Version 1.

Canonical Forms for Multidimensional Isostatic Frameworks, Determinate trusses, Linkages and R-tensegrity Systems.

Dr. Offer Shai,
Department of Mechanics Materials and Systems
Tel Aviv University, Israel
shai@eng.tau.ac.il, http://www.eng.tau.ac.il/~shai

TABLE OF CONTENTS

Abstract

Current paper serves two main goals. First is to develop the canonical form underlying a mechanism for generating isostatic structures. The canonical form comprises several hierarchical levels each associated with specific graph theoretical construction rules.
The second goal is aimed in the opposite direction - decomposition of the given structure into the basic building blocks. The mathematical basis for the presented results include Assur graphs – a recently developed mathematical concept borrowed from mechanical engineering, pebble game algorithm and some other advanced topics of discrete mathematics.
In the final chapters of the paper, a reference is made to possible applications to rigidity theory, tensegrity structures and geometric constraint systems.

1 Introduction

Current paper first presents a short overview on the novel mathematical concept of Assur Graphs recently developed in (Servatius et, al., 2008a,b). Assur graphs were shown to be the basic building blocks of isostatic frameworks possessing a variety of special combinatorial properties elaborated in the previous and the current publications.
Section 3 is dedicated to the main result introduced in this paper, by showing the development of an infrastructure referred as 'canonical form for isostatic structures'. The canonical form enables generating and organizing feasible isostatic structures. Staring from a basic simplex graph associated with the dimension of interest, the canonical form provides a methodology for arriving at structures of any complexity through execution of a series of graph theoretical construction rules. The approach is thoroughly demonstrated both for the two and the three-dimensional systems.
Section 4 shows that the known Pebble game algorithm can be used to identify whether the graph to which it was applied is the pinned AG or not.
Section 5 provides the connection between the Assur graphs and the Lamman graphs, providing a powerful platform for decomposition of Lamman graphs into the underlying Assur graphs and its subsequent simplified analysis.
Sections 6 and 7 are dedicated to possible the practical applications of the proposed approach, dealing with the tensegrity systems and geometric constraint systems respectively.
It should be noted that the material appearing in this paper is organized in the preliminary preprint form. Additional material will be added prior to engaging the peer review process and publication in the professional literature. The added material would include deeper elaboration of mathematical proofs provided in the paper, more thorough and more fair literature review and others.

2 Overview on the Assur Graph (AG) concept.

The Assur groups, occasionally referred as Assur Structures are widely used in the kinematical community, particularly among Russian scientists. Leonid Assur (Assur, 1952) has developed these basic structures in order to make it possible to decompose any linkage into components of zero mobility, and for each one, to develop special methods for analysis of locations, velocities, accelerations and other physical properties.

The concept has been reformulated for the first time in rigidity theory terminology in (Servatius et, al., 2008a,b), where it was defined as an isostatic pinned framework, for which deletion of any vertex results in non-rigid framework.

In the current work, the main issue is to establish the canonical forms, with Assur graphs being the central tools for achieving that. For this reason, we employ a slightly modified definition of Assur graphs from that appears in (Servatius et, al., 2008a,b), as clarified in the following:

Pinned Assur graph – is a pinned isostatic framework where the deletion of any vertex makes it not rigid. In (Servatius et, al., 2008a,b) it is termed an Assur graph. Figure 1a, depicts a graph, which can be seen as a pinned Assur graph since it is a pinned isostatic framework, and deletion of any vertex results in a non-rigid graph.

