drive the solution towards structural configurations with minimum average induced reactions. This can lead, if a forces-to-deformation model is implemented, to achieve solution with minimal energy deformation.

The medium was originally modelled as two-dimensional hexagonal packing of circular blocks, and it can be easily extended for a 3 dimensional cubical packing, which is how the structure of the individuals is made of.

Fig 18, left side, shows 4 blocks connected through a series of joints j1,j2,j3, which are represented as pink points, located at the middle position between the area centroids of each pair of blocks. Joints can be found, hence, at the corner of a block and at the middle of its edges. Joints and blocks can bear a limited amount of load until their mechanical properties, axial and torsional strength, are exceeded by the induced forces. The external load,here represented by the orange vector shown in fig 18, is the force caused by the weight of the internal block. If, propagating through the blocks, it causes structural failure in some of the joints, the structure is considered to be not stable.



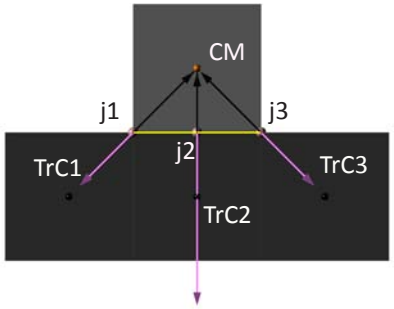
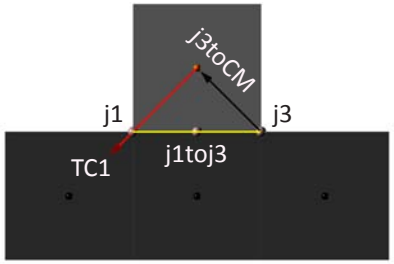


Fig 18 gravitational load vector | translational components



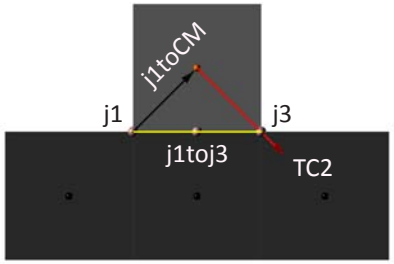
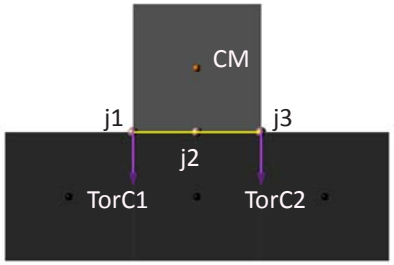


Fig 19 torsional forces



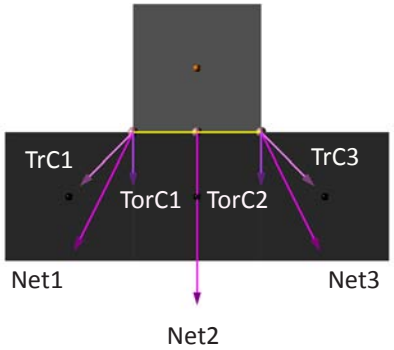


Fig 20 torsional component | net forces on joints

The one block problem

The one block problem (Galasyn 2008) can be represented by having three dark 'boundaryBlocks", which act as an infinitely rigid constraint for the structure, the lighter grey one being an "internalBlock". The whole structure is indeed a rigid body in equilibrium. CM is the center of mass which, in this case, is coincident with the area centroid of the internal block. The load is transmitted to the boundary blocks by means of the joints j1,j2,j3. The translational components of the induced forces lie parallel to the vectors connecting the joints to CM. Fig 18 shows the force components represented by violet vectors TrC1, TrC2, TrC3. The torsional components TC1 and TC2 lie orthogonal to the vector joining j1 with CM and j3 to CM, respectively. The action of TC1 causes induced torque moment on j1. This torque induces a rotation around j1 which has to be counteracted by the other joints being S a rigid body in equilibirum. For this reason j3 exerts a reaction which is assumed to be translational, TorC2 in fig 20, and directed orthogonal to the vector joining j1 and j3, the yellow line in fig 19.
The same applied for the other joint, j1, which has to bear the translational force TorC1 induced by the torque moment caused by the action of TC2. The resultant forces by the sum of translational and induced-torque components are the net reactions acting of the joints, Net1, Net2, Net3 in fig 20.

Many blocks and 3D extension

The one block problem can be easily extended to configurations that feature more than one internal block such as the one shown in fig 21. When having more than one layer of internal blocks, the distribution of forces is computed by slicing the structure layer by layer. Starting with the closest layer to the boundary block the structural stability of this first substructure is assessed. Every time a new layer is analysed, the already examined layers are treated as boundary (Galasyn 2008). Fig 21, right side, show three successive substructures with their gravitational load vector.
If in two dimensions all the forces resolve into translational components, in 3 dimensions the load propagates with both translational and torsional forces accumulating on each joint. Both the induced net translational and torsional forces are, hence, tested against the axial and torque strength of the joints.
Although the mechanical properties of the joints do not have any relation with real material mechanical properties, being them just arbitrary numbers, the model is able to predict structural failure due to over pronounced cantilever arms, such as the one shown in fig 22. The darker blocks are the boundary of the structure and the red points show that structural failure occurred at that position in the structure.

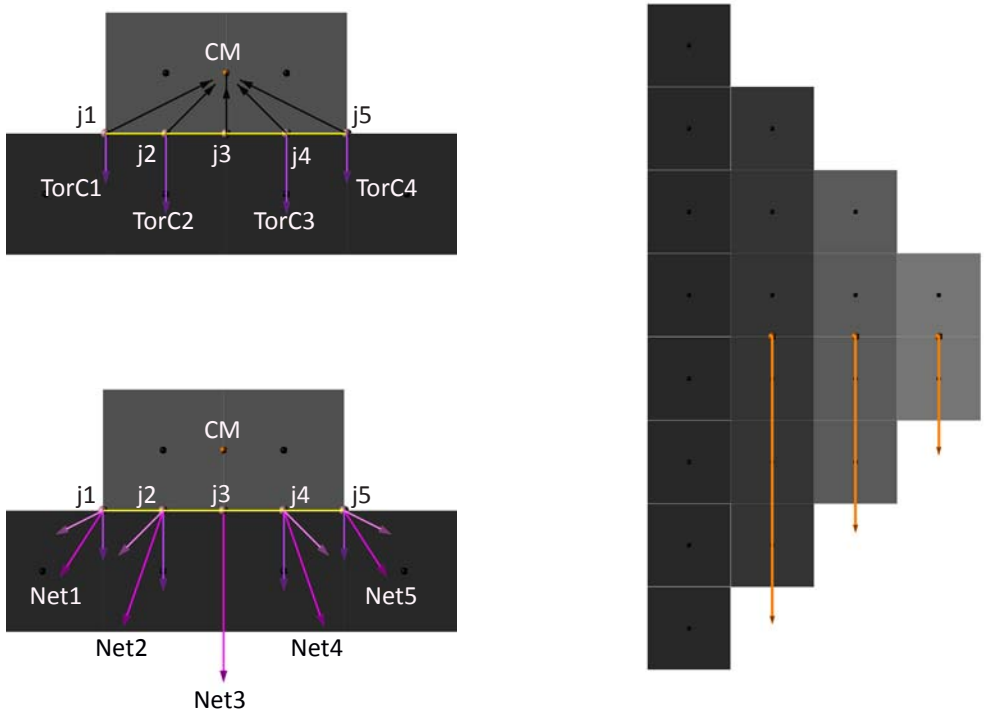


Fig 21 2 blocks problem | successive substructures with load vectors

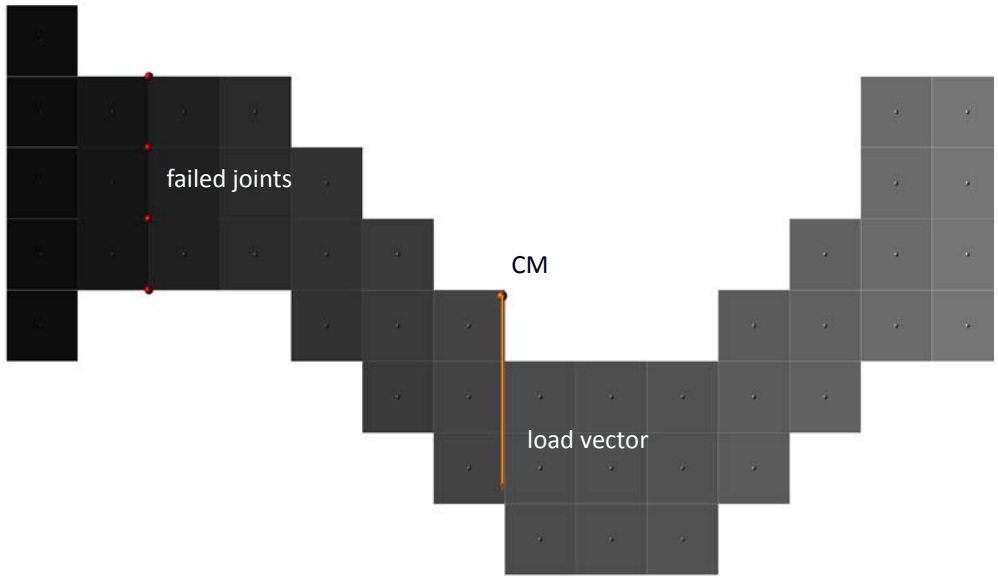


Fig 22 red joints show structural failure

Morphogenesis under load

The structural feedback that comes out of the analysis performed by the load propagation model can be compared to one achievable by static analysis which is usually done on much simpler structures than the one shown in fig 23. The automatization of the process allows to experiment the method on highly complex configurations having a quick feedback on their structural behaviour under the specified loads. Analysing the structure in its final configuration would give information on its stability only at this stage. If the individual has gone through structurally unstable states it is not inferable from this type of analysis. I will use the analogy between the process of interpretation of the L-System genotype, indeed the placing of the building blocks, to the organic developmental process of an organism. None of the parts of a living organism can grow if the whole body is not able to bear the loads that it is subjected to. If instead that analysing the individual at its final configuration we let the applied loads to propagate while it grows, the two processes become quite close. This is possible to achieve by checking the structural stability of the individual different times during its development, working with temporary substructures instead that the final one. If one of these substructures report structural failure, the whole solution is automatically discarded. If it does not, it is allowed to continue its development. It is worth remembering that for the L-System genotypic representation, building blocks can be placed only starting from already existing block in the structure. This makes possible to simulate a developmental process under loads. The procedure puts additional constraints, further decreasing the size of the region of the solution domain to search through (which nonetheless will be still immense). Not only an individual has to be stable at his final state but also during all the phases of it development. It means, in other terms, that these structures, within the limits of the simulated environment and the mechanical properties of their constituents, are autonomous in their morphogenesis without needing any additional support. Fig 23 shows one the evolved configurations with the flow of induced translational and torsional forces in its body. Fig 24 shows a close up of one of the boundary and the part of the structure where the bifurcation occurs. The translational and torsional components have been rendered using the following colours gradients:

< torq strength/7	< axial strength/7
< torq strength/6	< axial strength/6
< torq strength/5	< axial strength/5
< torq strength/4	< axial strength/4
< torq strength/3	< axial strength/3
< torq strength/2	< axial strength/2

“torq” and “axial strength represent the maximum value of load that a joint can bear, torsional and translational component respectively. These two values can be changed to test different structural response to loads and, consequently, trigger different morphogenesis.

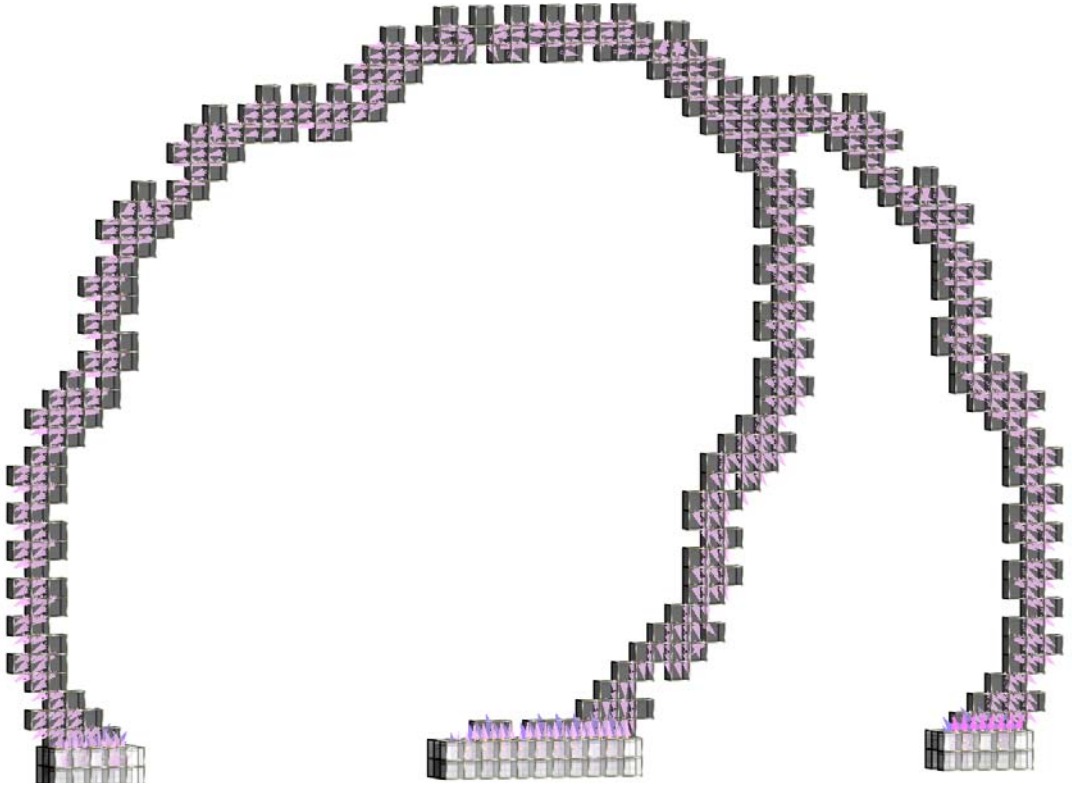


Fig 23 flow of induced translational and torsional forces in the structure

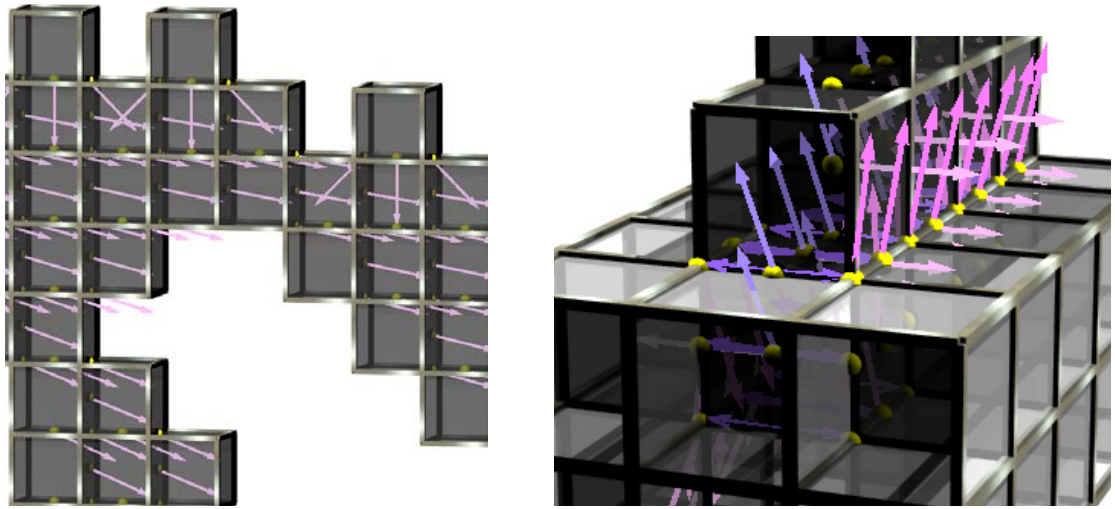


Fig 24 translational components | forces at boundary

Selection : an Elitist approach

As already said in chapter 1, the selection operator has the duty to rank the solutions according to some criteria with the objective to preserve good genetic material, still allowing differentiation. In order to achieve an effective searching into the solution domain, the possibility to incur local maximum should be, as much as possible, eliminated by exploring multiple directions.
The roulette selection, developed by Goldberg (Goldberg 1989), provides a method of selection where the probability for an individual to be chosen is proportional to its fitness value. It gives a high probability to fitter individuals but also leaves a certain possibility to the less fit. The slices of the pie chart shown in fig 25, represent fitness values of the individuals for a generation, the darker the bigger. More precisely each slice indicate the percentage of each fitness value in respect to the sum of all the values. To chose a solution, a random fraction of this sum is taken, acting as a threshold. The fitness values are summed in random order until the threshold is reached. The last value added corresponds to the solution to be selected, favouring in this way individuals that have bigger fitness values being the threshold reached faster by adding big values, fig 25 right .
An advantage of this technique is to encourage multiple searching directions but, as a drawback, good genetic material is often lost over generations (Kramer, Kunze 2005) .
One of the major problem when using Genetic Programming is the disruptiveness of its genetic operators (crossover and mutation), which can easily lead to the loss of good solutions when recombining their genotypes (Bentley 1999). It is worth mentioning that with GP the structure of the body plan is not embedded but is evolvable. This means that even when a very good solution is found, there are no guarantees its genotypic representation will be kept over generations. For this reason a selection operator based on elitism, rather than on selection proportionate to fitness, seems to be a more sensible approach. In contrast to the previous strategy, elitism entails the copying of a certain number of best performing members into the new generation. In this way good qualities will not be lost through mutation and cross over. These members remain unaltered until better performing individuals have been found, acting as a source of good genetic material. The diagram in fig 26 shows a typical workflow through a series of generations. A fixed or variable number is chosen to be the size of a subset of each generation, the “parents”. Those are the best performing solutions within a generation which is created by recombining their genotype as many times as the total number of individuals is. The magenta ellipses in fig 26 represent those solutions that were chosen to be parents and also manage to survive from one generation to another because no other solution outperform them. Over generations, there is an increase of “long living” solutions which can cause the searching to be restricted to a very specific region of the solution domain. To avoid this drawback, it is possible to set a maximum number of generation an individual can go through, being it discarded when this limit is reached.

