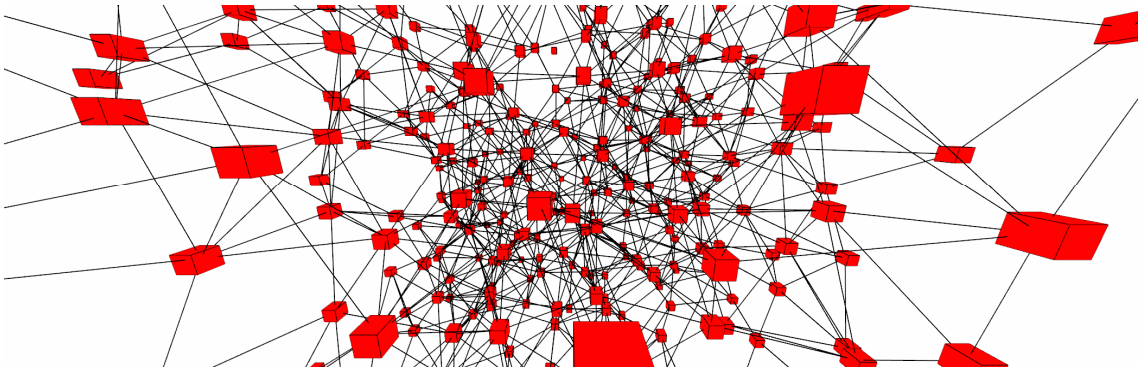
Topological Self-Organisation:

Using a particle-spring system simulation to generate structural space-filling lattices

Anastasios Kanellos



This dissertation is submitted in partial fulfilment of
the requirements for the degree of Master of Science in
Adaptive Architecture and Computation from the
University of London

Bartlett School of Graduate Studies
University College London

September 2007

Abstract

The problem being addressed relates to the filling of a certain volume with a structural space frame network lattice consisting of a given number of nodes. A method is proposed that comprises a generative algorithm including a physical dynamic simulation of particle-spring system. The algorithm is able to arrange nodes in space and establish connections among them through local rules of self-organisation, thus producing space frame topologies. In order to determine the appropriateness of the method, an experiment is conducted that involves testing the algorithm in the case of filling the volume of a cube with multiple numbers of nodes. The geometrical, topological and structural aspects of the generated lattices are analysed and discussed. The results indicate that the method is capable of generating efficient space frame topologies that fill spatial envelopes.

Acknowledgements

I would like to thank my supervisors:

Sean Hanna for all his guidance, advice and patience

Alan Penn for providing inspiration and encouragement

I would also like to thank:

Chiron Mottram for assisting with a preliminary version of the algorithm

Tristan Simmonds for his suggestions on possible applications and advice on structural aspects

Christian Derix for his critical viewpoint

Table of contents

List of illustrations

Appendix I

1.0 Introduction

The problem being addressed in this study relates to the design of lattice structures that can fill space, usually referred to as space frames or space grids. The proposed method of approach is contained within the framework of physical dynamic simulation and comprises a generative algorithm using a particle-spring system.

1.1 Physical Dynamic Simulation

Computational design models used in the field of architecture have more often than not corresponded to parametric systems. As far as form representation is concerned, CAD modelling software have made possible the simulation of form based on geometrical properties, operating under a parametric scheme derived from Object-Orientated-Programming (Kanellos, 2004). Recently, new types of design software have allowed the modelling of associativity between objects, making possible the specification of hierarchies and interdependencies between them, apart from their geometrical attributes. The parameter space of design objects can thus be considered to have been expanded from one encompassing only geometric features to a space of topological relationships facilitates the creation of more complex forms, while being disengaged from suitable. Nevertheless, associative geometry can't account by itself for the physical constitution of the produced forms. In fact, exactly because associativity physically-based generative rules, it often leads to the requirement of considerable post-processing and rationalisation in terms of constructability.

According to Manuel DeLanda (2002), algorithmic design can benefit from incorporating three ways of thinking that derive from Deleuze's philosophy: population, topological and intensive thinking. Considering a computational parametric design model, one may reflect on how it can encompass these three components. Population thinking may relate to parameters of a geometric nature that result in the specification of a population of formal instances of a design object. Topological thinking can be manifested by associative parameters that establish a topological interrelationship layout between design objects. These are the two ways of thinking that computational approaches to architectural design have mostly been concerned about. Both refer to a static snapshot of a system's properties. Intensive thinking however, is of a different nature and involves physical quantities that are indivisible. Intensive quantities feature another important characteristic*: "a difference in intensity spontaneously tends to cancel itself out and in the process, it drives fluxes of matter and energy…differences of intensity are productive differences since they drive processes in which the diversity of actual forms is produced"* (DeLanda, 2002). It could be argued that computer-based physical dynamic simulation is

capable of introducing intensive thinking into algorithmic design. By accounting for physical properties and embedding them into a parametric model, intensive quantities can be modelled as the forces that set a design object into a state of equilibrium in itself or with its environment.

Modern computer processing capacity has made possible the real-time simulation of physics of considerably complex environments. This feature can be exploited for the modelling of physical properties of design objects. The parameter space of objects in parametric design can be further expanded so as to incorporate parameters that control their "behaviour" under simulated forces, apart from describing their geometrical and topological features. The term "behaviour" is of significance, as it partly bridges the gap between representational space and the physical realm. Computer-based generative design methods based on physical dynamic simulation can in many ways substitute a physics experimentation table, such as the one that Frei Otto (Otto and Rasch, 1995) used to perform the famous bubble experiments that eventually led to the conceptualisation of tensile structures. Allowing simulated design objects to become carriers of behavioural information could reinstate their relation to the physical environment, where they are destined to be constructed. It could also account for the establishment of D'Arcy Thompson's "diagram of forces" (Thompson, 1961) as an explicit design method bringing form into an equilibrium state with its context (Alexander, 1966). It is believed that an inadequately explored potential lies in physical dynamic simulation both in terms of solving existing problems but also as a creative instrument for researching formal and spatial properties that remain to be conceptualised.

1.2 Space Frames

Space frame structures are three-dimensional spatial networks that consist of two types of elements, namely the nodes and the edges or struts, which connect the nodes together. According to Chilton (2000), they were discovered by Alexander Graham Bell in 1903 for the purpose of kite construction, but were not used in architectural applications until the introduction of the MERO system in 1943. During the 1950's and 1960's they became more popular because of their attractive features of modularity, load sharing, robustness and ease of erection among others. At the same time, Richard Buckminster Fuller, following his study of the close packing of spheres, developed the octet truss system, which has been applied extensively ever since.

For the most part, space frames have been implemented in the form of the double-layer grid for the construction of long-spanning roof structures. The double layer-grid has been used with certain topological patterns of connectivity between nodes, such as the octet truss (also known as square on square offset), the square on diagonal square, the triangle on hexagon, etc. Since

their discovery, these engineered topologies have been used repeatedly in the same form. Permutations of these configurations have mostly been based on geometrical aspects like member lengths or construction details, so that demands such as easier erection, different spanning lengths and curved forms could become achievable. Apart from few cases like the speculative projects by J. François Gabriel (1997), where mega-structures and high-rise buildings have been proposed using octahedral-tetrahedral multi-layer lattices (Figure 01), configurations with multi-layer grids have seldom been used. It can be claimed that one possible reason for this is that the specific, uniform space frame topological layouts that have been discovered in the past have been unsuccessful in adapting to a wider spectrum of spatial necessities demanded by applications other than long-spanning structures.

In most applications of space frame technology, a finite palette of potential topological layouts has been made available to designers. Therefore, the overall design of any construct implementing such a technology has had to be adapted to fit the constraints posed by this narrow spectrum of possibilities. For instance, while the octet truss has repeatedly been applied to geodesic dome design (Figure 01), its pre-determined topology rigidly confines the possible domes that can be designed to a limited set.

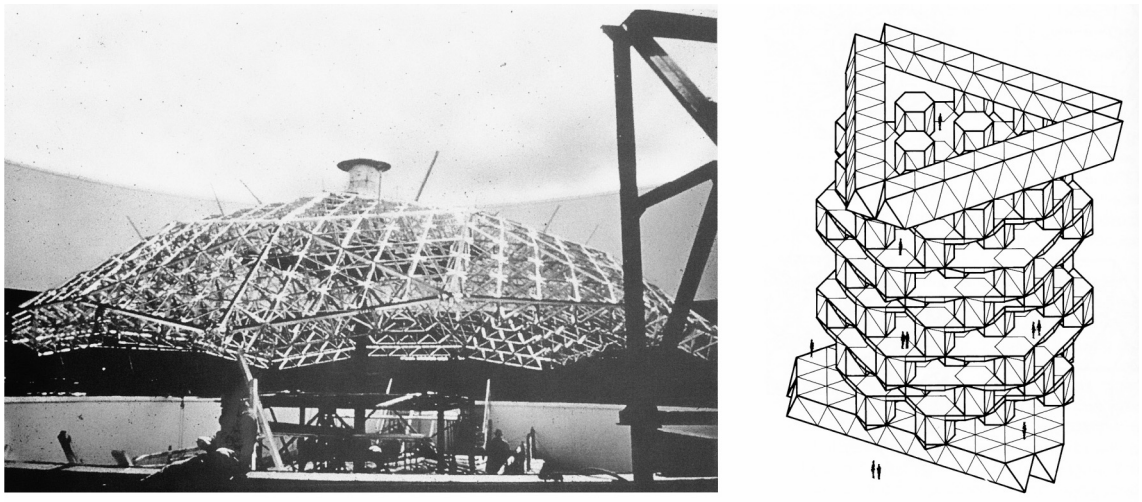


Figure 01. Space frames in architecture
Left: Geodesic Dome of the Ford Rotunda Building, Deaborn, Michigan, USA using the octet truss system developed by R. Buckminster Fuller (Source: Chilton, 2000, p.3).
Right: Speculative proposal by J. F. Gabriel for an eight-storey building using the multi-layered "hexmod" system (Source: Gabriel, 1997, p.470)

1.3 Problem Definitions and Thesis Aims

A question that naturally emerges is if the limitations posed by space frame topological preconceptions can be overcome in order to allow the adaptation of the structure to functional, formal or other spatial demands.

In this study, only the case of adapting space frame topologies to specific formal demands will be investigated, but the results may prove to be applicable to other design priorities. The study case is very much related to a problem that the sculptor Anthony Gormley faced when working on one of his projects. Gormley's *Body/Space/Frame* (Figure 02) is a project consisting of a 25 metre high open steel lattice in the shape of a human crouching figure (Hanna, <http://www.sean.hanna.net/bodyspaceframe.htm>). After the pattern of the shell had been decided upon, the question was how to fill the interior with a structurally stable space frame network.

Figure 02. Anthony Gormley's "Body/Space/Frame" (Source: Hanna, <http://www.sean.hanna.net/bodyspaceframe.htm>)

Given a certain volume, it is not always a straightforward task to construct a space frame network that uses a specific number of nodes for effectively approximating the form and filling the interior of the volume's spatial envelope. Attempting to arrange a given number of interconnected nodes in such an envelope, two aspects must be considered, namely the geometrical (the positioning of the nodes) and the topological (the connectivity pattern). No standard method appears to exist for dealing with such a problem in the relevant literature. Typical engineered space frames rely on grids for the arrangement of nodes in space and repetitive connectivity patterns for the structure's topological layout. These features constrain them to specific configurations and render them incapable of filling volumes that are not exact multiples of their grid size.

The aforementioned problem can be rephrased as the search of structural network lattices that are able to fill space and are subject to the following considerations:

(1) The space frame should consist of a fixed number of nodes that will be given. Existing engineered space frame topologies are constrained to numbers of nodes that are given by their respective grids. In this approach, it will be attempted to investigate topologies that can be generated with intermediate numbers of nodes.

(2) The nodes should be as evenly distributed in space as possible and the number of equal-length edges connecting the nodes should be maximised. In other words, more isotropic arrangements are preferred as this contributes to the facilitation of the manufacturing process and to a more uniform appearance.

(3) The space frame should have as few connections as possible and at the same time be structurally efficient, being able to withstand its self-weight under gravity. In other words, it should be as minimally rigid as possible.

(4) The internal angle between two edges springing from the same node must be sufficiently big. This also relates to an economy of connections used and to the more isotropic arrangement of connections in space.

(5) Any two edges must not cross each other too closely in space, not to mention intersecting each other. This relates both to formal aesthetic demands and to structural criteria, as closely placed intersecting connections that do not intersect at one of the nodes have a higher risk of breaking each other when the space frame sustains external forces.

(6) The topology should approximate the form of the given boundary envelope as best as possible.

The above problem statement contains multiple objectives that need to be addressed. Most of these considerations are consistent with Gormley's aesthetic viewpoint and represent properties that were sought in the development of the project *Body/Space/Frame*. Furthermore, they are general enough to be assumed as universally acceptable criteria, satisfied by efficient space-filling network structures.

Some of the aforementioned objectives are conflicting with others, which renders the solution landscape of the problem rather intricate. If the problem were to be approached through a top-down optimisation method, these would constitute the multiple objective functions that would determine the fitness of solutions and would thus have to be treated in an equal respect or be weighed according to importance. In order to map the solution landscape and arrive to optimised solutions from initial random ones, one would have to implement some form of a heuristic algorithm. Siavash Haroun Mahdavi and Sean Hanna have recently taken such an approach for a similar problem in *Microstructure Optimisation* (2003).

In this investigation, however, a different viewpoint is adopted. The algorithm proposed for dealing with the problem of structural space-filling uses a bottom-up numerical method based on the physical dynamic simulation of a particle-spring system. Instead of explicitly specifying objective functions within the definition of the algorithm, a self-organising system was established and was subjected to certain constraints. The system was found capable of generating the geometry and topology of space frame structures that exhibit a fair amount of advantageous properties.

The purpose of this study will be to examine the appropriateness of the proposed method by documenting and critically assessing its results and thus attempting to map a territory of the possible solution space. To this end, the problem considerations mentioned previously will serve as guidelines when evaluating and discussing the results. Specifically, (1), (2) and (3) will be used for measuring the proposed algorithm's performance, while (4), (5) and (6) will be purposefully handled by the algorithm through the specific rules specified in the definition of the bottom-up system.

1.4 Structure of the thesis

In section 2, completed work that relates to the stated problem and the proposed approach will be referred to. Following that, in section 3, the implemented method will be presented, the algorithm used will be explained and the results of the preliminary testing that allowed a more precise specification of the algorithm will be reported. Section 4 will involve the description of the formal experiment that was set up to examine the performance of the algorithm and the exhibition of its results. In section 5, some conclusions will be drawn based on the findings and concerning the potential of the method. The overall approach will be assessed and some possible directions for its further development will also be mentioned. The final section will present an overall review of this investigation.

2.0 Review of related work

2.1. Dynamic Relaxation and Tensegrity Structures

Dynamic relaxation is a technique that has been used in structural engineering applications, especially for form-finding membrane and cable net structures (Tibert and Pellegrino, 2003). A notable example is the dynamic relaxation algorithm developed by Chris Williams for form-finding the Great Court Roof of the British Museum (Williams, 2001). Dynamic relaxation is used on network structures with predefined topologies comprising interconnected linear elements for optimising the geometry of the connected nodes positions. In brief, it involves applying external forces to the system, such as gravity, and performing a relaxation process that entails iterative fine adjustments to the positions of the nodes until the total potential energy of the connections in the system is minimised.

The term tensegrity was coined by Richard Buckminster Fuller and is an abbreviation of tensional integrity (Ariel Hanaor in Gabriel, 1997, p. 385). Tensegrity structures consist of both tensional cable elements and compressive strut elements and rely on both for stability. They were discovered independently by Georges Emmerich in France and by the sculptor Kenneth Snelson while being a student of R. Buckminster Fuller (Jáuregui, 2004). Emmerich, Snelson and Fuller have all filed patents for tensegrity structures. Since then, there have been several attempts to provide precise definitions of tesegrity. One of the most recent definitions has been proposed by René Motro (2003) that takes into account the previous ones. Motro draws a distinction between the patent-based definition of Emmerich, Snelson and Fuller and an extended definition which is as follows: *"A tensegrity system is a system in a stable self-equilibrated state comprising a discontinuous set of compressed components inside a continuum of tensioned components"*.

Tensegrity structures have a short history and have not been implemented in many actual engineering applications since they still present some unresolved problems that need to be further researched. One of these is the very form-finding of the structures, which is complicated partly due to the nature of the geometry and the need for a precise specification of the pre-stress in the tensile members. Several methods have been used for form-finding (Tibert and Pellegrino, 2003), one of them being the dynamic relaxation method. However, almost all attempts for form-finding tensegrities take initially for granted a topology of connections between nodes and the tensile/compressive nature of the connections. Thus, this has allowed for almost only regular topologies to be researched, which have already been documented and classified thoroughly by Connelly and Black (1998) through the use of Group Theory. Zhang et

al (2006) have made progress in form-finding nonregular tensegrity structures through dynamic relaxation (Figure 03), but the initial specification of the topology is still required in the approach. An attempt for generating irregular tensegrities has also been undertaken by Paul et al (2005), in which a genetic algorithm is used for finding tensegrity topologies that are subjected to dynamic relaxation and then have their fitness evaluated according to an objective function of maximum occupied volume in order for the genetic algorithm to produce new generations of optimised solution populations (Figure 03).

Tensegrity structures are different from typical space frame structures mostly because of their nature of having only discontinuous compressive members. However, their particular nature was of relevance in this investigation, as their characteristically clear distinction between tensile and compressive members provided a conceptual background for the proposed generative procedure of lattice topologies.

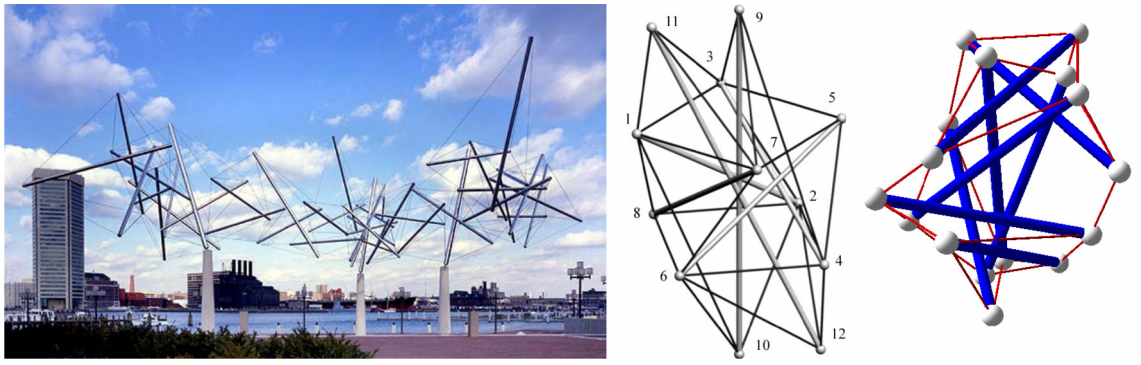


Figure 03. Tensegrity
Left: Kenneth Snelson's "Easy Landing", 1977 in Baltimore, USA. (Source: <www.kennethsnelson.net>)
Middle: An irregular topology that has been found with the method of dynamic relaxation by a group of researchers at the Laboratoire de Mécanique et Génie Civil, Univ. Montpellier led by René Motro (Source: Zhang et al, 2006)
Right: An irregular topology that has been found with the method of dynamic relaxation and a genetic algorithm by a group of researchers at the Mechanical and Aerospace Engineering Department of Cornell University (Source: Paul et al, 2005)

2.2. Particle-Spring Systems

*"The term **particle system** refers to a* computer graphics *technique for simulating certain fuzzy phenomena, which are otherwise very hard to reproduce with conventional rendering techniques. Examples of such phenomena which are commonly done with particle systems include fire, explosions, smoke, flowing water, sparks, falling leaves, clouds, fog, snow, dust, meteor tails, or abstract visual effects like glowing trails, etc."*
(<http://en.wikipedia.org/wiki/Particle_system>)

*"A **particle system** is a collection of point masses in 3D space possibly connected together by springs and acted on by external forces"*
(Bourke, <http://ozviz.wasp.uwa.edu.au/~pbourke/modelling_rendering/particle/>)

Particle systems have been used among other things for computer graphics and animation as well for simulating complex physical phenomena such as the behaviour of gases or of cloth (Baraff and Witkin, 1998). Even though particle-spring systems were not originally conceived for use in structural applications, they are very similar in principle with the dynamic relaxation method.

Recently, a particle-spring system approach has been used in the field of architectural design. Axel Kilian and Ochsendorf (2005) have developed "*CADenary*", an interactive tool for structural form-finding that can be used for simulating and designing catenary structures (Figure 04). The tool allows the user to design a topology consisting of particle nodes and spring connections. The nodes can be anchored in 3d space and be subjected to a gravitational force. When under gravity, the unconstrained particles fall under their self-weight causing deformations to the elastic springs that can only sustain tensional forces. Eventually, the system reaches an equilibrium state, where particles are held in place by the deformed springs after having sustained the total amount of applied stress. The 3d model of the system can then be reversed along the horizontal plane. Similar to Gaudi's hanging chain models, the generated tensile springs are turned into compression elements and the resulting inverse topology constitutes a stable structure that optimally handles the distribution of stresses induced by gravity.

Another approach for structural form-finding using a particle-spring system was recently taken by Przemyslaw L. Jaworski in "Using simulations and artificial life algorithms to grow elements of construction" (2006). In this case, an algorithm produces a support space frame structure for a certain volume by introducing particles iteratively (Figure 04). Each introduced particle is connected by springs to three others, thus forming tetrahedral arrangements. The particles grow from certain seeds and are directed into place by following trails of simulated agents. By interacting through forces of attraction and repulsion to each other, the particles maintain equal distances between them and settle into foam-like topologies that constitute a support structure able to withstand the weight of the volume placed upon them.

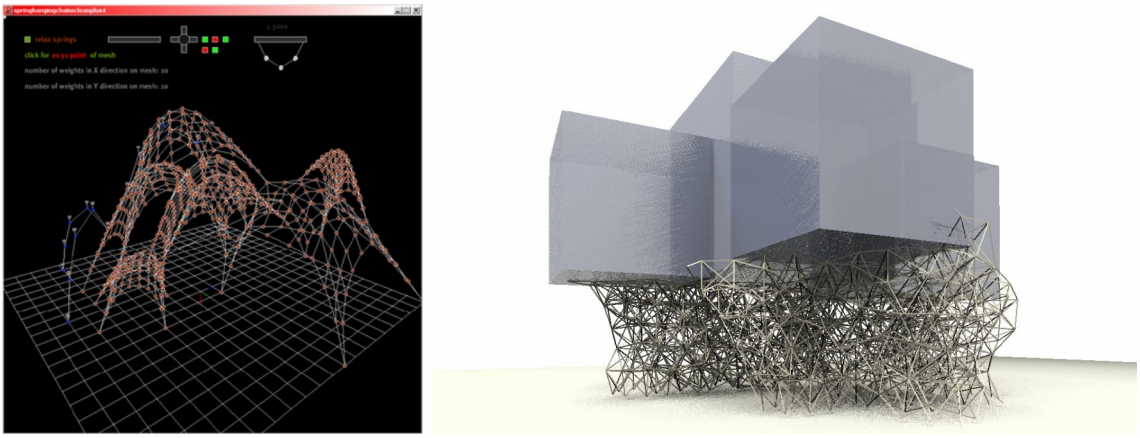


Figure 04. Particle-System approaches in architectural design
Left: Screenshot from the program "CADenary", developed by Axel Kilian and a group of researchers at the Department of Architecture, Massachusetts Institute of Technology. (Source: < http://destech.mit.edu/akilian/projectpages/cadenary.html>)
Right: Support structure generated by a particle-spring system algorithm, developed by Przemyslaw L. Jaworski for the MSc thesis "Using simulations and artificial life algorithms to grow elements of construction" at the Bartlett School of Graduate Studies, UCL (Source: Jaworski, 2006)

2.3. Close packing of spheres

"Close-packing of spheres is the arranging of an infinite lattice of spheres so that they take up the greatest possible fraction of an infinite 3-dimensional space".
(<http://en.wikipedia.org/wiki/Close-packing>).

Assuming that the centres of close-packed spheres are connected with edges to the centres of other spheres they are in contact with, a lattice structure can be formed that consists of members of equal length. The most common arrangements for sphere packing are the face-centred cubic (FCC) and the hexagonal close packing (HCP) (Figure 05). When connected, the centres of spheres in FCC packing produce a topology that has been used extensively in double-layer space frame structures for architectural and engineering applications. This is also known as the octet truss system or by the term Isotropic Vector Matrix (IVM) that was coined by Richard Buckminster Fuller (Urner , <http://www.grunch.net/synergetics/ivm.html>)

The average density of both the FCC and HCP packings when infinitely expanded in Euclidean space is equal to $P=\pi/3\sqrt{2}\approx0.74048$. In both arrangements each sphere is in contact with exactly 12 other spheres (sphere valence). Another common arrangement is the orthogonal simple cubic packing (SCP) (Figure 05), where the average density of infinitely stacked spheres is $P\approx0.524$ and average node valence is 6. Demonstrations with ball bearings in a box and computer

simulations have shown that when spheres are packed randomly, the packing density is around P≈0.64 (Beals et al, <http://www.tiem.utk.edu/~gross/bioed/webmodules/spherepacking.htm>).

"The Kepler conjecture formulated in 1611 states that the density of face-centre cubic or hexagonal close packing P=0.74048 is the maximum possible density for both regular and irregular arrangements."
(<http://en.wikipedia.org/wiki/Kepler_conjecture>).

The octet truss/FCC/HCP topology might have the optimal packing density, but this is so only when considering infinitely packed spheres. If the spheres were to be contained within a specific envelope, finding the optimal packing with the highest density might lead to other topologies. A simple example for this is the existence of a threshold before which the SCP topology is better in filling a cubic volume, while past this threshold the FCC/HCP topologies become more efficient than the SCP. More details for this are presented in Appendix I, §1.

The issue of close packing of spheres inside a cube is a standing mathematical problem. The problem consists of finding the positions of *n* congruent hard spheres placed inside a cubic container, at which the length of the minimum distance between them is maximised. This problem has been approached with numerical methods and computer dynamic simulation, where spheres are simulated as non-overlapping colliding entities, known as "billiards systems" (Lubachevsky, 1991). In such systems the algorithm initiates randomly placed non-jammed spheres in the container and gradually increases their radii until they settle into a jammed packing (Figure 05). Optimal packings have been reported using such approaches for different numbers of spheres in a cubic container (Gensane, 2004), but not all have been proven to be globally optimal.

The lattice topologies derived from the close packing of spheres are isotropic, having nodes equally distributed and connected by members of equal lengths. However, the rigid constraint of complete isotropy comprising a single length between nodes might prove to be incapable of producing topologies that best describe the volume of the envelope. Introducing a certain amount of "fuzziness" (Kennedy and Eberhart, 2001, p.37) might be required as a compromise in order to fill space with a space frame topology more descriptively. In some ways, the proposed algorithm is similar with the billiards simulation, but a main difference is that the simulated balls are conceived as more "soft" and are allowed to partially penetrate each other's volume. Furthermore, optimal packings do not necessarily yield structurally stable topologies as often the number of connections and their topological layout do not suffice for the achievement of structural stability.

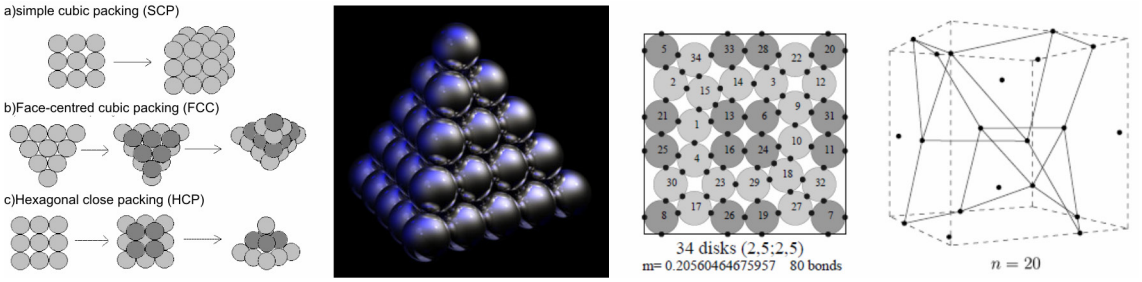


Figure 05. Sphere close packing

First: The three most common packings of spheres that correspond to existing engineered space frame topologies (Source: Beals et al, http://www.tiem.utk.edu/~gross/bioed/webmodules/spherepacking.htm)

Second: FCC packing of spheres (Source: (<http://en.wikipedia.org/wiki/Close-packing>).

Third: Optimal packing of disks in a square found by the "billiards simulation" algorithm, developed by R. L. Graham and B. D. Lubachevsky (Source: Graham and Lubachevsky, 1996)

Fourth: Optimal packing of spheres in a cube found by the "perturbed billiards simulation" algorithm, developed by T. Gensane (Source: Gensane, 2004)

2.4. 3d Mesh subdivision

In the field of computer graphics, the method of "*bubble-meshing*" (Shimada, 1995) has been used to subdivide three-dimensional solids for the purpose of producing suitable discretised models that can be analysed by the Finite Element Method. The bubble-mesh method uses particles that are simulated as connected bubbles interacting with each other to find an optimal placing inside the volume. In brief, the algorithm (1) takes an initial guess for node placement inside the volume using hierarchical spatial subdivision, (2) defines proximity-based repulsive/attractive forces and (3) performs dynamic simulation for a force-balancing configuration, while (4) adaptively controlling the bubble population. Because of the more relaxed constraint on the distances between bubbles, it is capable of fitting a mesh inside a volume that describes the volume more accurately. However, isotropy is also a desired property, so solutions are ranked according to member length deviation from a single ideal length and the bubble population control method adjusts the number of bubbles to yield fitter solutions.

The algorithm proposed in this investigation for the purpose of space filling shares several similar features with the bubble-mesh method. Differences in the proposed method include the use of a simpler physics simulation, the lack of pre-specified initial positioning of particles, the lack of a pre-specified connectivity pattern and the structural nature of the generated lattice topologies.

3.0 Method

3.1 Overview of algorithm

Taking inspiration from natural bottom-up systems such as boids and the behaviour of matter at a molecular level in metal die-casting (discussed in more detail in Appendix I, §2), the proposed method for dealing with the problem of space filling relies on a numerical method of a an algorithm that dynamically simulates a particle-spring system. The algorithm was developed using the Processing programming language (<http://processing.org>). Its main characteristic is the use of forces between finite point elements called particles that interact with each other to produce crystal-like lattices. No explicit description of the resulting lattice topology is given to the algorithm. Instead, simple, bottom-up, local rules of interparticle interaction are implemented and the particles are able to generate the forms through self-organisation. Unlike other approaches using a particle-spring system, the connectivity pattern between the particles is not pre-determined, but is dynamically established by using suitable particle proximity constraints. Through the subjection of the particles to proximity-based interparticle spring forces, they are able to optimise their relative and absolute position, forming temporary bonds between them. In an analogy to metal die-casting procedures, the space whose volume is to be filled operates very much like a mould. Its boundaries are impenetrable by the particles and define an outer shell, while its interior is explored by particles until they manage to settle in an equilibrium state, balancing the developed forces between them.

3.2 Description of the particle-spring system

A simple particle system is programmatically established. Particles are programmed as instances of a class of point elements which at any time hold two quantities, a position and a velocity vector. Each particle is assumed to have mass of zero, which makes it negligible for any practical purposes in calculations. The system is considerably simpler than what is considered to be a typical particle system, as for example, there is no use of accelerations, viscous drag or viscous damping.

The system is initiated with the particles at random positions inside the bounding envelope and with zero velocities at the first iteration. In each of the following iterations, however, the particle preserves a part of the velocity from the previous iteration (this is further discussed in Appendix I, §3). For this reason, a temporary position of all particles is calculated as the sum of its current position and velocity vectors and stored by the particle (Figure 06). This temporary

position is used in calculations at the following stage of inter-particle interaction, so that invalid movements can be adjusted before they occur.

The step that follows is the creation of a topology of connections between particles according to a pairwise proximity check. All possible pairs of particles are examined and the distance d_{ij} that separates the temporary positions of each pair is calculated. If d_{ij} is below a certain threshold D a spring connection is established between them (Figure 06). After all appropriate springs have been created, the algorithm iteratively calculates the force that each spring will exert to the two particles it is connected to. At any time, all springs have the same ideal stable length l_0, which is set to be smaller than D. If the distance d_{ij} between the particles is less than l_0, ($d_{ij} < l_0$) then the spring is in compression and thus exerts a force that takes the two connected particles away from each other. In this case, the particles sustain a mutual repulsion. If the distance d_{ij} is more than l_0 ($l_0 < d_{ij} < D$), then the spring is in tension and thus exerts a force that brings the two connected particles closer to each other. In this case, the particles sustain a mutual attraction. If the distance d_{ij} is equal to l_0 ($l_0 = d_{ij}$), then the spring has an ideal length and is in equilibrium. In this case, no force is exerted to the particles by the spring (Figure 07). More details about the nature and the calculation of the force as well as the stable distance l_0 and its proportional relation to D are presented in Appendix I, §4.

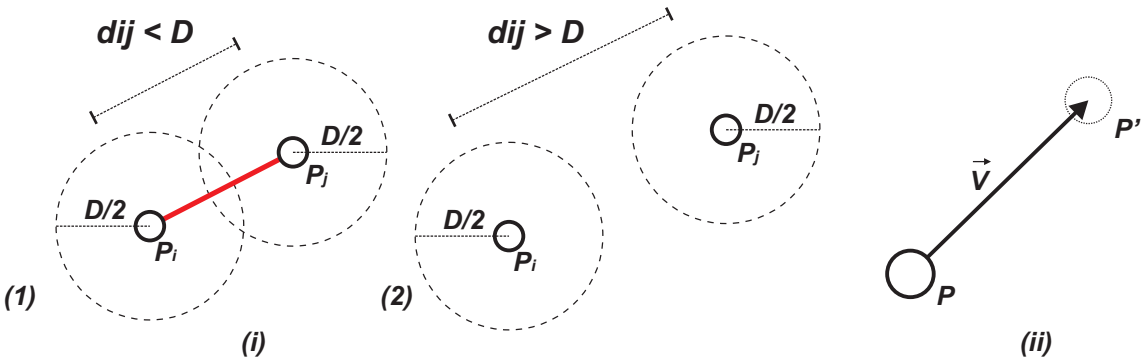


Figure 06. Inter-particle spring establishment and Temporary Position Calculation
(i) A spring is established between two particles i,j when their distance is less than the threshold **D.**
(ii) The new temporary position is calculated as the sum of the position and velocity vectors.