Assur Graph (AG) – Here the definition of Assur graph is also a bit different from the use in the engineering society and the one used in (Servatius et. al., 2008a and b). Assur graph is defined recursively as appears in section 3, thus we first define the canonical procedure to construct it and only later (in more details in the forthcoming paper) we define its special properties in higher dimensions. In general, we define the Assur graph through the orientation of the edges yielded from application of the pebble game algorithm on a graph. If the algorithm results in a directed graph where there is no directed cut-set and the pebble game terminates without assigning one edge only, then the graph is an AG. The remaining pebbles at the termination of the pebble game indicate on how to construct the corresponding pinned-AG from it. Figure 1 depicts different types of graphs used in this work, all of them in 2D, however, the same types exist in other dimensions as well. Figure 1a is an Assur Graph, Figure 1b is a pinned Assur graph, while Figure 1c is not a pinned AG, since one can remove vertex A with its two edges and remain with a dyad – a pinned isostatic framework, which contradicts the definition of pinned-AG to be minimally rigid relative to the vertices.

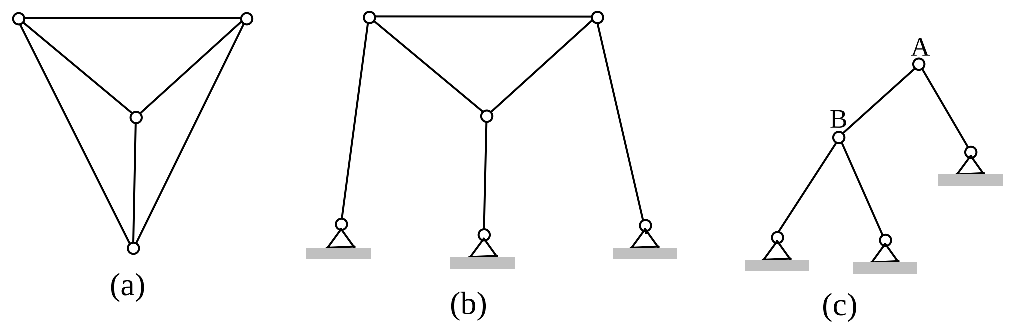


Figure 1– Different types of graphs in 2D.

a) Assur Graph. b) Pinned-Assur Graph. c) Pinned isostatic framework but not a pinned-AG.

Since the characterization of the Assur graphs is based on the results of applying the pebble game, Figure 2 depicts the graphs from Figure 1 with the oriented edges resulted from the application of the pebble game. In Figure 2a there is one edge, which is not assigned due to the one time redundancy in the edges, and there are three pebbles left, indicating how to pin the graph to the ground as is explained below. Since there are no directed cut-sets, this graph is of the Assur graph type. In Figure 2b, the pebble game was applied to the pinned framework and since the inner edges form a well connected sub-graph thus it is concluded that it is a pinned AG. On Figure 2c it can easily be verified that there are two directed cut-sets indicating that it is not a pinned AG, and the order of the decomposition using the directed cut-sets indicates the order of the decomposition of the pinned isostatic framework into pinned AGs.

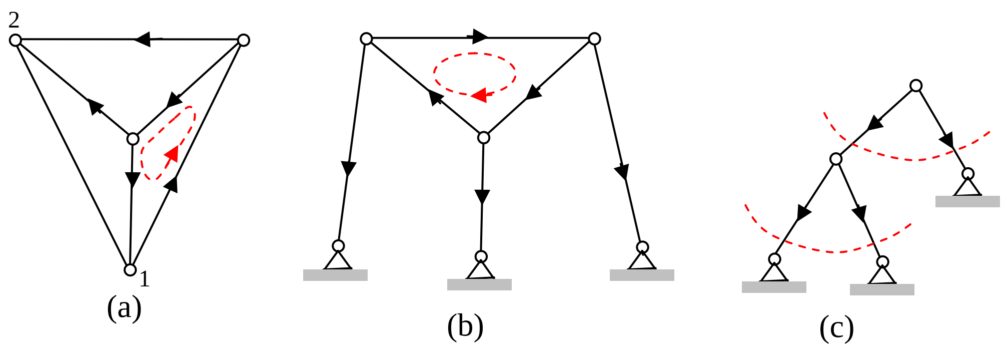


Figure 2 – Different types of graph yielded after applying the pebble game.
a) Assur graph with no directed cutsets. b) Pinned AG whose inner edges form a well connected graph. c) Pinned isostatic graph consisting of two pinned AGs, each defined by the directed cutset.