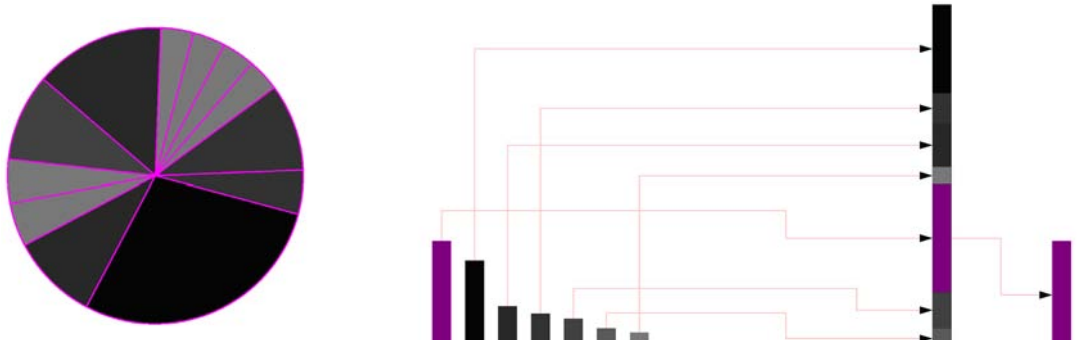


Fig 25 Godberg’s roulette wheel | selection proportionate to fitness

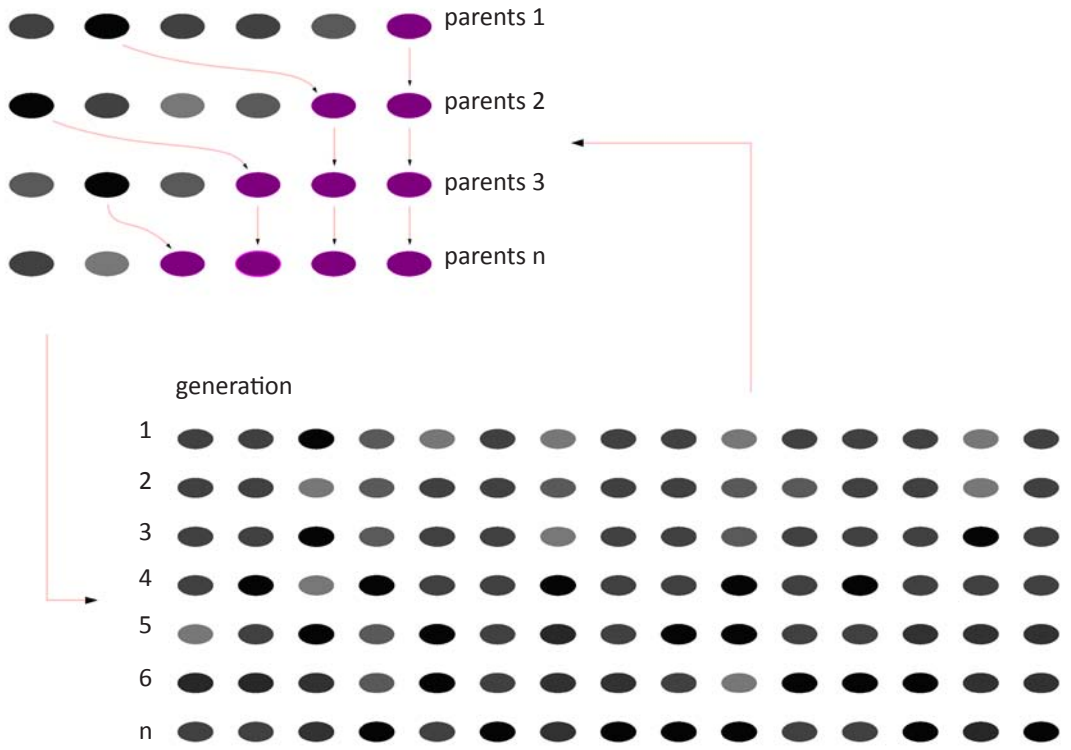


Fig 26 elitism | best performing solutions are kept over generations

Genetic Operators : CrossOver and Mutation

Recombining the genotypic representations of pairs of selected individuals is usually performed by swapping two or more parts of their genotype, making two or more new genotypic representations. This shuffling of information ends by generating a new population of configurations that will share same traits with their old parents and present completely new features. When using explicit or implicit embryogeny, the difference between the individuals of two successive generations is very neat at the beginning of the evolution, while it generally tends to be less pronounced over generations. The inherent disruptiveness of the genetic operators, when working with Genetic Programming, is mainly due to the tree data structure nature of the genotypic representation. Fig 27 shows the typical "one-point" crossover technique which starts by selecting a random node position in both of the genome that are going to be bred. All the other nodes below the selected position are designated to be exchanged with node coming from the other genome, which have undergone the same process. It is clear how dramatic is the change in the structure of the two new genome compared to the parents' one.
This change in topology of the genotypic representation is fully reflected in variability of "body plans" in the phenotypic representation. If only a small set of symbols changes in the L-system strings bracketed expression, its interpretation is completely altered. What said so far has always been seen as a serious problem which can effectively disturbs the evolutionary endeavour. Small variations of the genotype should reflect in small variation of the phenotype (Bentley 1999). On the other hand, this is also what gives to an evolutionary system an immense source of material to work with. To tackle this problem, different type of genetic operators have been developed by GP practitioners, amongst with, a one point crossover only between genotypes that share similar hierarchical structures (Poli, Langdon 1997).
The approach that has been taken here instead, is to work only with the productions rules rather that with the whole genotype. A production rule is the sequence of symbols that is recursively replaced, when matching characters occurs, to form the string expression. These rules can be parsed into tree data structures, being them part of the larger string that they generate, which is why crossover can be performed on them rather than directly on the genotypes. In this way breeding can happen in an abstract space without being affected by the structure of the genotypic representations.
Mutation is here implemented as a chain of events whose happening is stochastic. The most common mutation, when working with tree data structures, are the addition, deletion or copying of new branches as well as the mutation of single nodes. Fig 28 shows a series of mutation such as the inversion of branch a, whose branches c, and e have been eliminated, the deletion of branch b which has been substituted by d, a deeper level branch of a.
Mutation is particularly important to introduce new genetic material when the degree of variability between two successive generations has reached a very low value.

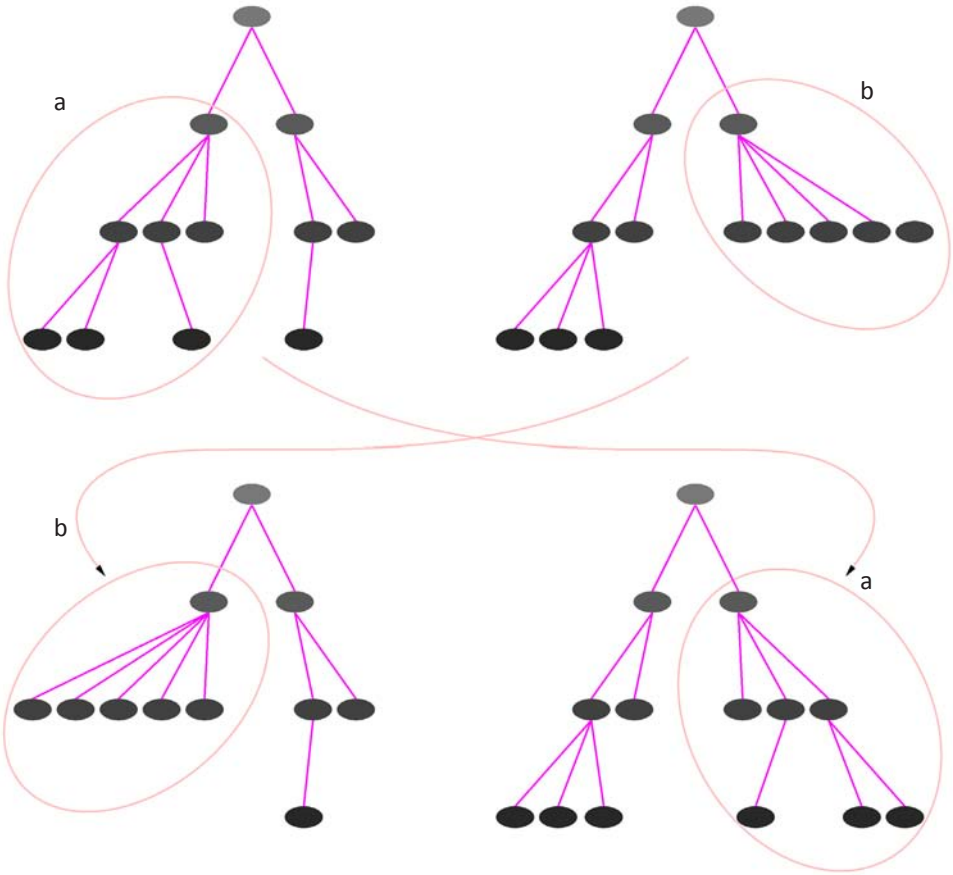


Fig 27 cross over of two tree genotypes

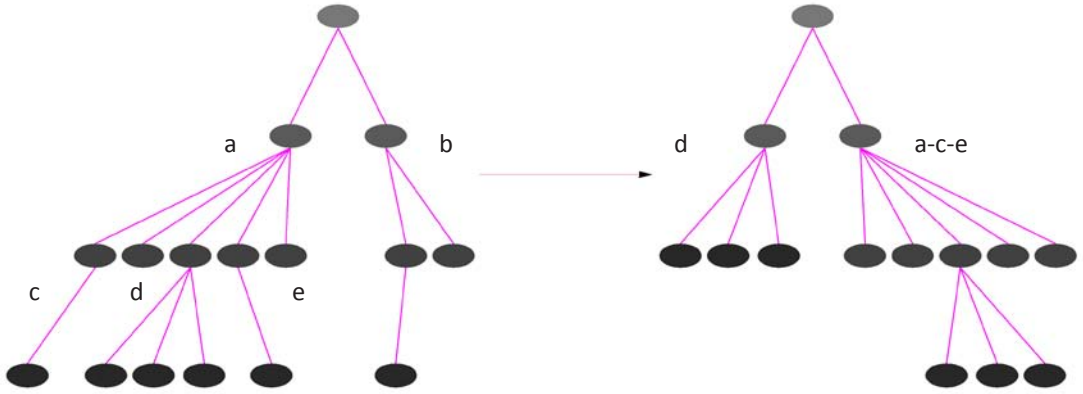


Fig 28 copy, delete and add branch mutation

The software workflow

EvolvedStructures, the name of the evolutionary system here discussed, was first created by James Galasyn (Microsoft) who took as model the experiments carried out by Coates at CECA (Coates et al. 1999). I have further developed this system by modifying part of its structure, adding new features and building a new layer which has brought it to be converted in to a plug in for Rhino 3D (McNeel). The application has been written in C # using the .Net Platform launched by Microsoft in the year 2000. C # is an Object Oriented Programming language, an object being a structure containing data and methods that manipulate other data. What follows is a small overview of the key concepts of an Object Orientated language which will ease the explanation of the software workflow avoiding to confuse the reader with a terminology who he/she might be not familiar with. The main concepts are :

• "A ***class*** is simply an abstract representation of a type of object; think of it as a blueprint that describes the object. Just as a single blueprint can be used to build multiple buildings, a class can be used to create multiple copies of an object" (Classes and Structs , C# Programming Guide).

• A ***property*** can be thought as an attribute that belongs to the class, and indeed to the object that reference it. For instance, each node in the tree data structure genotypes is an object that references the class "node". An object node can retrieve different type of information such as which other node is its sibling or which one is its parent. These are both properties of the class "node".

• A ***method*** is something that can be quite closely compared to a "function" in Visual Basic or script environment. Through methods, objects perform actions on data, other objects or on themselves. For instance one of the methods of the class *EvolvedStructures* is responsible to interpret the bracketed string expressions that comes from the L-System, generating the series of building blocks that will form the structure of the individual.

• ***Events*** tells objects to perform an action if a specific occurrence has taken place. To give an idea clicking one of the bottom of the mouse is an event.

• ***Encapsulation*** means that a group of related properties, methods, and other members are treated as a single unit or object.

• ***Inheritance*** is the ability to create new classes based on an existing one. In this way the new class inherits all the properties, methods and events of the base class. For instance the class *BlockStructure,* which is responsible to organise the body of the individuals in three interlinked parts, is inherited by *BlockStructure3D,* which adds specific methods and properties for the 3D case.

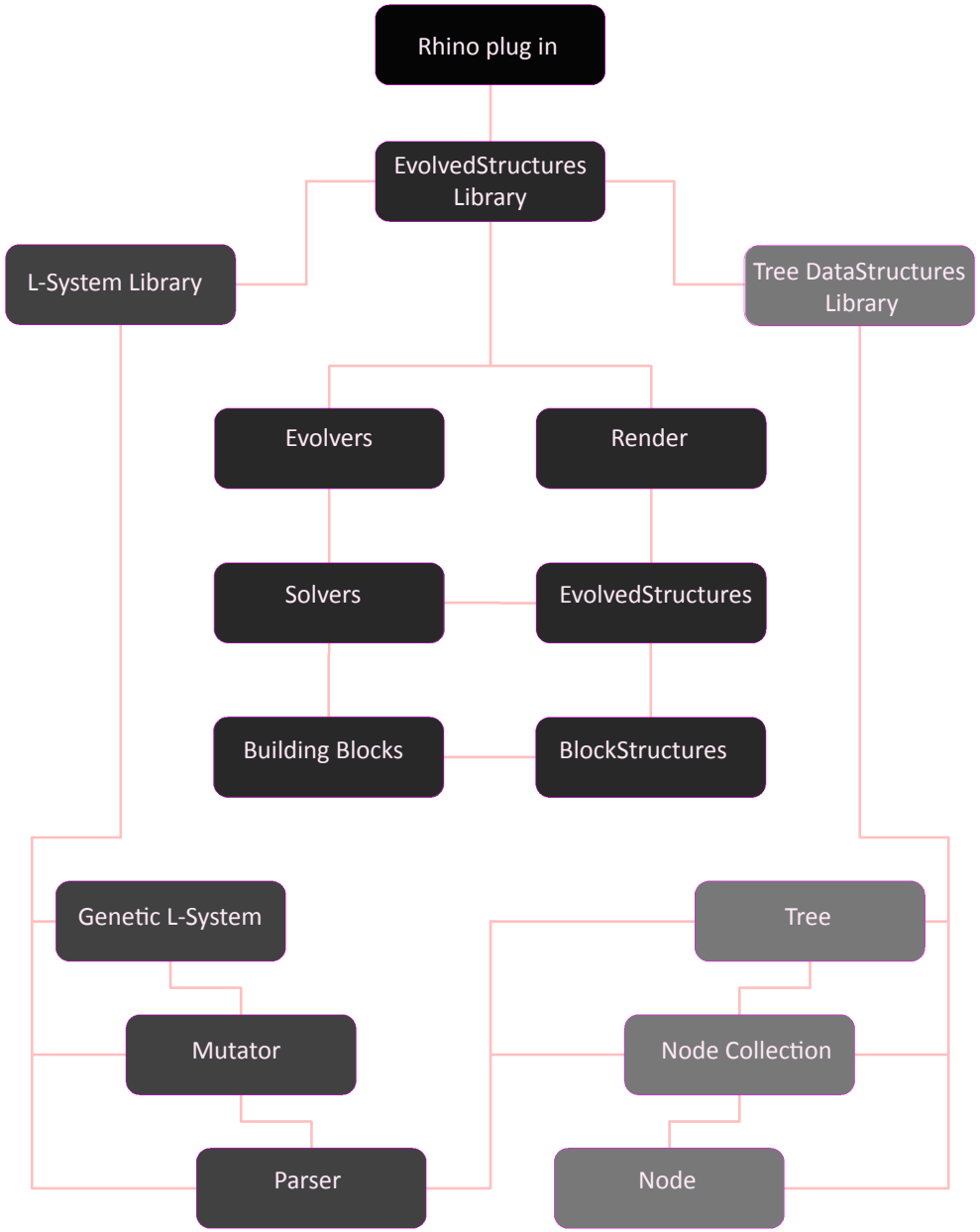


Fig 29 abstract workflow of the system

ents classes although they implement the same methods, properties and events in a different way. This is extremely convenient especially in big application because it allows to use same name of items without worrying about the particular object that is in use.