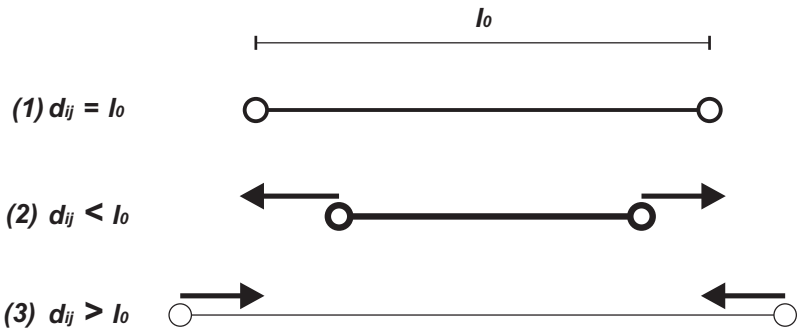


Figure 07. The three cases of the spring force.
(1): The spring is at the stable length l_0 and does not exert any force to the particles
(2): The spring is in compression and exerts a repulsive force to the particles
(3): The spring is in tension and exerts an attractive force to the particles

The total force that is to be sustained by each particle from its connected springs is summed and accumulated as a temporary force vector stored locally by each particle. Only after the algorithm has applied all spring forces, is the temporary force vector added to the velocity vector of the particle. This is done so as to avoid errors that would emerge if the calculation of the spring forces took into account distances of corrupted particle positions resulting from sustained forces from other springs during the iterative process of force calculation.

In the following step of the same iteration, the algorithm calculates a new temporary position for each particle by adding the updated velocity to the original position. This new temporary position is used for collision detection between the particles and the boundaries. The boundary collision detection is further explained in Appendix I, §5. A response vector is calculated and added to the velocity vector of particles that are found to be in a collision state with the boundaries of the envelope.

Finally, the resulting velocity vector from the above boundary collision calculations is added to the original position of the particle and the particles are drawn at the new positions along with the pairwise spring links that have been established.

In each of the following iterations, the above process is repeated, but the topology of spring connections that has been established in the previous iteration is discarded and a new one is created according to the updated positions and velocities of the particles.

Given enough iterations the system converges to an equilibrium state, where all velocities are at a number close to zero ($|\mathbf{v}|<0.01$). According to the number of particles, the ideal spring length and the boundary given, the system will result into a settled space frame topology.

3.3 Preliminary testing

The above description covers all the basic rules used for attributing the overall behaviour to the system. Such an approximate overall behaviour was sufficient to allow the system to generate network topologies of particles and spring connections. These topologies persistently exhibited certain desired properties that could be easily observed, such as an even distribution of particles in space and sufficiently big angles at nodes between connections.

The initial testing of the algorithm was performed by manually increasing and decreasing the spring ideal length $\boldsymbol{l_0}$, and thus the spring creation threshold **D**, it being a function of $\boldsymbol{l_0}$. Through the observation of the system's behaviour and its response to the adjustments of $\boldsymbol{l_0}$, a better understanding of its performance was achieved. According to these observations certain assumptions were made that led to the more precise specification the spring force used, described in Appendix I, §4 and some of the other steps of the algorithm described in the following section.

Throughout this preliminary testing several boundary envelopes were used ranging from primitive shapes such as a cube, a tetrahedron and a sphere (Figure 08) to arbitrary concave meshes (Figure 09). The algorithm was found to be robust in terms of the diversity of spaces it can fill, as it could handle the volume of any closed, ordered, triangle-faced mesh provided.

An interesting observation in the case of the tetrahedron was that the system converged to a lattice consisting mostly of tetrahedral arrangements. When the number of points used were not exactly the one required to fill the tetrahedron by only tetrahedral configurations, the algorithm seemed to generate "fuzzy" tetrahedral configurations that included degenerate parts. In the case of the sphere, if enough particles were used in the generative process, the system would arrange the particles in concentric sphere configurations. In the case of the cube, depending on the node number used, the system would converge to orthogonal or octahedral topologies.

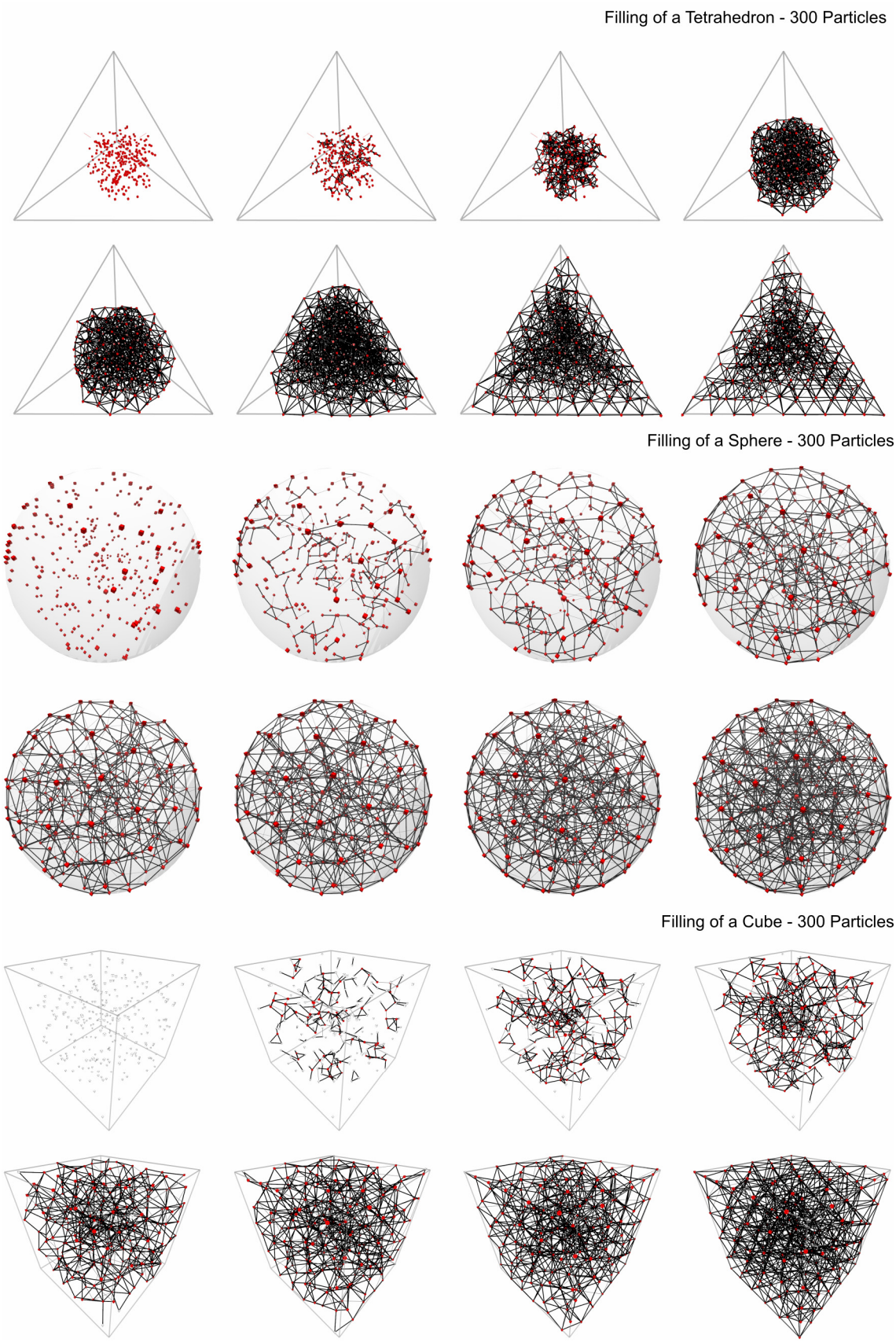


Figure 08. Process of the gradual increase of the spring ideal length and its effect on the particle-spring system in the preliminary manual testing. The iterative generation of a topology that fills the volume of a tetrahedron, a sphere and a cube is demonstrated.

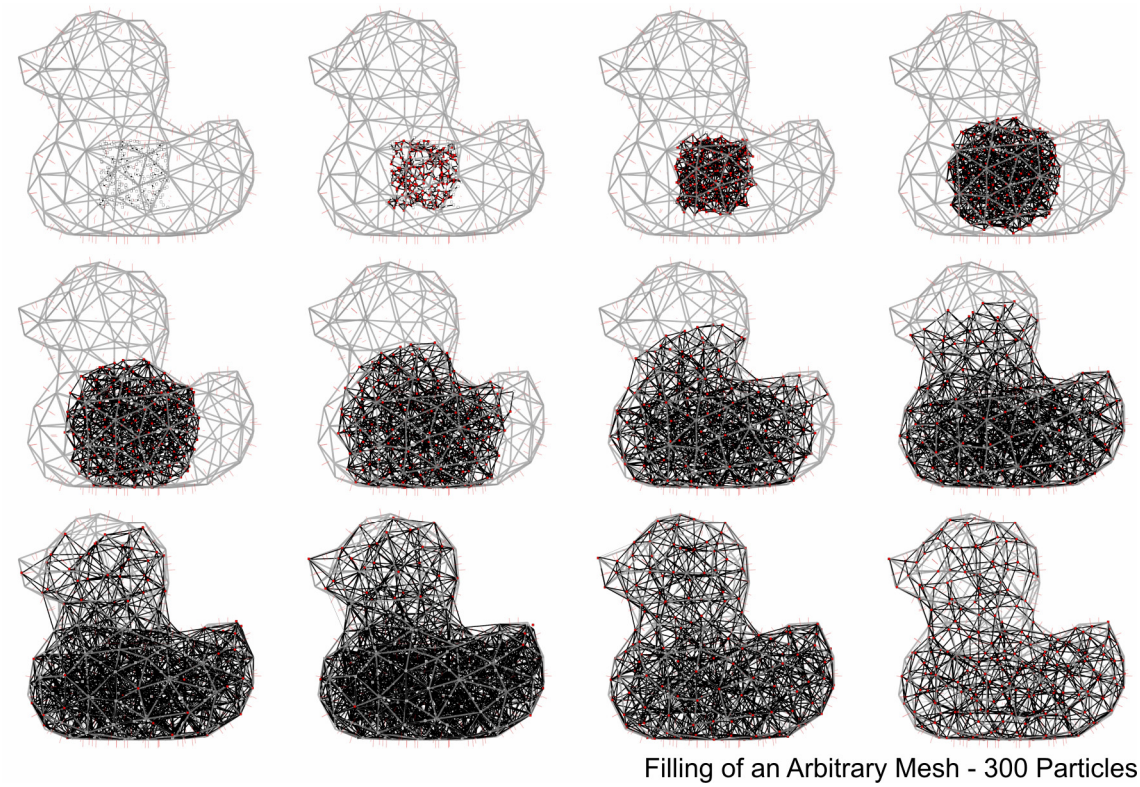


Figure 09. Iterative generation of a topology that fills the volume of an arbitrary mesh.

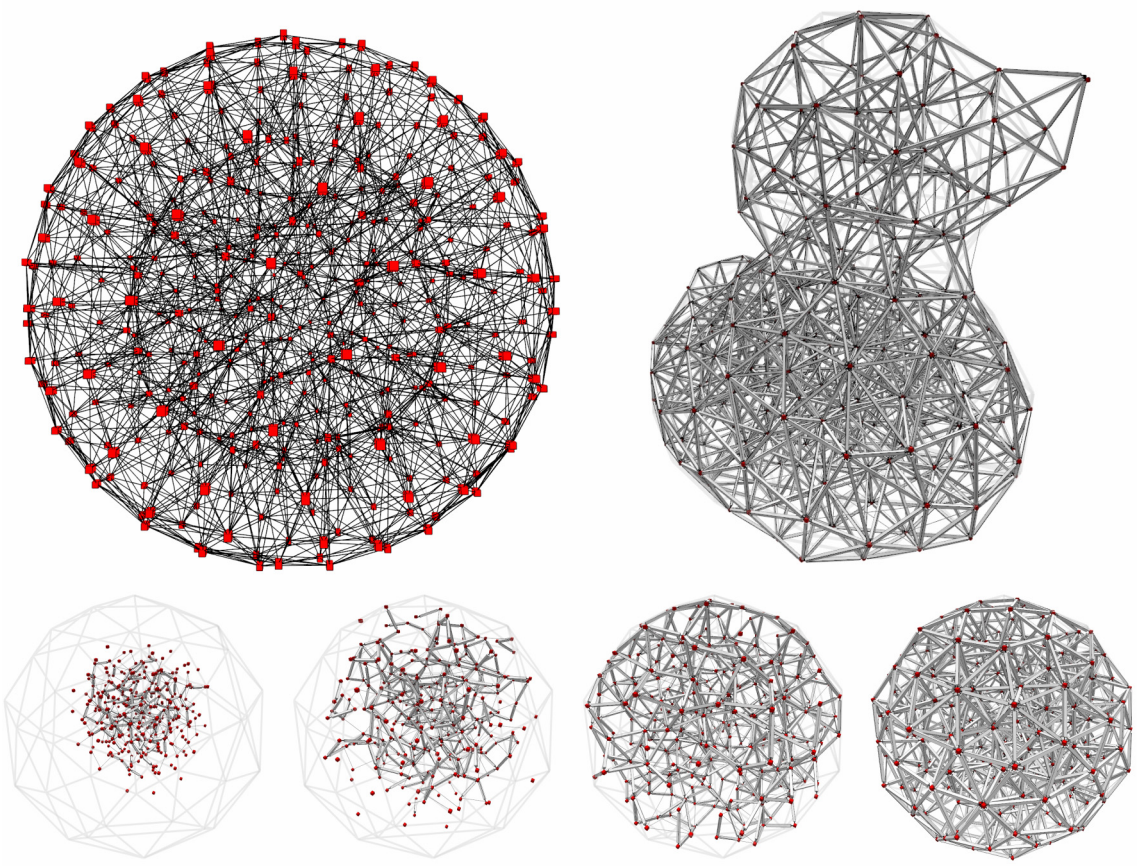


Figure 10.

Top: Generated Topologies that fill the envelopes of a sphere and an arbitrary mesh

Bottom: Iterative generation of a topology that fills the volume of a geodesic sphere.

3.4 Specifying the physical parameters and the algorithm process for the formal experiments

In order to examine the behaviour and test the performance of the algorithm more thoroughly, the study case of filling the volume of a simple cube was selected. This choice was made on the basis that the cube constitutes a primitive abstract geometric entity defined by axes in Cartesian coordinates. Space frame topologies that are used in architectural applications often have to adapt to constraints of Cartesian orthogonal coordinates, the octet truss being such a case. Furthermore, the resulting structures could be more easily examined and evaluated since they can be easily projected onto three planes. Finally, there is a formulated body of reference regarding the close packing of spheres within a cube that could serve as a basis for comparing and evaluating the results.

Even though informal observations from the preliminary testing showed that the algorithm was capable of generating space frame structures with a fair amount of good properties, some choices were made so that these would be more purposefully induced. Thus, the algorithm had to be tuned to produce as optimised topologies as possible according to the specificities of the problem of this investigation. Furthermore, the generative process would have to be automated and not rely on manual adjustments, but on explicit rules. This was not a straightforward task, as it involved a top-down approach of finding a good combination of physical parameters and a more exact specification of algorithm steps. This process was very much like adjusting the material properties of a physical system and was based on assumptions resulting from the feedback from the preliminary manual testing. Its analogies to other bottom-up systems is discussed in Appendix I, §2.

The observations from the preliminary testing led to an automation of the increase/decrease process of the spring ideal length l_0. This was set to be performed by the algorithm in order to maintain the valence of all nodes within the range of 3 and 12. The valence upper limit of 12 was chosen as it is the maximum that can be achieved when close packing spheres in Euclidean space, and was thus considered to be sufficient for any resulting topology that has as equal member lengths as possible. This tactic seemed to cause the system to converge to topologies that in most cases were sufficiently and not excessively connected and had expanded enough to reach the boundaries of the cubic envelope. More details about this can be found in Appendix I, §6.

In order to further increase the chances of optimisation, a certain form of heuristics was used that was intrinsic to the nature of the system and inspired from the metal die-casting under

pressure. The hypothesis was made that an application of compressions to the system would optimise the topologies further, as it would cause the particles to fill any existing gaps and also induce a self-organisation of the system rendering it more able to resist vertical forces. The resistance to a vertical force was considered to be a desired property as the resulting topology should be eventually able to withstand its weight under gravity when tested structurally. The application of a gravitational force instead of compressions during the generative process was also considered and experimented with, but was abandoned as it had a negative effect on the system's stability. The algorithm was thus set to make three total compression applications when the system reached an equilibrium state. More details about this can be found in Appendix I, §7)

4.0 Testing and Results

In order to thoroughly examine the performance of the algorithm three main aspects of the generated cube-filling lattices had to be considered, namely the geometrical, the topological and the structural. In order to gain some conclusive insight into what the features of the lattices are, a fairly large number of test samples was required and thus the results that emerged had to be treated statistically.

The experiment that produced the test samples involved using 82 different cases of node numbers ranging from 4 to 240 for generating lattices that fill the volume of a cube with a side of 1,000 units (Figure 11). The samples were taken at an increment of one node for the range of low node numbers (less than 30) and with an increasing increment after that. For each number of nodes the algorithm was executed five times and at each time with a different random initial condition of node positions. A more complete list of illustrations of the generated lattices can be found in Appendix II. In each run, a log file was being recorded at a fixed time interval that kept track of several measurements. For most measurements the state of the system before the first compression and the one after the last compression were used. The results from the five runs for each number of nodes were averaged out to create a mean solution that corresponds to a single member of the total population of generated solutions.

Apart from the lattice topologies generated by the algorithm, three other idealised engineered topologies were purely geometrically produced and tested, so that they could provide a measure of comparison. These were 1) the face-centre cubic space frame topology (FCC) (also known as the octet truss and the square on square offset), 2) the hexagonal close packing topology (HCP) and 3) the orthogonal simple close packing topology (SCP). The samples from these topologies were chosen on the basis that they could be contained within the same cubic envelope as the generated topologies (Figure 12).

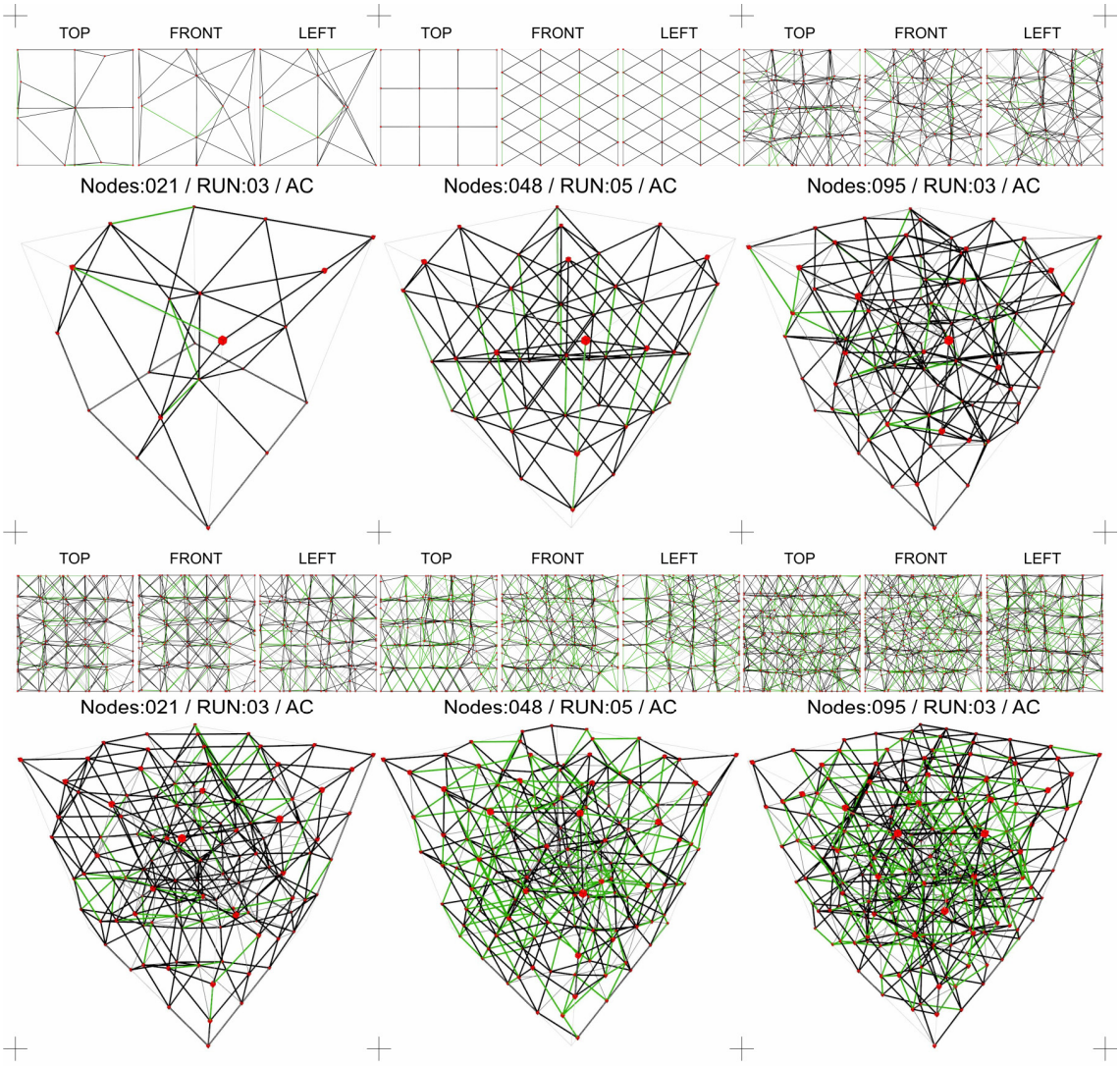


Figure 11. Indicative generated samples. AC denotes "After compression state". Green edges denote springs that have very small deviation from mean length.

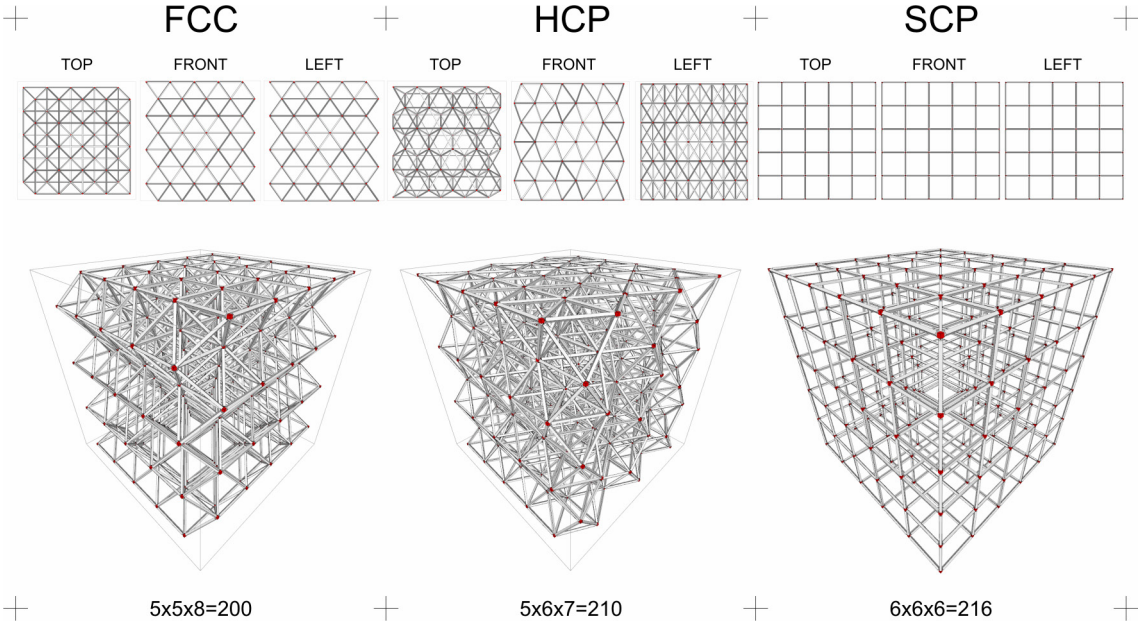


Figure 12. Samples of the three engineered topologies that were used for comparing the results against

4.1 Geometrical Analysis

The performance of the algorithm in terms of the connection lengths and their deviation from a common ideal length was measured by calculating the average deviation of all springs from the common mean length. In each algorithm run, the mean spring deviation starts from zero, as no connection is present and as the spring ideal distance increases, the deviation increases as is seen in Figure 13. This continues up to a point where the valence limits are reached and the system is allowed to relax. The following effect of the compression process is clearly marked by the three spikes that the deviation forms in the relevant diagrams. Between compressions, local minima of the deviation are formed, as the system is allowed to relax after each compression and before the next. No distinguishable pattern seems to emerge from observing the relation of minima between compressions. The time at which the compressions occur differ for each run and for each number of points. In overall, the runtime increases with bigger numbers of particles and that for smaller numbers of particles there is more consistency at the times of the compression for the different runs of the same member of the population (Figure 14).

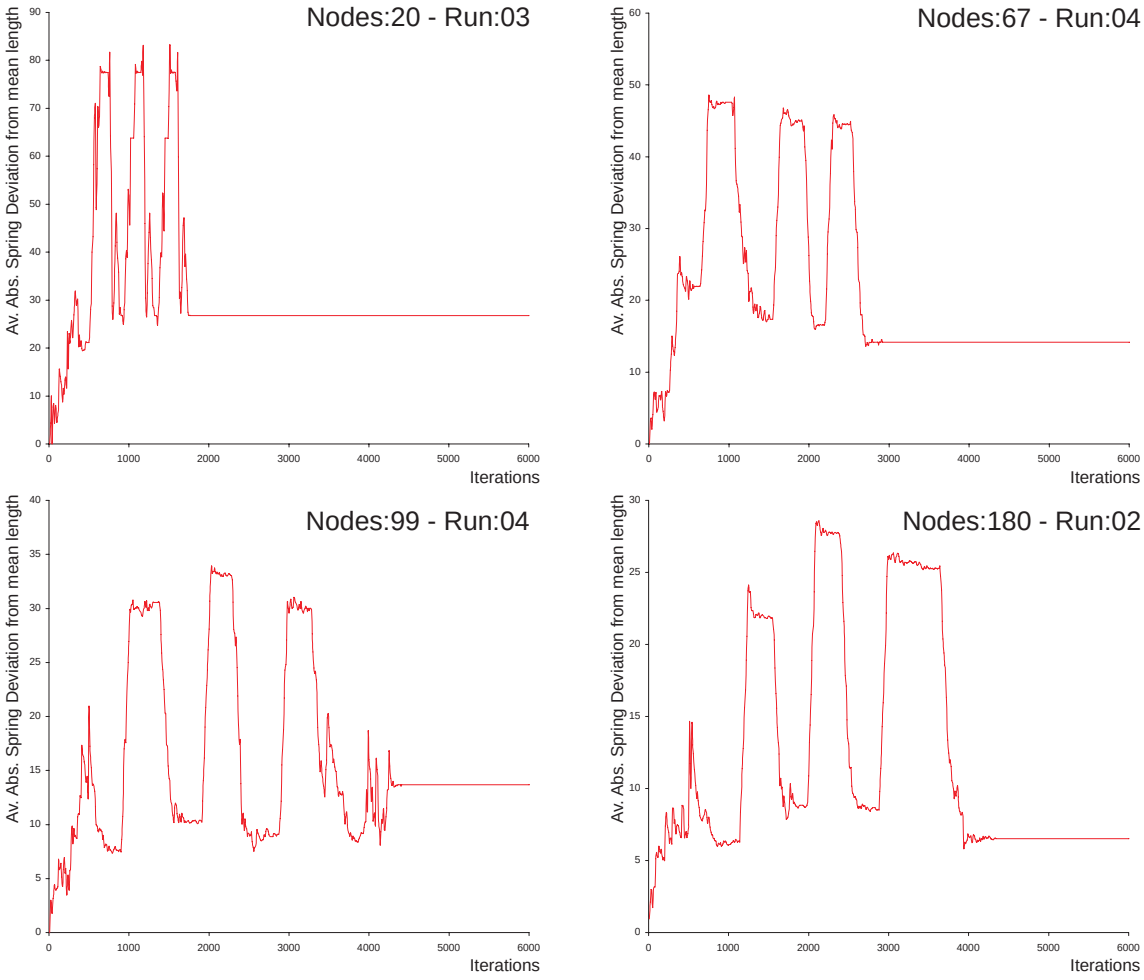


Figure 13. Absolute average spring length deviation from mean length during runtime of the algorithm for single runs of members with 20,67,99 and 180 nodes.

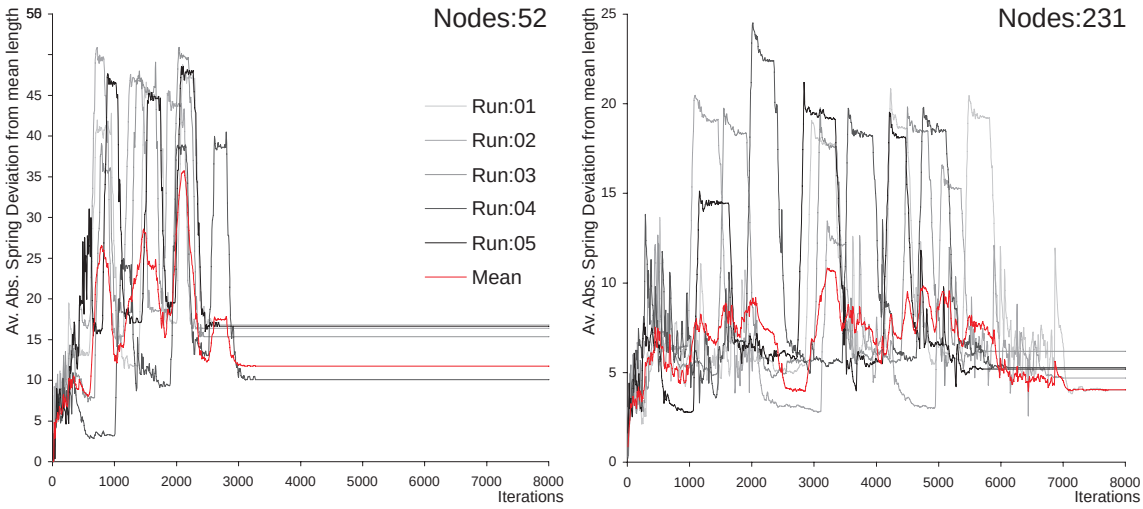


Figure 14. Absolute average spring length deviation from mean length during runtime of the algorithm from five runs. The mean deviation of the two generated population members from the five individual runs is also shown.

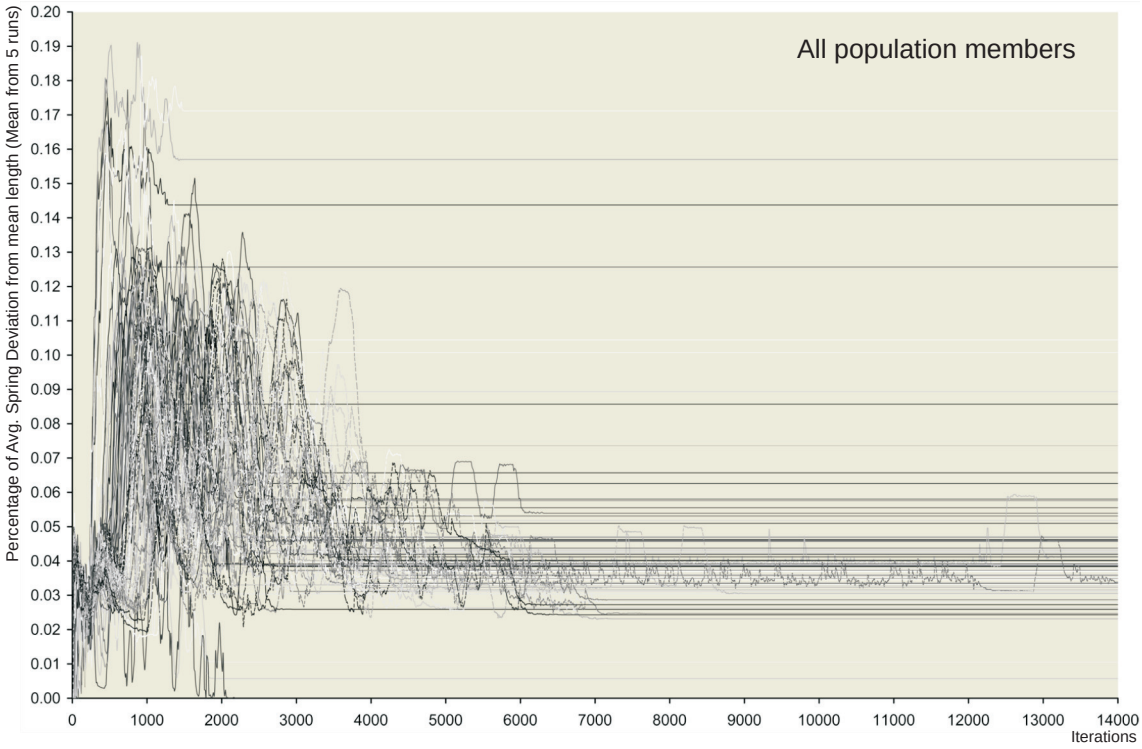


Figure 15. Percentage of the mean absolute average spring length deviation from mean length during runtime for all population members.