3 Canonical form for the rigid Isostatic frameworks

The current section shows a canonical form enabling to generate and classify possible rigid isostatic structures. The canonical form constitutes a hierarchy built of 5 levels (see Figure 3). The lowest level in this hierarchy enumerates the possible rigid isostatic structures. The rest of the levels are arranged in order of the topological complexity, as follows: simplex graphs, atomic Assur graphs (atomic AG), fundamental Assur graphs (fundamental AGs) and Assur graphs (AGs). Figure 3 depicts the hierarchy of the canonical forms accompanied by the corresponding representative examples.

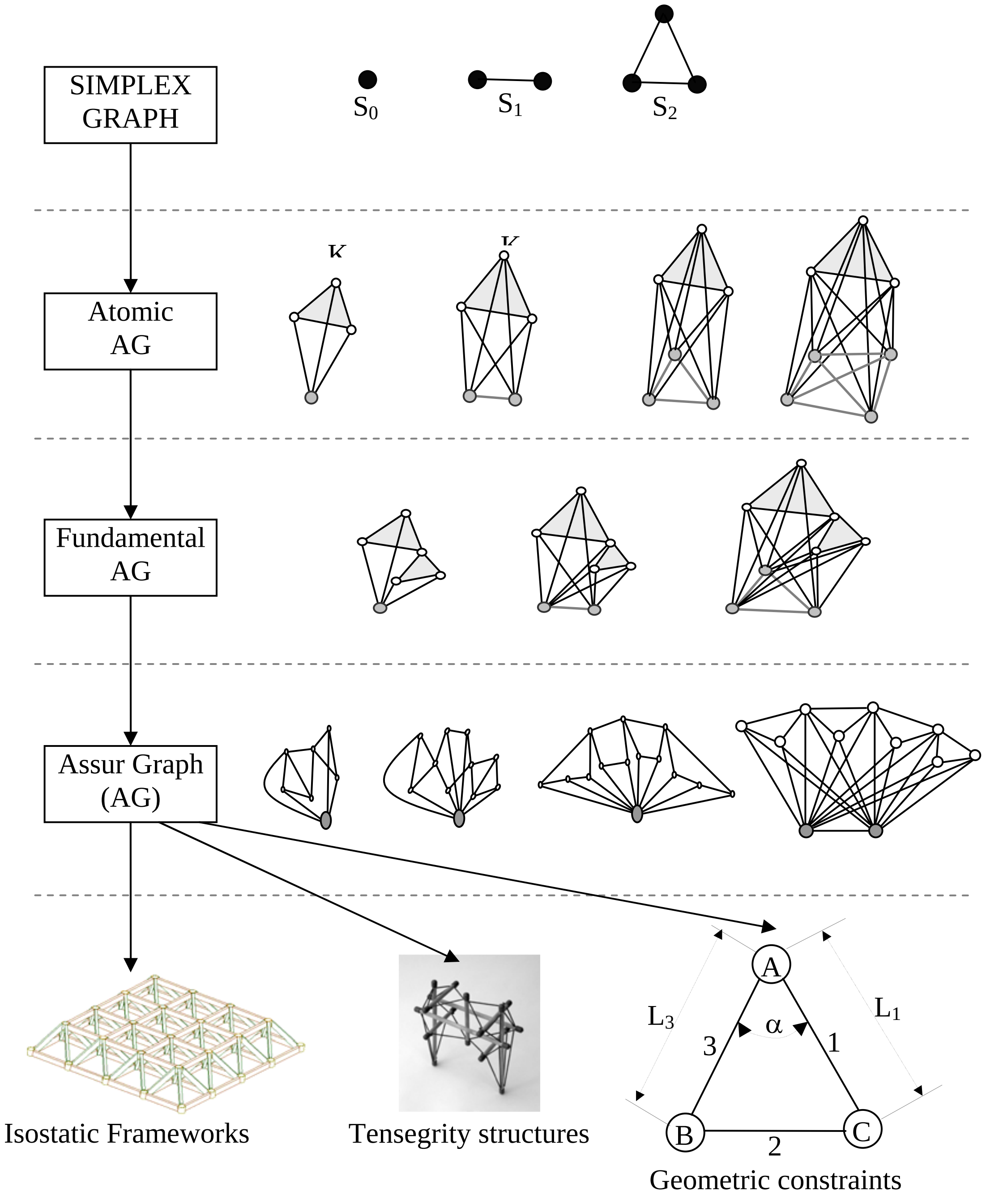


Figure 3. The 5 levels of the canonical form for rigid isostatic frameworks.

The following sub-sections provide further elaboration upon the canonical forms, gradually building it up, starting from the lowest simplex graph level – to the highest level of the isostatic structures.

3.1 The atomic AG

For each dimension there corresponds a so-called atomic Assur Graph (atomic AG), which is the basis for the generation of the fundamental AGs, which in their turn, as will be explained in section 3.2, are the basis for the development of the Assur graphs and engineering systems.