The diagram in fig 29 show how the different modules of the application are linked together to build the evolutionary system. The software is mainly composed of three libraries :

• the ***Tree Library*** hosts all the classes that it is necessary to implement to work with tree data structures. *Node*, *Node Collection* and *Tree* are the main classes. An object of type Node is the elementary unit of the genotype, a collection of these objects make a branch (Node Collection) and a series of node collections forms an entire tree.

• the ***LSystem Library*** contains everything concerns the production of the bracketed strings rewriting technique. Part of this library is also the series of classes that forms the ***Mutator*** operator and the implementation of the ***Parser***. This last module is the one responsible to convert the genotype expressed as strings into tree data structures.

• the ***EvolvedStructures Library*** constitutes the core of the application being the place where evolutionary engine is implemented. It comprises in 6 main modules, a module being a series of classes. *Buiding Blocks* for the definition of the units that form the body of the individuals. *BlockStructures* that organise the blocks in different system of blocks, each having a specific role such as boundary, internal and envelope. *EvolvedStructures* is responsible of the graphic interpretation of the bracketed string expressions coming from the *Genetic L-System* module. *Evolvers* and *Solvers* are series of classes whereby the design criteria are encoded into the system. *Evolvers* represent the definitions of the criteria, *Solvers* test the generated individuals against the criteria. Finally, the module *Render* takes the information enclosed in each phenotypic representation and feed the 3D CAD package (Rhino 3D) for their visualization.

I designed the application to be user-interactive making what can be called a “collaborative evolutionary system”. The user is able to exert his judgement independently from the selection performed by the system. In this way if there is disagreement on the chosen evolutionary path, the user can deviate it leading the system to explore other regions of the solution domain. This is mainly done by surfing a generation of individuals and altering the score of a configuration that, according to user’s criteria, has been underestimated by the system. It is also possible to completely eliminate more than one solution from the current generation if needed. In order to monitor the process data are sent to an Excel spreadsheet while the application runs, allowing the user to have statistical feedback in real time on the fitness trend.

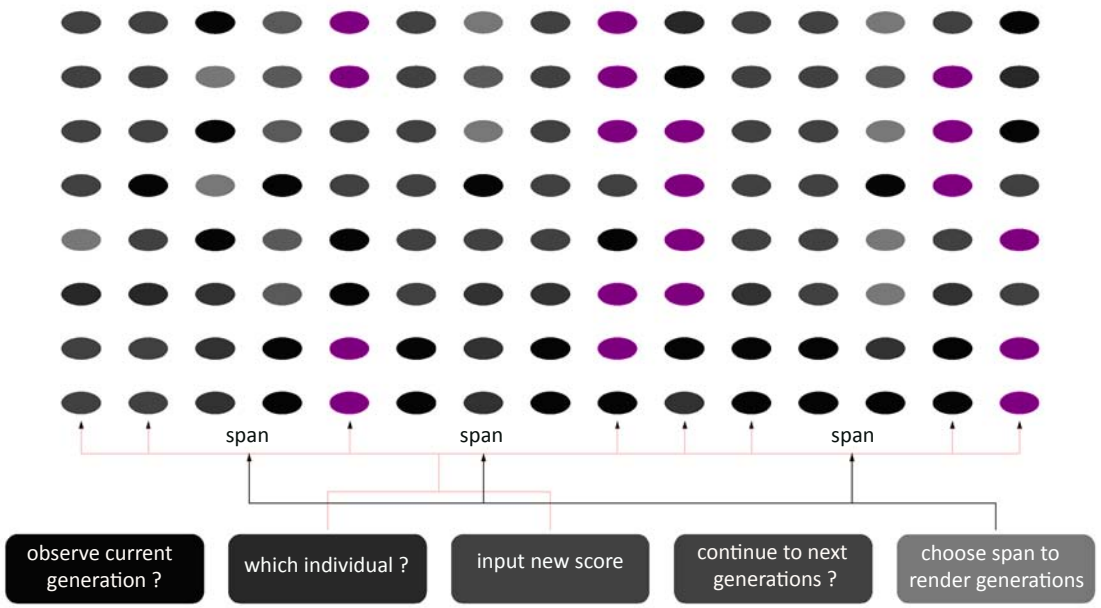


Fig 28 user interaction

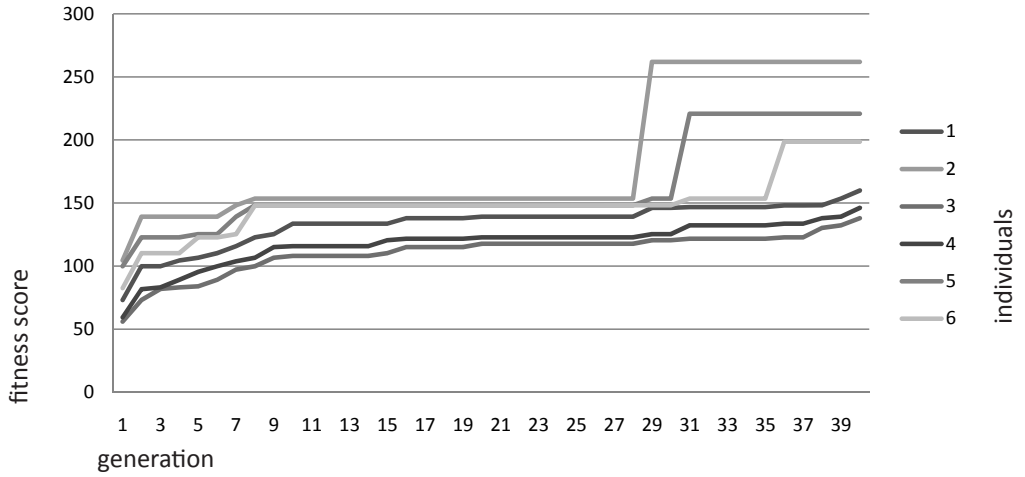


Fig 29 monitoring the fitness trend

In this chapter I present some of the experimental results obtained so far testing different design criteria scenario such as the evolution of bridge like structures and tall configurations. The way in which we should interpret the results is, in my opinion, threefold:

• we should analyze the outcomes under a technical perspective and check if there is any positive feedback between what the system produces and what is embedded in the design criteria that we specify

• we should be able to identify possible expressions of architectural design in the solutions that system continuously present us to fit them in different scenario (this is of course the tricky part)

• we should leave the system enough open for it to occasionally surprise us by presenting to our eyes something that we would have never considered as feasible option.

The word system, from now on, will always comprise the presence of the user, being the feedback created by the interaction between the two parts that generate experience and can potentially lead to novel expression of design.

Early experiments: the emergence of symmetry, bridges and helices

The first series of experiment is conducted without encoding any particular design criteria but just having structural feasibility as design constraint. This means that any solution that presents structural feasibility will be admitted to be a "parent" for the next generation until the number of parents per generation has been reached. It is worth remembering that a parent is just a solution that perform well (in this case which is just structurally stable) which is extracted by a generation after having evaluated all its elements. Parents are the ones that create next generations by recombining and mutating their genetic material. Although a universe of possible body plans would be available just by having structural feasibility as design constraint, the emergence of quite recognisable pattern exhibits frequently. Amongst the others, individuals featuring symmetry and sometimes bilateral symmetry are worth reporting the most (fig 1). What should be highlighted is that symmetry arises spontaneously out of a process of evaluation that test "design proposals" against an environment. The environment here is constituted by a vector field of forces that simulate the presence of gravity and the consequent induction of an internal vector field of reactions that counteract it(as explained in Chapter 2, Load Propagation Model). In so doing, a solution that happen to be symmetrical, is most likely to be stable, certainly having balanced weights. This holds until the induced forces do not exceed the mechanical properties of the

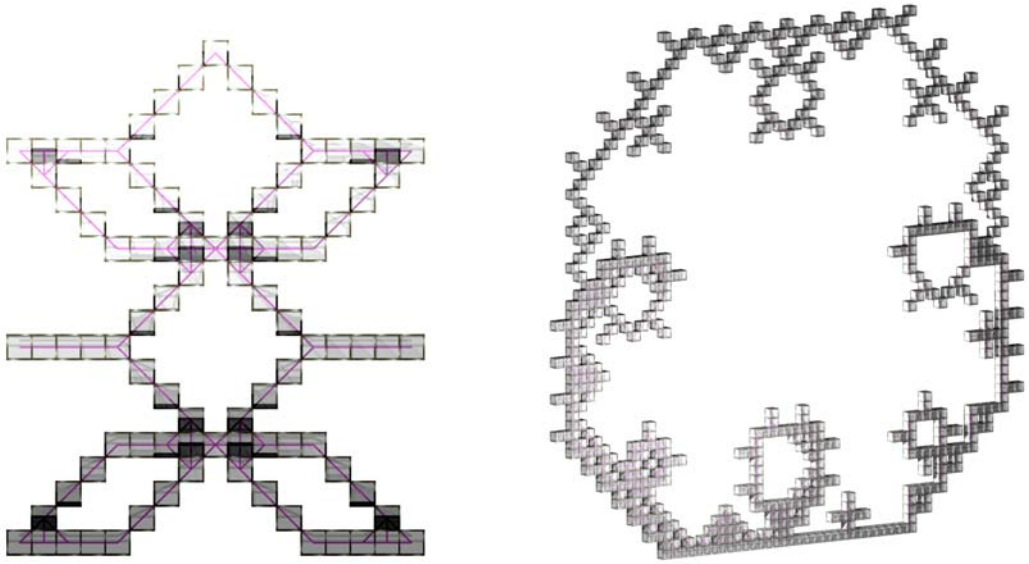


Fig 1 emergence of symmetry

building blocks and the joints. By combining structural feasibility with simple evaluators based on geometrical measurements, the formation of many other patterns can be encouraged. Bridge-like structures are not difficult to be found if, beside structural feasibility, the individuals are rewarded based on the size of their span along a direction orthogonal to the gravity vector, for instance along the x or y axis. Along with this, the individuals are also rewarded for having the smallest possible number of building blocks positioned at the “ground level” i.e. z=0. The score of the solutions will be, hence, directly proportionate to these figures leading to the formation of truss-like structures. If the obtained score is made inversely proportionate to the total number of building blocks, the structures tend to feature slenderness (fig 2), this triggering an economization on the amount of used material. Examples of this topology are the bridge like structures shown here in fig 2 and the three “legs” arch shown in fig 23 in Chapter 2.

If the structural feasibility is coupled with fitness proportionate to the height of the individuals, helices patterns exhibit very frequently. The structures, being encouraged to grow in the opposite direction of the applied load, the gravity, find convenient to develop in an helicoidal pattern rather then straight. This is because helices tend to minimise the torque momentum induced by the presence of groups of blocks that are not aligned with the Z axis. Fig 3 shows a generation of evolved single branch helicoidal structures that have developed starting from not more than a bunch of amorphous pattern of blocks. The emergence of multiple branches spiralling along the Z axis does exhibit frequently as shown in fig 4.

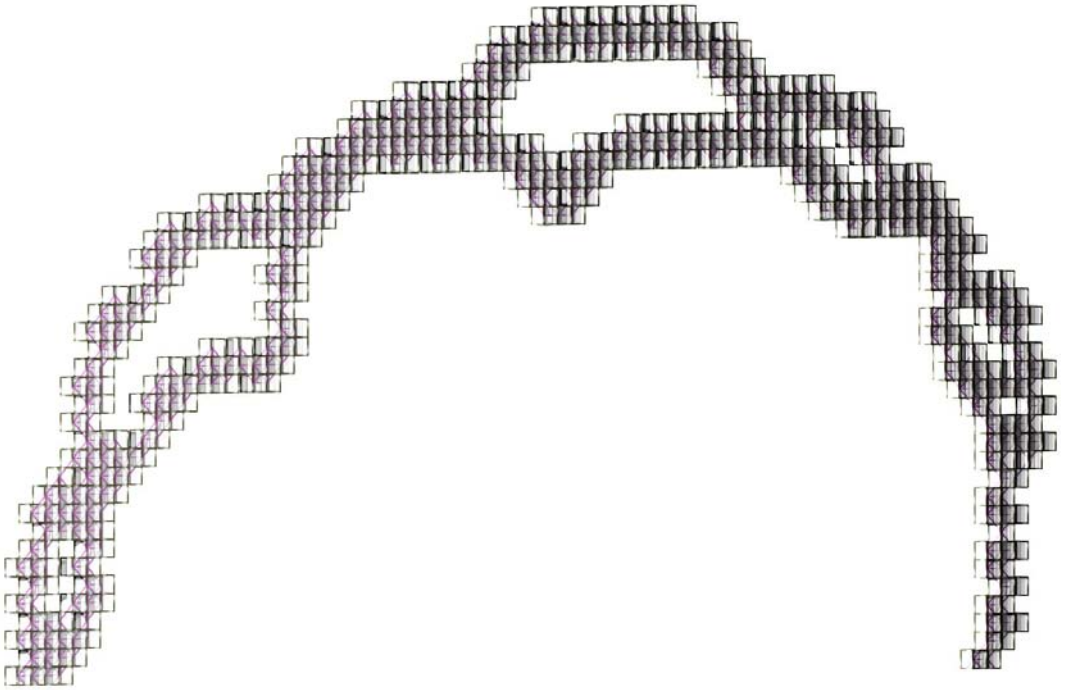


Fig 2 emergence of bridge structures

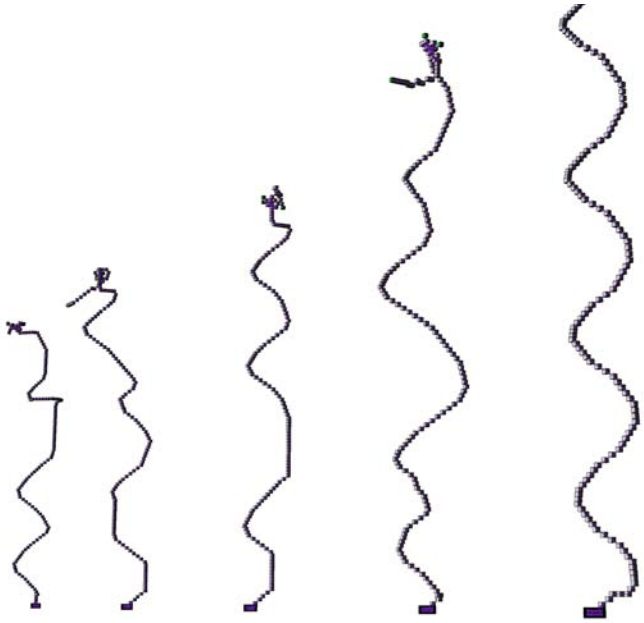


Fig 3 emergence of helices

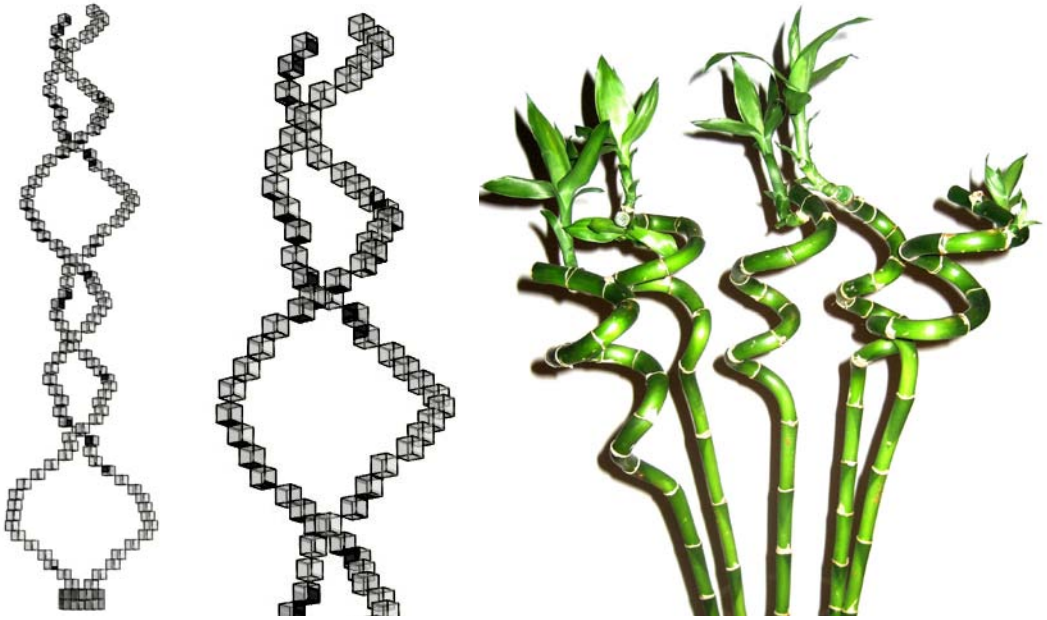


Fig 4 double helicoidal branch | bamboo flowers

Next series of experiments are more concerned to develop an environment which, coupled with the structural feasibility, can lead to the formation of outcomes that can be framed in an architectural scenario. Along with structural stability and fitness proportionate to geometric measurements, I introduce here two procedures:

• ***local accessibility analysis***, is a simple method to assess the topology of the neighbourhood of each building block during the development of the individual. The type of information are related the degree of “accessibility” that each block has through its immediate neighbours. A block that shares a face with one of its neighbour, is certainly more accessible from it than it would be from another neighbour sharing an edge of a vertex with it, fig 5. In this way accessibility assumes here a local meaning, testing the local connection of each block with its neighbours. The higher is the number of neighbours that share faces with a block, the higher is its local score. The sharing of edges and vertex is still allowed determining a smaller value of local score for the block. The structure is traversed in order for each block to check the type of connectivity that share with its neighbours. A global fitness value comes out of this analysis which is an index of connectedness of the spatial configuration as a whole.