As far as the number of connections is concerned, they seem to increase linearly with the increasing number of nodes used, when the results from the five individual runs are averaged out (Figure 16). The same occurs for the engineered topologies, with FCC and HCP highly coinciding and SCP having considerably fewer nodes. What is of interest is that the plotted trendline of the connections over nodes for the generated topologies seems to be contained between the FCC/HCP and SCP respective trendlines. This may indicate that, in overall, the

generated topologies might possess features of the engineered. The Laman condition of Rigidity Theory states that a minimally rigid structure should have edges equal to E=3*N-6 (Graver, 2001). This is a required, but not sufficient condition for a structure to be rigid, as a suitable topological layout should also be present. As the graph in Figure 16 shows, the generated topologies have, in all cases, more connections than those specified by the Laman condition.

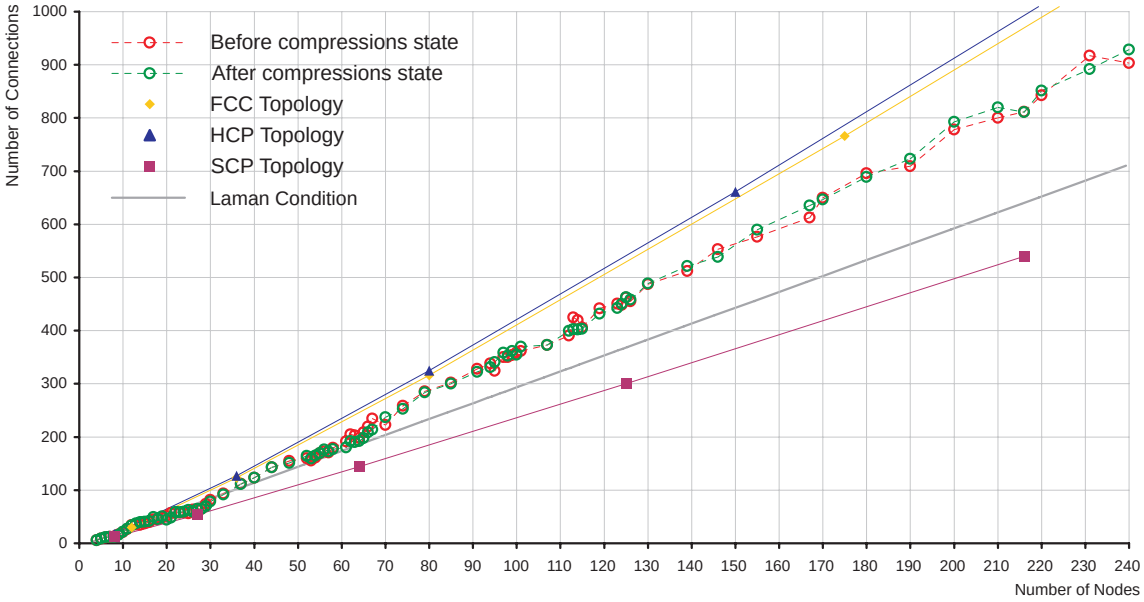


Figure 16. Number of connections of all population members.

Considering the absolute mean connection length of the generated topologies averaged from the five individual runs, it can be seen from the plot in figure 17 that it decreases for higher numbers of nodes, as would be expected. Apparently, it is higher in overall than the connection lengths of the engineered topologies, but for smaller node numbers it coincides with the SCP topology. In addition, it appears that the states after the compression process have achieved marginally greater mean lengths than the ones of the states before the compression process. In this plot the threshold past which the FCC and HCP topologies become more efficient in filling the same cube is also visible. Since all samples of the generated topologies have less nodes than those required for the FCC/HCP topologies to become more efficient, this may serve as a possible explanation as to why the generated topologies coincide more with the SCP.

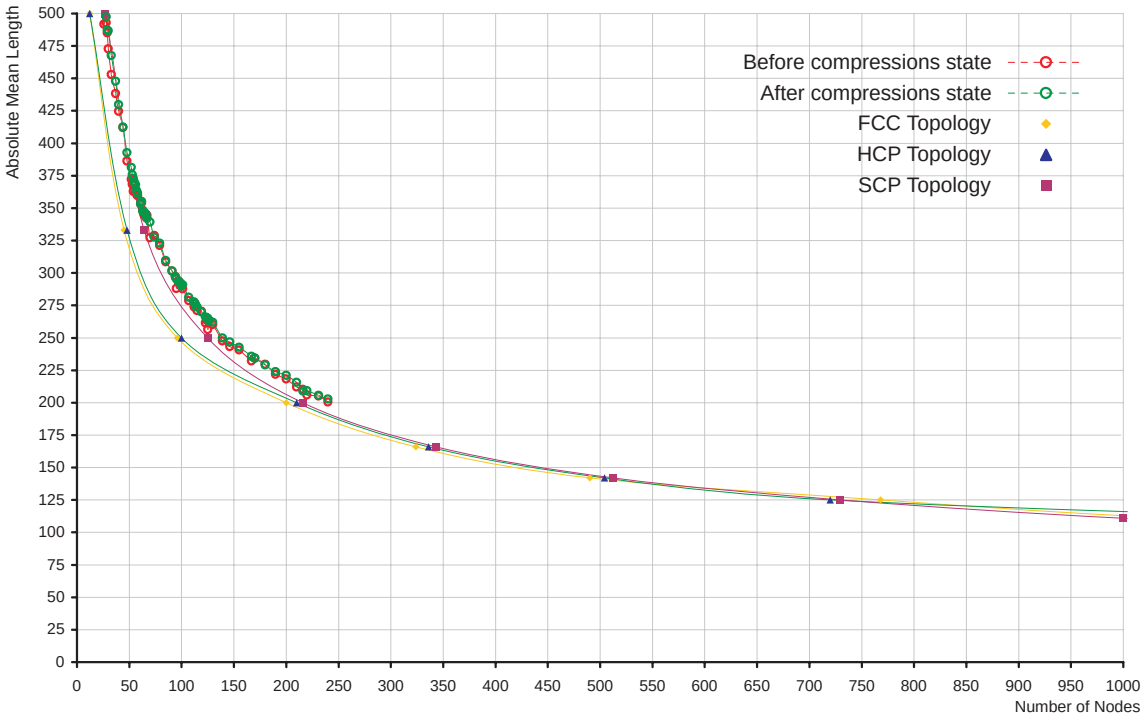


Figure 17. Absolute mean length of all population members & Sphere packing threshold of engineered topologies.

In the case of percentage of the average deviation of member lengths from the mean length regarding the individual solutions of all runs for all numbers of nodes, it can be seen in Figures 18 and 19 that the spectrum of deviation seems to decrease for higher numbers of nodes, which suggests that the algorithm produces results more consistently for higher node numbers. In the relevant plot of the average of the five runs in Figure 20, it is clear that there was an overall increase in the deviation of springs from the mean length after the compression process. The deviation, however, appears to be decreasing in total when the number of nodes used increases. The local deviation minima found in the same graph appear to be close to the area of points where the engineered topologies test samples were taken from. It also seems that the lowest minima are found closer to the area of the samples of the SCP topology, a fact that may serve as an indication of the algorithm being more able to converge to that topology. This is even more apparent in the before compressions state. Given that the SCP topology within this sample range is better at filling a cubic space, this fact may be considered as a good feature of the algorithm's performance.

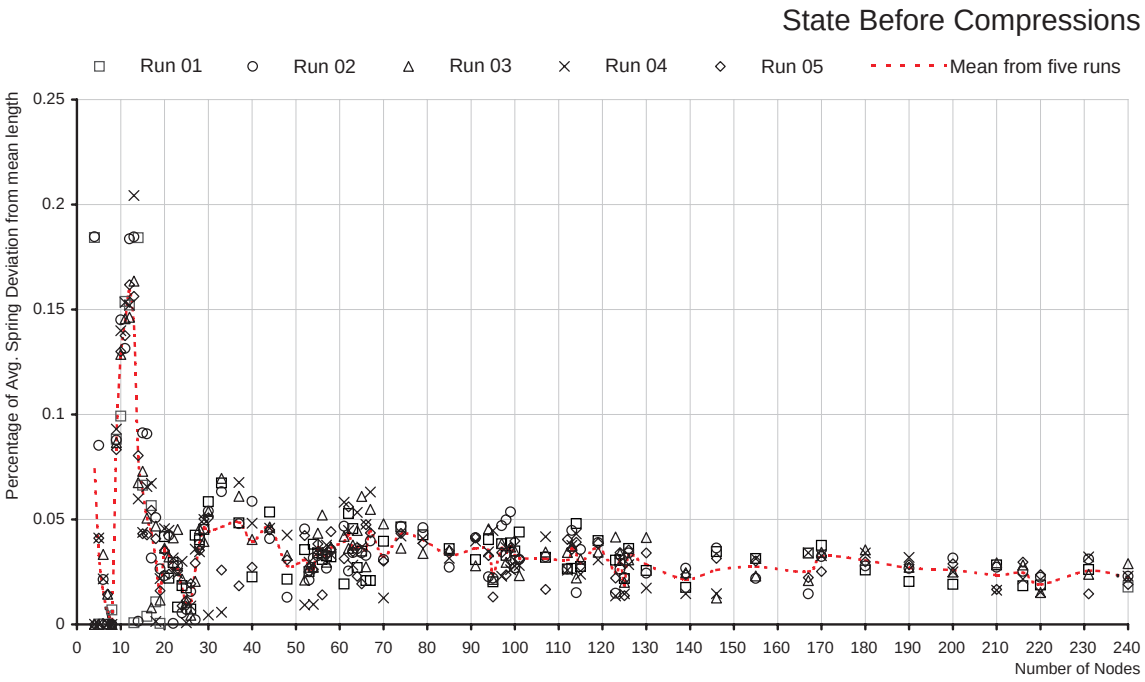


Figure 18. Percentage of Average Spring Deviation from mean length in the before compressions state of all runs of all population members.

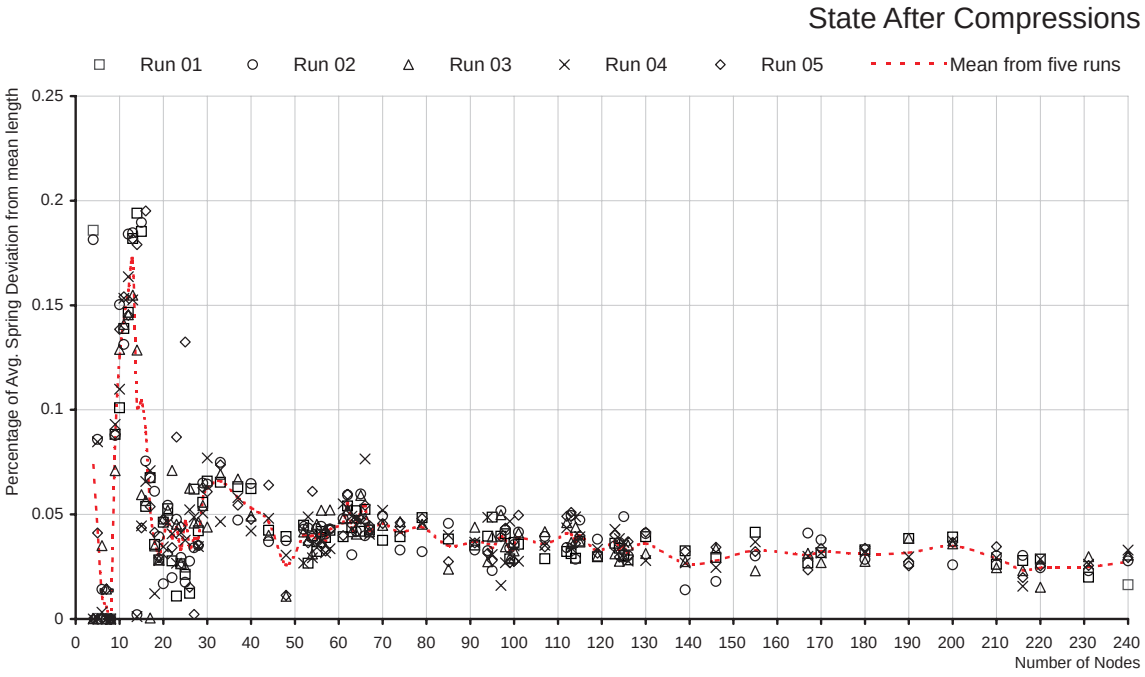


Figure 19. Percentage of Average Spring Deviation from mean length in the after compressions state of all runs of all population members.

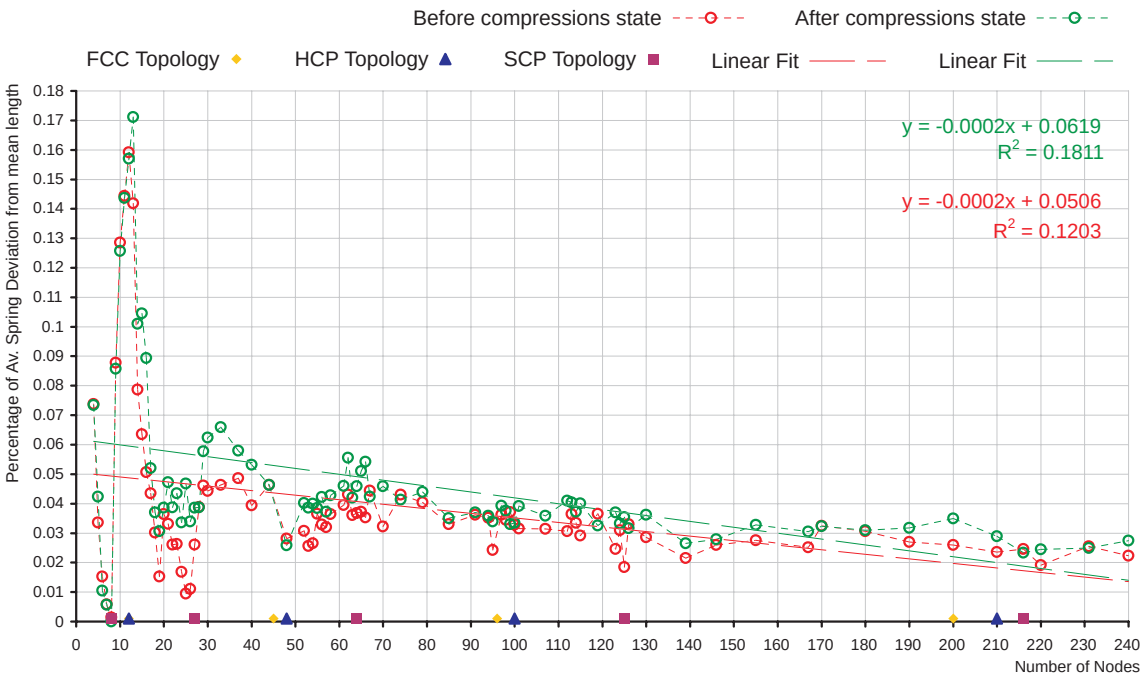


Figure 20. Mean Percentage from five runs of Average Spring Deviation from mean length of all population members.

4.2 Topological Analysis

The topological features of the generated lattices can be analysed by examining the valence (number of connections) of the system's nodes. This is performed both on a level of valence distribution among nodes for standalone members of the mean solution population and on a level of the average valence from each, so that a more clear comparison between all members can become possible.

The valence distribution diagram in Figure 21 shows the resulting average distribution of nodes from the five runs according to their respective valence for all the cases of node numbers used in the consecutive executions of the algorithm. Some more distinct patterns are revealed when inspecting the diagrams in groups of population members. A more detailed assessment of the results in groups of similar valence distribution is presented in Appendix I, §8, where the generated samples are also compared to the engineered topologies. The results for both the before and the after compression process states are very similar, having only an identifiable differentiation in the magnitude of the peaks. Another notable feature is that as the nodes increase, the overall distribution seems to move from lower valence areas to higher ones, but this is to be expected as in the cases of more nodes the structures' nodes in the periphery that have a lower valence are less in proportion to the nodes in the structures' interior that have higher valences. In addition, it should be stressed that even in the samples with higher numbers of nodes the higher valence areas were not found to be as occupied as was expected. This might

be considered as a malfunction on the part of the generative algorithm, as it didn't succeed in establishing sufficient connections for enough nodes to receive the maximum valence of 12.

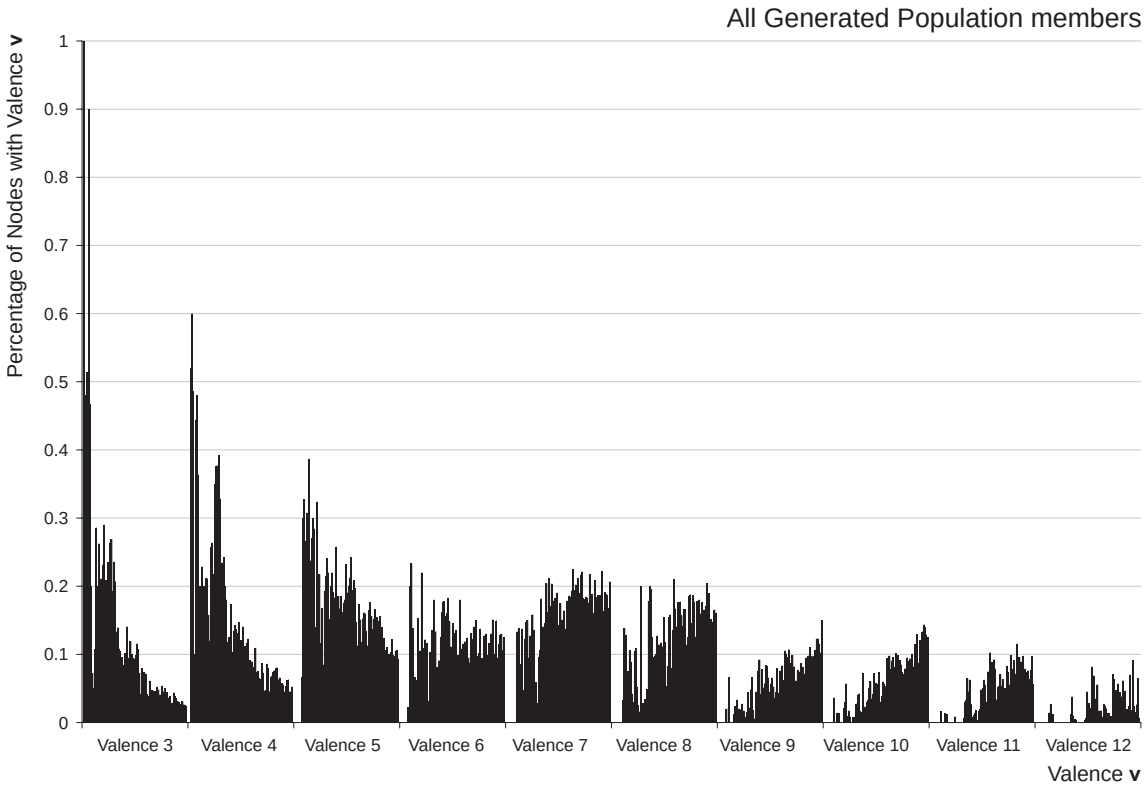


Figure 21. Valence distribution between nodes for all population members in the after compression state..

Observing the achieved average valence for each member of the generated population in figures 22, 23 and 24, it can be noted that it seems to increase at a pace of an approximate logarithmic function as the number of nodes increases. The same holds for the engineered topologies, where the increase of the average valence seems to be on a par with the increasing number of nodes and runs approximately parallel to the plot of the logarithmic curve of the generated topologies. For higher numbers of nodes, the average valence seems to settle just below the valence of 8, but it can be speculated that it may further increase for even higher numbers of nodes than the maximum of 240 tested. This is to be expected, as statistically, when the total number of nodes increases, the nodes at the boundaries become less in proportion to the nodes inside the volume that possess a higher valence. The average valence trendlines of the FCC/HCP topologies are closely matched and above the one of the generated topologies, while the one for the SCP topology is much lower than the one for the generated topologies. It can also be seen that the generated topologies converge closer to the FCC/HCP topologies for higher numbers of nodes. Another interesting finding is that the local minima and maxima of the average valence for the generated topologies fall into the areas of the numbers of points that were used for sampling the engineered ones and the curvature of the trendline of generated topologies seems to fluctuate between them. In effect, the curvature appears to be attracted periodically by either the samples

of the FCC/HCP topologies or the SCP topology. More details on this are given in Appendix I, §9. This overall pattern serves as another indication for the conjecture that the algorithm produces hybrid topologies, which tend to become more similar with engineered topologies when the numbers of nodes used in the generative algorithm are the same or close to the ones of the engineered samples.

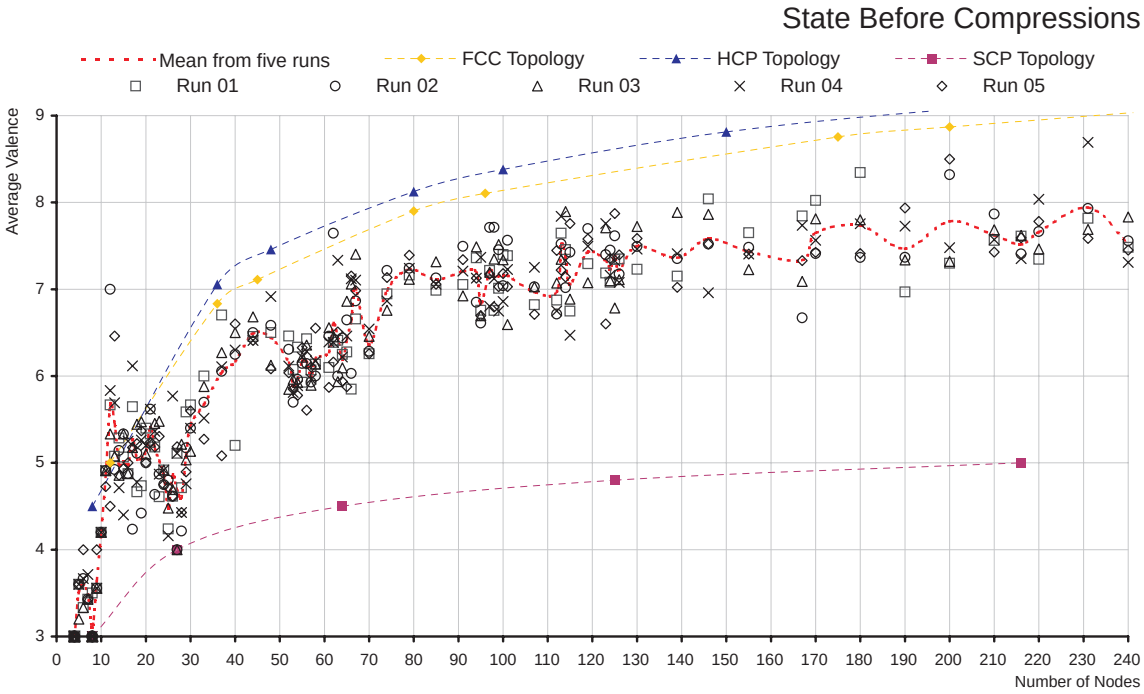


Figure 22.Average Valence of all runs of all population members in the before compressions state

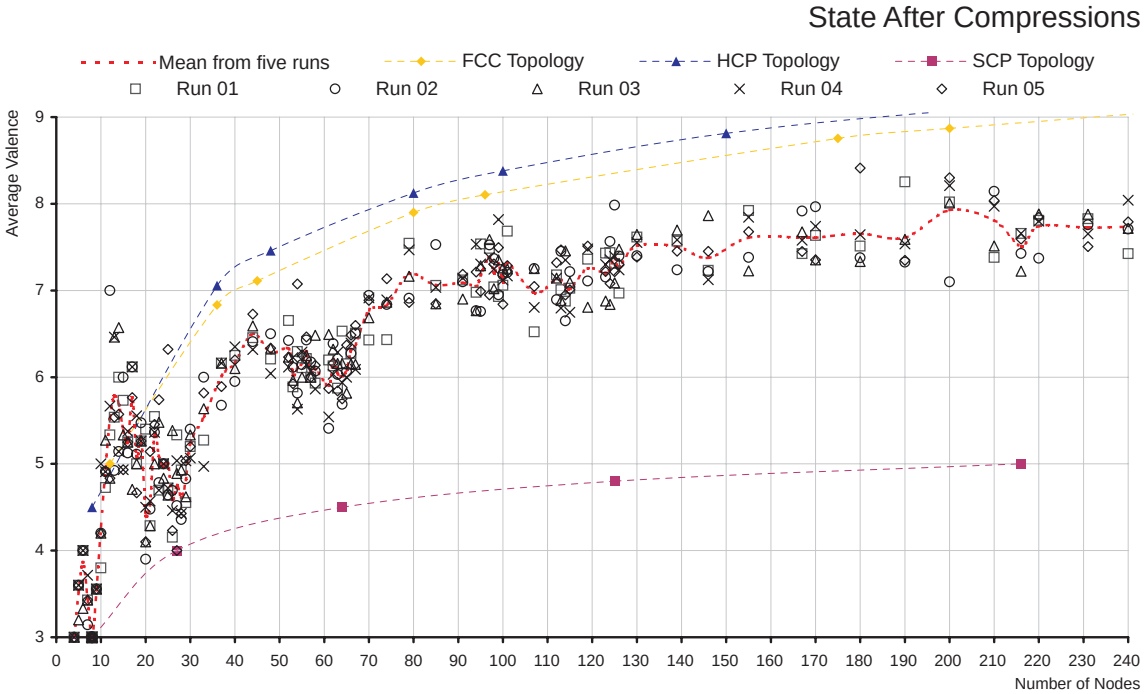


Figure 23. Average Valence of all runs of all population members in the after compressions state

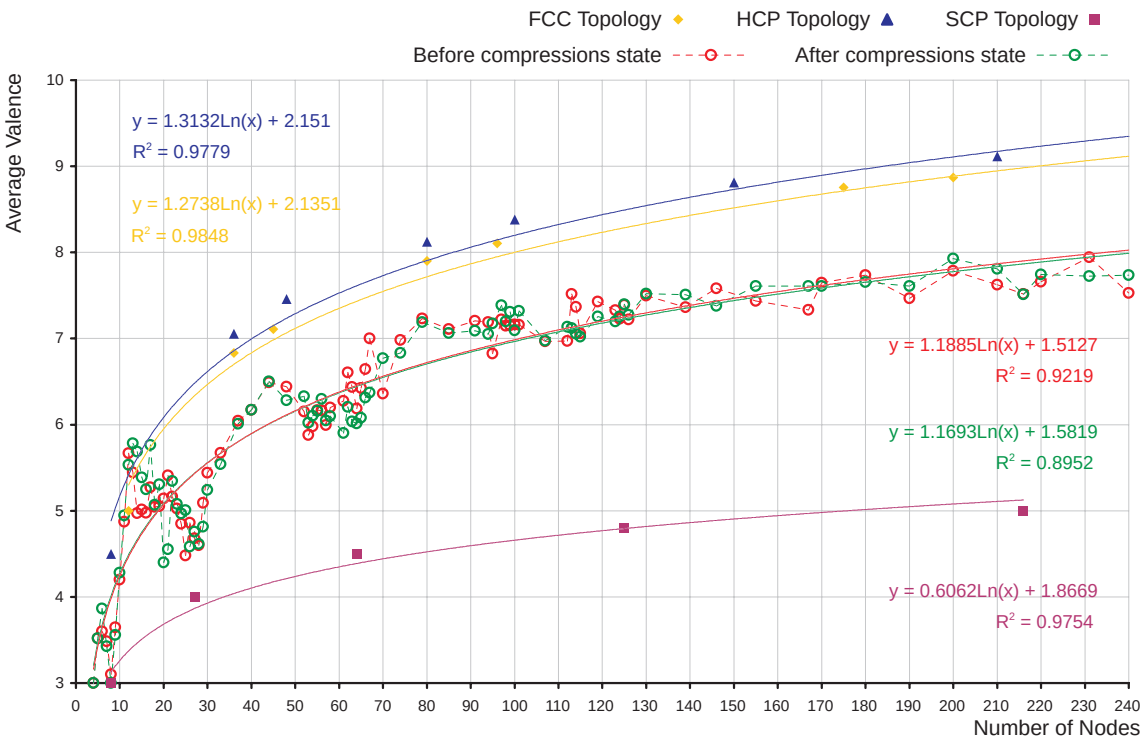


Figure 24. Average Valence of all population members.

A principal component analysis (PCA) was conducted for more efficiently managing the valence distribution among nodes for all members of the generated population (Figure 25). From this plot, it appears that the first principal component is very close to the valence of 8 with valences of 7 and 9 being very close as well and the second principal component almost coincides with the valence of 5. As is indicated by the plot, the members of the population with a low node count are quite scattered in the lower valence area. However, the plot also makes apparent two coherent clusters of nodes. The first contains members with a node number between 37 and 67 and lies on top of the second principal component of valence 5. The second cluster lies in the area of higher valences and contains members with a node count higher than 67. This seems to be a diffused cluster of the population from members with lower numbers of nodes towards members with higher numbers of nodes along the valences of 10, 11 and 12. The argument that the algorithm performs more consistently for higher numbers of nodes is applicable in this case as well and is supported by the diffusion of solutions in the second cluster according to node numbers. The fact that the clustering is so apparent in the PCA plot also gives way for the claim that the measurement of valence distribution among the nodes of a topology is a property that can effectively describe the topologies of the solutions, as the same principal components capable of inducing the maximum distribution of members in the space of valences are also capable of clustering solutions together. It should also be noted that the clustering pattern becomes much more apparent for the structures that resulted after the compression process when comparing them to the ones from before the compression. From this we could conjecture that the compression process causes topological transformations that contribute to the

consistency of results. In addition, it can be assumed that the clustering patterns must in general be very coherent, as they were found to exist in a projection of the solution space on the plane of the principal components. Furthermore, the fact that two different clusters exist separated at a specific area of node numbers provides the grounds for the assumption that a threshold exists in that area, at which a significant transformation is caused in the topology of the generated lattices. Finally, the existence of such a threshold leaves room for speculating that similar thresholds of topological transformation might exist and be achievable by the algorithm for higher numbers of nodes than the ones tested in this experiment.

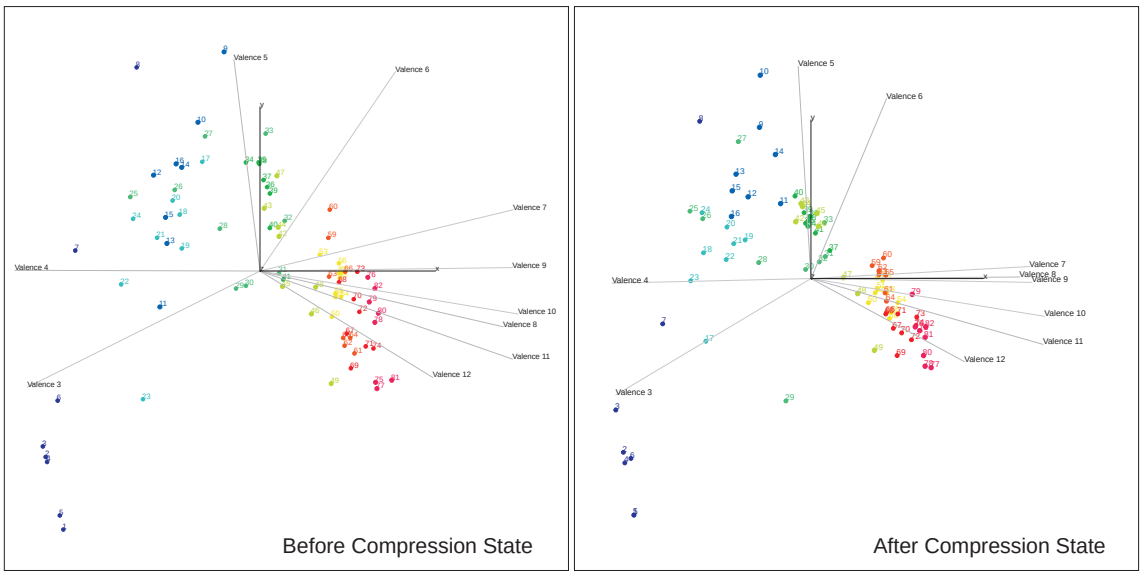


Figure 25. Principal Component Analysis of population according to valence distribution. Note that the index next to the samples is not the actual number of nodes, but each is the respective sample's index of order according to number of nodes.

4.3 Structural Analysis

The generated structures were tested for stability under gravity in a separate script that was an adaptation of the generative one. The positions of nodes and their respective connections were exported from the generative script both at the state before and after the compressions for all individual runs of all numbers of nodes and were imported into the structural analysis script. More details on the structural analysis script and relative problems that were presented are given in Appendix I, §10.

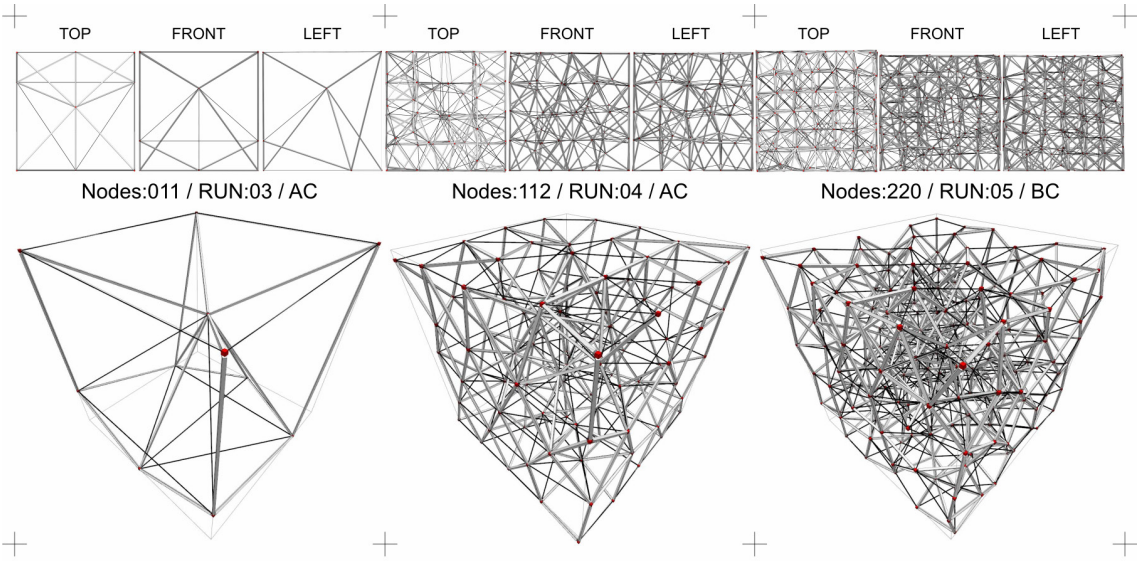


Figure 26. Indicative population members after they have sustained gravity in the structural analysis. Bars denote members in compression and lines denote members in tension.