• the introduction of three interwoven systems : boundary , internal and envelope blocks. ***InternalBlocks*** are the direct result of the mapping procedure from genotype to phenotype. ***BoundaryBlocks*** and ***EnvelopeBlocks*** depend partially from the topology of the structure and partially from its context. The ***BoundaryBlocks*** constitutes a group of blocks whereby constraining the structure which acts, hence, as its foundations. The ***EnvelopeBlocks*** is an additional layer of blocks that wraps around the internal ones, which is created locally by each block. Once again the structure is traversed and each block checks for the number of neighbours that surrounds it. The creation of the envelope can be triggered by making a condition based on this number. In the case of the structure shown in the following pictures, when the number of neighbours for each block is smaller than two, the block will fill completely all the available positions of its neighbourhood. The addition of the envelope alters to a great extent the morphogenesis of the structure, being the envelope blocks considered as structural elements.
The internal blocks becomes indeed network of paths through which every unit of the boundary and the envelope can be accessed.

The spatial configurations shown in the following images are extracted from different experiments where the individuals are rewarded with fitness proportionate to their height, or more precisely to the Z position of each block in the structure, and to the degree of local accessibility for each block. These come alongside the constraint of structural stability and the user judgement.

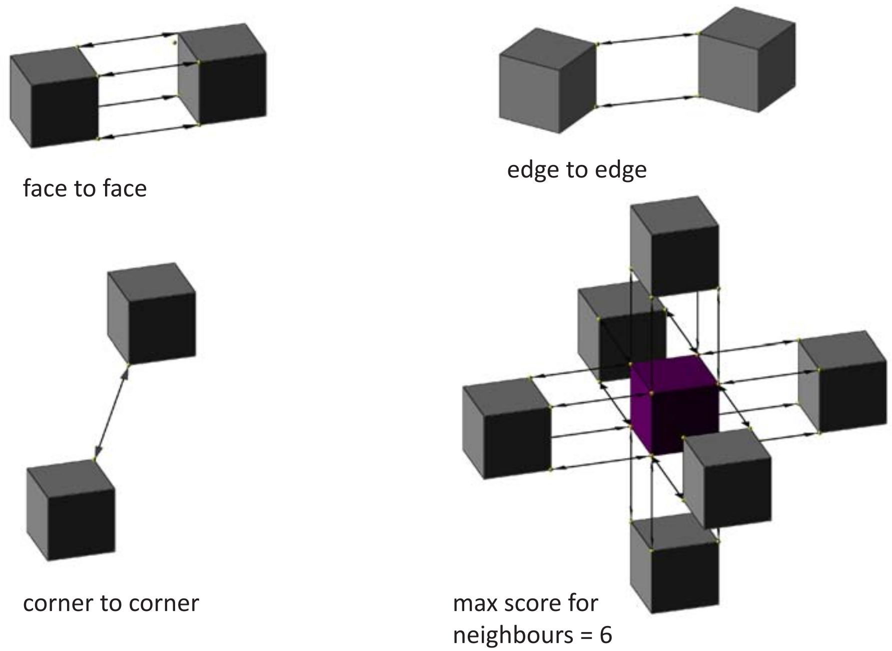


Fig 5 neighbourhood topology

axiom —> F[^^|FF|F+&FF|+FF^F|^FF-F]
n. interations —> 4
sequence length —> 795

production rule 0 | context free | operand —> F | sequence —>

```
++F-^-[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F--^^|F&FFFF|F+F^++]&^
```

genome sequence —>

```
++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^[^^|++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^++F-^-[&F|+-
FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^|++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^+&++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F-
-^^|F&FFFF|F+F^++]&^|+++F-^-
[&F|+FF&FF|&F-FFF]&FF+F&[FF^|^FFF|F&|F--
^^|F&FFFF|F+F^++]&^++F-^-[&F|+FF&FF|&F-
FFF]&FF+F&[FF^|^FFF|F&|F--
^^|F&FFFF|F+F^++]&^^++F-^-[&F|+FF&FF|&F-FFF]]
```

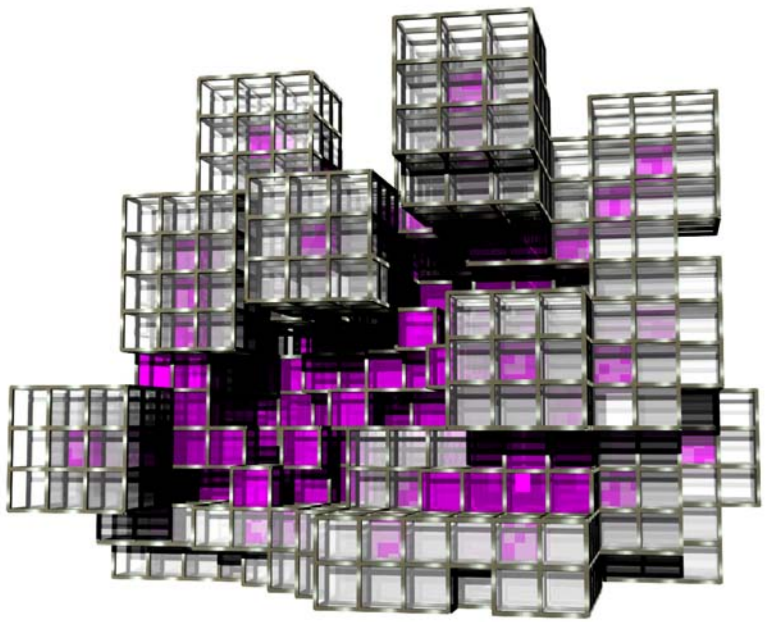


Fig 6 design criteria : local accessibility

Weights and mechanical properties

The fitness function is implemented for a multi-objective evaluation and is also supported by the eye-ball test (user judgement) which is required every time a generation of new individuals is rendered. Usually, in order not to make the process too slow, generations are rendered within a certain range that can be specified by the user or when there is a drastic improvements in the fitness values of one or more of the individuals (fig 28 chapter 2). In this way the user can express his/her judgment only on a small part of the solution space which comprises the best performing individuals available at the time of the evaluation.
When having more than one design criteria, they need to be properly combined in order to make a unique value that can be used to evaluate the individuals. This can be done by identifying regions of the search space that are of particular interest and using their boundaries to normalize each fitness value. For instance, for the development of high rise structures, the height of the individuals needs to be evaluated. This can be done by simply taking the span of the structure along the Z axis or by giving a score to each block which is proportionate to its Z position. Both these two values can be scaled for the maximum height. This value can be derived by assuming that all the symbols in the L-System genotype tells the block to pile on top of each other, as many time as the length of the string is. For instance, if the maximum length of the string is 400 and the edge of each cubical measures 1 unit, the structures can not be taller than 400 units. Dividing the value of the height of each individual by this figure, returns a non dimensional fitness value expressed in terms of percentage which is ideal to be combined with other values. The result that comes out of the local accessibility analysis (which is already without dimensions) and the score for height can be combined by multiplying each for a factor and by summing the obtained figures. The factor is usually called “weight” and it is introduced to bias the searching towards desired region of the solution domain. For instance, very tall and slender structures present a low value of integration, while less tall and blocky ones , fig 7, have a low height value. By tuning the fitness function with the weights, it is possible to reach intermediate position between these two extremes.
In order to favour the emergence of high rise structures, not only the weight relative to the height but also the value of the axial strength of the joints need to be increased. Being the structural evaluation the first design constraint to satisfy, the individuals can increase their height only within the limit of the mechanical properties of the blocks. In addition to this, the emergence of different structural topologies can be encouraged by altering these values. For instance, a low value of the torsional strength of the joint does not allow the structure to develop pronounced appendices, while a very high value usually lead to “hairy” configurations. To make the fitness function dynamic and explore the search space in multiple directions, the values for the weights and the ones for the mechanical properties can be changed over the course of the evolution. In this way the user is not only able to judge the generated solutions on the base of aesthetic criteria but also to radically change the region to explore .

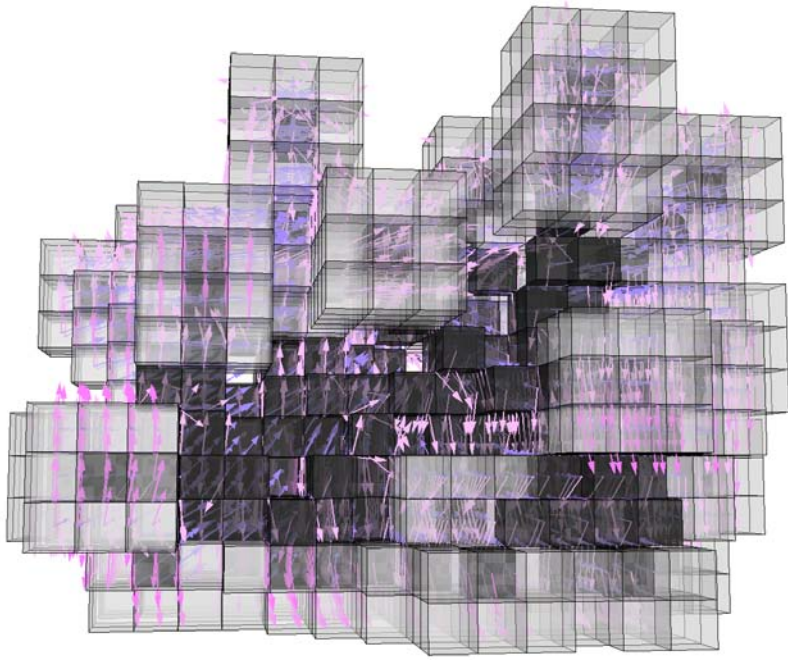


Fig 7 translational and torsional component of induced forces

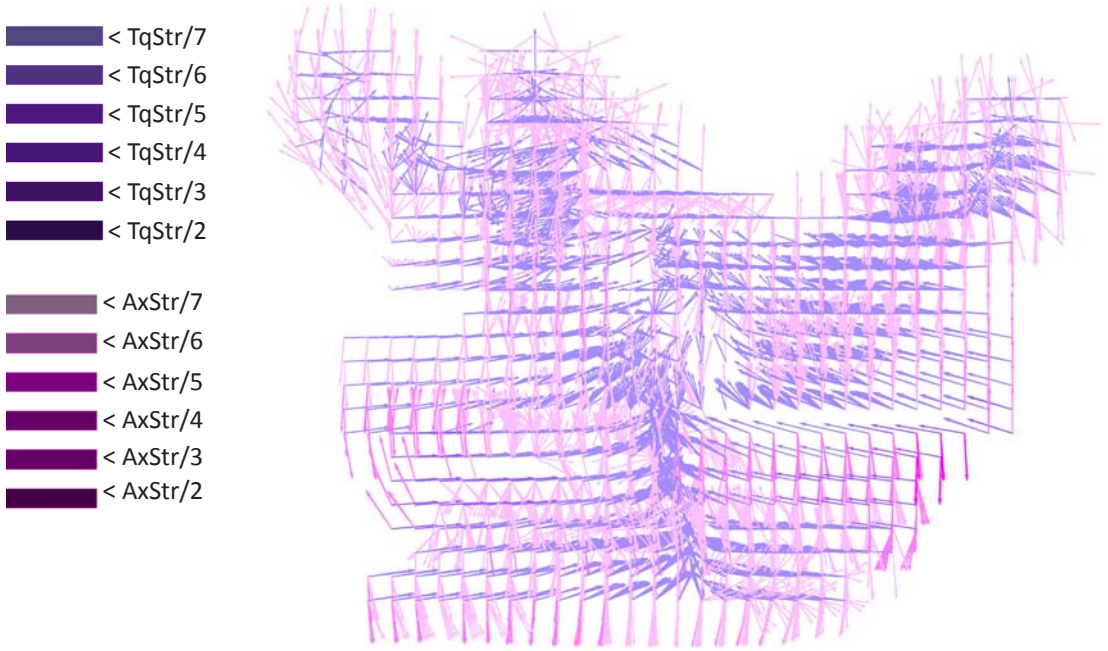


Fig 8 translational and torsional component of induced forces

axiom —> F

n. interations —> 4

sequence length —> 1590

production rule 0 | context free | operand —> F | sequence —> F-FF+&-+++[F^F^F
^F^F^F^F^F^F^F^F^F^F[F&|F^&|+^F[FF^H|&+|F-HF|-FF|&FFFF|F-+F&|FGH-+|HFF&F|F^F^]-FHF[FF^|GGHF|+F-F|^-H+F]-
&|F&H[+HFFFG|F&+F+&|F+HF|F+F&|&F-G&&|+^FFHG|+F+HH|G-+]F^&[+-&HF]F^&|F^&|F^&[-H|H-|FF|FF|G&FG-|FH&+|FFF]
F^&|F^&|F^F^F^F^F^F^&|F^&]G-^[-G|FFF|GF|FFF--]++F|G+FF[&F]FF|FFFFFFFF+FFFFFFFFF+GFF|FG|&+|-FFG[F^+|-FH|FF&F|-
F-|F^+|GF-H+|+F+|H-+G-H]H+FFFG-|HF|F&F|FFFHF|F+FFH|-FG^+^|FFF^G|FFF^G|FFF^G]FH&-|-HFF+G|-FF[+&|F^FFF|FF-
|GFHF+&[F&+|-F&H&F|FF&FGH|F^FF|FFHH+|F&|FFF&G|&G]-F|-F-[HG|+FH+|^F-F&|-F|H&F+|---^]+H&FF[FHF|FFH|-
F|F^GF|+FGHF|F+FFH|HF|&F|F++|H+|+FGHF|+FGHF|+FGHF]FHF[^+|HF|FF&F^|FH-|FFGF|FG+^F-|^^^^|-F]-FFFFFFFFFFFFFF+]
HG[-F|GFFG|FH+|F++[&^HH|^FF&|F^]G&F|F|F+[FF^H&F|^+F&FF|G&&|&G-&G]FH+|FH+[^-FFH|FF|FFF|H^|H&|FF-H|F-
&FFH|+F-GFF]FH+[GFF^FF|FHF|^&^FG&|F^]FH+|FHH-^[&+FH+|GHG-|FF^|HH&|GHG-|GHG-|GHG-|GHG-|GHG-|GHG-|GHG-
|GHG-]&&&&&&&&&&&&&&&&&-|F+|&G&HF|FFF[&FF]F-[FHH|-GGF|+FF-F&|F-GF|FFFF|-G-]GFFG|GFFG|GFFG[FF]
GFFG|GFFG|GGFG|GF-G]G^G[^&+|&&F[HF|FFF&+F|GG+H|FHFFHF|&H|FFFF&|-+GG^F]HHFF-F|GFFFH|FH---&|FFG|FG-FHF|+-
&F|^-G-F|FF&F|HF|-H|+&FF^F]FF-++H[&F+F+F]-F[FF&FF&|FF[FF&GHF]-&[HF-FF+|FF|F^F-FF|F^|FFF+^G|F^F-FF|F^F-FF|F^F-FF|F^F-
FF|F^F-FF|F^F-FF|F^F-FF|F^F-FF|F^F-FF|F^F-FF]HF|FH[G-|+&FF+|+F+GFF|^GFF&F|G^|FFH^F|GFFH|FFFGHF|&+FFF^]F-F|FFFFF[
GG|&FHF^&|FF+HG+|H&F|&FHF^&]G-[^F&F]F++[FF^G^|FGF^|FF^^|^H^GG|FFF^]-F-&+]F-FF[^FH+FF|^FF^F|-^+&+-|^H+|HFFG-
-|FFF|^+|^+|^+|^+|^+|^+|^+|^+|^+|^+]FFF|F^[F&++F&]FF+G|FGHF|&&F-F-[+F|FHFF-|F-HFF|HFF-&|^FG-|FGH&F|FF&|GFF&|-
^FFF&|-^FFF&|-^FFF&|-^FFF&|-^FFF&|-^FFF&|-^FFF&]&G&FF[HF^FF|+FF|F&F&&]-G|^--FF|FF^|FFFFFFFFFFFF|FFH^FF

production rule 1 | context free | operand —> F | sequence —>
-++-&+F^FG[^F+F&|FFF[FHF-+F|FG^|FF&G|FFFFFF|FG|FF|FF^F|FF^HF|FG^|FG^|FG^]FFF[+HHHHHHHHHHHHHHF|F^^&-
|^FF|&&[+F^|^FH|FF|&F+F|F&+]&&|+&F[--FFF]HF^FF&[F&F|FF^GF|FFFFFH|FG&&|FFF]FFF&FF|+FHGF|F-|GG+|-H[FH-
|F+^F|GG+++&|FFF-FF]FF^F[&F^+|G^F-&|&FG|-&+^|+HF-|GFFF^G|-+&^|&^-H|F+|GF-F&F]FF^|FG[GG|G+]+FG|GFFFF^[^FHF&]
FF++F|-G+FH|FGFFF|HF^GG&|-F|&^|GFHF+|GFG|F&GF^F|-F|-F|-F|-F|-F|-F|-F]FFG&[FFFFF^|FFFGHG|&FHF&F]+F[
HHGF|&FF[FGG]F^|GF^[++FH|F+FFF|GHH|F-FGF+|^-F^F|^-F^F|^-F^F|^-F^F]F&F----------------&+[HF&HF|F&F^&F|GF-
F|GH&GFF|FF+|GG^GF+|FF^G|F-F]^F&F[F&FF|-F|FHF-^|&HHF&G|-GHFG^|-F|&FF|^HF&|F+GF&&]FF|F&F-&+[--
|FGFF^|&F|F&F+|+&FFF^|HGF|^-FHF|+F&F^&|G&HF|FG+|^-FHF|^-FHF|^-FHF|^-FHF|^-FHF|^-FHF|^-FHF|^-FHF]HGF--|GFF-
&F|FHHHHHHHHHF[FFF&F]HFFG&|FFFF|+&F|HFF++F|GFHFH[FFGF|&-^F|FF-F-F|F^GF|-+|-F|FFF^F|F-F|-H]F+|HFH+^|FF|FF|F++FF|
HH&G|F&HH|FH&&|HH-FF|-&&|^F&^F|-G+-]FF+[G&|FFHF|++|&^|+-FFHF|GGFGFF|^+F&|+F+GFF|^FF|GGFGFF|GGFGFF|GGFGFF|G
GFGFF|GGFGFF|GGFGFF|GGFGFF|GGFGFF]+^F[FFF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|FF^&HF|F
F^&HF|FF^&HF]-G|F^&&&&&&&F|-&^F[F&FGF^|F+F^|+&&-|FF^F]GF&G[G
GFGF|+^+FG|&GF|FFFFG^]^F|G^GF-+|FF|^^-F&[+F|^HF]FGFFF|FFFF+|^FG|-FF^GF|FF|+F^^-|-F]

production rule 2 | context sensitive | operand —> F | left context —> “G” | right context —> null | sequence —> F-^F+&-+-G[^F+G-[&F^|G-^FG]&FH[-
^GF[FGGF^-|FFG|+&F^|^GFHF+|GGF+GG|HF|FF|H-^^|F^|HFF]&FF[F-F|^F&F+H|HHFF]FF+FF[&F]F+--|HF[&FFF^-]-
G^G|H+G[HFFFG|GFFFF|-&F|FF]^+FFFF|+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+&FF+
&FF+&FF+&FF+&FF&[-FFF-|-FF^^|F&|HFF|FF|HH^&|+FF^|FFGF+&]+-|FFHF[&H^G|&FH-&-|F+F|FH|GG&+|HFF|-G-|F+
|F+F^|F^&G|F^&G|F^&G|F^&G|F^&G|F^&G|F^&G|F^&G]FFF|-FHF|FF^--F|&G]FFF|FFFFFFFFFFFFFFFFFFFFFFFFFG&[-
F^^[F-FFFF|^GGF|HGFG^F|^F|GGFFF]F^FF+[&F+F^|F+&FF&|-F|-&F|FH&|FHFFF|FF^F|FHFFF|FHFFF|FHFFF|FHFFF|FHF
FF|FHFFF|FHFFF|FHFFF|FHFFF|FHFFF]FH|H^H^+-[F^F^|F--|+FFFF|^F|-&FH|^FFFF|FF+HF^|-+|GG|HF]+H[FF+|+G|++|-
^F|^GF|+H|FF-FG]-&F[FF^F++]FF^|FF[-FH|H--+&F|FHFFHG|GHHH-F|FH+&|FH|FGFH++|FG^|FH+F^^|GHGFG]
FF]+F[FHF&H-|+F-+&|G-F&[G+|^F&-F+|FG+^FH|H&F-|F^&F|F+F|GFFFFF|HFFF^]-F|-^FG[-H|FG|-FF|FGG|+F|-&-F|&-]
F^|FFGFF|FFFFF[-F-FF^|-HFFH|FF|H-H|H-H]-^FG[^FH&F|&F|FFG-^|F&|F+HFH|FF^|F^|F^|F^|F^|F^|F^|F^|F^]-^FG|^-FG|^F-
|F&&|+H&F|F+FH^|F^-F+|+-FG^|G&&-|+-FG^|+-FG^|+-FG^]FF+|+^F|-G|+F^|F&[F&-FF|FF-F|F-^H|F+FF|&H|FFF|F^+&]&FFFFFFFF
FFFFF^[FF+|HH&H|-+^|^F+]+FF&^[&GH|F-HFF|FFF|&+|^&GH|-F^FHF|F^FG&F|FF^]FF&FGH]-[+FF+F[^^F^F+|^&F&F|FG+FFF|FF&|-
F|F--FGH|FGF^|+H]FF[&G&F-]FF[FFFF|F+FHFF|-HH|FFFF|FF-|FFG|HFFH|&FG]&-FHFHFHFH-[-&^F+F|^F-&FH|G-F|HHG|G+G|FF|GG]
FF|-+-F+|&F|&-F|FGFFFF|FGFFFF|FGFFFF]&-GGF-GGF-FF|[&-&FF^--&F^|^FF+&&|&FFF&|&+|FF+|+FF^[HH&|^FF|F-+F|FFG-
^|+F|-FFHFH|FFGF-|F-GF&|-+FHH|+--FF|FFGF-|FFGF-|FFGF-|FFGF-|FFGF-]FF-&[FF|FH|&F|F&F^&|G+H+&|^F+F]FFF[GF|-
^^FFG|-&G+FG|-^^FFG|-^^FFG|-^^FFG|-^^FFG|-^^FFG|-^^FFG|-^^FFG]&F-]^^^|FG++[+F+[F^FHF|-F|^-H|F+&HHG|&GH|FF
F&H|^F&FF|GF^G&G|^GF]^-^&|-HF|FF+&G[-H|FF&H|&F+F|F++^FF|^FGFG+|GFG&|F+FG-F|^HFFF-|F&|G+]FFF|F^[GHFF|F-
|FG&|GHFFHF|FFG|F^&FF^|F-|F-|F-|F-|F-|F-|F-]--FF[&F-H|FFFGFF|F^&F^|FF&^|H++FF|FFF|&FF-FF|+F+^^^|-F|FF--^F]F&F|FF&|-
FF--|H&|+-&^|HFGG+|-&F[F+]^&FG]GFF&FF[GG&FF|-FFF|&H^|G^&FHF|^F^^+|H&^|FGH|FGH|FGH|FGH|FGH|FGH|FGH|FGH]
FG++[F^|^-F-|GGFF|^F&+G|F^&FF|^HFFF|+F|HG^&|F+F|&+^]FG++|FG++[^^FG|FF^F^+|HFH^|FFF^|HFF|F+|^GGF|GFG|^^|GFG|G
FG|GFG|GFG|GFG|GFG|GFG]FG&F[FF]FGGGGGGGGGGGGGG+&GGF+[GFHFGF|^^-FF+|H^F^F|G+^^FF[FFF+|FHH|+&-|+&-|+&-|+&-
]--F[G+&F-G|G&&&H|+F-H+^|G-GF-F]FGGH|-F&H[FF^&|FF&]^FG&F^|&-[F-FFF^|HG|GFFF+|+HFFF|+GF|FGHF^|FFHH|+^+&F-|+HF
FF|+HFFF|+HFFF]&GF&GF&GF&GF&GF&GF&GF&GF&GF&GF&GF&GF&GFF[FHGFH+|^FF|F+|FFF|F-]FH&F[GF|GF|FF+F&|FH&|G^F
F|FF|G&^&F|FF]+FG|FG[HFF&F+|FF&F|FF|FFF|^F|^+|&G&&G|G-H|F&HFF+|&^FF]FFH|F&|FF+-]FG+[FHFH[FGF-F|&F|-H|-H|-H|-
H|-H|-H|-H|-H|-H|-H|-H]HF[^+HH-|+-F|+FHFF|HF+&G|FFGF|H^+F|H-FG&G|&&+^]+F-G[FF|FF|-&G&H|FGG-+^|H^F|GFGFF&|-
FG-|^FF]^-[H-F^|FF|G^GFF|^+G^|&F-|^H^H|F^-F|G^GFF|G^GFF|G^GFF|G^GFF|G^GFF|G^GFF|G^GFF|G^GFF|G^GFF]^^[FF++FF]
HF|HF|+GH|++^F|^G^GH+|F^F+F-|F&FF|F^F+F-|F^F+F-|F^F+F-|F^F+F-]+GF+[&F^F&|FFH-+[F+F^F|^-&^HF|F++FF|^F|FFFHFH|^&^^-
G|&+^FF|F-+H+^|FFFG+|-GF^]+FF]FG++[&&G+|FF^|G+FHF&|FH|^&+FF|FF|+FF|G^&F]FG++[FH^F|-F^FFG|G^&G|FF|HG^|FF-
|FFGFF-]FG++|&HHG+-|+F|-+FF|FFH&F&|-&&|HF|FF|F^[HF+-G|^FHF|F+F|GGF&|F&^|&G-|FG&^+F|-^&FFF|FG|F-H]FHF