The edges of the generated topologies appear to have suffered less strain than the FCC topology but more than the HFC and the SCP, as can be seen from the average strain diagram in Figure 27. The total amount of strain seems to be increasing for all topologies as more nodes are used to construct them. This is due the initial assumptions used in setting up the experiment as the amount of gravity applied is always proportional to the number of nodes. Thus, as the node number increases, more internal "levels" are formed inside the structure and the connections at the lower levels suffer much greater strain as they have to support the accumulated weight of all the levels above them. But, since the method of testing was the same for all topologies the results can be assumed to be comparable. Nonetheless, an attempt to overcome this unevenness in the measurements was performed by dividing the average strains of each topology by their respective number of nodes in order to normalise the effect of gravity being greater due to bigger numbers of nodes. The results of this workaround are of course highly questionable, as a different amount of gravity for each system and proportional to the number of nodes might have had a different effect in the overall non-linear behaviour of the system. Even so, they are

presented here for a more complete overview of the system's performance. Using this workaround of normalising the effect of gravity, it can be seen from the relevant diagram in Figure 28 that the generated topologies seem to perform much better than all of the other pre-defined topologies.

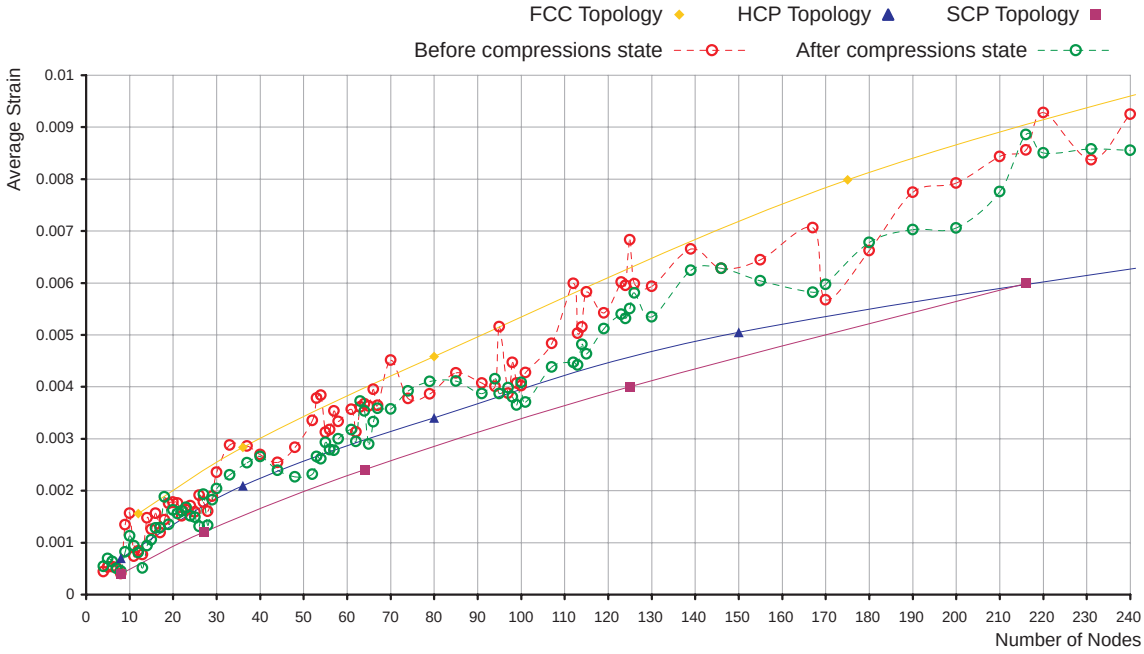


Figure 27. Average strain - All population members

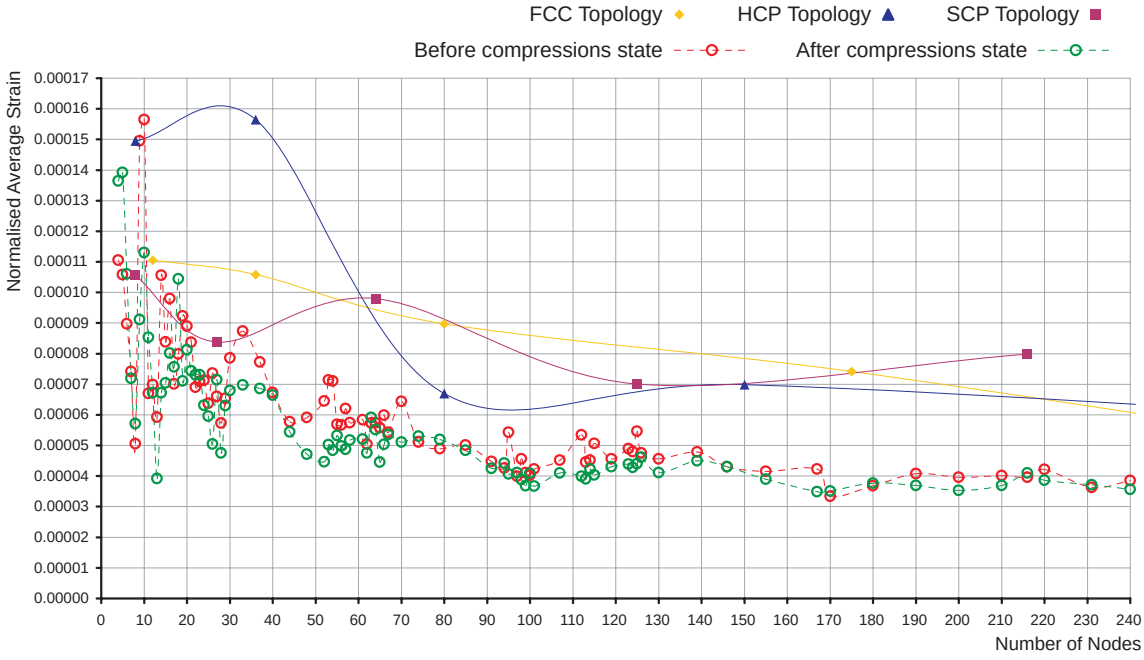


Figure 28. Average strain with a normalised effect of gravity - All population members

The displacement of nodes seemed to be much greater in the generated topologies than the pre-defined ones when gravity was applied, as can be seen in Figure 29. This might partly be explained by the fact that some of the generated topologies were not completely even at the bottom side of the cube and the nodes at the bottom, suffering the accumulated stress from the

ones above them, were displaced greatly until they reached the floor level. The centroid displacement graph in Figure 30 exhibits a better performance of the generated topologies. This measurement is less dependent on the number of nodes used and can thus be considered to be more reliable, though the problem of initial states persists. Even though in this case, the results are comparable to the FCC topology, the generated topologies continue to perform worse than the engineered ones in overall. As it can also be seen, the compression process does appear to have a beneficial effect to the structural performance of the generated lattices, as far as the centroid displacement is concerned.

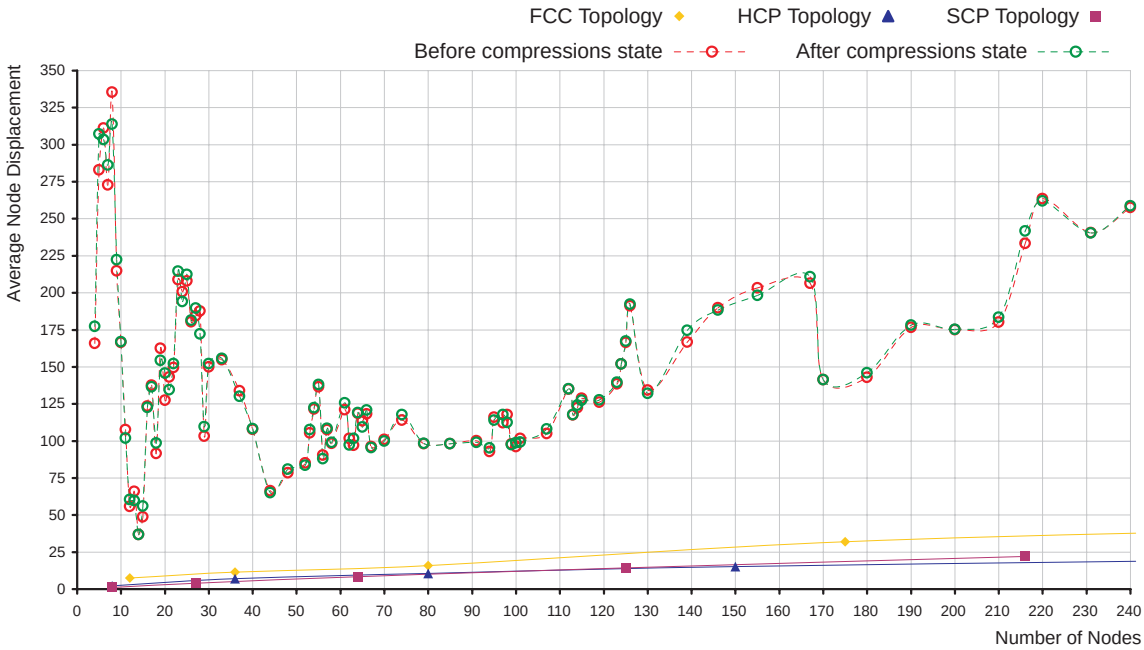


Figure 29. Average Node Displacement under gravity – All population members

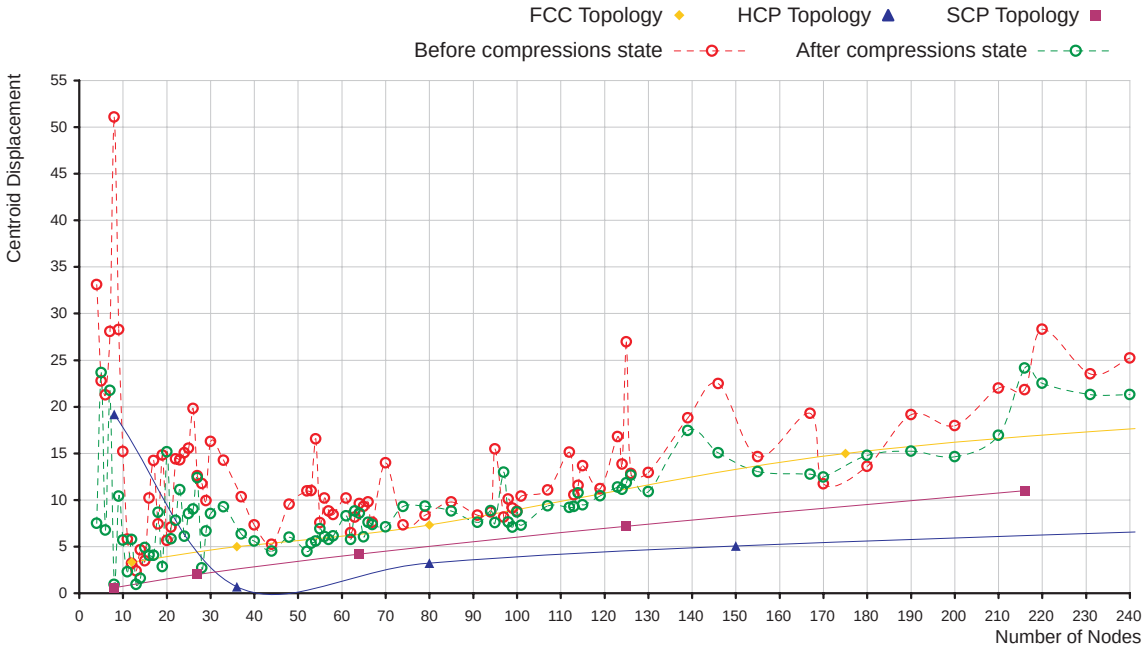


Figure 30. Centroid displacement under gravity – All population members

The ratio of members behaving in compression to the ones behaving in tension when gravity is applied to the system was also measured and is demonstrated in Figure 31. As all connections are assumed to have ideal length before the application of gravity, they initially exhibit no compressive or tensile behaviour. After the gravity was applied, the springs react to the sustained force and are either elongated or compressed according to the distribution of stresses. The results from the SCP engineered topology have not been plotted, as the fact that they consist only of vertical and horizontal members led to only their vertical members being compressed and the horizontal receiving no stress. The spectrum of the ratio of resulting compressive to tensile members was logged and was found to be narrowing down and converging to 1.5 for the generated structures as the number of nodes increased. Furthermore, once again the spectrum of results was found to be between the FCC and HCP topologies which seem to converge to a ratio of 1.75 and 0.95 respectively. The achieved ratio of the generated structures might relate to the determination of the spring force function and thresholds in the generative process. Perhaps the fact that the compressive behaviour of springs was more favoured in relation to the tensile in terms of maximum magnitude and range of effect could explain this finding, but this can't be confirmed, as more testing would be required. In addition, the fact that both the generated topologies and the FCC topology exhibit a ratio higher than the unit suggests that they rely more on compressive members to maintain structural integrity, as opposed to the HCP topology that converges to a ratio of less than the unit, which in turn suggests that in this case the tensional elements contribute more to stability.

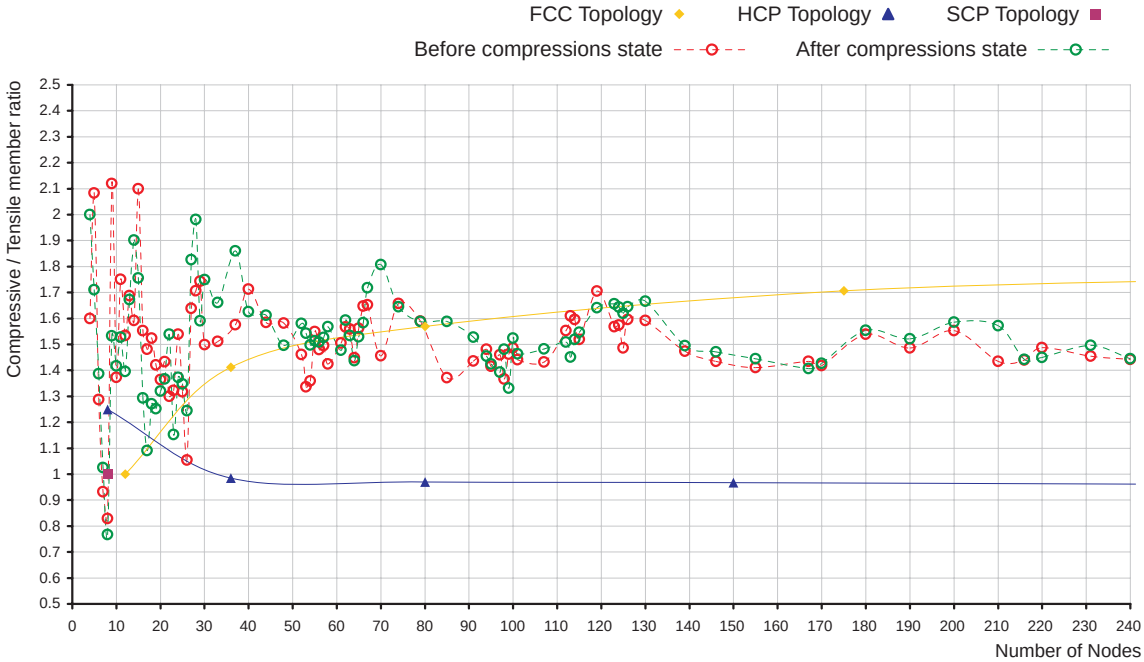


Figure 31. Compressive / Tensile connection ratio – All population memebers

4.4 Correlation of geometrical, topological and structural features

A correlation scatterplot matrix of the various measurements served as an additional instrument for understanding the overall performance of the algorithm and for evaluating the reliability of measurements. Only a matrix of pairwise correlation values is given below in figure 32, while the actual scatterplot matrix along with a presentation of the findings that can be derived from it can be found in Appendix I, §12.

	Nodes	**Connections**	**Mean Length**	**Deviation**	**Valence**	**Strain**	**Displ.**	**Centroid Displ.**	**Compr. / Tens.Ratio**
Nodes (6/8)	1.0000	0.9984	-0.7538	-0.4256	0.8458	0.9860	0.9134	0.6568	0.0431
Connections (6/8)	0.9984	1.0000	-0.7247	-0.4101	0.8340	0.9798	0.8966	0.6659	0.0354
Mean Length (5/8)	-0.7538	-0.7247	1.0000	0.4690	-0.8918	-0.7935	-0.9005	-0.2757	-0.0783
Deviation (0/8)	-0.4256	-0.4101	0.4690	1.0000	-0.2785	-0.4439	-0.4462	-0.4330	0.2621
Valence (5/8)	0.8458	0.8340	-0.8918	-0.2785	1.0000	0.8602	0.8808	0.3262	0.1515
Strain (6/8)	0.9860	0.9798	-0.7935	-0.4439	0.8602	1.0000	0.9339	0.6622	0.0811
Node Displ. (6/8)	0.9134	0.8966	-0.9005	-0.4462	0.8808	0.9339	1.0000	0.5105	0.1316
Centroid Displ. (3/8)	0.6568	0.6659	-0.2757	-0.4330	0.3262	0.6622	0.5105	1.0000	-0.0678
Compr. / Tens. Ratio (0/8)	0.0431	0.0354	-0.0783	0.2621	0.1515	0.0811	0.1316	-0.0678	1.0000

Figure 32. Pairwise correlation values. A good correlation is accepted as one that is more than 0.7 or less than -0.7. The numbers next to the measurement titles in the first column correspond to the number of other measurements they correlate well with.

5.0 Discussion

5.1 Overview of findings

The results show that the algorithm has been capable of generating a multiplicity of cube-filling space frame structures with various topologies that seem to encompass a certain amount of good properties.

One of the persistent findings was that the generative algorithm seems to perform more consistently for higher numbers of nodes, producing more similar results. It was also conjectured on several occasions that the algorithm produces hybrid topologies featuring a combination of properties found in the engineered topologies that they were compared against. This assumption was based on the fact that for several measurements the values corresponding to the generated topologies were found to be contained between the engineered topologies' respective values.

The geometrical analysis showed that the algorithm performs more optimally in terms of minimising the member length deviation from the mean length when higher numbers of nodes were used. The total connections were found to be less than those of the FCC/HCP topologies, which was an advantageous property, but has also most likely contributed to the poor performance of the generated topologies in terms of structural integrity.

Concerning the topology of the generated space frame structures, the valence measurement was found to be a competent instrument of analysis that was capable of mapping the topological space of the generated solution population. Through the analysis of the samples' average valences, it was found that they periodically resemble the average valences of either the engineered FCC/HCP or SCP structures when the number of nodes used for generating structures was close to the ones used for sampling the engineered. This was more evident for lower counts of nodes of up to 100 and less apparent for greater counts. This finding confirms the initial observations from the preliminary testing and it can thus be claimed that the algorithm is capable of adapting the generated topology to the volume it is provided with and generating a topology that can fit it in it according to the given number of nodes. Another important discovery was that the valence distribution analysis coupled with a PCA showed that two clusters of generated samples exist with different topological features. The conjecture was made that a threshold exists in the population area of 67 nodes, at which a significant topological transformation must be taking place. However, more research would be required for a more thorough interpretation of this finding and its extensions.

The structural analysis of the generated topologies appeared to be giving mixed results. In terms of average member strain, the generated topologies performed better than the FCC topology and worse than the HCP and SCP cases. When dividing the average strain with the number of nodes to overcome the effect of gravity being depended on them, the generated topologies seemed to perform better than all other topologies, but this result is highly disputable. In terms of average node displacement, the generated structures seemed to perform much worse than all the engineered topologies. The centroid displacement measurement yielded results that continue to be worse than those of the engineered topologies, but are more comparable to them.

It should once be again pointed out that due to the problems inherent in the way of holding the structural analysis experiments, their results should be treated with due caution, as their validity can't be confirmed. An aspect of the topologies that was also missing from the structural analysis was the testing against lateral forces, such as wind loading, which might have altered the results. One verifiable proof of this is that the orthogonal SCP topology, which lacks any diagonal bracing and is known to be inefficient for lateral force stresses, was found to perform much better than other topologies.

The compression process used in the generative algorithm was found to be having a transformational effect on the topology of the structures, as was shown by the valence measurements. The hypothesis that the compression process would cause an optimisation of the topologies seems to be confirmed when considering their structural behaviour in terms of strain and centroid displacement. On the other hand, the compression process was also found to be causing an increase in the deviation of member lengths from the mean length, although this can be viewed as an acceptable tradeoff enhancing the structural aspects of the topologies.

5.2 Reformulation of hypotheses according to feedback from results

The cases where the algorithm converged to one of the engineered topologies, when numbers of nodes were used equal to the ones of sampling the engineered, should be viewed as a good property. However, in the formal experiment this occurred on much fewer occasions than in the preliminary testing. Furthermore, by evaluating the results, the algorithm was often found to have been incapable of producing sufficient connections to achieve the specified upper valence limit of 12. On further reflection of this fact, the very assumption of setting a valence limit can also be seen as a contradictive preconception to the bottom-up nature of the algorithm. These observations led to a re-consideration of certain steps taken by the generative process. Therefore, some final informal experiments were conducted that aimed in confirming the hypothesis that the algorithm could perform better without a valence limit constraint. A key re-

consideration related to the actual measurement of geometrical spring length deviation. It was found by documenting the actual distances between nodes instead of the lengths of the established springs that the algorithm does appear to optimise the structures' geometry in terms of even distribution of points in space to a much a higher extent. However, for this to occur, the valence limit of 12 had to be exceeded in most cases and particles had to be connected not only to adjacent ones, but also to others in a bigger range. This also means that it might be beneficial to consider the geometrical and topological aspects of the structures as more distinct issues in the algorithm definition. It can be speculated that better results could become feasible by allowing the algorithm to optimize the node positioning and having it consider the topological connections of the structure as two separate processes.

5.3 Critical assessment

It should be noted that most of the conclusions are based on a statistical treatment of the results after the measurements from the five algorithm runs for each number of nodes had been averaged out to create a population of mean solutions. Obviously, the statistical treatment partly dismisses the value of individual cases, but this tactic was deemed as necessary for the management of the large available data set in order to gain a more complete impression of the overall performance of the algorithm. Had the results been observed on a more individual basis, specific solutions with even better properties might have been discovered on several occasions. This can be verified by the scatter plot of deviations for individual solutions emerging from single algorithm runs in Figures 18 and 19. From this fact, we can draw the conclusion that the efficiency of the final system is dependent upon the starting conditions of initial particle positioning and other non-quantifiable factors deriving from the non-linear behaviour of the particle-spring system. However, the examination of the performance of the algorithm based on the statistical average of the generated solutions has even so been found to yield results with advantageous attributes, which suggests that the algorithm performs fairly well, regardless of initial conditions.

In respect of the initial problem statement of this study, presented in section 1.3, only the considerations (1), (2) and (3) have been more thoroughly investigated through the geometrical, topological and structural analysis. The performance of the generated topologies according to considerations (4), (5) and (6) has not been sufficiently examined and this should be more centrally included in any further investigation. However, it can be asserted that consideration (4) of maximisation of angles between edge members has been achieved in almost all cases, being the outcome of the even geometrical distribution of nodes in space that were appropriately interconnected. Consideration (5) of non-intersecting edge members was also accomplished in almost all cases, as this was dictated by the nature of the spring creation threshold purposefully

used in the definition of the particle-spring system. Finally, consideration (6) of adequately describing the bounding volume is the most questionable attribute of the generated topologies, as its applicability relies mostly on subjective aesthetic criteria and thus no standard method for evaluating it through quantifiable measurements could be found. However, it has been visually observed that the steps taken by the generative algorithm for maintaining the valence of particles within the specified limits, led the particle system to fully expand up to the boundaries of the envelope in most cases.

It can be argued that one of the most important features of the proposed method in that it can effectively fill the space of a given volume with a lattice of any specified number of nodes, contrary to engineered topologies, where node numbers derive from their respective grid multiples. Furthermore, as has already been demonstrated in the very simple case of a cube, only the SCP topology can completely describe its form in every detail and this is in fact only so when the numbers of nodes used are third powers of integers. Other engineered topologies such as FCC and HCP are sub-optimal in filling the entirety of the cube, as their respective grids are not compatible with its geometry and only allow them to approximate it with a few combinations of node numbers. The rules of "fuzzy isotropy" embedded in the generative algorithm allow it to overcome the constraints of engineered topologies and produce results that respond more efficiently to the problem at hand. Therefore, the method can be seen as one of interpolation between the numbers of nodes deriving from the grid multiples of the engineered topologies.

Another important aspect of the method is the robustness of the algorithm in terms of envelopes it can handle. A reasonable claim is that not much practical scope exists in filling the volume of a cube in terms of architectural or artistic aspirations. The choice of the cube as a study case has already been explained and was only made on a basis of facilitating the analysis of the algorithm's performance in order to determine the fitness of the method. Cases of arbitrarily shaped volumes would not have been able to provide with as much insight in regards to what the algorithm is capable of. Taking this into account, the proposed method has indeed been found to exhibit overall commendable qualities in the study case of the cube. From this finding and the preliminary testing of the algorithm in cases of other spatial envelopes, it can be inferred that the algorithm should perform equally as well in other cases. Besides that, the fact that very simple rules have been used in the algorithm definition contributes to the more universal applicability of the method, as the rules are independent of the envelope used. Of course, the performance of the method in other cases remains to be confirmed through more thorough testing in possible future endeavours.

The value of a robust method for filling arbitrarily shaped volumes with structural lattice networks can be appreciated when considering projects like Gormley's *Body/Space/Frame*. As far as artistic or sculptural applications are concerned, the proposed method has proven to be a competent one. Even though this might appear to be sufficient justification for the investigation of this method, it should nevertheless be noted that this study has remained purely focused on resolving space frame structures that meet principally formal demands. Selecting to limit the investigation only in this field was predominantly due to the simpler and fewer implications that were contained within the problem definition, so that a preliminary examination of the method's capabilities would be possible. However, even if some of the generated samples were found to possess certain architectural extensions, the overall investigation did not purposefully consider spatial, functional or other requirements that an actual architectural design approach would encompass. Therefore, an open question for future reflection would be how the proposed method could be seamlessly integrated into a design process of finding structural frameworks that meet the demands of architectural priorities in more specific study cases.

5.4 Further investigations

The study presented in this thesis is far from complete in respect of various issues and may only serve as a preliminary examination of the fitness of the proposed method for filling spatial envelopes with structural lattices. Some aspects that need be further explored have already been mentioned. A more complete list with such considerations is outlined below, while some are presented in more detail in Appendix I, §12.

- Re-consideration of issues in the specification of physical parameters.
- Re-consideration of algorithm steps and termination.
- More thorough structural analysis, perhaps using FEM.
- More thorough evaluation of the generated structures according to all of the considerations in the problem definition.
- Resolution of speed issues
- Resolution of various issues relating to non-convex arbitrary mesh boundary envelopes and evaluation of algorithm performance for envelopes other than the cube.
- Use of more than one spring ideal length and consideration of ellipsoidal packing.
- Attempt to minimise the number of angles at which connections are attached to nodes.
- Investigation of the use of multi-dimensional vectors for the generation of space-filling quasicrystal topologies.
- Investigation of viability of the proposed method for generating space-filling tensegrity topologies
- Consideration of design priorities for space-filling other than formal ones.

6. Conclusions.

This study attempted to address the problem of filling spatial envelopes with structural lattice topologies using given numbers of nodes. A method for tackling the problem was proposed that involved a generative algorithm including a physical dynamic simulation of a particle-spring system. The aim of the thesis was set to determine the appropriateness of the method. Preliminary manual testing of the algorithm showed that it was able to robustly generate structures with observable good properties. Certain assumptions were formulated based on these observations that led to the more precise definition of the system's physical parameters and algorithm steps. A formal experiment was conducted that involved filling a cubic envelope with a population of structures generated by the algorithm. Some quantitative features of the generated space frames were documented, reflecting the system's performance. The geometrical, topological and structural aspects of the generated structures were analysed using statistical methods and compared to three engineered structural topologies. The evaluation of the results confirmed that the algorithm is capable of producing lattice structures with advantageous properties. The limitations of the method were reported, ways to overcome them were proposed and other considerations that could be included in future development were mentioned.

The outcome of this investigation is supportive to the claim that architectural design can benefit from incorporating physical dynamic simulation in parametric and algorithmic approaches of certain design problems. In this study, by modelling the elements of design as carriers of behavioural information that physically interact with each other, a region of possibilities in space frame topological layouts has been mapped, containing solutions that would have less likely been conceptualised and discovered in a conventional, top-down design approach. Even though the problem that this thesis addressed relates more to sculptural or artistic formal pursuits, the method can very likely be proven to be applicable to other architectural considerations through appropriate adaptations. Besides, the field of art is not irrelevant to the field of architecture. It should not be forgotten that art, not being restricted by the constraints of architectural priorities, has in the past managed to process a wider spectrum of possibilities in design. Under these terms, by serving as a principal force in testing new conceptions, it has repeatedly provided architectural design with feedback that has led to the progress of architectural ideas, forms and methods. In this context, it might well be the case that the proposed approach can receive more architectural extensions and constitute an instrument for the research of space and structure relationships.

7. Appendices.

Appendix I

I.1. Sphere packing threshold

When comparing the orthogonal simple cubic packing to the FCC packing in the case of filling the volume of a cube, a threshold exists before which the simple cubic packing is more efficient and past which the FCC becomes more efficient.
Considering the simple cubic packing, one can empirically verify the above by trying to fit $\mathbf{k = n^3}$ spheres inside a cube of side length **a = n * d**, where **d** is the sphere diameter. For **n=2,3,4,5** and therefore **k = 8,27,64,125**, the most efficient way to pack them is not in a cubic face-centre arrangement, but in a simple cubic lattice. However, there is a threshold for **n = 6** and **k = 216**, where 216 spheres of diameter ***d*** can be packed in a cube of side **a = 6 * *d*** in a simple cubic arrangement. Trying to pack spheres of the same diameter ***d*** inside the same cube in an FCC arrangement layer by layer, one can verify that 231 such spheres can fit. Based only on the above empirical testing, we can conjecture that it is quite likely that for $\mathbf{n < 6}$, $\mathbf{k = n^3}$ spheres can be packed more optimally in a simple cubic arrangement and that for $\mathbf{n \geq 6}$ the most efficient way to pack $\mathbf{k = n^3}$ spheres is in an FCC configuration.

I.2. Algorithm Analogy to Natural Systems

The proposed algorithm takes inspiration from two natural systems, namely the behaviour of matter at a molecular level in metal die-casting processes and the behaviour of agent groups, known as boids.

When forging metal parts through die-casting procedures, the metal is liquefied under high temperatures and then cast into a mould under pressure. The molecules of the metal in liquid form exhibit high mobility and form temporary bonds between them. The intermolecular forces responsible for the development of such bonds are known as the van der Waals forces (<http://en.wikipedia.org/wiki/Van_der_Waals_force>). As the temperature cools down, the mobility of the molecules decreases and their position becomes more constrained due to the developed intermolecular bonds. Eventually, the system settles into an equilibrium state, forming rigid crystalline topologies with permanent intermolecular bonds. The final piece of solid metal receives the form of the mould and is sufficiently rigid to withstand external stresses after the mould has been removed. The process of the initial exposure to extreme temperatures

and the gradual cooling in order to induce certain desired properties to metal alloys is also known as the annealing process in metallurgy.

The term "boids" refers to collections of autonomous agents that appear to have group behaviour, such as the one of flocks of birds, schools of fish, etc. Simulations of boids introduced by Craig Reynolds (1987) have brought about a discussion on the behaviour of complex organisms. Reynolds, by dynamically simulating a collection of boids that locally adhere to a set of very simple rules, managed to establish a convincing simulation of the overall behaviour of meta-organisms. Simply put, the implemented rules make each boid being attracted by others and at the same time being repelled by others. By finding a balance between these two forces, an even distribution of boids is achieved while the overall cohesion of the group is maintained. Gary William Flake (1998, p. 276), commenting on the Reynolds boids simulation, states that *"each of the rules can be viewed as an individual behavioural agent that competes and cooperates with the other rules to yield an emergent property that looks a lot like bird intelligence. Thus, the boids are composed of behavioural agents, which gives another depth of recursion in how emergence and self-organisation can come about"*.

Considering the process of physical parameter and algorithm step specification, an analogy can be found with simulations of other bottom-up dynamic systems. In these, their rules in general produce a consistent behaviour of the system, but usually, more specific rules are set up so that the system produces desired behaviours. In the case of boids for instance, many combinations of parameter values and adaptations of the steps taken might always result into a behaviour that will resemble one of a group of agents. However, if the objective is the behaviour of a specific kind of birds, then a more precise determination is required both for the parameters and for the specific steps taken by the algorithm. In the case of this investigation, there was no prior knowledge as to how the various adaptations of the algorithm would affect the results and thus the final algorithm that was tested had to be based on informal assumptions.

I.3. Preservation of velocity

Three options were investigated regarding the preservation of the velocity of particles from one iteration to the next. The first option was to eliminate all velocities and calculate the new velocities at the next iteration as the result of only new inter-particle forces, boundary collisions and gravity. This option is more stable, but the algorithm performs at a lower speed. The second option is to preserve the velocity and keep adding the new forces to it. This option decreases time of convergence to final equilibrium states, but also compromises stability and thus some equilibrium states might be overlooked. A more accurate modelling would require loss of kinetic energy and thus of velocity when semi-elastic particle-particle or particle-boundary

collisions occured, but this was not pursued. Instead, a trade-off between the first two options was considered, where a small portion of each particle's velocity is discarded (V.scale(0.95)) before advancing to the new iteration. This configuration yields stable algorithm runs at a fairly good speed. In the case of concave 3d meshes however, the elimination of velocities option is used from one iteration to the next due to the complexity of the boundary collision calculations.