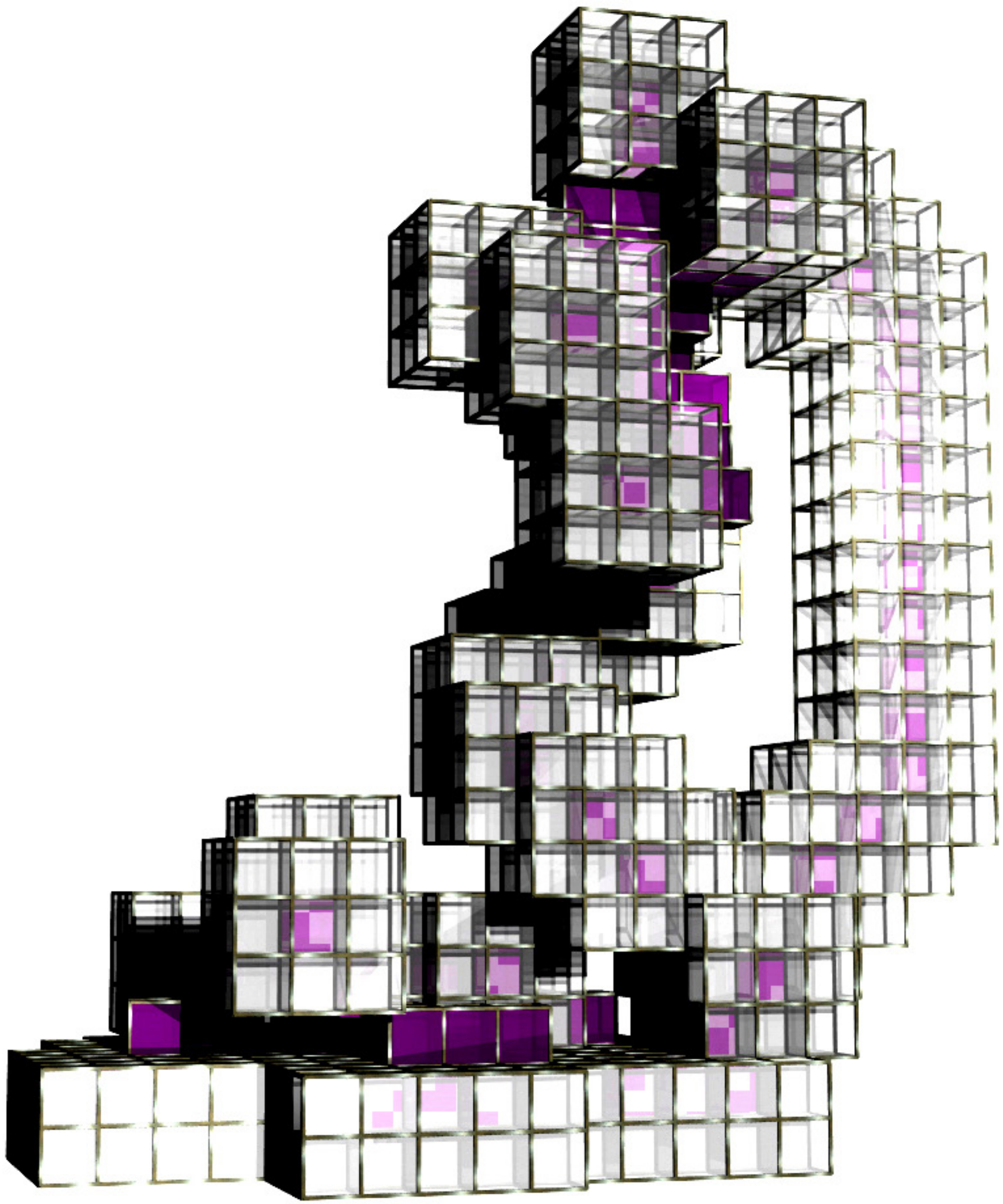


Fig 9 design criteria : stable high structure | connectivity analy-

Fig 11 design criteria : stable high structure | connectivity analy-

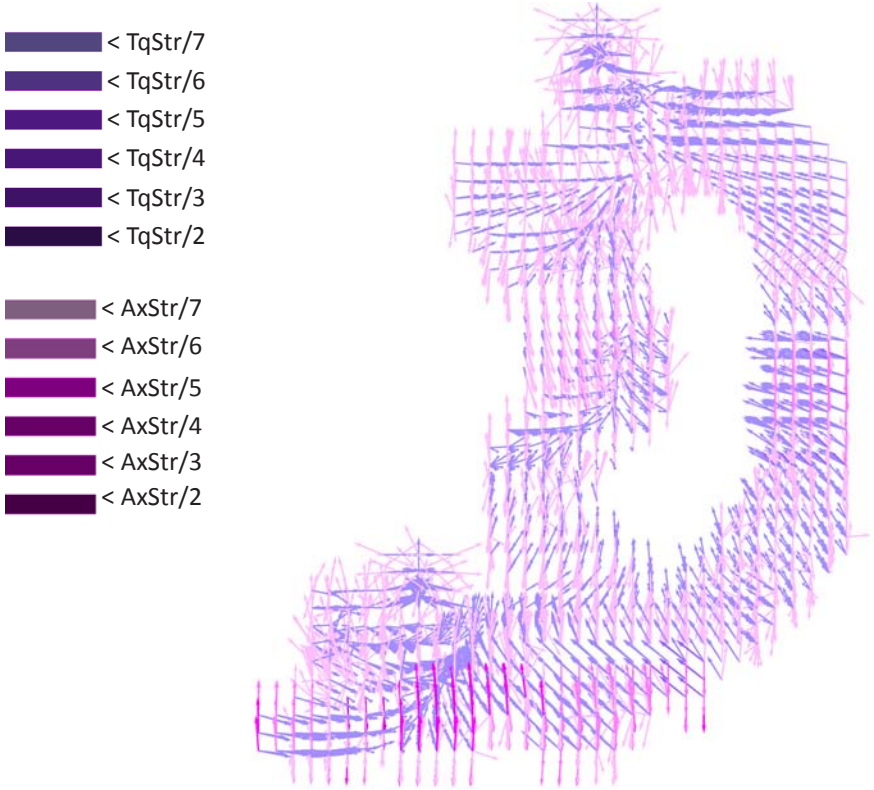


Fig 10 translational and torsional component of induced forces

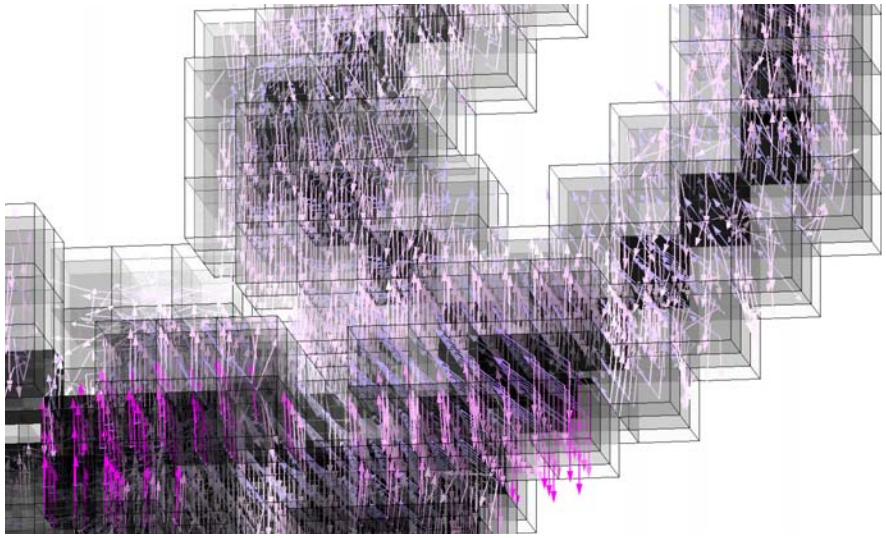


Fig 12 forces at bifurcation

axiom —> F

production rule 0 | context free | operand —> F | sequence —> F&[^+[FFFFFF|&F&-
|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-
FF+^|-F|F^F^+F|FF|-FF]^^-]

genome sequence —> +-F&[^+[FFFFFF|&F&-|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F
&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]F&[^+[FFFFFF|&F&-|F--
&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-
F|F^F^+F|FF|-FF]^^-]&&[&F&[^+[FFFFFF|&F&-|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&
F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]F&[^+[FFFFFF|&F&-|F--
&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-
F|F^F^+F|FF|-FF]^^-]^|F&[^+[FFFFFF|&F&-|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F-
&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]+F&[^+[FFFFFF|&F&-|F--&^F|+F]
F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]+F&[^+[FFFFFF|&F&-|F--
&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-
F|F^F^+F|FF|-FF]^^-]|F&[^+[FFFFFF|&F&-|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F-
&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]^|&F&[^+[FFFFFF|&F&-|F--&^F|+F]
F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-
FF]^^-]^F&[^+[FFFFFF|&F&-|F--&^F|+F]F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F&F[
^++F|&++|&F&&|FF^|F-FF+^|-F|F^F^+F|FF|-FF]^^-]]

n. interations —> 3

sequence length —> 1197

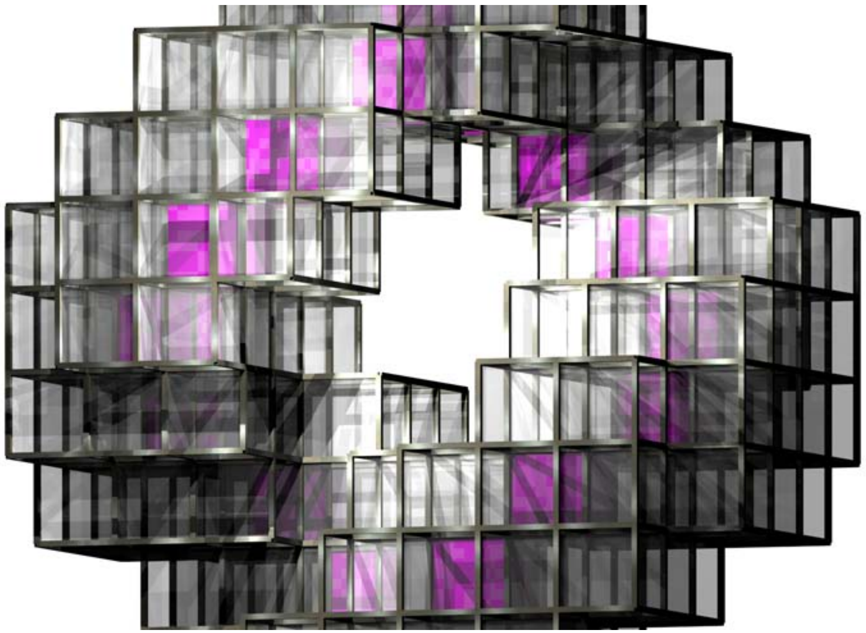


Fig 13 close up | joint between the two helicoidal arms

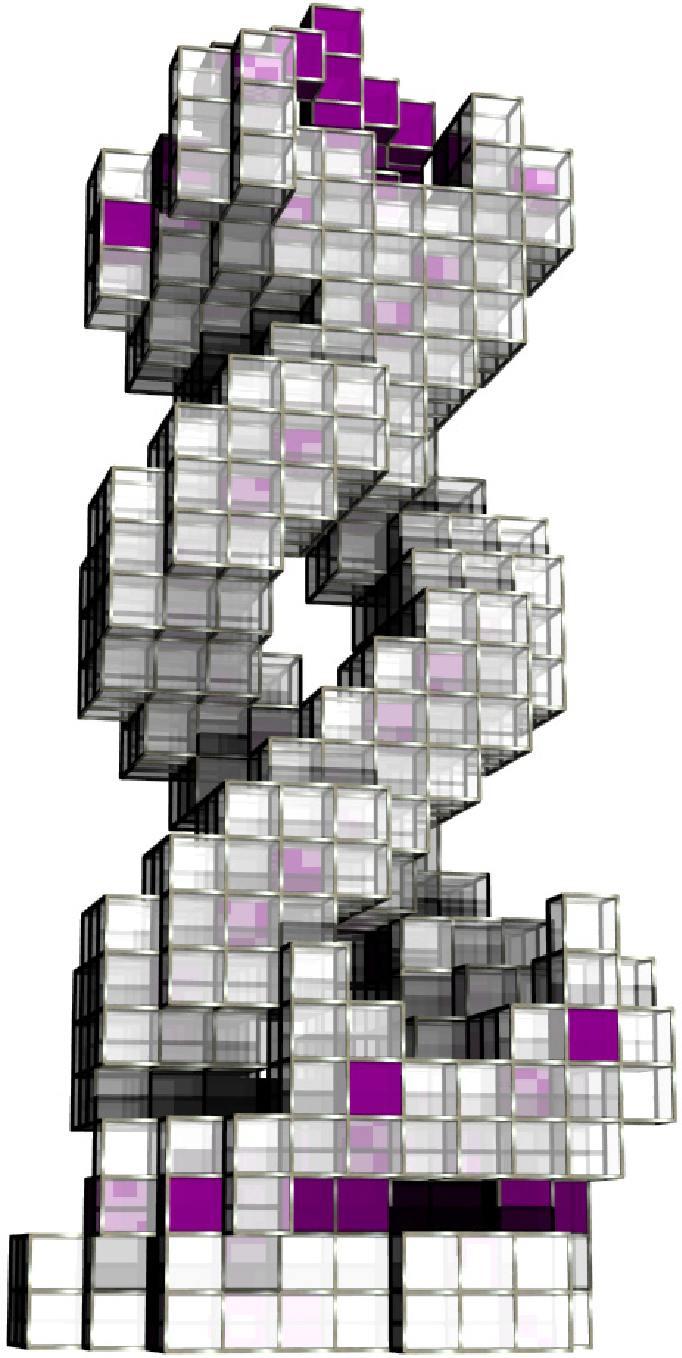


Fig 14 design criteria : stable high structure | connectivity analy-

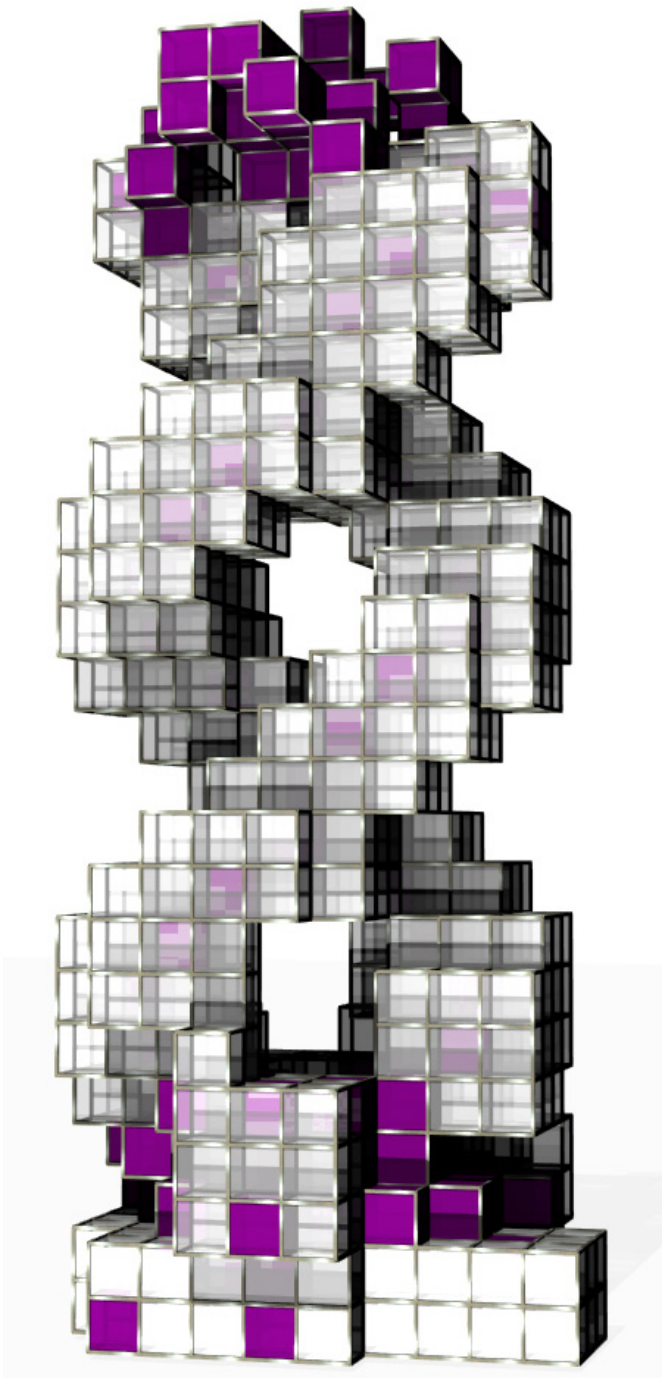


Fig 16 design criteria : stable tall structure | connectivity analysis

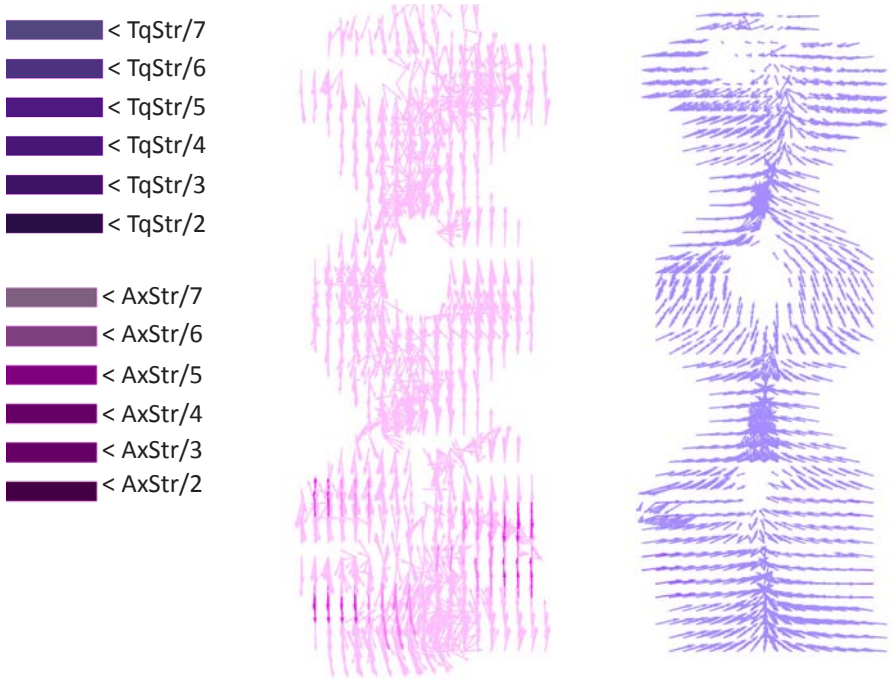


Fig 15 translational and torsional component of induced forces

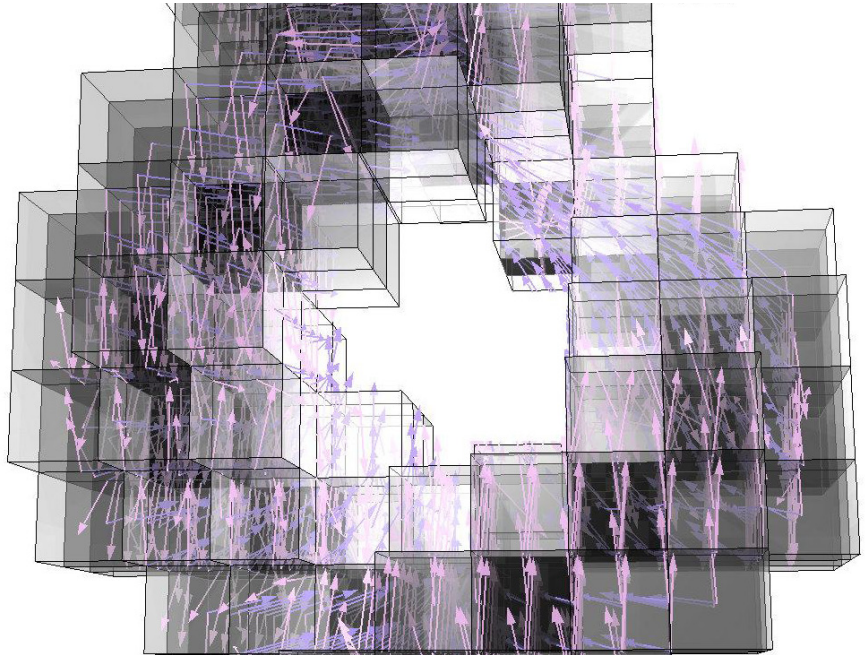
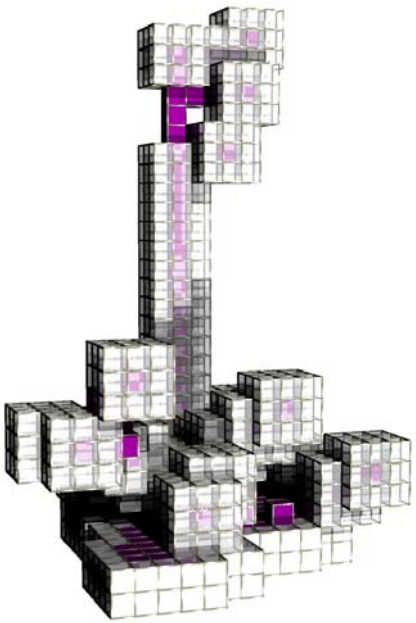