I.4. Spring Force

The interparticle force is a subject that was examined extensively. Unlike the billiards simulation used for close packing of spheres (Lubachevsky, 1991), where sphere collisions are modelled in great physical accuracy, a simpler, physical simulation of inter-particle interactions was used. The approach is more similar to the bubble-mesh method (Shimada, 1995), where a force is developed between pairs of particles according to their distance and to the van der Waals force that is developed between physical molecules. The billiards simulation uses collision between hard spheres and the result is an exchange of velocities between the involved colliding spheres, whereas in the bubble-mesh method an attraction/repulsion force is developed between bubbles when they are their proximity demands it. To better illustrate the spring force method, one could consider soft spheres that behave elastically and have a certain allowable range of interpenetration.

The applied spring force is a vector parallel to the line connecting the position vectors of the two particles. The magnitude of the force is a function of the distance threshold $\boldsymbol{D}$ that allows a spring to be established between two particles, the actual distance $\boldsymbol{d_{ij}}$ between the two connected particles and the spring stable length $\boldsymbol{l_0}$. A number of different configurations were attempted for the determination of the spring function.

The following features were examined that were found to have an impact on the overall algorithm performance:

- Nature of the force: repulsion only or repulsion/attraction.
- Relation of the distance threshold $\boldsymbol{D}$ to the stable length $\boldsymbol{l_0}$.
- Linear or polynomial form of the spring force function.

Initially, when modelling the system after physical springs, a linear equation model was used with a single spring stable length $\boldsymbol{l_0}$. The spring force is given by the function of Hooke's law $\boldsymbol{F= k*\Delta x= k*(d_{ij} - l_o)}$, where $\boldsymbol{k}$ is a spring constant that relates to the speed/stability of the system and $\boldsymbol{d_{ij}}$ is the actual distance of the two connected particles $\boldsymbol{i}$ and $\boldsymbol{j}$. When $\boldsymbol{d_{ij}>l_o}$, the force $\boldsymbol{F}$ is positive and when $\boldsymbol{d_{ij}<l_o}$, F is negative. The sign of the force sets the direction of the vector, so when it is positive, the force leads to the particles being repelled and when it is negative, it

leads to the particles being attracted. The particle distance d_{ij} is always smaller than the distance D, as no spring would otherwise have been created between the particles in the first place.

Initially, a simple linear spring equation was considered, only allowing springs to exert repulsive forces to connected particles. This was more similar to the sphere collision approach, as particles would be pushed away from each other if one would violate the minimum distance that would have to be maintained between them. In this case, the spring creation threshold D would be equal to l_0. Therefore, all created springs would either be in equilibrium state ($d_{ij}=l_o=D$) or in compression ($d_{ij}<l_o$). Two problems were observed, however, when using this approach. A first issue was that particles reaching the boundaries would cling to them and not ever get the chance to fall back into the interior of the volume. This would be true even if the stable length l_o was being reduced, as the boundary would not exert any force to the particle to push it further than the boundary limit itself. This issue could have been overcome by changing the boundary collision response method, but that would have had an effect on the overall stability of the system when it would be converging to equilibrium. A second issue that was observed at a later stage of the program development was that by using attraction together with the repulsion, more structurally stable configurations would emerge. This is of course partly due to the increase of the number of the system's total connections when using a threshold D greater than l_0. In general, the sheer number of connections is not sufficient to bring about structural stability as it has to be accompanied by a suitable connectivity pattern. But, statistically, an increase in total connections, coupled with some suitable constraints on the form of the topology, will eventually act beneficially to the system's structural efficiency. Nonetheless, the structural optimisation will seize after a certain threshold, past which the number of present redundant members will also start increasing. There is another explanation for using attractive forces, relating more to the analogy between the algorithm and physical systems. The lack of attraction would mean the absence of forces that maintain the cohesion of the overall system. Similar to metal die-casting, the bonds that molecules develop when the hot metal is cast inside the mould are what maintain the consistency of the final overall form once the metal is cooled and the mould is removed. Further, when considering boids simulations, the cohesion between them is due to "invisible forces" that don't allow a single boid to wander too far away from the group. Considering these issues, it was decided that an attraction should be included in the force along with the repulsion.

The scalar parameter k of the linear spring force equation was also considered. This parameter is mostly related to the speed and stability of the system. Higher values cause a more swift response from the system, but if the value is too high, the system can be destabilised and entirely collapse. It was observed that this value is related to the particle valence, which is the number of springs connected to a single particle. A threshold value up to which the algorithm

performs stably was approximately found by observing the performance of the system for different values of ***k***. Initially, this threshold was set to be inversely proportional to the sum of valences of the spring's connected particle, being ***k=1/(node1.valence+node2.valence)***. However, that meant that the springs that were connected to particles with many springs attached to them were weaker than those connected to particles with less attached springs. Attempting to remove this inconsistency, the average valence from all particles was calculated in each iteration. This was then used in the determination of ***k*** to ensure the congruent behaviour of all springs. Thus, the eventual value used for ***k*** was ***k=1/mean_valence***.

It was observed that when using the same constant ***k*** both for the attractive and the repulsive cases of the force, the system would always converge to very tight clusters of particles, reducing the number of springs that could achieve a stable length. Testing indicated that a lower ***k*** value should be used for the attraction force to enhance performance. This can also be explained via the intermolecular bond analogy, where the attraction of the van der Waals force between molecules is much weaker than repulsion. The impact of a higher attraction force can be witnessed through the increase of average valence of the particles that is to be avoided as it is matched with an increase of redundant connections. In general, it was found that better results are obtained when the attraction is less or equal to one tenth of the repulsion. In practice, an even lower value was used that seemed to further enhance the results. The value chosen was $\mathbf{k_{attract}}=0.025*\mathbf{k_{repel}}$, with $\mathbf{k_{repel}}$ being calculated as was mentioned above for ***k***.

The distance threshold ***D*** that allows the creation of a spring also plays a fundamental role in the generation of topologies. In the bubble-mesh method (Shimada, 1995) and in (Jaworski, 2006) the threshold where the force becomes effective is $\boldsymbol{D}=1.5*\boldsymbol{l_o}$, but no further explanation is given as to why this is chosen. After thoroughly testing the system, it was discovered that this parameter relates to geometrical aspects of the topology. This becomes more apparent when considering a simple system consisting of four particles operating only in two dimensions.

In such a system where $\boldsymbol{D}>\sqrt{2}\boldsymbol{l_0}$, the particles are allowed to settle into a configuration of a square with diagonals equal to $\sqrt{2}\boldsymbol{l_0}$, $\boldsymbol{l_0}$ being the side of the square. In this case, each particle will be attracted by another one at the opposite corner. The resulting X configuration is of course structurally stable in 2d with the sides of the square being in compression and the diagonals in tension, but seems to violate some of the objectives of the algorithm: the two diagonals are longer than the sides (no equal member length) and the diagonals intersect each other (Figure 33). Furthermore, once this configuration has been generated, it is persistent and any change in $\boldsymbol{l_0}$ will only scale the square's total size, having no effect on the topology. Another disadvantage of the X configuration is that once it has been generated, none of the edges will ever be able to reach the distance $\boldsymbol{l_0}$. The sides of the square might well be equal to

each other and the diagonals equal to √2 times the side length, but the sides will not be able to reach l_0. This is because the tensile diagonals keep pulling the opposite corners, and the capacity of compression that the sides can withstand does not suffice for them to ever become equal to l_0. Therefore, the structure can't minimise its potential energy, which is another way of measuring its value.

On the other hand, four particles can be used to generate a more advantageous configuration, consisting of two adjacent equilateral triangles sharing a common edge. This configuration is also structurally stable in 2d and at the same time has all edges equal to each other and equal to l_0. Moreover, there are fewer connections than in the X configuration and there is no intersection between edges. Such a configuration can be obtained by the algorithm if $D<\sqrt{2}l_0$ and $D>l_0$. In this case, when four particles form a square there will not be any springs generated at the diagonals, thus avoiding the X configuration (Figure 33).

The value of disfavouring X configurations can be better appreciated for the three-dimensional case, when the octahedron is considered, a shape found in the octet truss topology. The octahedron is a minimally rigid structure, having exactly the number of edges that are required for structural integrity. By observing its geometry, one can easily see that three squares are present. If the value for D was $D>\sqrt{2}l_0$ and l_0 was the side of the octahedron, then three redundant additional springs would have been established connecting opposite corners of the octahedron, rendering the shape a completely connected graph, where every node is connected to every other. Using $D<\sqrt{2}l_0$ would help the system to converge to the octahedron (Figure 33).

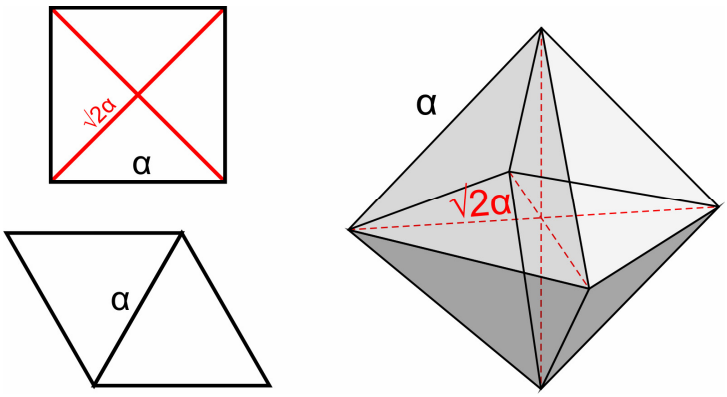


Figure 33. Configurations taken into account for determining the spring force
Top Left: The square is stabilised with diagonals, resulting into an X configuration, where the X edges are longer length than the sides of the square
Bottom Left: Two equilateral triangles with a common edge is a stable shape, with all edges equal and fewer edges than a square with diagonals
Right: The octahedron is a rigid shape by itself. If the corners of the squares that are contained within are connected with diagonal edges, then it is redundantly rigid and has edges that are not all equal to each other

Any value for $\boldsymbol{D}$ between $\boldsymbol{l_0}$ and $\sqrt{2}\boldsymbol{l_0}$ seemed to work, with smaller ones resulting to configurations with less attraction springs and larger ones to more attraction springs.
The exact value chosen for $\boldsymbol{D}$ was $\boldsymbol{D}$=1.15*$\boldsymbol{l_0}$ as it seemed to produce good results.

Apart from the linear form of the spring force function, a polynomial form of the function was also considered. A parametric cubic polynomial function was created using Lagrange interpolation. The parametric equation allowed the specification of different values of $\boldsymbol{l_0}$ and $\boldsymbol{D}$ as well as scaling the max/min limits of the equation to obtain suitable values that corresponded to the limitations mentioned previously. One of the benefits of the polynomial equation was the ability to diffuse the magnitude of the force smoothly from zero to zero between $\boldsymbol{l_0}$ and $\boldsymbol{D}$. This means that as the distance $\boldsymbol{d_{ij}}$ approaches $\boldsymbol{l_0}$ or $\boldsymbol{D}$, the attraction gets smaller and is bigger in between. In this case even the value $\boldsymbol{D}$=$\sqrt{2}\boldsymbol{l_0}$ seemed to produce good results, as when springs were created at diagonals of squares they would be weaker and have more chances of breaking.

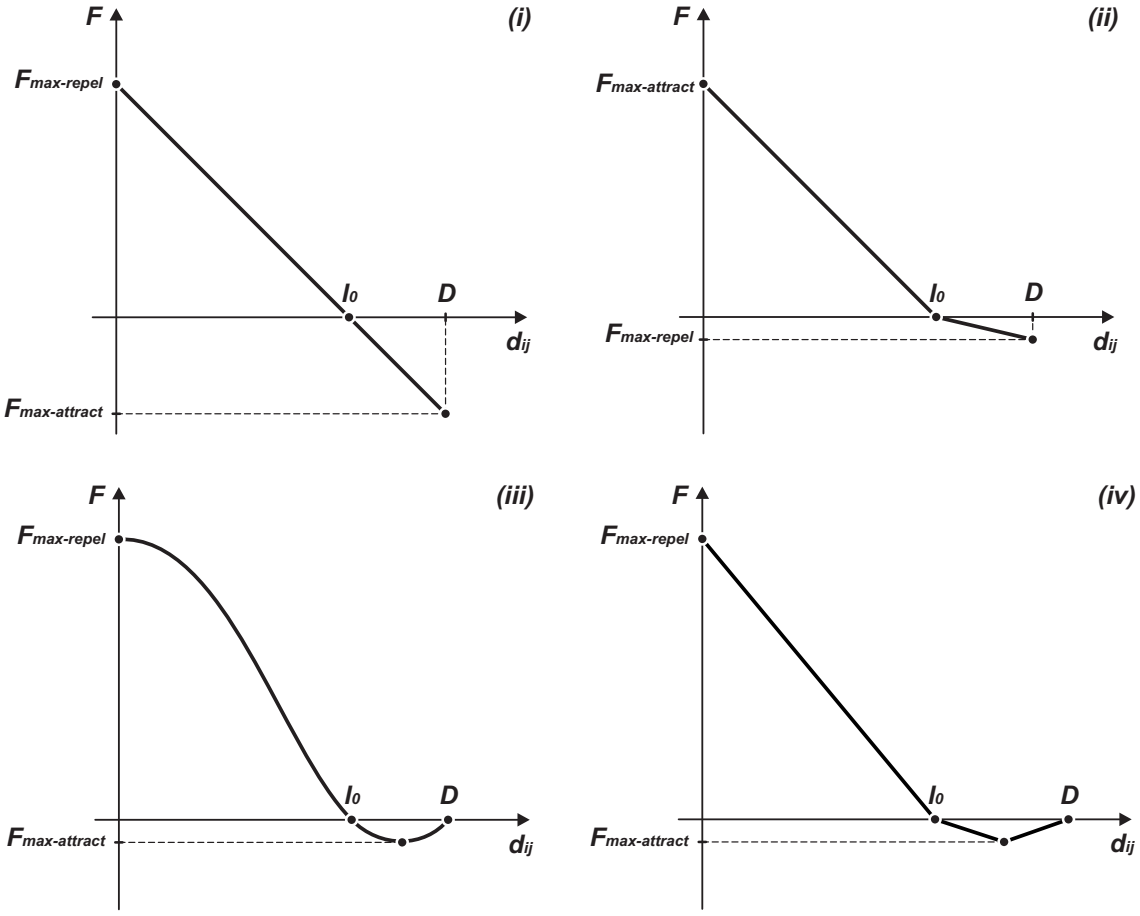


Figure 34. Graphs of different functions for the determination of the spring force. Positive values denote repulsive force being exerted at the particles, while negative denote attractive force.

(i) Linear function with attraction being equal to repulsion ($\boldsymbol{k_{attract}}$=$\boldsymbol{k_{repel}}$)

(ii) Linear function with attraction force having less magnitude than repulsion ($\boldsymbol{k_{attract}}$<$\boldsymbol{k_{repel}}$)

*(iii) Polynomial function with attenuation of attractive force towards threshold **D**.*

*(iv) Linear function with attenuation of attractive force towards threshold **D**. This is the case that was eventually implemented.*

I.5. Boundary Collision Detection

As was mentioned in the overview of the algorithm, before the collision detection, a temporary position of particles is calculated and used to check if the particle with its current velocity will be positioned outside of the envelope. All the collisions with the boundaries are considered to be perfectly inelastic, which means that the particles don't bounce when coming into contact with a boundary, but come to a stop just on the plane dividing the inside from the outside.

The particle's position is iteratively considered against each one of the planes that constitute the hull of the mesh of the bounding envelope. In the case of the cube this is simple: each such plane is a quad face of the cube. For a more complex 3d mesh, each plane is one of the triangle faces of the mesh. A piece of parser code was written for importing meshes that was able to read geometry data from .obj files. After the import of the mesh in the beginning of the algorithm, the centroid C and normal vector N that points to the outside of the envelope was calculated for each face and stored locally.

The vector operations for collision detection and response are graphically represented in Figure 35. The vector of the particle's temporary position P is subtracted from the vector of the centroid C of each face to give the vector Q (Q=C-P). A scalar value ***s*** is calculated as the dot product between the vector Q and the face normal N (s=Q.N). If the scalar value is positive, the particle is inside the envelope. If it is zero then it lies exactly on the face considered. If the value s is negative the particle is on the outer side of the considered face and thus a response vector must be calculated by the algorithm to maintain the particle inside. The response vector F is equal to the inverse of the face normal scaled by the dot product s (F=-N.scale(s)). The vector F is parallel to the face normal and its magnitude is equal to the perpendicular distance of the particle to the face, which is also the shortest distance. The vector R is added to the velocity vector of the particle. In effect, it negates the part of the component of the velocity vector perpendicular to the face that is beyond the plane of the face, thus causing the particle to move up to exactly the plane of the face and no further. If the particle is found to be violating the boundary of more than one face it will eventually receive the sum of the response vectors from each of such faces.

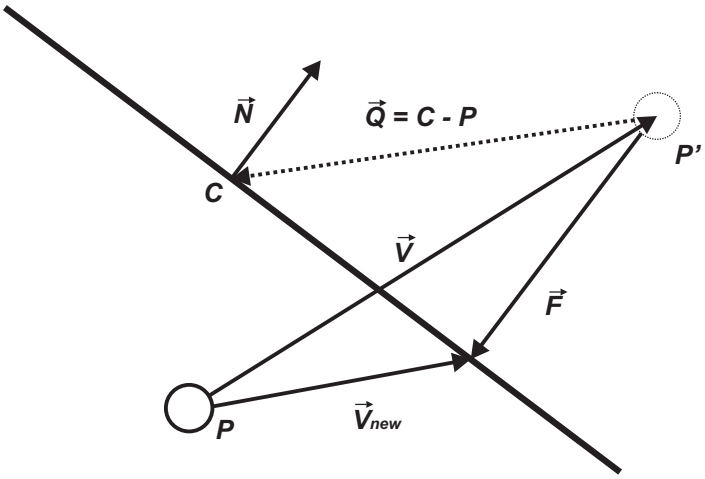


Figure 35. Vector calculations used to detect collisions with boundaries and determine the response once a collision has occurred.

The method described above works effectively for the cube, convex three-dimensional triangle-face meshes and the sphere after some slight adaptations. For the case of concave three-dimensional triangle-face meshes however, some additional calculations were required that were computationally expensive. These considered the relative proximity of the particle to the faces among other things.

I.6. Lo increase/decrease Automation

At each iteration the spring ideal length l_0 is increased or decreased by a small amount (l_0**_step**=0.1) and thus so is the spring creation threshold **D**, as it is a function of l_0. The amount of l_0**_step** was chosen on a basis that it was low enough to give the system enough time to self-organise between adjustments, but also not low enough to cause an excessively long algorithm run time. As all nodes are required to be connected into the final topology and in fact with at least three springs, so that they can be sufficiently constrained in 3-space (have zero degrees of freedom), the value of l_0 is incremented until this occurs. At the same time, the algorithm does not allow any particle to be connected with more than 12 springs and it proceeds to a decrement of l_0 when at least one particle is found with a valence of more than 12 and until no particles have a higher valence than that. If all nodes have a valence between 3 and 12 and at least one particle is moving, the algorithm increases the value of l_0, so that there are more chances for the particle system to expand up to the boundary and sufficiently describe it. If it is found that at the same time at least one particle has a valence of less than 3 and at least another has a valence of more than 12, then the system uses a bigger l_0**_step**, chosen randomly within a certain range. This is to avoid situations where the algorithm is jammed in a continuous l_0 increase/decrease that doesn't alter the overall state of the system. If all nodes have a valence between 3 and 12

and the velocities of all particles are very small (|V|<0.01), then the algorithm considers the system to be in equilibrium.

I.7. Compression process

The compression process begins when the system reaches an equilibrium state of valence between 3 and 12 and negligible velocities. A virtual pressure pad starting from the upper side of the cube is then incrementally lowered at a steady pace similarly to the l_0**_step**, until it reaches half the height of the cube., At that point, the algorithm waits again until all particle velocities have become negligible, after which it raises the pressure pad again at the same steady pace, while the particle system relaxes back to fill the entirety of the cube. When a new equilibrium is reached, the compression process is repeated. This continues until a total of three compressions have been applied.

I.8. Valence Distribution

Since all valences range from 3 to 12, we may consider a certain scale inherent to the system to facilitate the discussion of the results. The valences at 3,4 and 5 can be considered to be low, the valences of 6,7,8 and 9 can be considered to be mid-ranged and the valences of 10,11 and 12 can be considered to be high.

By examining the valence distribution in the engineered topologies, it can be seen that the FCC and HCP topologies have a more uneven distribution of nodes than the SCP. In the case of the FCC topology (Figure 36), when the sample includes lower number of nodes the percentage of nodes with a low-level valence is higher. When the sample's number of nodes increase the percentage of nodes at the low-valence area decreases. The opposite holds for the high-valence area. For lower node number samples, the percentage of nodes at the high-valence area is low and when the sample includes more nodes, the system's nodes with higher valence increase. There also seems to be a constant aggregation of nodes at the mid-level valence area for samples of all node numbers.
The HCP topology appears to have a single identifiable peak close to the low-level valence area for the samples with lower numbers of nodes that seems to shift towards the high-valence area as the sample's nodes increase. Halfway through this shift there are states with a more even distribution of valences that peaks marginally at the mid-level valence area (Figure 36).

The valence distribution plot for the SCP topology seems to be contained in a bell-curve, whose peak shifts from areas of low-level valence to mid-level ones as samples' number of nodes

increases. Of course the SCP can only have a maximum valence of 6 and it is only expectable that as the number of nodes in the topology increases the peak of the bell-curve valence distribution diagram will converge more to the valence of 6, as the total number of nodes at the boundaries of the cube with lower valence will decrease in proportion to the ones at the inner parts of the topology which have a valence of 6 (Figure 36).

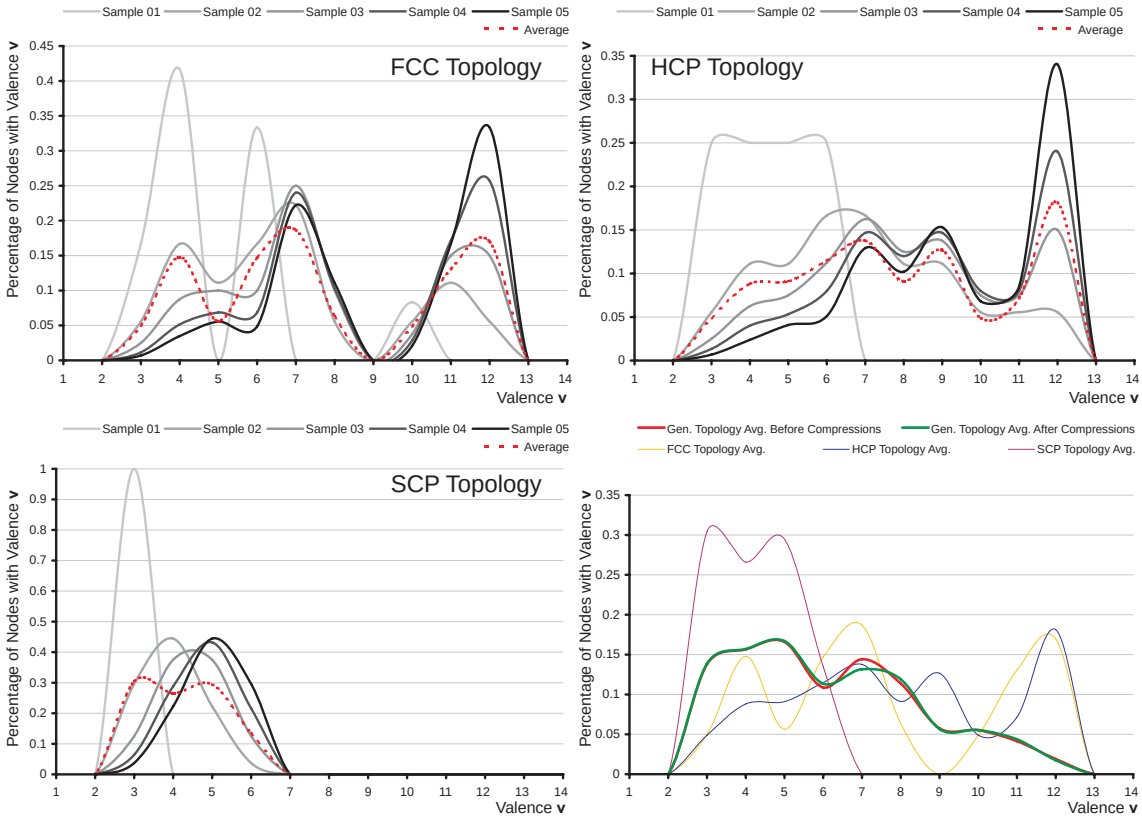


Figure 36. Valence distribution of engineered topologies.
Top Left: FCC Topology valence distribution.
Top Right: HCP Topology valence distribution.
Bottom Left: HCP Topology valence distribution.
Bottom Right: Plot of average valence distributions of engineered topologies and average valence distributions of generated topologies in the before and after compression state.

The generated topologies seem to have an uneven distribution that doesn't appear to yield some global recognisable pattern on first inspection (Figure21 and 37). Topologies with lower numbers of nodes have a distinct peak at the low-valence area, whereas the high–valence areas seem to be disfavoured in overall for all samples. This is not the case even for samples with high numbers of nodes that were expected to have higher percentages of nodes with higher valences. This may indicate the presence of inefficiency on the part of the generative algorithm, as it appears that it didn't allow enough nodes to gain higher valences in overall. Another explanation might be that the number of nodes used in all generated samples were below the threshold of better sphere packing by the FCC/HCP topologies and thus the algorithm might

have in essence applied permutations the SCP topology finding it to be more often advantageous in filling the given envelope. However, the latter is not likely to be the case, as the inefficiency of the algorithm to produce topologies with a higher percentage of high valence nodes can also be confirmed by the reduced number of established connections in comparison to the FCC/HCP topologies (Figure 16).

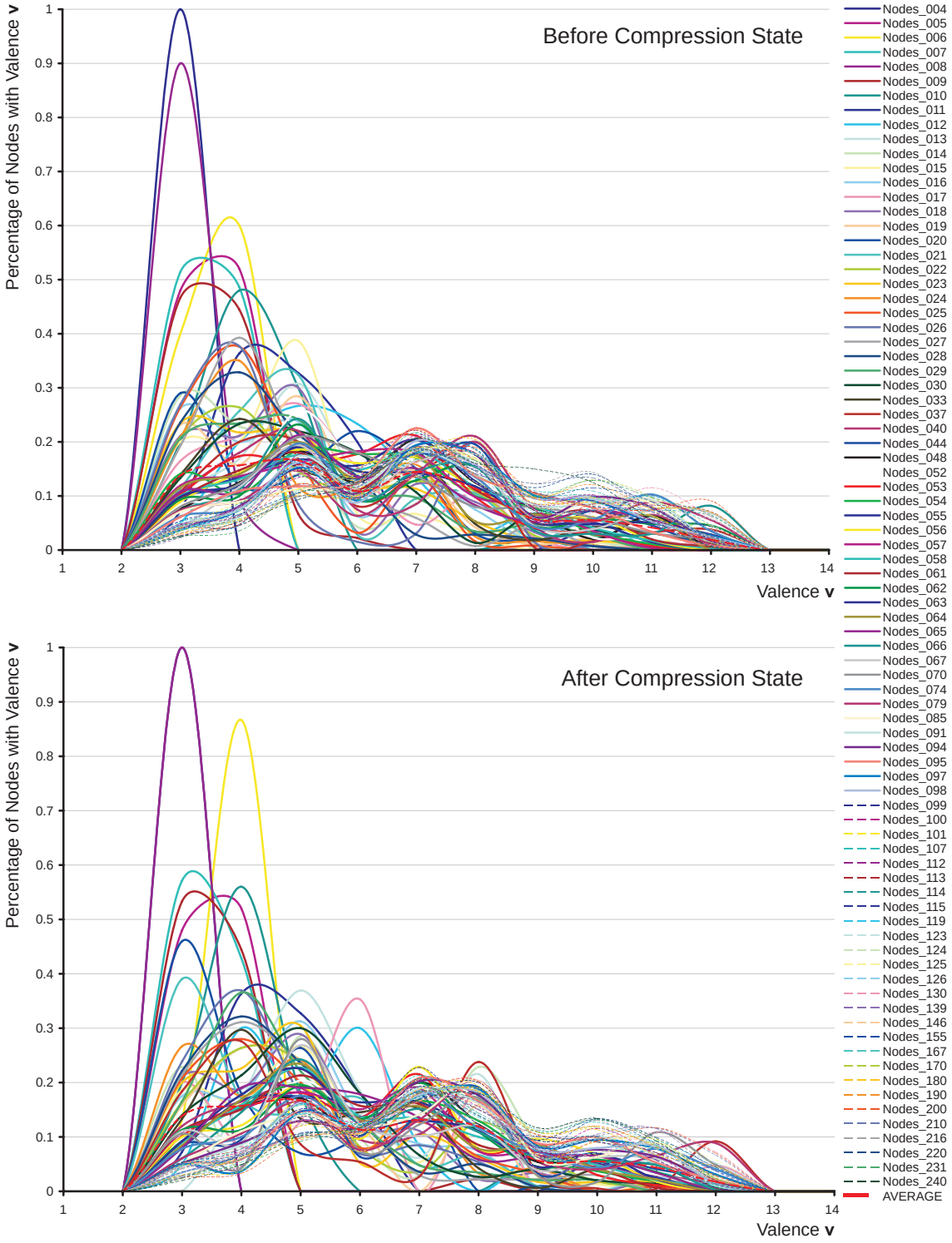


Figure 37. Valence distribution among nodes of generated samples in the before and after compression state.

By examining the valence distribution diagrams in groups of samples (Figure 38), the topologies with less than 15 nodes seem to have their single or most identifiable peak at the low-level valence area. This is to be expected, as for samples with a low count of nodes, all of the nodes are on the periphery and do not connect to many others. The highest peaks seem to shift to the higher-level valence area as the number of nodes increases. The samples with numbers of nodes close to 27 seem to have similar valence distribution diagrams as the SCP, which is confirmed by the fact that the algorithm has generated almost entirely orthogonal lattices at that range. For numbers of nodes greater than 40 the valence distribution diagrams retain some persistent characteristics. Specifically, they seem to encompass three peaks. The peak in the middle seems to persistently remain close to valence 7. For numbers of nodes from 40 to 74, only two of the peaks are more distinguishable, one close to valence 5 and one close to valence 7. Close to 40 nodes, the peak at valence 5 is higher, but as the node number increases towards 74 the peak at valence 7 becomes higher. At this range a much lower third peak close to valences 9 and 10 also exists.

Past 67 nodes a persistent high peak exists between valences 7 and 8 along with two lower peaks towards the lower and higher-level valence areas. In respect of the two lower peaks, the one at the area of 5 is higher than the one at the area of valence 10-11 for samples with nodes closer to 67, whereas the opposite holds for samples with larger numbers of nodes.

The samples with 4 to 15 nodes seem to have varied valence distribution diagrams, while some groups exist whose members share similar valence distribution graphs. These are approximately for nodes 15 to 25, 25 to 40, 40 to 67 and 67 to 240. The last two groups appear to be the most consistent as the valence distribution diagrams remain almost the same and the increase in nodes affects almost only the magnitude of the formed diagram peaks.

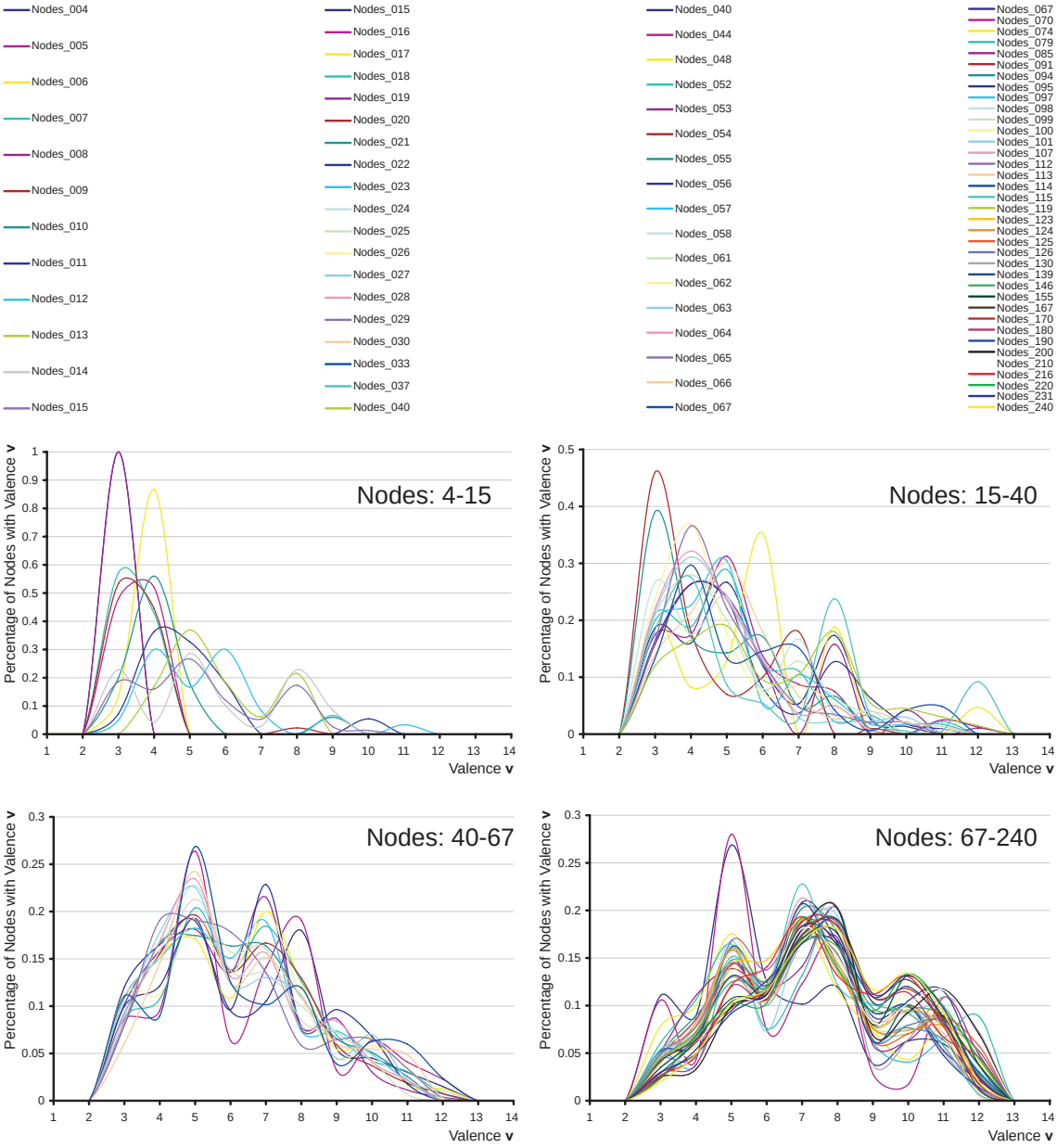


Figure 38. Valence distribution among nodes of generated samples in the after compression state divided into groups of samples with similar distribution graphs.