Fig 17 induced forces close up



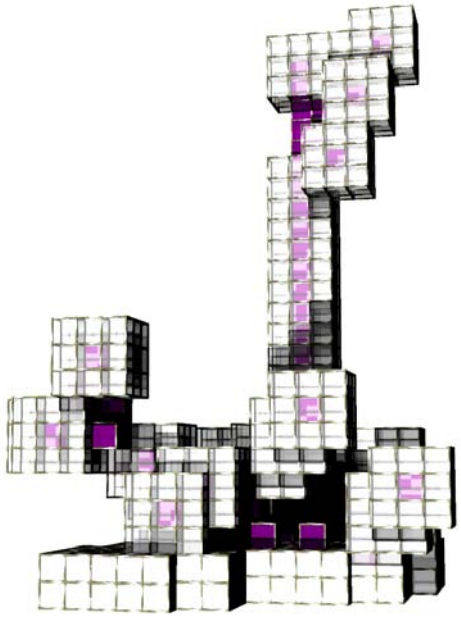


Fig 18 design criteria : stable high structure | connectivity

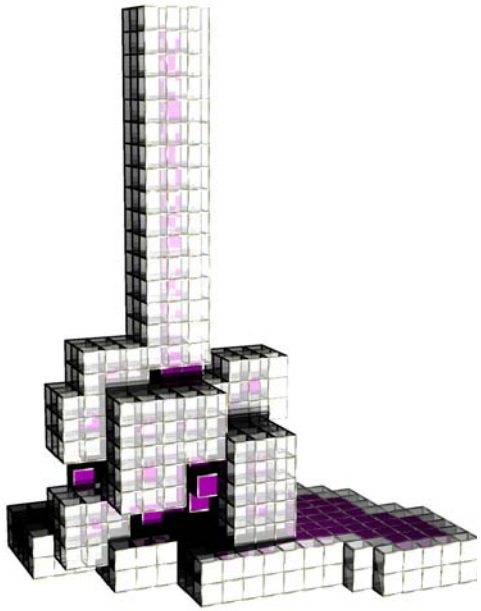
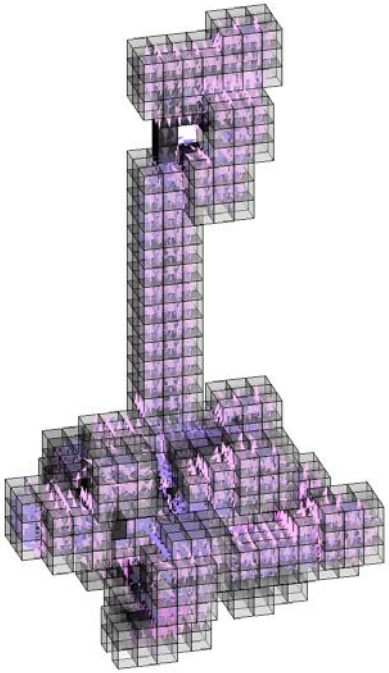


Fig 19 design criteria : stable high structure | connectivity



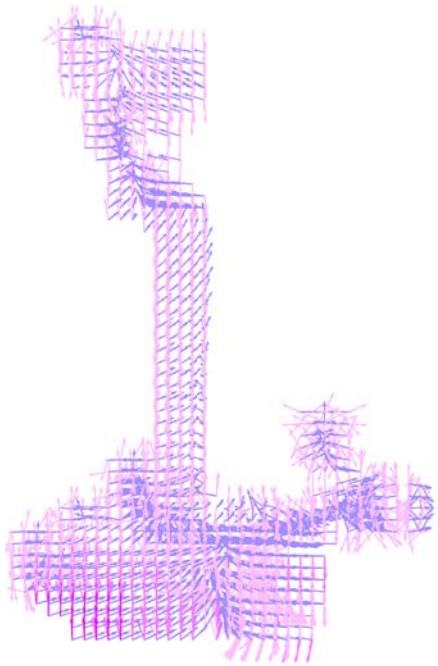
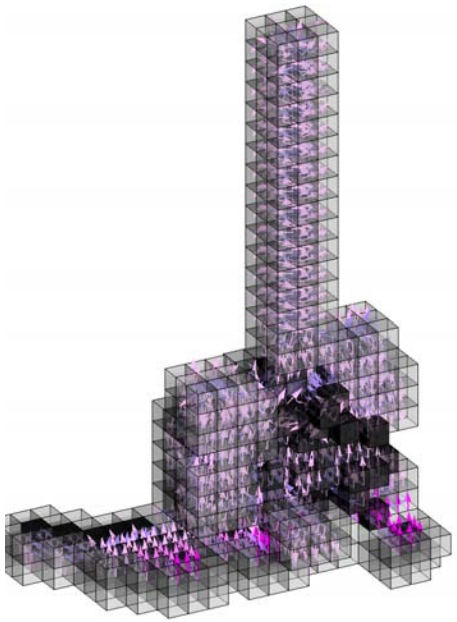


Fig 20 map of translational and torsional component of induced forces



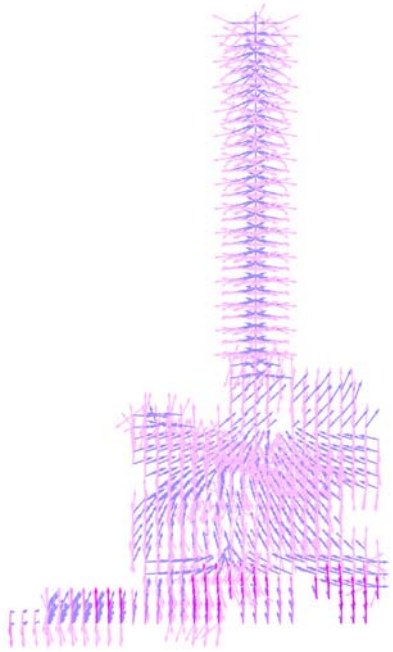


Fig 21 map of translational and torsional component of induced forces

The genotypic representation that has been used so far implies the use of a one to one mapping procedure between the search space and the solution space. One genome is formed, interpreted and mapped to make the phenotype.
An alternative approach is to consider the genotype being made by multiple parts, which are eventually mapped onto one unique phenotype. Different parts of the individual have their own genotypic representation but, during their growing process, interact to form one individual.
Two or more systems are co-evolved in a process where each system constitutes the “biotic” environment for the other parts. Each representation is ,hence, under the pressure of both the biotic (the other systems) and the abiotic environment. The abiotic environment is made by the design criteria/constraints that have to be observed by all the systems present in the individual. The biotic environment is instead pertinent to each part in the system and can be considered as the function that each part performs respect to the others. The experiment of “enclosure” and “space” carried out by Coates (Coates et al. 1999) and Jackson (Jackson 2002) show a possible applications of such a coevolutionary approach.
The experiments conducted in this work explore the structural coevolution of multiple parts within the same spatial configuration. Two genotypic representations are mapped in two different regions of the 3D cartesian space and let interact. The interaction mainly consists in their cooperation in order to achieve pattern formation along the z axis, structural stability and the accessibility of their constituent elements. The structures start developing separately but, during their growth, can happen to share commons spaces which are merged to create a unique individual. Fig 22 shows the two distinct parts of the spatial configuration featured in fig 25. These join in three different locations, in this way supporting each other and reaching structural stability. The two structures would not be stable taken separately .
The configuration shown in fig 26 is composed by a structure that constitutes its main body, which is not stable taken singularly (fig 25), and a second part that acts as a base in which the first is interlocked. This gives to the whole structure the capacity to counteract the weight of its cantilevering appendices .
One of the biggest challenge that such a co-evolutionary approach presents, is a highly pronounced difficulty for the individuals to reach better performances. The multiple representation of the genotype might comport further discontinuity in the search space in addition to what already caused by the use of an explicit embryogeny.
The experiment related to this approach are at a very early phase. The procedure should go major development to enrich the way the structures interact with each other and to find new definitions of multiple representation.

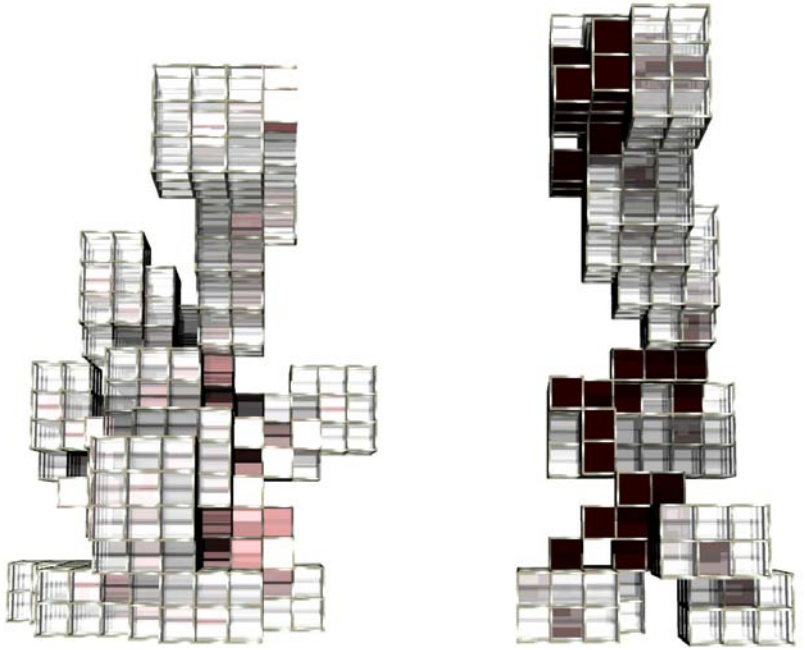


Fig 22 the two structures merge to make one stable configuration

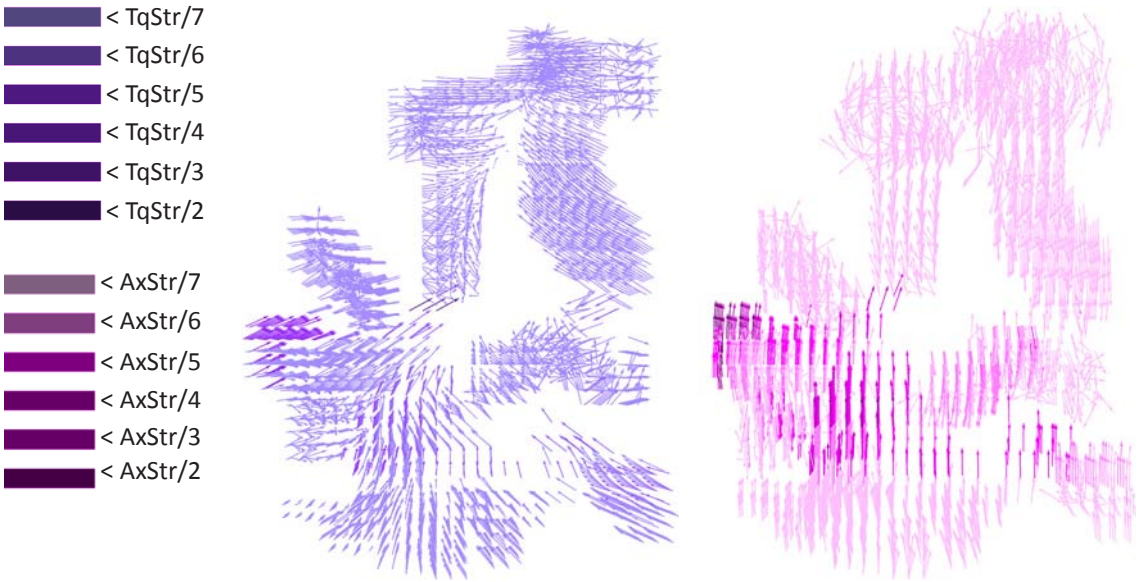


Fig 23 torsional and translational component of induced forces

Structure 1

axiom —> F

production rule 0 | context free | operand —> F | sequence —> ^-+F[+$+£F|£££--
F|$F|FFFF|+F&FF$]

genome sequence —> ^-+^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$][+$+£^-+F[+$+£F|£££--
F|$F|FFFF|+F&FF$]|£££--^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]|$^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]|^-
+F[+$+£F|£££--F|$F|FFFF|+F&FF$]^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]^-
+F[+$+£F|£££--F|$F|FFFF|+F&FF$]|+^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]&^-+F[+$+£F|£££-
--F|$F|FFFF|+F&FF$]^-+F[+$+£F|£££--F|$F|FFFF|+F&FF$]$]

n. interations —> 2

sequence length —> 386

Structure 2

axiom —> FF-F-|F+F|$F-F|FF[FFF+|F-|£FF|F&F$+£]

production rule 0 | context free | operand —> F | sequence —> ++F-£|£^+F£F[+^F|FF^&
&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-
£F[-$F-£|+£F|+F^-]

genome sequence —> ++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-
[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]++F-
£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-
|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]-++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]
F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-
]-|++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]
FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]+++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F
$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-
£|+£F|+F^-]|$++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-
$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]-++F-£|£^+F£F[+^F|FF^&&|FFF|£F
^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-
£F[-$F-£|+£F|+F^-]|++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-
$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]++F-£|£^+F£F[+^F|FF^&&|FFF|£
F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-
£F[-$F-£|+£F|+F^-][++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-
$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$
+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-
£|+£F|+F^-]++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]
FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]+|++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|
FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-
]-|£++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]
FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]
F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-
]|++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-
|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]&++F-£|£^+F£F[+^F|FF^&&|FFF|£F^|$+&F$F|FF+-&]F|F-F£-
[&F£+F|F&F&FF|F£+F|&F-$-$|$^^F-&]FF$|F£F|F-|$&F[+F+|$-FFF&|FFF]F+&F|-£F[-$F-£|+£F|+F^-]$+£]