I.9. Average Valence diagram

In the average valence plot of figure 24, it can be seen that for 8 nodes the average valence is exactly the same as the SCP topology, which is confirmed by the fact that in all five runs for 6 particles, the topologies generated always converged to the exact topology of the cube. Regarding the fluctuations of the trend curve, one of the greatest local minima is found at the area close to 27 nodes, where the trend curve makes a big drop and gets very close to the SCP topology, which is confirmed by the fact that the algorithm produced almost entirely orthogonal topologies at that area. For node numbers close to 45 and 48, the trend curve approaches more towards the FCC/HCP topologies. At this range the FCC/HCP topologies were capable of filling

the cubic volume. For node numbers close to 64, the curve approaches the SCP topology. It should be noted that with 64 vertices, the SCP topology is able to fill the cube exactly. A similar pattern continues for the samples with more nodes, but with less magnitude. Thus, at 96, 100 and 200 nodes the curve approaches more towards the FCC/HCP topologies, while for 125 and 216 nodes the curve approaches the SCP topology.

I.10. Structural Analysis Script

A key difference of the structural analysis script from the generative one was that there was not a single ideal length $\boldsymbol{l_0}$ of springs involved. Instead, the individual lengths that the springs reached at the generative process before they were exported were considered to be the new ideal stable lengths of the imported springs. Further, the spring force equation was set up differently and in this case the attraction force when the spring was being stretched was equal to the force that was exerted to the particles when the springs were being compressed. A scalar value of the spring force at 1/12 was implemented as 12 was the maximum valence that was allowed for any node at any topology to obtain 12 ($\boldsymbol{k_{attract}} = \boldsymbol{k_{repel}} = \boldsymbol{k}$ = **1/maximum_valence = 1/12**). This was enough for the algorithm to behave stably and was the same for all tests performed to ensure the equal strength of all springs. Imported springs were not pre-determined as performing only in tension or compression, but were considered to be pin-jointed struts that can equally sustain tensile and compressive forces. In addition, during the run of the structural analysis algorithm, the velocities that the particles reached at the end of each iteration were discarded in the next and were re-calculated. In the structural analysis script the bounding envelope of the cube was removed and only a horizontal plane at the height of the lowest node of each topology served as a boundary collision limit that provided the resistance to the weight of the structure. Finally, a small amount of gravity was applied to all nodes at each iteration during the run of the algorithm. This caused particles to move vertically and springs being deformed past their new individual ideal lengths. The application of gravity lasted for 1,000 iterations, after which the deformation of springs and their tensile or compressive behaviour were recorded as well as the displacement of nodes.

The strain of connection members was calculated as the percentage of the deviation from the original length at the end of the gravity application. The displacement of each node was also calculated and then the summed total displacement was divided by the number of nodes in each topology to get the average displacement. In addition, the displacement of the centroid of each structure was also measured. The results from the five runs from each number of nodes were averaged out to get the mean strain and displacements for each population of a specific number of nodes.

A problem that was presented when considering the experiment of testing the structural aspects of the generated topologies relates to the measurements of displacements. In some cases, the generative algorithm didn't manage to produce lattices that filled the entire volume of the cube. This is more evident in the cases with fewer numbers of nodes, where there was a higher chance of the resulting lattice floating inside the volume and not forming a clear planar face touching the bottom side of the cube. In order to overcome this problem, the structural analysis algorithm was set to assume that the lowest level that provided the resistance for the structure under gravity was at the level of the lowest node found. This did not rectify the problems in all situations, as in several cases the orientation of the lattice inside the cube allowed it to perform a partial free fall or rotation when the gravity was applied. However, this problem was only found in the cases with a very small number of nodes (less than 10) and for other cases the generative algorithm was able to produce lattices with a clear base touching the bottom side of the cube and thus providing the necessary resistance to the structure's weight.

I.11. Measurement Correlations

From the scatterplot correlation matrix in figure 39 and the value matrix in figure 32, it appears that the number of nodes is the most correlated quantity with all other measurements. This can be seen as a measure of the consistency of the algorithm, as it signifies that the behaviour of the topologies generated is in most cases a function of the number of nodes used. This is also supported by the fact that another mostly correlated quantity with all other measurements is the number of connections that increases linearly with the number of nodes, as it has previously been shown.

The measurement of mean length seems to correlate fairly well with most other measurements, but this is less so when samples with smaller numbers of nodes are concerned.

The fact that the percentage of the average deviation of member lengths from the achieved mean length of the resulting topologies is poorly correlated with other measurements may indicate that it does not constitute a suitable quantity for measuring the geometrical aspects of the generated lattices. However, upon closer inspection, for higher numbers of nodes, the deviation correlates with more measurements, as can be seen from the linearly clustered samples with high numbers of nodes in the scatterplot matrix. This is once again supportive to the previous argument that the algorithm appears to perform more consistently for higher numbers of nodes.

The average valence measurement, chosen for the analysis of the topological features of the produced lattices seems to have a fairly good correlation with many of the other measurements as well. Upon closer inspection of the sample range in the scatterplot matrix, it is apparent that

also here the average valence correlates better with more measurements when samples of higher node numbers are concerned.

As far as the structural analysis is concerned, the average member strain and the average displacement of nodes appear to be as highly correlated with other measurements as the number of nodes is. However, the problems related with the specification of the structural analysis experiment and the dependence of the effect of gravity to the number of nodes have already been mentioned. Therefore, this finding can only be accepted with some due scepticism. This limitation is reinforced by the fact that the average centroid displacement measurement, which should be regarded as more reliable, is quite poorly correlated with other measurements and in fact it is poorly correlated with the very measurements of average member strain and node displacement.

The average ratio of members with compressive behaviour to members with tensile is the most poorly correlated measurement, not having a good correlation with any other measurement. This finding can't be any further interpreted, but is nonetheless presented here to preserve the completeness of the overview of the algorithm's performance.

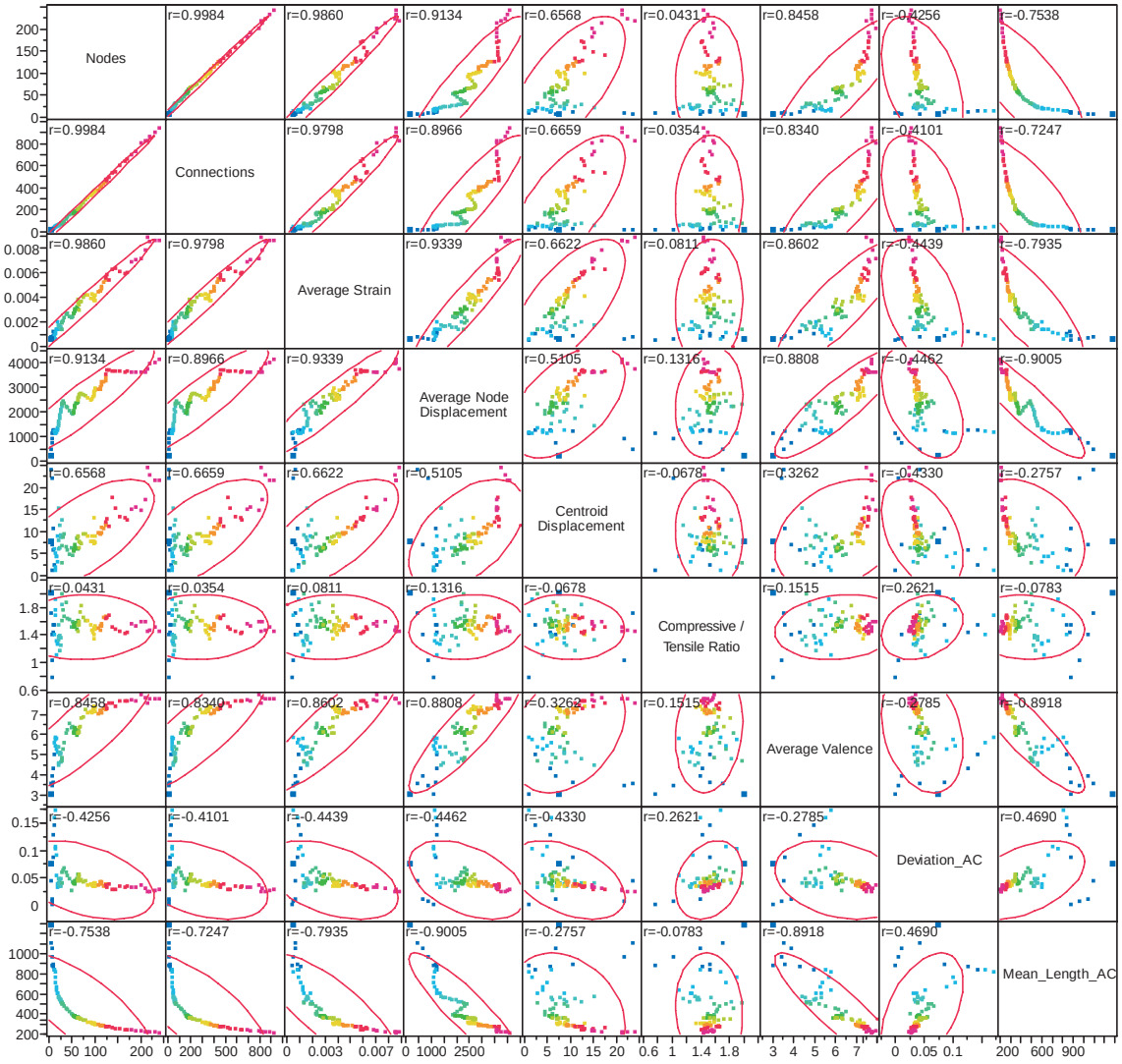


Figure 39. Scatterplot correlation matrix of measurements of the generated samples in the after compression state

I.12. Further investigations

Re-consideration of issues in the specification of physical parameters

The linear function used for the specification of the spring force could be replaced with a more smooth curved function. Preliminary experimentation with a curved function did not yield very different results, but only polynomial functions with Lagrange interpolation coefficients of up to four degrees were used, so that the curve could be fitted into specific roots derived from the system's thresholds. A parametric definition of a curve with interchangeable coefficients that can fit adjustments to the magnitude of the force and the spring creation thresholds might serve as a better function that could produce a more continuous behaviour of the system. A cubic spline interpolation might serve as such an improvement.

In addition, a more accurate physical simulation that would include accelerations, viscous drag and damping as well a more realistic collision response might be worth investigating.

Re-consideration of algorithm steps and termination

The results of the final node distribution experiment that was conducted after the evaluation of the results of the formal experiments showed a promising direction for developing the algorithm. Treating geometrical and topological aspects in separate processes in the algorithm could be further explored and might require the re-consideration of several of the generative algorithm steps, such as removing the valence limits .

Furthermore, in future development, the evaluation of certain aspects of the generated topologies should be explicitly linked to the generative algorithm as it runs. The results that emerged from the experiments have indicated the measurements that the algorithm could respond to in real-time.

Finally, the purpose of this investigation was to map the space of possible solutions of the algorithm so that suitable heuristics could be integrated in the algorithm run. The compression process was found to have a beneficial effect to the generated topologies, but perhaps others may yet prove to optimise the results even more. A possible heuristic method for arriving to more optimal solutions and in shorter times could be a simulated annealing method that could help the particle-spring system escape from local optima and find global ones.

More thorough structural analysis, perhaps using FEM

The problems inherent in the structural experiments have already been reported. A more thorough structural analysis would be required for a more reliable determination of the structural performance of the generated topologies. In addition, forces other than gravity should also be considered, such as the lateral force of wind loading. A Finite Analysis Method might also be useful and certainly more reliable than the structural script proposed here. Testing with a FEM should not prove to show many incompatibilities, as the models generated by the algorithm have a discretised simple geometry consisting only of edges meeting at certain points. A more complete structural analysis should also include embedded metadata regarding actual physical material properties of the generated structures.

Resolution of speed issues

As was seen in the review of the results, the algorithm seems to perform more consistently for higher numbers of nodes. In this context, achieving a faster performance will allow the testing of samples with higher numbers of nodes. This would also allow the investigation of sample topologies with numbers of nodes closer to the ones required by the threshold of the alteration of efficiency of FCC/HCP and SCP in terms of close packing spheres in a cubic space. The present operation of the algorithm is quite expensive computationally and can certainly benefit from improvements in terms of reducing the calculations it entails. A feature that can most likely yield improvements is the use of an Euler or a Runge-Kutta solver integration for the particle system that could advance the algorithm steps with much less computational cost and a negligible loss in precision.

Resolution of various issues relating to non-convex arbitrary mesh boundary envelopes

The preliminary testing of the algorithm on concave arbitrary mesh boundary envelopes showed some of its vulnerabilities. A more efficient collision detection method should be developed to account for the issues caused by irregular face meshes. In addition, a problem that was observed had to do with geometry bottlenecks of concave meshes, where the algorithm had difficulties in getting enough particles to pass through. Again, this problem requires some further attention to the collision detection.

Use of more than one spring ideal length and consideration of ellipsoidal packing

Recent advances in close packing studies have shown that ellipsoidal shapes yield better packing density than spherical ones (Donev et al., 2004). By using this principle for the behaviour of the particle-spring system, topologies based on ellipsoidal packing could be created and investigated.

Optimisation of node angles

The whole approach pre-supposes that the edges of the generated topologies have an acceptable deviation from a single ideal length. In this respect, they are less advantageous than the engineered topologies that have a single ideal length for all members. However, it is this feature of "fuzziness" that allows the proposed method to fill a certain space with any given number of nodes that is not subject to the grid constraints of the engineered topologies. Furthermore, it is achievable construction-wise due to the progress of mass customisation manufacturing techniques.

Another issue that the generated topologies are less advantageous in when compared to the engineered is the individual nature of the nodes. As the algorithm considers the structure to be pin-jointed, the edges get attached to nodes at arbitrary angles. One of the most costly features of space frames the construction detailing of the nodes. Having each node individually customised to be able to receive struts at the angles specified by the generative algorithm would greatly increase the cost of the structure. Therefore, an issue that could be taken into account in future development of the proposed method would be to use constraints on the angles at which springs can be attached to particles during the generative process. Minimising the angles at which struts get attached to nodes would lead to a better standardisation of the nodes and would have a very beneficial effect to the overall cost.

Use higher dimensions in vector specification and research the possibility of the production of quasi-crystal topologies

Experimentation with higher dimensional vectors of particles might lead to the system being able to generate quasi-crystal topologies. This would involve a rather simple adaptation, as the main algorithm body would remain the same. The main difference would be in the position and velocity vectors of particles. Spring creation and responses would take into account the multi-dimensional geometry. By having particles equally distributed in n-dimensional space, the three-dimensional projection of the generated structure would be one of a quasi-crystal topology.

Investigation of viability of the proposed method for generating space-filling tensegrity topologies

An issue worth exploring would be to allow the algorithm to determine the structural behaviour of the generated springs, so that tensegrity topologies could emerge. This direction could be pursued if suitable constraints were embedded in the algorithm that forced compressive members to be established discontinuously.

Consideration of design priorities for space-filling other than formal ones

The method for structural space-filling could be adapted so as to incorporate design considerations other than filling the volume of a specific three-dimensional form. Spatial demands and relationships requested by a design brief could provide the grounds for an investigation of fitting a structure to a topological layout of interrelated spaces.

Appendix II

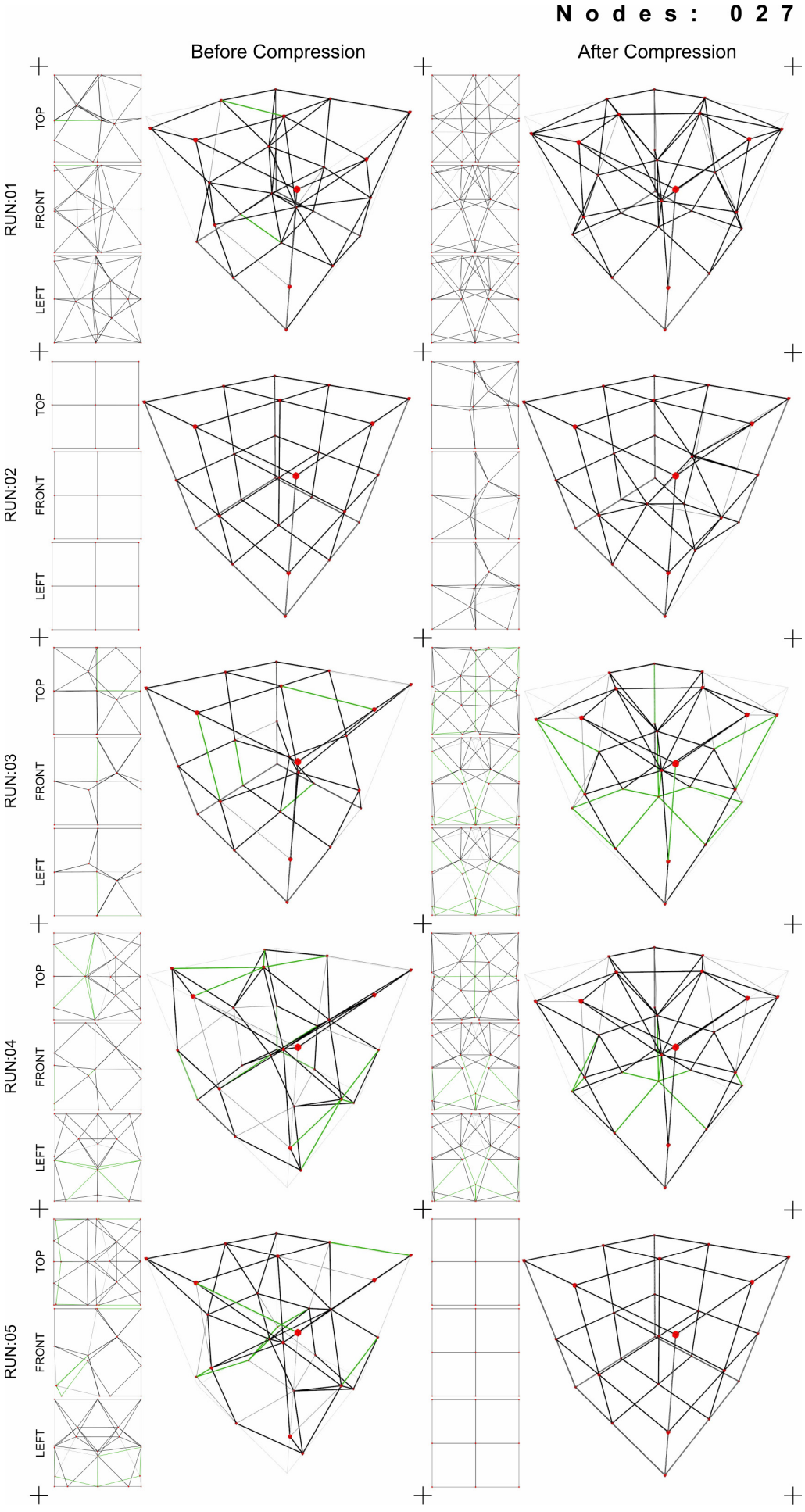
Illustrations of generated lattices

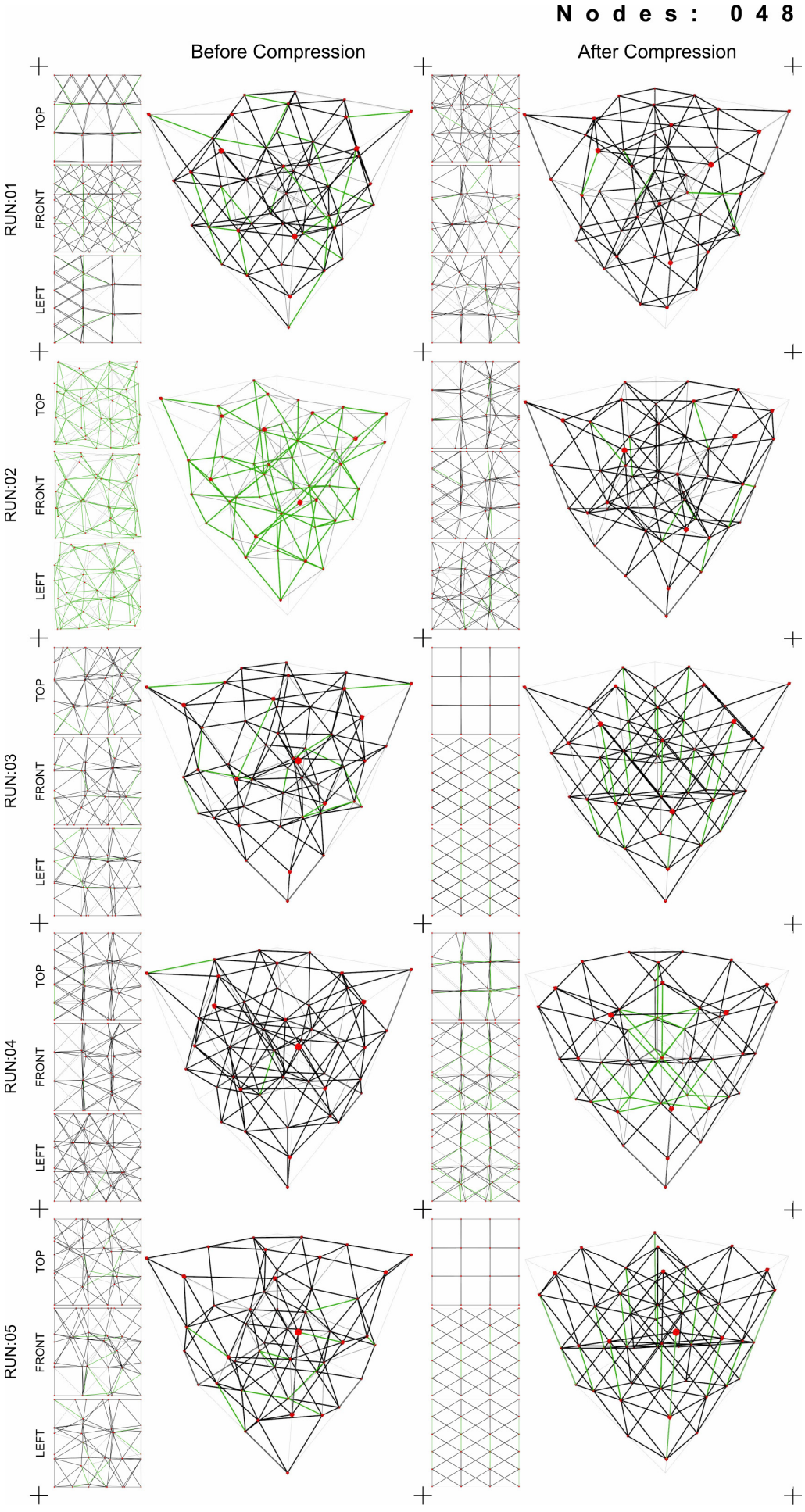
Complete set of illustrations of five population members (pages 73-77)

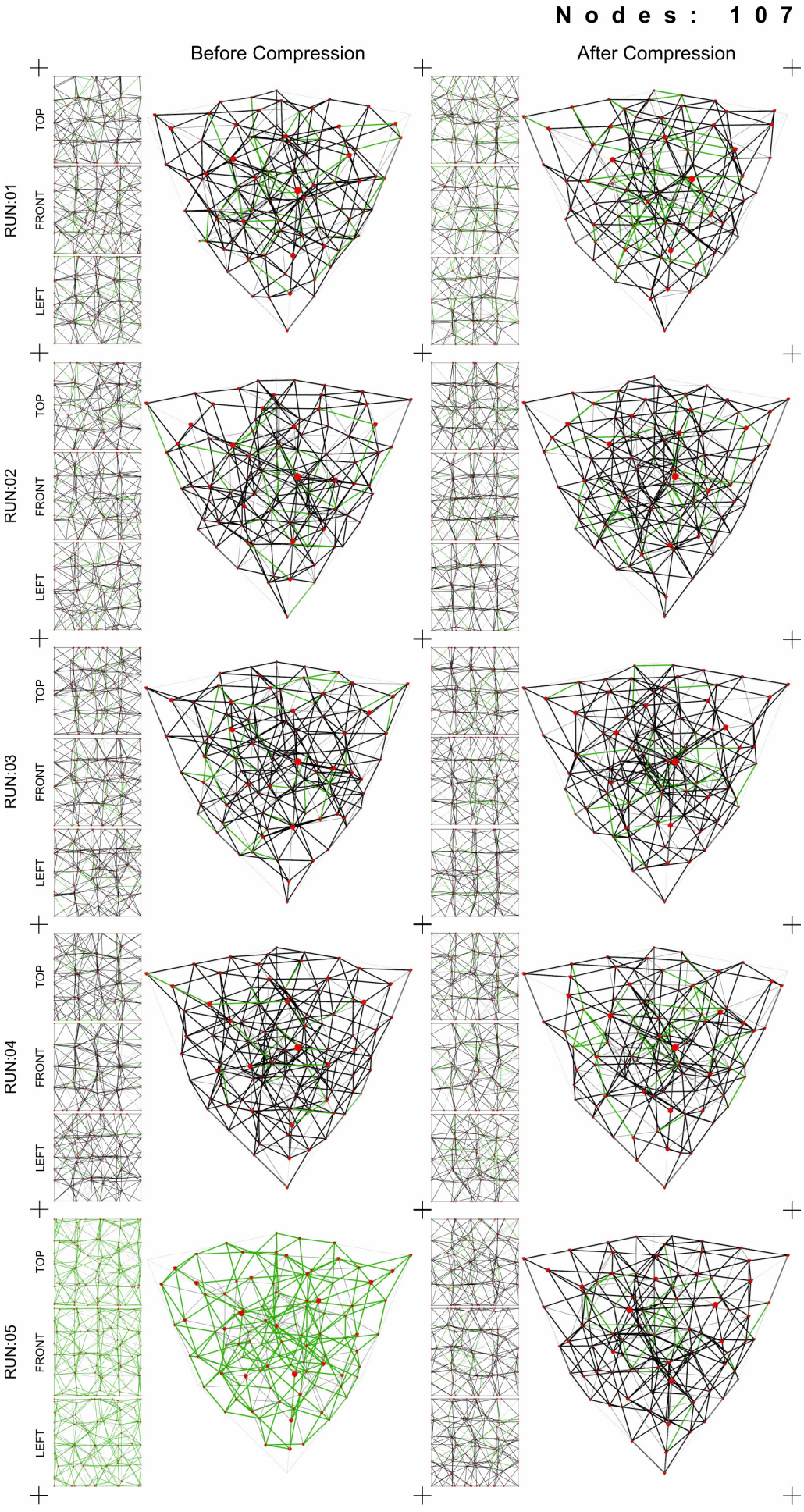
This list includes the generated lattices from the five runs of the algorithm before and after the compression process. Thick lines denote springs that are longer than the ideal spring length. Thin lines denote springs that are shorter than the ideal spring length. Green lines denote springs that are very close to the mean length of all springs.

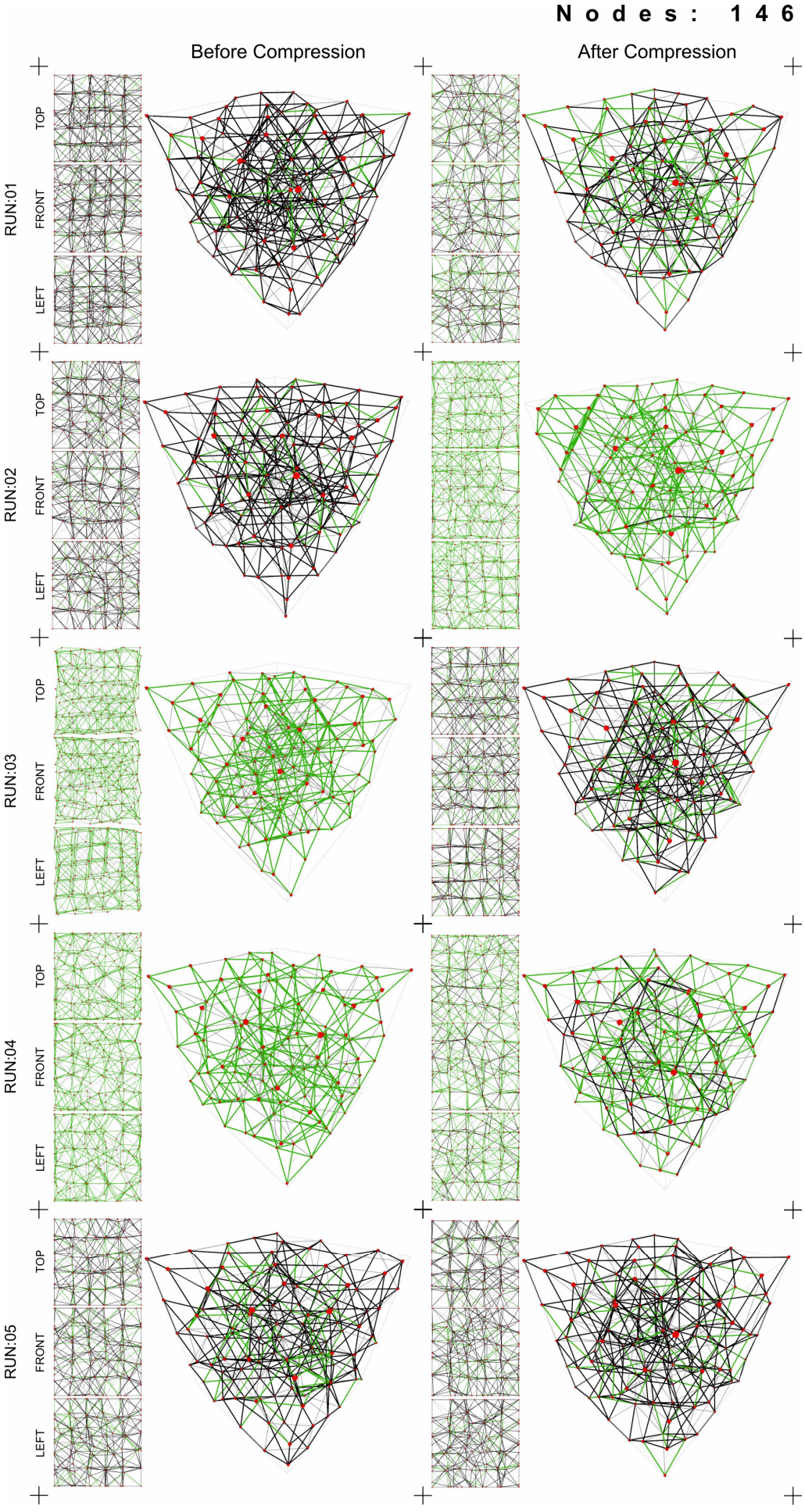
Illustration of one indicative state from all population members when stressed under gravity (pages 78-86)

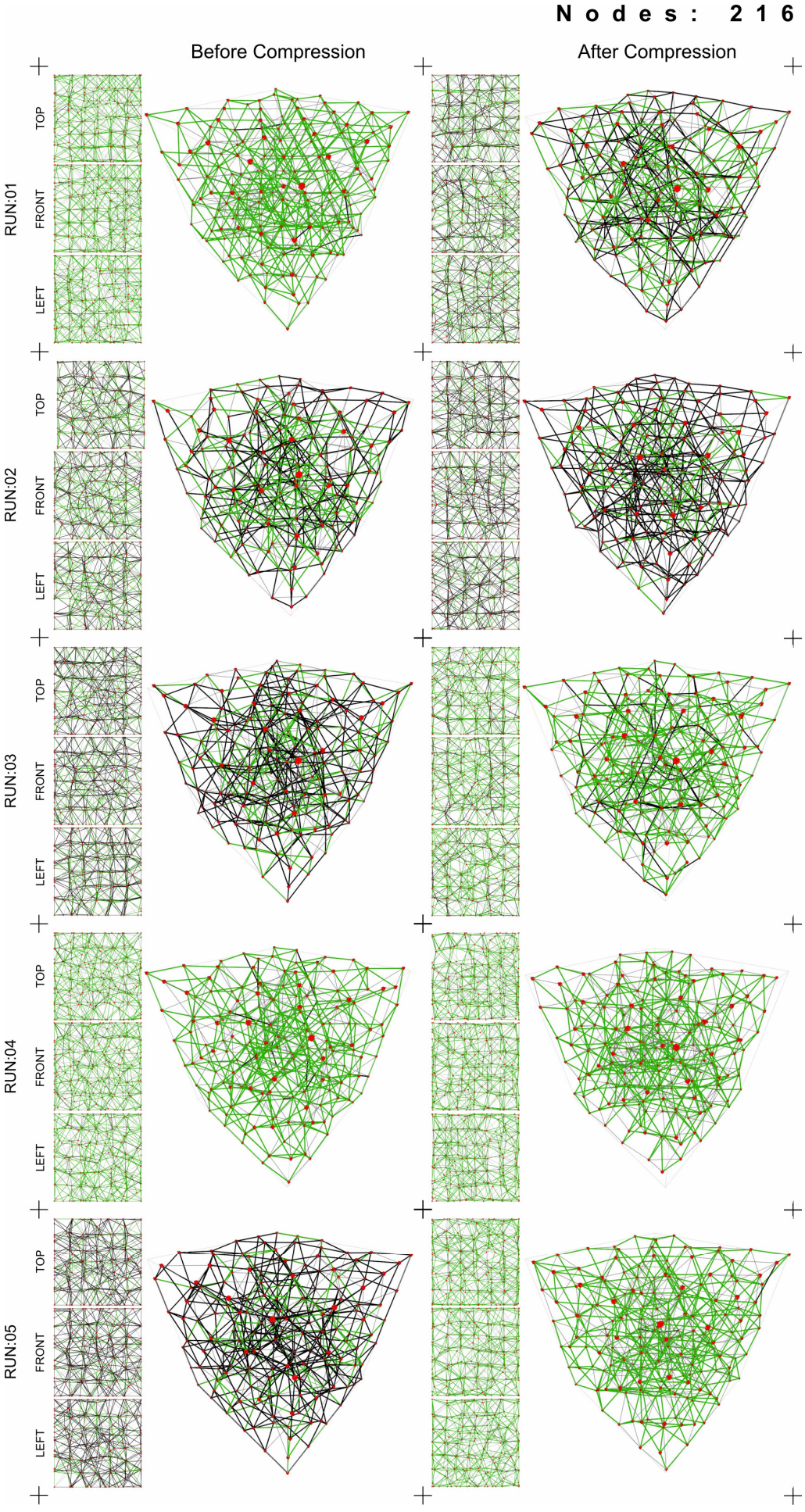
This list includes a single image from each population member. All images are from generated lattices that resulted after the compression process. They are illustrated at the state they converged to after gravity had been applied to them in the structural analysis script. Struts denote connections in compression and lines indicate connections in tension