n. interations —> 2

sequence length —> 2367

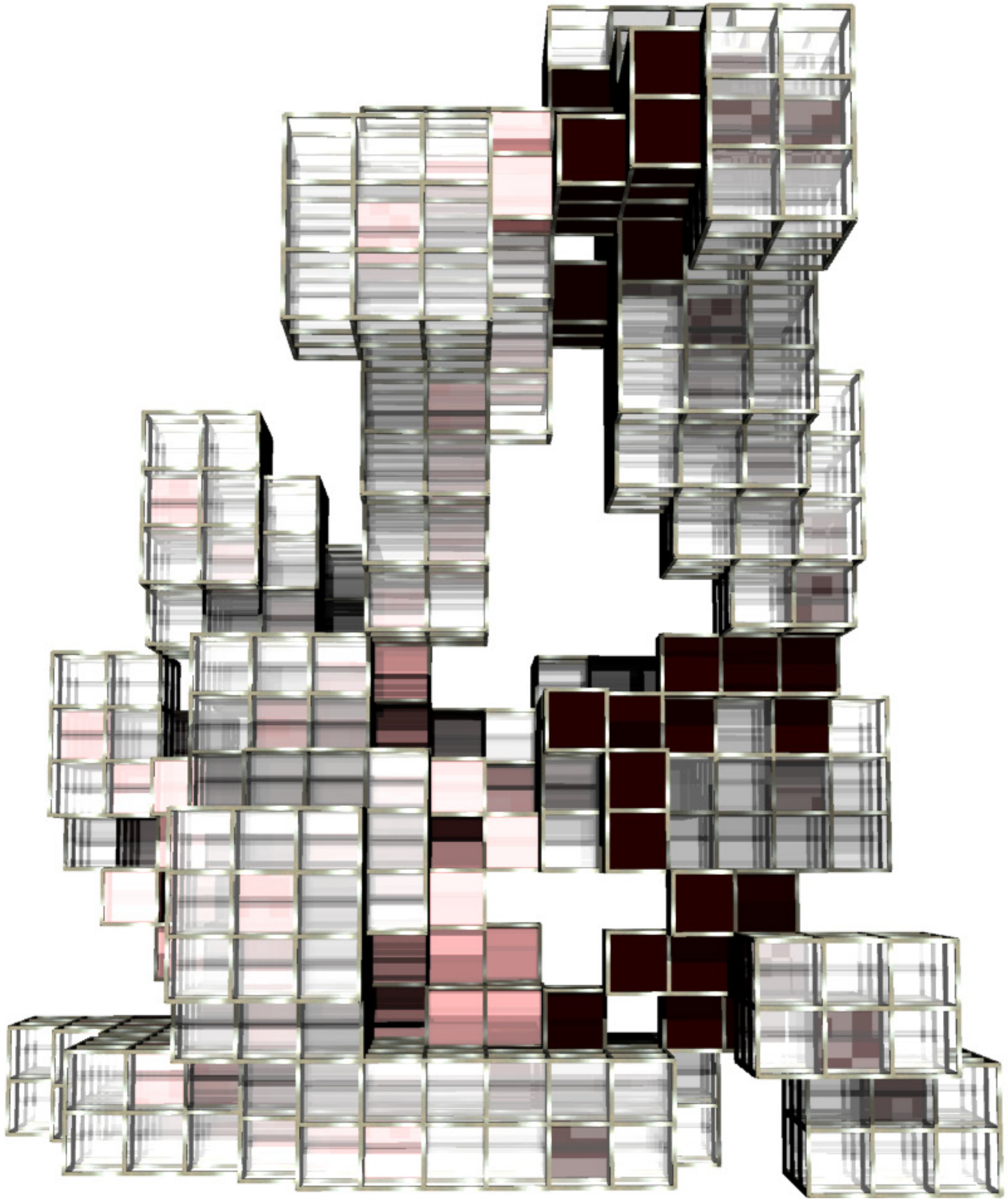


Fig 24 design criteria : stable high structure Coevolver | connectivity analysis

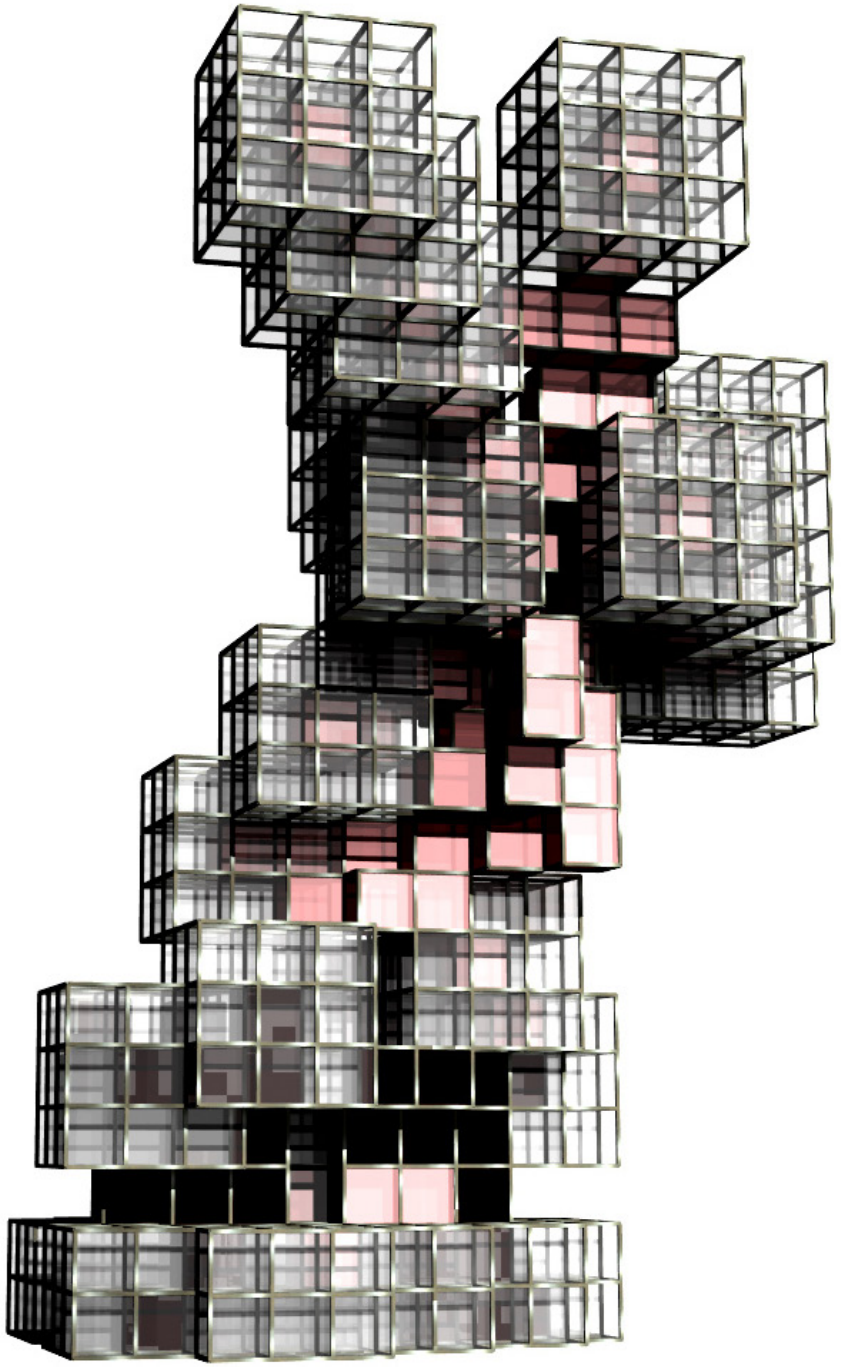


Fig 26 design criteria : stable high structure Coevolver

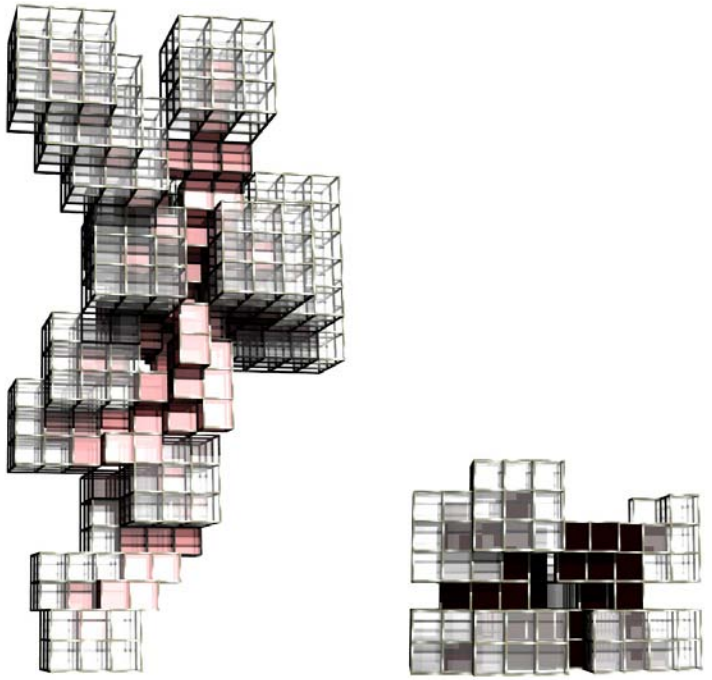


Fig 25 structure 2 acts as a base for structure 1

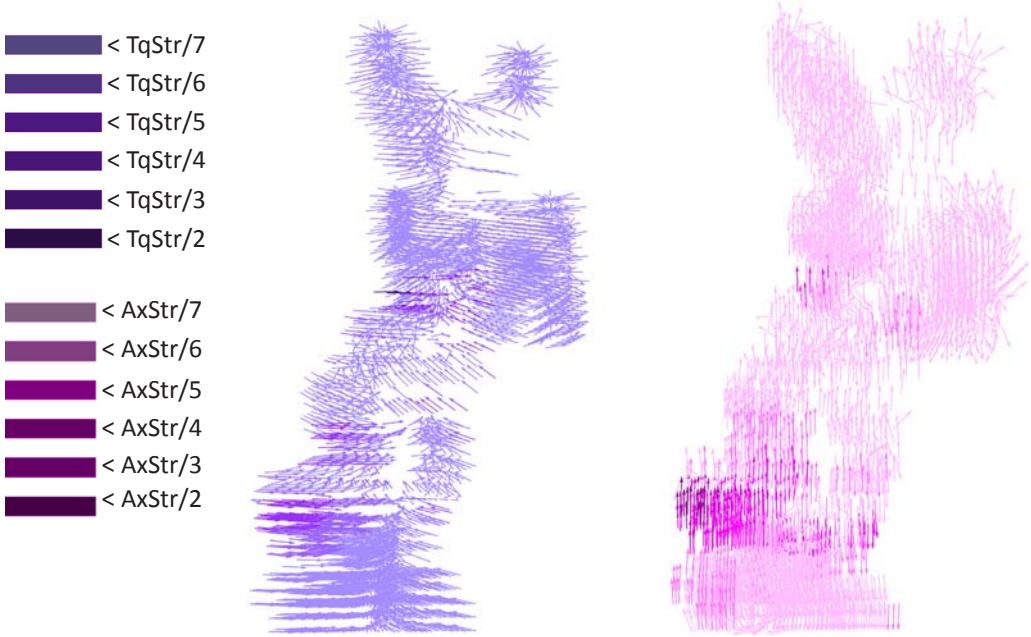


Fig 27 torsional and translatioanl components of induced

Conlcusions

The system tries to generate spatial configurations out of a process of searching, based on the specification of design criteria/constraints, and evaluation of the outcomes. To make this possible it needs the combined action of different subsystems. A generative rules system creates the initial definition of the problem, trying to embrace the largest possible region of the solution domain. A searching mechanism continuously restructures the representation of the search space to generate new possible solutions. A mapping process translates the information encoded in the genotypic representation into geometrical entities. The evaluation of the outcomes drives the process towards specific sub-spaces of the solution domain, in order to accommodate the requirement expressed in the encoding of the design criteria.
The initial proposition, which this work moved from, was to create "geometry" through "topology", to generate formal expressions of design criteria by only specifying the relations between the components of the artefact being designed. The discussed system tries to respect this proposition by evolving the structure in which these relations are framed. This does not have to be intended as the designer knew nothing about what he designs, but rather as the design process started with an ill-definition of what the final outcome is going to be. The creator of the system, hence the designer, brings in all his /her background, being a certain number of assumptions to be made in order to build the system itself. In addition, the interaction between the designer and the system gives the possibility to observe unforeseen solutions to the problem posed by conflicting design criteria. The "*affordancy"* (Gibson 1977 cited in Laughlin 2008) of the system mainly relies, hence, in this interaction, when it reveals to us solutions that we would have never considered but that are nonetheless still feasible according to the evaluation criteria. *Affordancy* is a term coined by the psychologist James J. Gibson to explain the interaction between experience and reality (Laughlin 2008). We, as human being build models of what reality provides. The process of recognising things is based on the properties of the thing and on the subject that recognises it, who is mainly influenced by his cultural heredity (Laughlin 2008). When surfing a population of configurations generated by the system, the user starts to recognise familiar patterns which features symmetry, self similarity within their components and that are close to his/her aesthetic criteria. On the other hand, the feedback coming out of the evaluation of the performances of the proposed configurations, help him/her to consider non-conventional formal expressions that perfectly fit within the context framed by the design criteria. No doubt, hence, the design process still owns its subjectivity, which mainly belongs to the social-cultural background of the designer.
Having said that, it is worth pointing out that the mutual feedback between the designer and the system goes much deeper than the judgement of temporary solutions that need to be evaluated. By interacting with the system, the designer becomes able to understand the behaviour of the system as a whole which gives him the capacity to make structural changes to its framework. This makes a multi-level feedback loop where the designer not only is part of the process (evaluating what the system produces) but eventually becomes the environment for the system (the designer making or altering the structure of the system). The system is, hence, organised in hierarchical layers of subsystems where one can be subordinate to another in some aspect, but superordinate to it for other aspects, in a "heterarchical" structure (Heylegen 1989).
Whether the outcomes shown in previous pages could reach the realm of design briefs, it will definitely require further implementation of the system and further elaboration of the simulated environment. The system should undergo major developments in order to be able to contextualize the environment in which the evolved configurations are placed.
Possible ways to achieve this and future way of development could be :

• The implementation of an implicit embryogeny (Bentley 1999) would make the mapping process adaptive during the development of the phenotype. Parallel processing, conditional iteration, subroutines would be, in so doing, inherently incorporated in the mapping without the need to design them by hand. A direct feedback between solution space and search space, phenotype and genotype would be , hence, created.

• The symbiotic approach whereby interwoven systems co-evolve together towards common better performances, has been explored only from one perspective, the structural stability of a unique configuration from the development of two separate structures. This implies the possibility to extend the process to other design criteria such as the one already experimented by Coates of "enclosure" and "space" (Coates et al. 1999).

• additional force fields should be added to consider the presence of physical agents, such as sunlight irradiation or wind exposure, as well as agents related to urban context such as the surrounding landscape

• the introduction of a context sensitive environment in which the morphogenesis of the solutions can be influenced by the presence of other entities. This will broaden the range of applications of the process which could be used to evolve design that adapt to different type of surroundings.

• the introduction of a procedure based on agents based modelling systems to test the efficiency of the layout of the generated configurations (Coates et al. 1999) would be a valuable evaluation criteria for the system to feature

Further developing the system will eventually lead to the making of a design methodology whereby forms, following forces, become expression of high level of abstractions such as design criteria and the plastic sensibility of the designer, potentially resulting in the formulation of design briefs.

References

Bard, J. (1990). Morphogenesis: *The cellular and molecular processes of developmental anatomy*. Cambridge University Press, UK.
Bentley P. & Corne D., 2002, *Creative Evolutionary System*. Morgan Kaufmann Publishers, San Francisco.
Bentley, P. (1999), *Evolutionary Design by Computers*. Morgan Kaufmann Publishers, San Francisco.
Coates,P.; Broughton,T.; Jackson H. (1999), *Exploring Three Dimensional Design Worlds using Lindenmayer System and Genetic Programming*. In Evolutionary Design by Computers,Bentley P. (1999). Morgan Kaufmann Publishers, San Francisco, p. 323-341.
Coates, P. (2010) *Programming Architecture*. London.
DeLanda, M. (2002), *Deleuze and the use of the Genetic Algorithm in Architecture, Designing for a Digital World*. Wiley-Academy, p.117-120.
Dollens, D. (2005), *Digital-Botanic Architecture*. Lumen, Santa FE.
Frazer, J. (1995), *An Evolutionary Architecture*. AA Publications. London
Galasyn, J.(2008)(unplublished). *Load propagation in a discrete two-dimensional medium*, http://www.leftopia.com/zs_arcology_studies.htm
Goldberg D., (1989), *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.
Heylighen F. (1989): "*Self-Organization, Emergence and the Architecture of Complexity*", In Proceedings of the 1st European Conference on System Science, (AFCET, Paris), p.23-32.
Hillier, B.(1996), *Space is the machine : A Configurational Theory of Architecture*. Cambridge University Press
Holland, J. (1992), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to biology, Control, and Artificial Intelligence*. MIT Press, London, p.1-19
Ingeborg, M. 2004, *Calculus - Based Form: An Interview with Greg Lynn*. In AD Programming Cultures: Art and Architecture in the Age of Software,76(4), p.89-95
Jackson, H. (2002), *Towards a Symbiotic Coevolutionary approach to Architecture*, In Creative Evolutionary Systems, eds. Bentley, P.; Corne, D. (2002). Morgan Kaufmann Publishers, San Francisco, p. 299 - 312.
Johansen, J. (2003), *Organic process*. In The organic approach to architecture, eds. D. Gans & Z. Kuz. Wiley-Academy, p. 95-104.
Kolarevic, B. (2003), *Architecture in the Digital Age*. Spoon Press
Koza, J.(1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*.The MIT Press, Boston.
Krämer, J.;Kunze, J.(2005). *Design Code*. Professor Finn Geipel Labor für integrative Architectur Techniscule Universität Berlin unpublished thesis,TU Berlin.
Kumar, S.; Bentley P. *Implicit Evolvability: An Investigation into the Evolvability of an Embryogeny,* http://www.cs.ucl.ac.uk/staff/p.bentley/KUBEC2.pdf [Accessed 22 Sep 2009]
Kumar, S.; Bentley, P. T*he ABCs of Evolutionary Design:Investigating the Evolvability of Embryogenies for Morphogenesis*,
http://www.cs.ucl.ac.uk/staff/P.Bentley/KUBEC1.pdf[Accessed 18 Sep. 2009].
Lynn, G. (1998), *Variations on the Rowe Complex, from Folds, Bodeis and Blobs*. Collected Essays, Books-by Architects Series, Bibliotheque de Belgique.
Poli, R.;Langdon, B.(1997).*Genetic Programming with One-point Crossover and Point Mutation*. In Genetic Programming 1997: Proceedings of the Second Annual conference on Genetic Programming, eds. Koza, J.;Goldberg, D.; Fogel, D.; Riolo, R.Morgan Kaufmann,San Grancisco, CA, p.278-285.
Prusinkiewicz, P.; Lindenmayer A.(1990), *The Algorithmic Beauty of Plants*. Spinger - Verlag, New York
Steadman, P. (2008), *The Evolution of Design*. Routledge, London, p.238 - 270
Stiny, G.; Mitchell, W.; *The Palladian Grammar*. In Environment and Planning B, volume 5, p.5-18
Thompson, John L. (1994), *The coevolutionary process*.University of Chicago Press
Turing, A.M. (1952). *The chemical basis of morphogenesis*. Phil. Trans. R. Soc., Series B, no.641, vol.237..