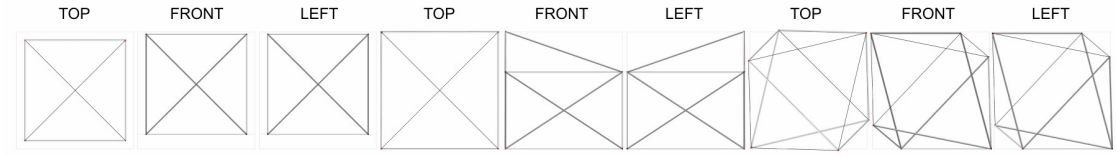

Nodes: 027
Before Compression
After Compression
RUN:01
RUN:02
RUN:03
RUN:04
RUN:05
TOP
FRONT
LEFT


Nodes: 048
Before Compression
After Compression
RUN:01
RUN:02
RUN:03
RUN:04
RUN:05
TOP
FRONT
LEFT

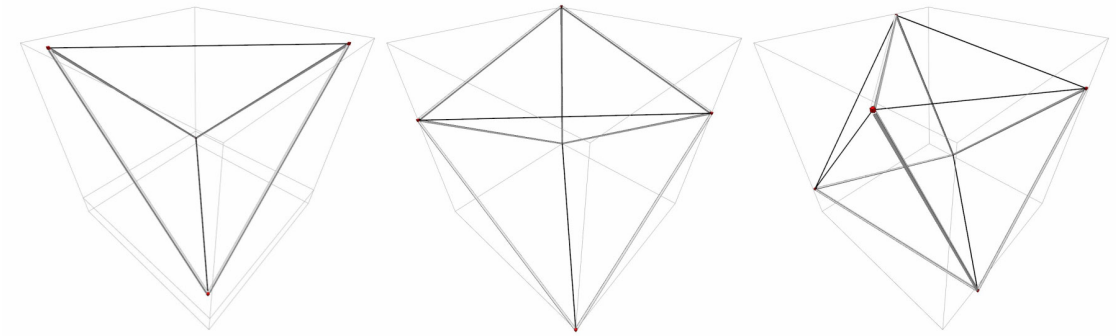

Nodes: 107
Before Compression
After Compression
RUN:01
RUN:02
RUN:03
RUN:04
RUN:05
TOP
FRONT
LEFT

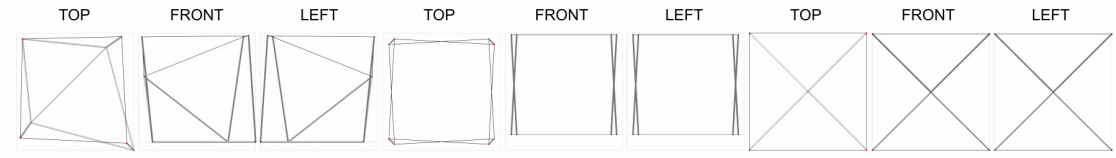

Nodes: 146
Before Compression
After Compression
RUN:01
RUN:02
RUN:03
RUN:04
RUN:05
TOP
FRONT
LEFT


Nodes: 216
Before Compression
After Compression
RUN:01
RUN:02
RUN:03
RUN:04
RUN:05
TOP
FRONT
LEFT

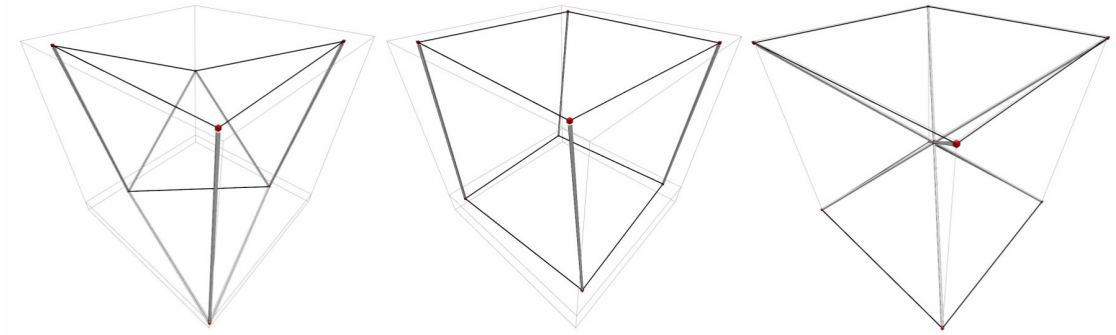


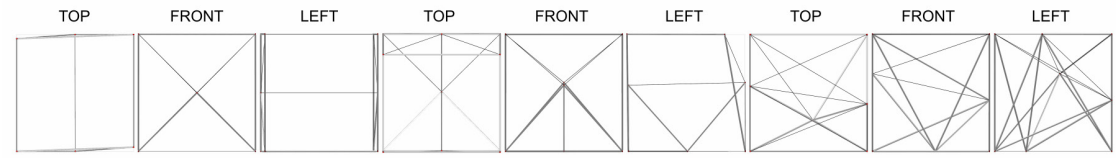
Nodes:004 / RUN:04 Nodes:005 / RUN:02 Nodes:006 / RUN:02



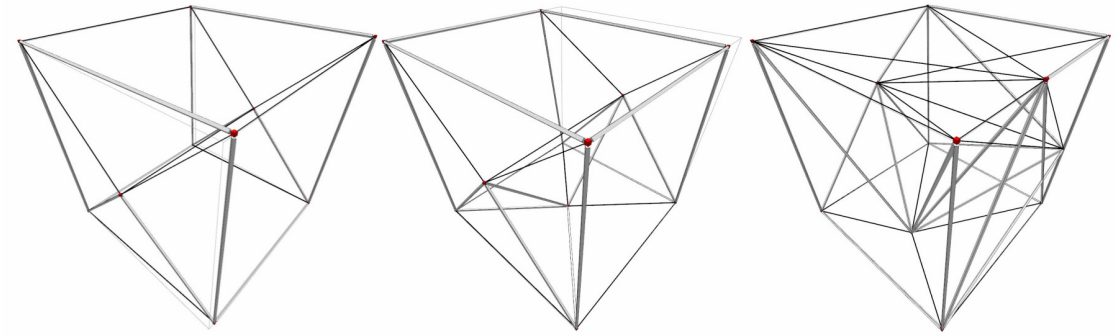


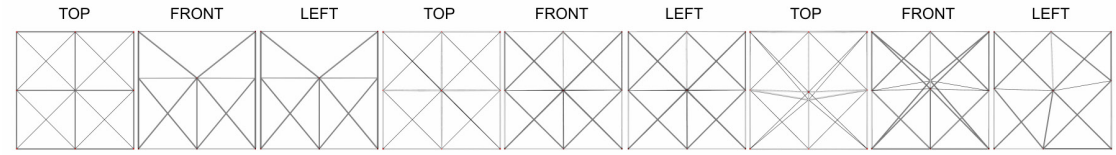
Nodes:007 / RUN:01 Nodes:008 / RUN:01 Nodes:009 / RUN:03



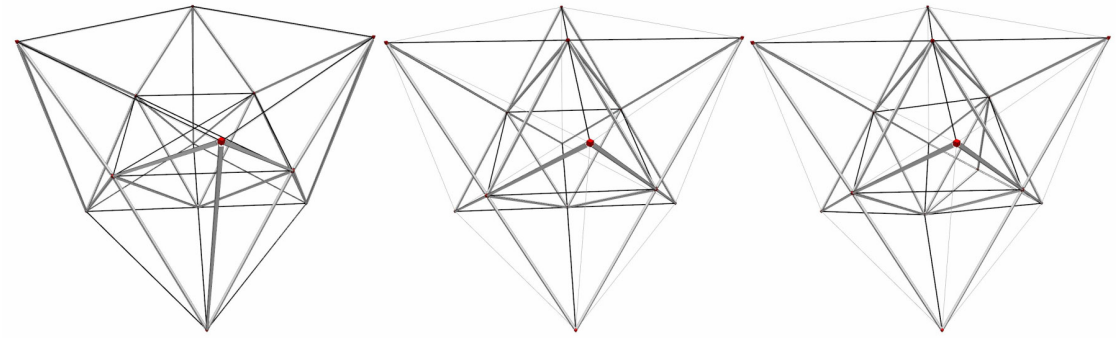


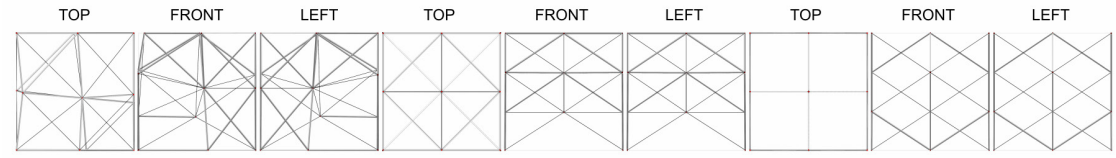
Nodes:010 / RUN:02 Nodes:011 / RUN:05 Nodes:012 / RUN:02



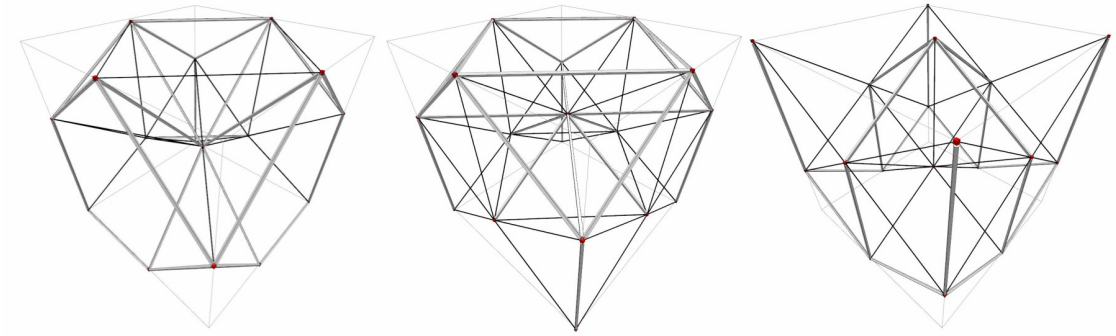


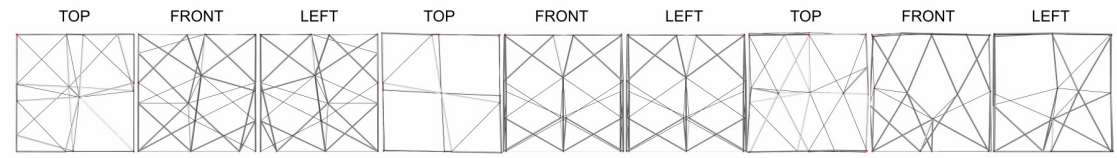
Nodes:013 / RUN:04 Nodes:014 / RUN:02 Nodes:015 / RUN:03



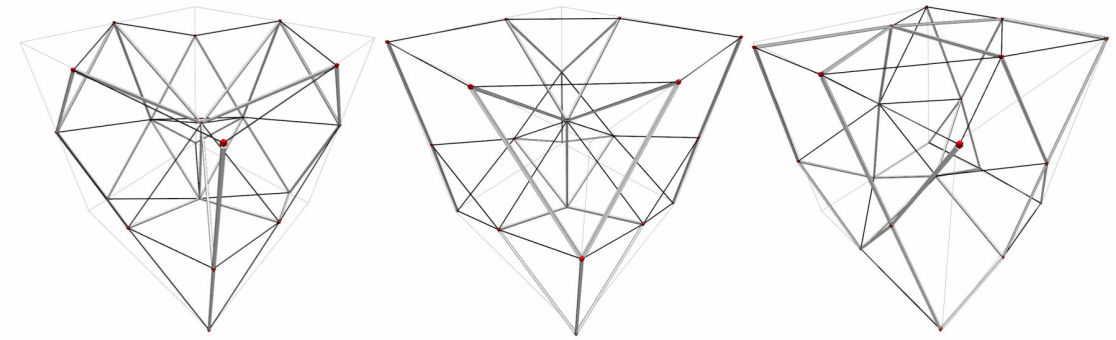


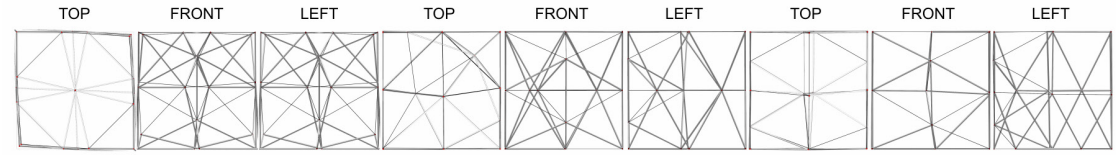
Nodes:016 / RUN:4 Nodes:017 / RUN:02 Nodes:018 / RUN:01



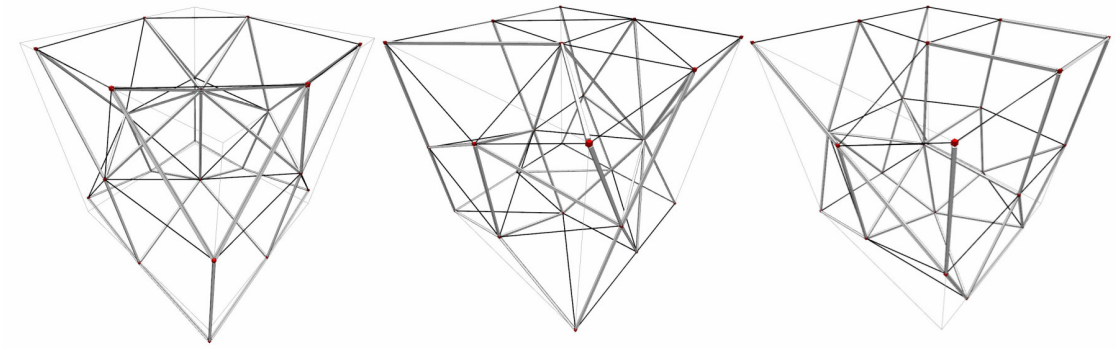


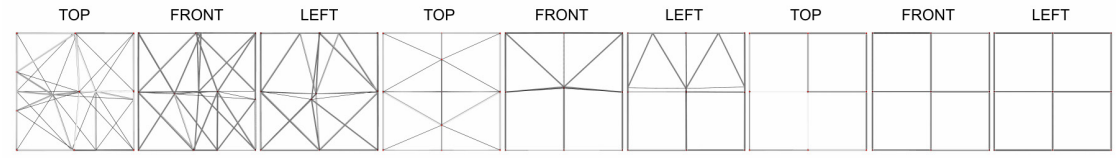
Nodes:019 / RUN:05 Nodes:020 / RUN:04 Nodes:021 / RUN:04



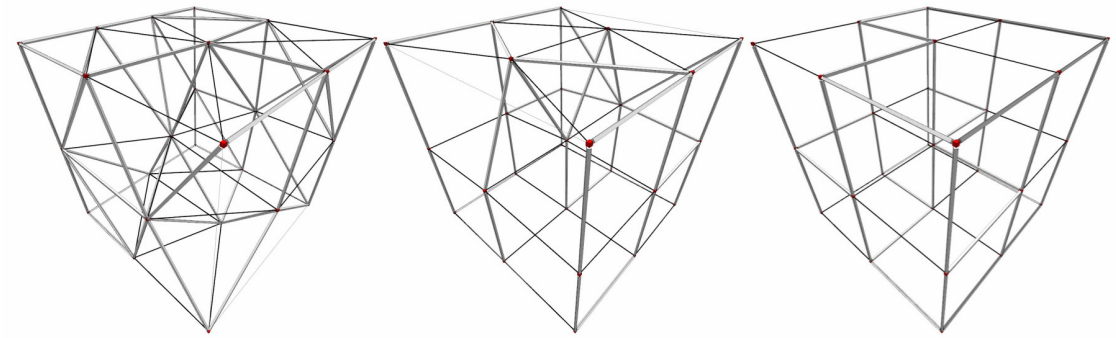


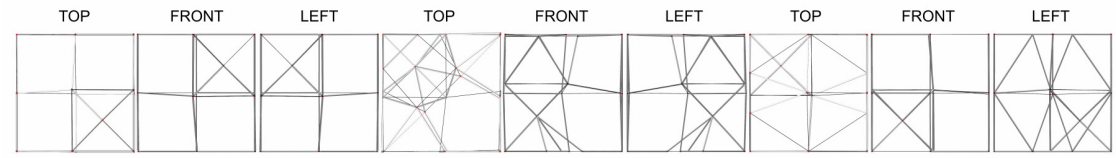
Nodes:022 / RUN:04 Nodes:023 / RUN:05 Nodes:024 / RUN:05



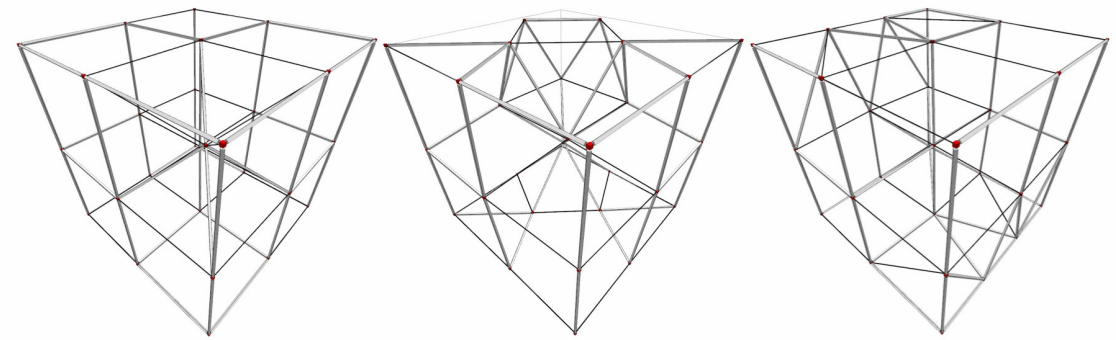


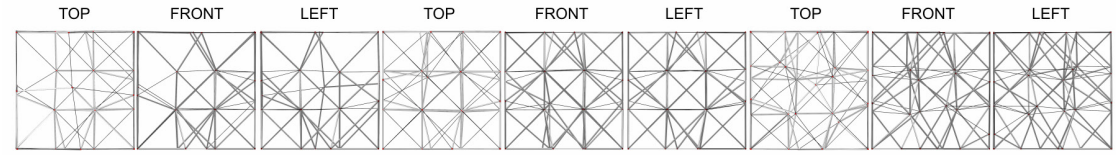
Nodes:025 / RUN:05 Nodes:026 / RUN:04 Nodes:027 / RUN:05



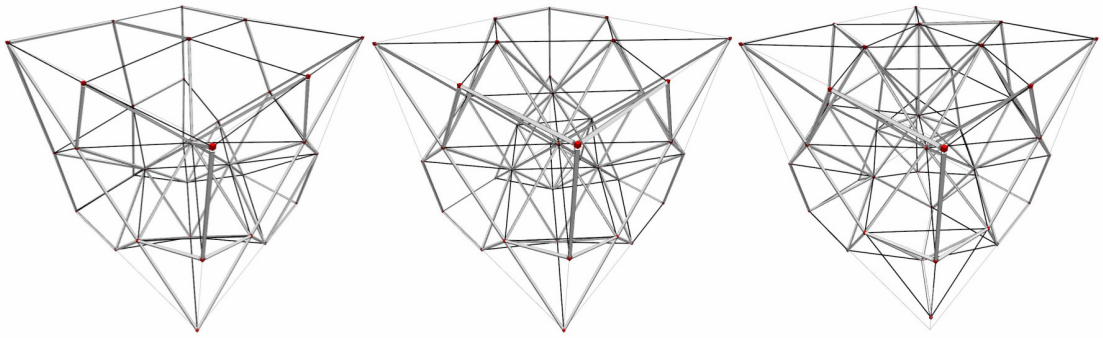


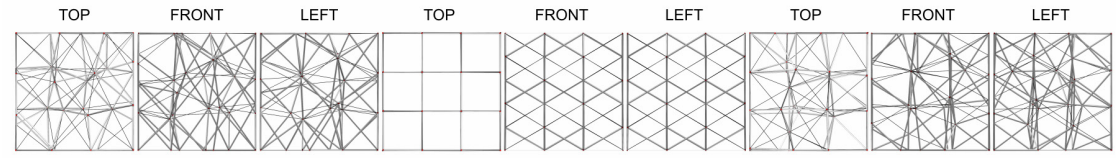
Nodes:028 / RUN:05 Nodes:029 / RUN:01 Nodes:030 / RUN:04



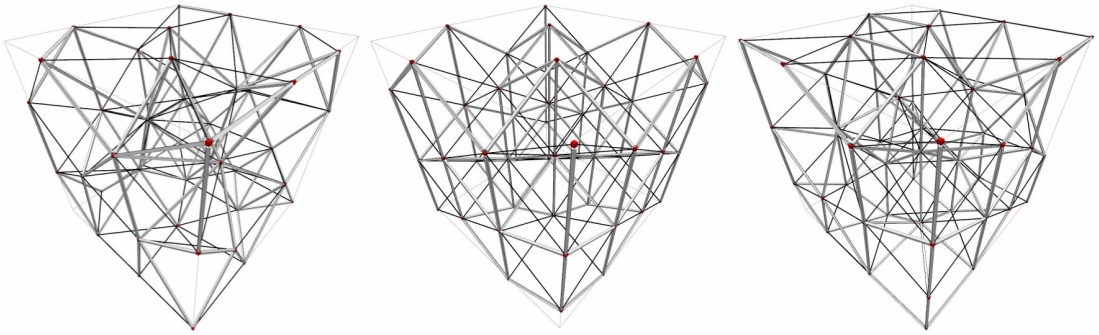


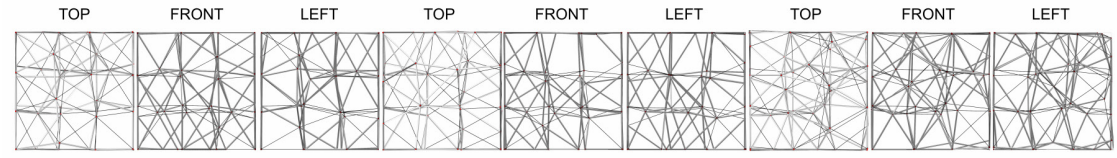
Nodes:033 / RUN:03 Nodes:037 / RUN:03 Nodes:040 / RUN:01



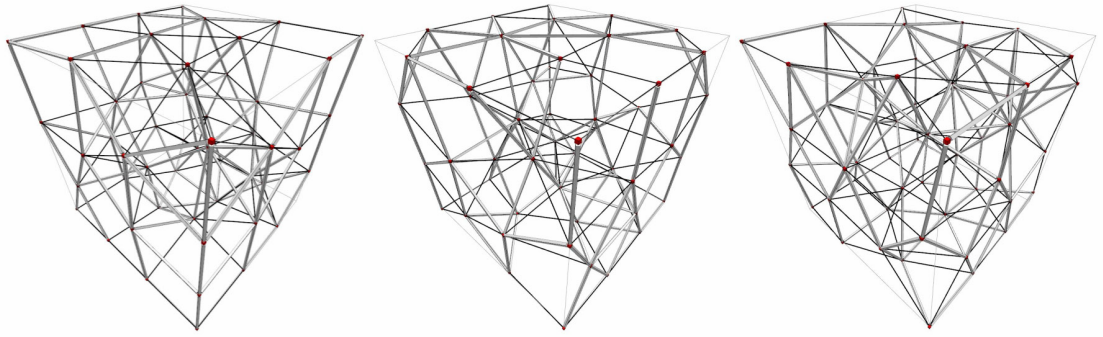


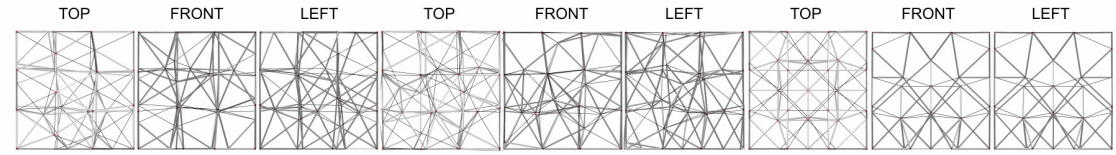
Nodes:044 / RUN:05 Nodes:048 / RUN:05 Nodes:052 / RUN:01



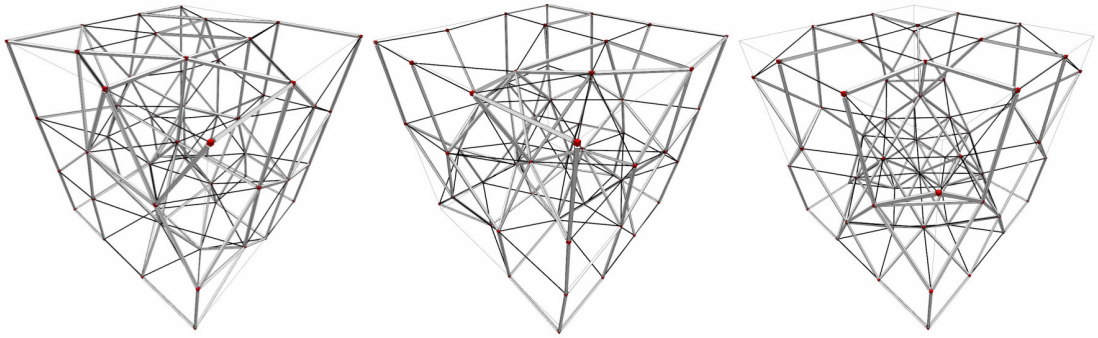


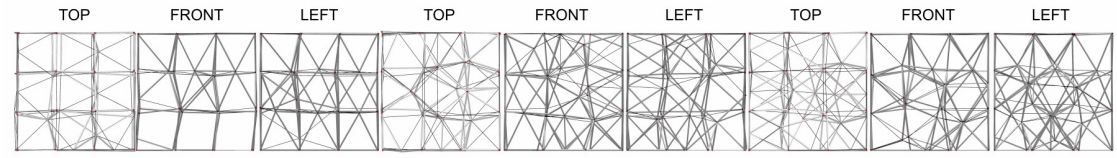
Nodes:053 / RUN:03 Nodes:054 / RUN:02 Nodes:055 / RUN:04



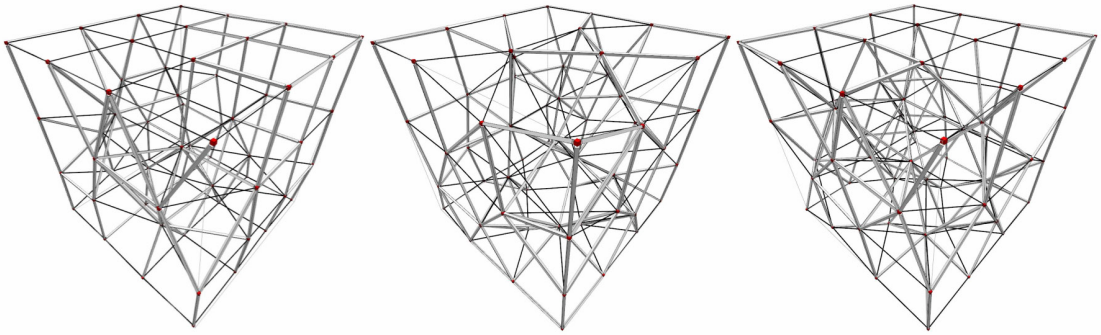


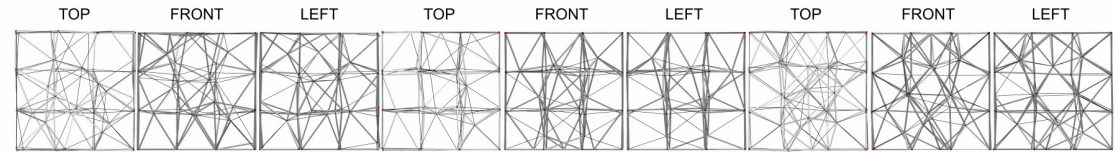
Nodes:056 / RUN:03 Nodes:057 / RUN:03 Nodes:058 / RUN:01



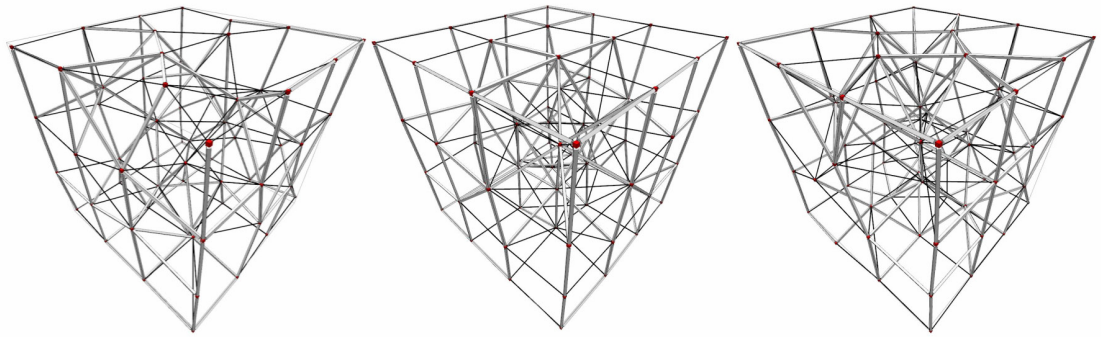


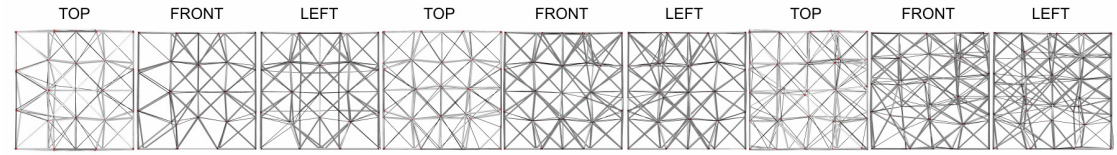
Nodes:061 / RUN:02 Nodes:062 / RUN:04 Nodes:063 / RUN:04



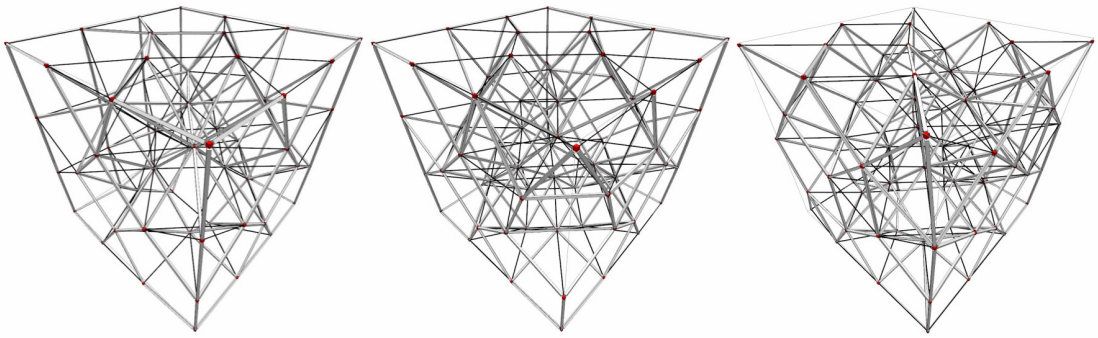


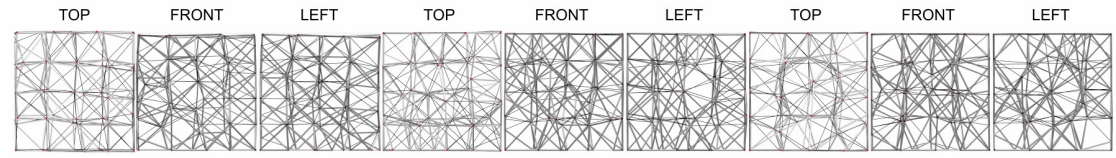
Nodes:064 / RUN:05 Nodes:065 / RUN:03 Nodes:066 / RUN:05



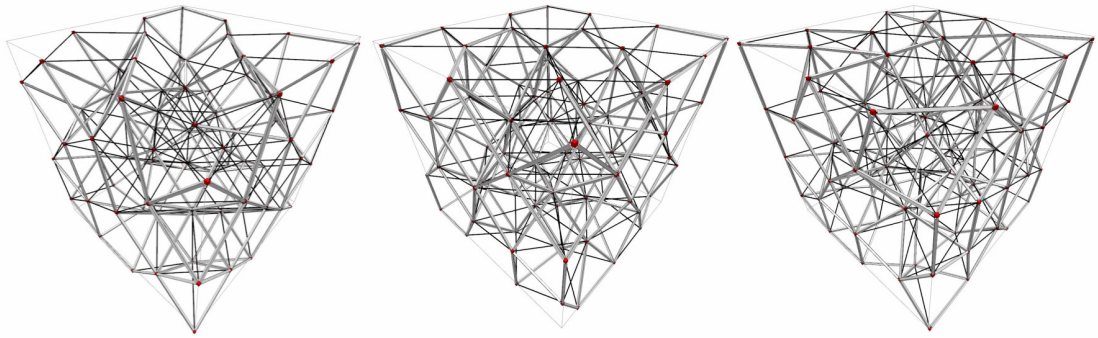


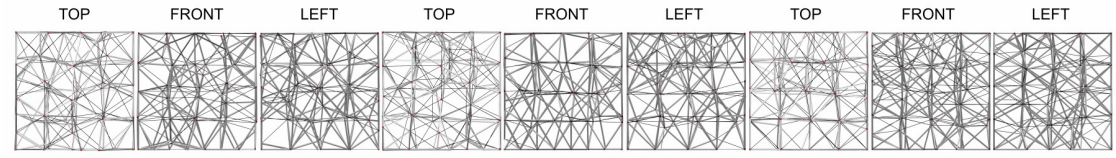
Nodes:067 / RUN:02 Nodes:070 / RUN:02 Nodes:074 / RUN:05



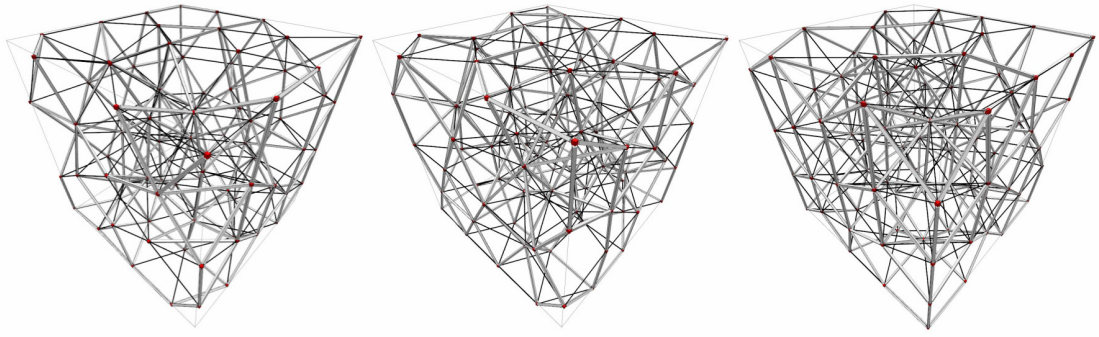


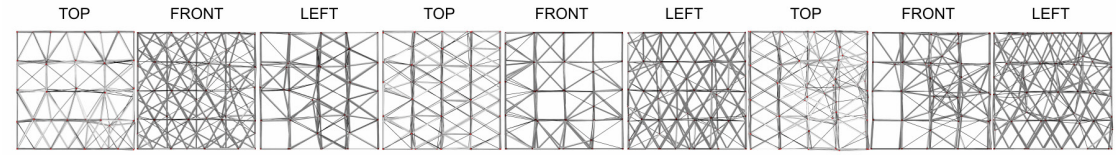
Nodes:079 / RUN:02 Nodes:085 / RUN:04 Nodes:091 / RUN:03



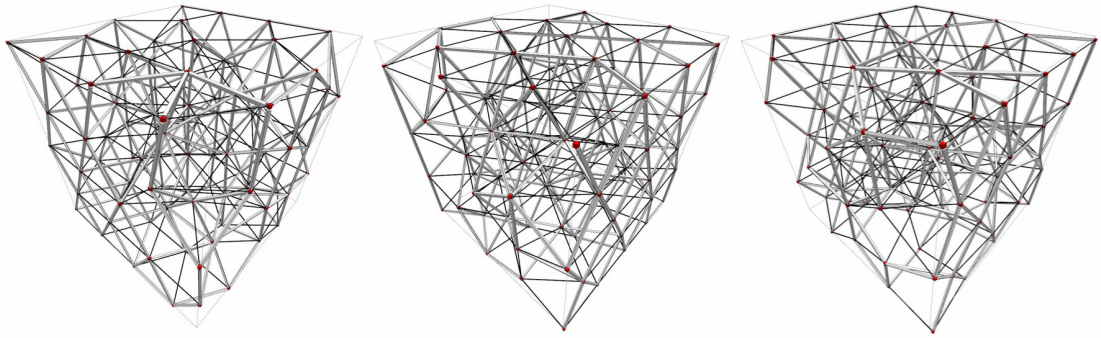


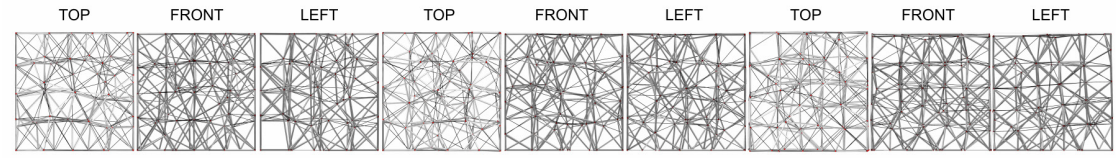
Nodes:094 / RUN:01 Nodes:095 / RUN:05 Nodes:097 / RUN:03



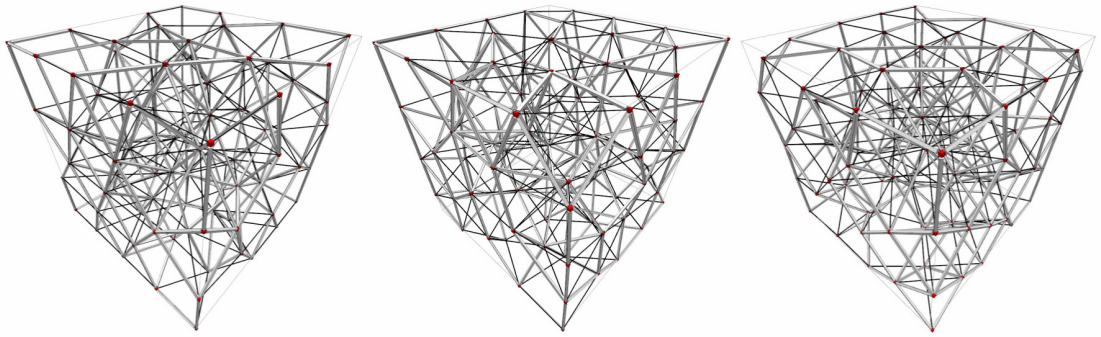


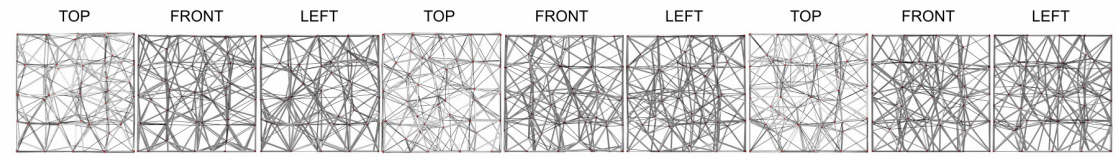
Nodes:098 / RUN:04 Nodes:099 / RUN:03 Nodes:100 / RUN:05



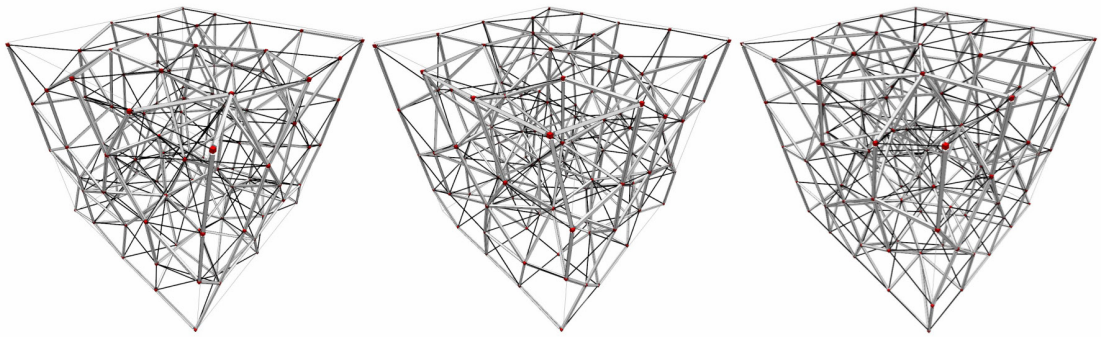


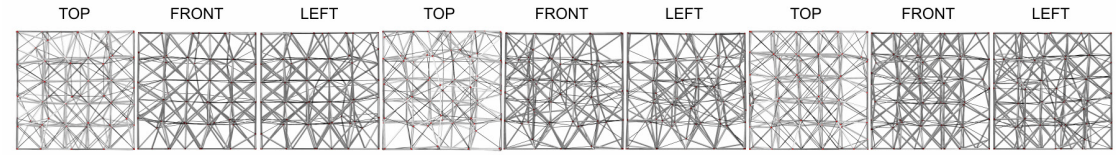
Nodes:101 / RUN:05 Nodes:107 / RUN:05 Nodes:112 / RUN:01



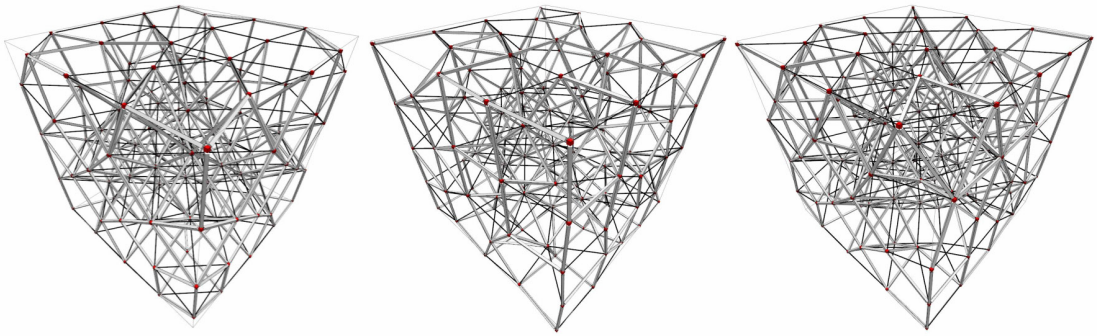


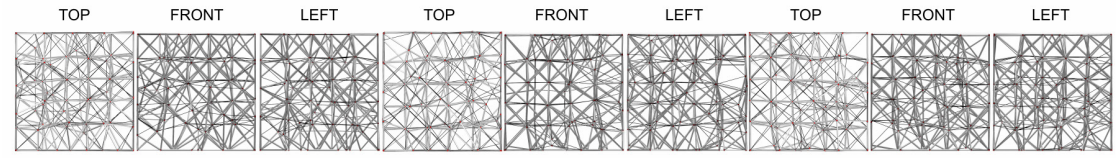
Nodes:113 / RUN:01 Nodes:114 / RUN:01 Nodes:115 / RUN:02



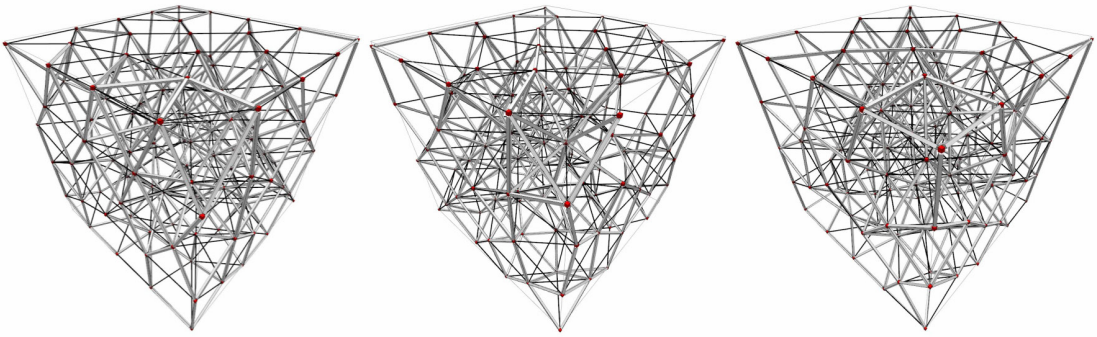


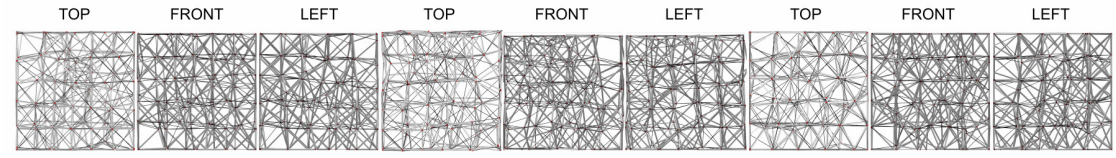
Nodes:119 / RUN:05 Nodes:123 / RUN:03 Nodes:124 / RUN:02



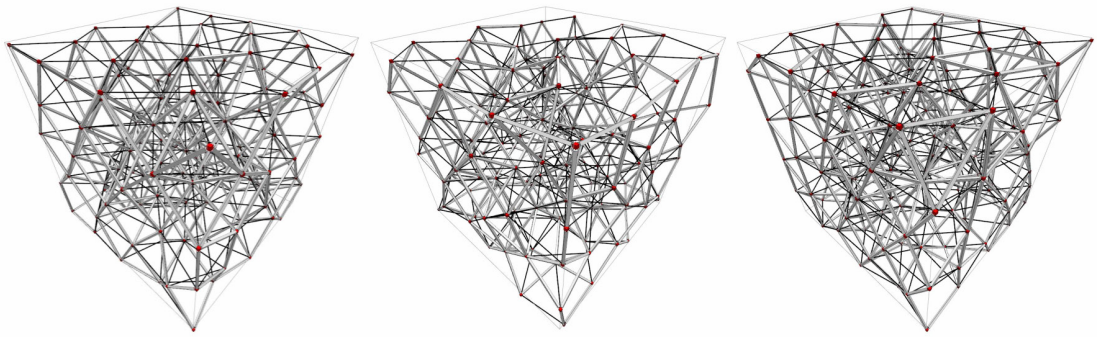


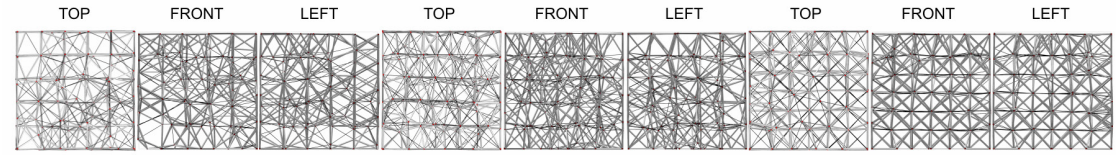
Nodes:125 / RUN:02 Nodes:126 / RUN:02 Nodes:130 / RUN:03



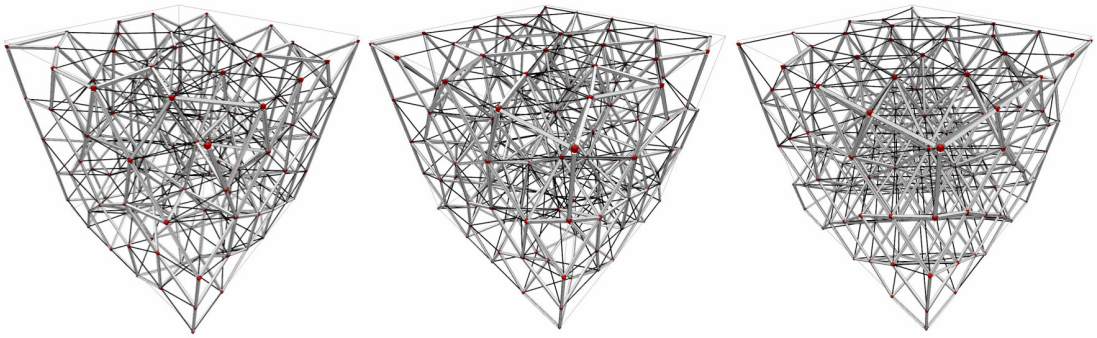


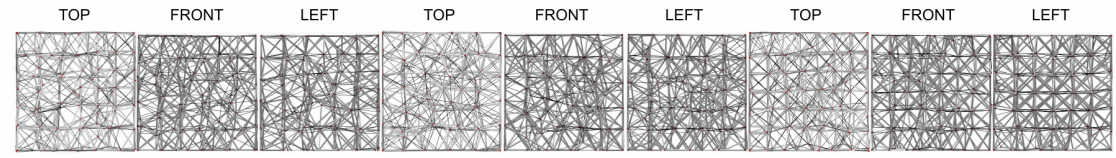
Nodes:139 / RUN:04 Nodes:146 / RUN:02 Nodes:155 / RUN:04



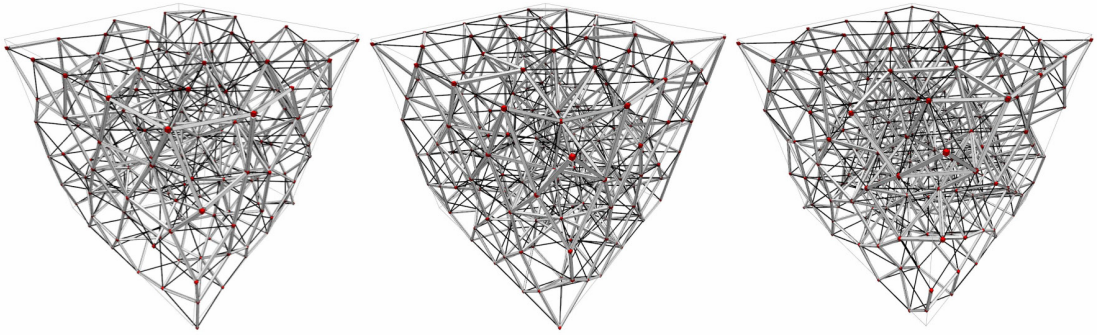


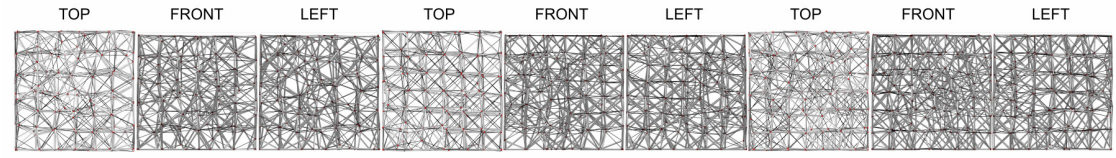
Nodes:167 / RUN:01 Nodes:170 / RUN:02 Nodes:180 / RUN:05



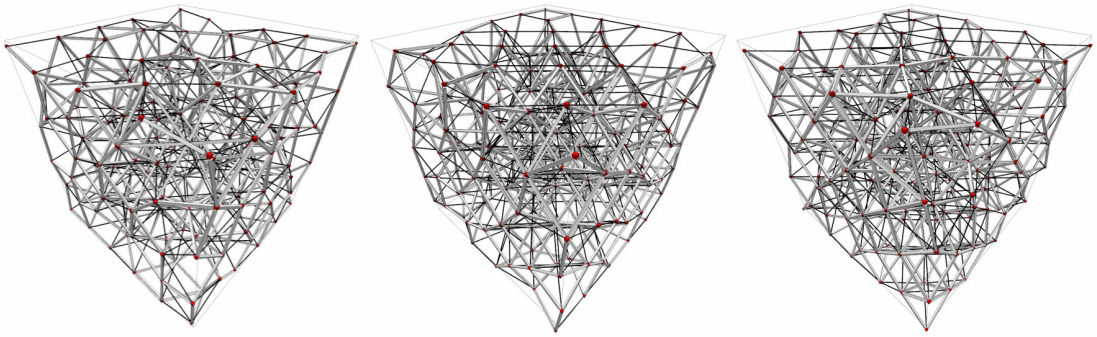


Nodes:190 / RUN:02 Nodes:200 / RUN:01 Nodes:210 / RUN:044





Nodes:216 / RUN:03 Nodes:220 / RUN:05 Nodes:231 / RUN:02



Appendix III

Pseudocode (after Processing API)

Basic functions of the particle-spring system

```
int particles = 100;
Node [] nodes= new Node[particles]; // Array of the particles
Vector springs = new Vector();
//The dynamic vector that holds the established springs
float total_val; // Total summed valence of all particles
float side = 1000f; //Side of the cube
float lo=0; // Spring Initial ideal length

void setup()
{
  for (int i=0; i<nodes.length; i++)
  {
    nodes[i] = new Node (random(-side/2, side/2), random(-side/2, side/2), random(-
side/2, side/2),0,0,0); // Create node and add it to the respective array
  }
}

void draw()
{
  springs.removeAllElements(); //delete all existing springs

  for (int i=0; i<nodes.length; i++)
  {
    nodes[i].valence=0;
  }

  for (int j=i+1; j<nodes.length ; j++)
  {
    Vec nowi = add (nodes[i].pos,nodes[i].dir);
    //Calculate temporary position of particle i
    Vec nowj = add (nodes[j].pos,nodes[j].dir);
    //Calculate temporary position of particle j
    float d_ij = disto (nowi, nowj);
    //Calculate distance between temporary position of particles i,j
    if (d_ij<=lo*spring_thr) //distance is less than a threshold
    {
      Spring a = new Spring (nodes[i],nodes[j]); //establish spring between nodes i,j
      springs.addElement(a); //add spring to the vector
      nodes[i].valence++; // add one to the valence of particle i
      nodes[j].valence++; // add one to the valence of particle j
    }
  }
```

```
  for (int i=0; i<springs.size();i++)
  {
    ((Spring)springs.elementAt(i)).go();
    // Calculate spring force and accumulate it in the temporary force vector of the
    // connected particles
    ((Spring)springs.elementAt(i)).draw(); //draw established springs
  }

  total_val = 0;

  for (int i=0; i<nodes.length; i++)
  {
    nodes[i].dir = add (nodes[i].dir, nodes[i].tdir);
    //add accumulated force vector from springs to the velocity vector of particle i
    nodes[i].tdir = new Vec(); // nullify node's temporary force accumulator
    total_val+= nodes[i].valence; //add node's valence to system's total valence
    nodes[i] = keep_in(nodes[i]);
    //Do boundary collision detection. Collision detection adds a response force vector
    //to the node's velocity if the node is found to be in violation of the boundaries
    nodes[i].move(); //Add final velocity vector to the particle's position
    nodes[i].draw(); //Draw the particle
    nodes[i].dir.scale(0.99); // Dump a portion of the velocity
  }

  av_val = total_val/float(particles); //Calculate system's average valence
  lo+=0.1; //Increase the spring ideal length
}
```

Basic functions of the particle class

```
class Node
{
  Vec pos;
  Vec dir;
  Vec tdir;
  int valence;
  Node(float x, float y, float z, float x_dir, float y_dir, float z_dir)
  {
    pos = new Vec(x,y,z);
    dir = new Vec(x_dir, y_dir,z_dir);
    tdir = new Vec();
    valence=0;
  }
  void move()
  {
    pos = add(pos,dir);
  }
  void draw()
  {
    pushMatrix();
    translate (pos.vec[0],pos.vec[1],pos.vec[2]);
    box(5);
```

```
    popMatrix();
  }
}
```

Basic functions of the spring class

```
class Spring
{
  Node node1; // Connected Particle 1
  Node node2; // Connected Particle 2
  float d; // Length of spring
  float scalar; //A scalar value that is used for scaling the spring force

  Spring(Node nodex, Node nodey)
  {
    node1 = nodex; // Input particle 1
    node2 = nodey; // Input particle 2
  }

  void go()
  {
    Vec now1 = add (node1.pos,node1.dir);  //Calculate temporary position of particle 1
    Vec now2 = add (node2.pos,node2.dir);  //Calculate temporary position of particle 2
    d = disto (now1,now2);
    //Calculate distance between temporary positions of particles 1,2
    Vec tdir_1 = sub(now1,now2); // Create a vector that repels particle 1 from 2
    Vec tdir_2 = sub(now2,now1); // Create a vector that repels particle 2 from 1
    tdir_1.normalise(); //Normalise the force vector 1
    tdir_2.normalise(); //Normalise the force vector 2

    scalar=0;
    if (d<lo) //Spring is in compression
    {
      scalar = -((d-lo)/(av_val)); // exert repulsive force
    }
    if (d>lo && d<=((spring_thr-1)/2+1)*lo) //Spring is tension, threshold case 1
    {
      scalar = -((d-lo)/av_val)*0.025; // exert attractive force
    }
    if (d>((spring_thr-1)/2+1)*lo) //Spring is tension, threshold case 2
    {
      scalar = ((d-spring_thr*lo)/av_val)*0.025; // exert attractive force
    }

    tdir_1.scale(scalar); //scale the force that is to be exerted to particle 1
    tdir_2.scale(scalar); //scale the force that is to be exerted to particle 2

    node1.tdir = add(node1.tdir,tdir_1); //accumulate the force locally at the particle
    node2.tdir = add(node2.tdir,tdir_2); //accumulate the force locally at the particle
  }
```

```
  void draw()
  {
    line (node1.pos.vec[0], node1.pos.vec[1], node1.pos.vec[2], node2.pos.vec[0],
node2.pos.vec[1], node2.pos.vec[2]); //Draw the Spring edge
  }
}
```

Basic functions of the boundary collision detection and response for the case of the cube

```
Node keep_in(Node nod)
{
  Vec new_pos = add (nod.pos,nod.dir); //Calculate node temporary position
  if (cube)
  {
    // Test if node is in violation of one the bounding sides of the cube. For each
    //violation calculate a response vector by the respective boundary and apply it to
    //the node's velocity
    if (nod.pos.vec[0]+nod.dir.vec[0]<=centre.vec[0]-side/2f)
    {
      Vec bound = new Vec(nod.pos.vec[0]+nod.dir.vec[0]+(centre.vec[0]+side/2f),0,0);
      nod.dir = sub (nod.dir, bound);
    }
    if (nod.pos.vec[0]+nod.dir.vec[0]>=centre.vec[0]+side/2f)
    {
      Vec bound = new Vec(nod.pos.vec[0]+nod.dir.vec[0]-(centre.vec[0]+side/2f),0,0);
      nod.dir = sub (nod.dir, bound);
    }
    if (nod.pos.vec[1]+nod.dir.vec[1]<=centre.vec[1]-side/2f)
    {
      Vec bound = new Vec(0,nod.pos.vec[1]+nod.dir.vec[1]+(centre.vec[1]+side/2),0);
      nod.dir = sub (nod.dir, bound);
    }
    if (nod.pos.vec[1]+nod.dir.vec[1]>=centre.vec[1]+side/2f)
    {
      Vec bound = new Vec(0,nod.pos.vec[1]+nod.dir.vec[1]-(centre.vec[1]+side/2),0);
      nod.dir = sub (nod.dir,bound);
    }
    if (nod.pos.vec[2]+nod.dir.vec[2]<=centre.vec[2]-side/2f)
    {
      Vec bound = new Vec(0,0,nod.pos.vec[2]+nod.dir.vec[2]+(centre.vec[2]+side/2));
      nod.dir = sub (nod.dir,bound);
    }
    if (nod.pos.vec[2]+nod.dir.vec[2]>=centre.vec[2]+pad)
    {
      Vec bound = new Vec(0,0,nod.pos.vec[2]+nod.dir.vec[2]-(centre.vec[2]+pad));
      nod.dir = sub (nod.dir,bound);
    }
  }
  return nod;
}
```

8. References

- Alexander, C., 1966, Notes on the Synthesis of Form. Cambridge, Massachusetts: Harvard University Press
- Baraff D., Witkin A., 1998, 'Large steps in cloth simulation', in Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 43-54
- Chilton J., 2000. Space Grid Structures, Oxford: Architectural Press
- Connelly R., Back A, 1998, 'Mathematics and Tensegrity', in American Scientist, vol. 86, no. 2, pp. 142
- Delanda M., 2002, 'Deleuze and the use of the Genetic Algorithm in Architecture', in Leach N., Designing for a Digital World, New York: Wiley
- Donev A., Cisse I., Sachs D., Variano E. A., Stillinger F. H., Connelly R., Torquato S., Chaikin P. M., 2004, 'Improving the Density of Jammed Disordered Packings Using Ellipsoids', in Science Journal, vol. 303, no.5660, pp. 990-993
- Flake G. W., 1998, The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems and Adaptation, Cambridge, Massachusetts and London, England: The MIT Press
- Gabriel F. J., 1997, Beyond the Cube: The architecture of Space Frames and Polyhedra, New York / Chichester / Weinheim / Brisbane / Singapore / Toronto : John Wiley and Sons, Inc.
- Gensane T., 2004, 'Dense Packings of Equal Spheres in a Cube', in The Electronic Journal of Combinatorics, vol. 11, no. 1,accessed September 2007, <http://www.combinatorics.org/Volume_11/PDF/v11i1r33.pdf>
- Graham R. L. and Lubachevsky B. D., 1996, 'Repeated Patterns of Dense Packings of Equal Disks in a Square', in The Electronic Journal of Combinatorics, vol. 3, pp.17, accessed September 2007, < http://www.combinatorics.org/Volume_3/PDFFiles/v3i1r16.pdf>
- Graver J. E., 2001, Counting on Frameworks: Mathematics to Aid the Design of Rigid Structures, The Mathematical Association of America: Dolciani Mathematical Expositions, Number 25.
- Jáuregui V. G., 2004, Tensegrity Structures and their Application to Architecture, MSc thesis, School of Architecture, Queen's University, Belfast, accessed September 2007, <http://www.alumnos.unican.es/uc1279/Tensegrity_Structures.htm>
- Jaworski P. L., 2006, 'Using simulations and artificial life algorithms to grow elements of construction', MSc thesis, Bartlett School of Graduate Studies, UCL, accessed September 2007, < http://eprints.ucl.ac.uk/archive/00002882/>
- Kanellos A., 2004, 'Parametric Design: Attributing parametric features to the synthesis, representation and physical level of architecture', Undergraduate Individual Research Project, School of Architecture, National Technical University of Athens.

- Kennedy J. and Eberhart R., 2001, Swarm Interlligence, London: Morgan Kauffman
- Kilian A. and Ochsendorf J., 2005, 'Particle-Spring Systems for structural form-finding', in Journal of the international association for shell and spatial structures: IASS, no. 148, pp. 77-84, accessed Septemper 2007, <http://designexplorer.net/newscreens/cadenarytool/KilianOchsendorfIASS.pdf>
- Lubachevsky B. D., 1991, 'How to simulate billiards and similar systems', in Journal of Computational Physics archive, vol. 94 , no. 2, pp. 255 - 283
- Mahdavi H. S. and Hanna S., 2003, 'An evolutionary approach to microstructure optimisation of stereolithographic models', in Proceedings of CEC2003, The Congress on Evolutionary Computation, Institute of Electrical and Electronics Engineers, pp. 723-730.
- Motro R., 2003, Tensegrity: Structural Systems for the Future, Hermes Science Publishing Ltd, an imprint of Kogan Page Ltd.
- Otto F. and Rasch B., 1995, Finding Form: Towards an Architecture of the Minimal, Axel Menges Edition
- Paul C., Lipson H., Cuevas F. J. V., 2005, 'Evolutionary form-finding of tensegrity structures', in Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO 2005, pp. 3-10
- Reynolds C. W., 1987, 'Flocks, Herds, and Schools: A Distributed Behavioral Model', in Proceedings of SIGGRAPH '87, vol. 21, no. 4, pp. 25-34, accessed September 2007, <http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/>
- Shimada K., 1995, 'Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing', in Proceedings of the third ACM symposium on Solid modeling and applications, pp. 409-419
- Tibert A. G. and Pellegrino S., 2003, 'Review of Form-Finding Methods for Tensegrity Structures', in International Journal of Space Structures, vol. 18, no. 4, pp. 209-223
- Thomson, D'Arcy, 1961, On Growth and Form. Cambridge: Cambridge University Press
- Williams C., 2001. 'The Analytical and Numerical Definition of the Geometry of the British Museum Great Court Roof', accessed September 2007, <http://staff.bath.ac.uk/abscjkw/BritishMuseum/ChrisDeakin2001.pdf>
- Zhang L., Maurin B., Motro R., 2006, 'Form-Finding of Nonregular Tensegrity Systems', in Journal of Structural Engineering, vol. 132, no. 9, pp. 1435-1440

Web documents (last accessed: September 2007)

- Bourke P. in <http://ozviz.wasp.uwa.edu.au/~pbourke/modelling_rendering/particle/> (Paul Bourke's definition of particles systems)

- Hanna S. in <http://www.sean.hanna.net/bodyspaceframe.htm> (Sean Hanna on Gormley's Body/Space/Frame)
- Urner K. in http://www.grunch.net/synergetics/ivm.html (Kirby Urner on Fuller's Isotropic Vector Matrix)
- (Beals et al in http://www.tiem.utk.edu/~gross/bioed/webmodules/spherepacking.htm (Beals et al on sphere close-packing)
- www.kennethsnelson.net (Website of the sculptor Kenneth Snelson)
- http://destech.mit.edu/akilian/projectpages/cadenary.html (Axel Kilian's web page on "CADenary")
- http://en.wikipedia.org/wiki/Close-packing (Wikipedia definition of sphere close packing)
- http://en.wikipedia.org/wiki/Kepler_conjecture (Wikipedia on the Kepler Conjecture)
- http://en.wikipedia.org/wiki/Particle_system (Wikipedia definition of particle systems)
- http://en.wikipedia.org/wiki/Van_der_Waals_force (Wikipedia definition of Van der Waals force)
- http://processing.org ("Processing" programming language website)