3.1.1 The canonical rule of building the atomic AGs in different n-domains

Definition. Simplex of dimension d, S_d, is defined as a basic graph corresponding to dimension d and can be obtained inductively by adding a new vertex and connecting it with edges to all the vertices in simplex S_{d-1}. The simplex of dimension d=0 is a single vertex. Figure 4a-e shows the simplex graphs corresponding to dimensions 0-4 respectively.

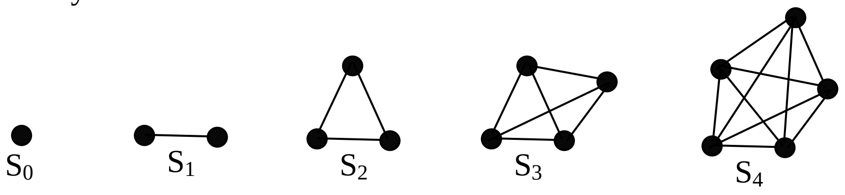


Figure 4– Examples of simplexes
a) Point. b) Line. c) Triangle. d) Tetrahedron. e) Pentachoron.

The principle of constructing the atomic AG is to connect a triangle with an S_{d-2}, simplex of dimension d-2, by adding an edge connecting each vertex of the triangle with each vertex of the simplex S_{d-2}. Figure 5 schematically depicts such a connection. The vertices of the simplex will be referred as the ground part of the AG, and will be designated with the gray color. One can easily verify that the atomic AG in dimension d is identical to the complete graph - K_{d+2}.

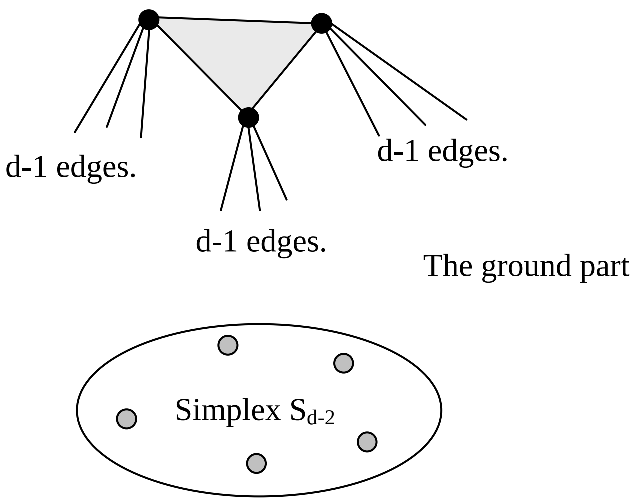


Figure 5 -The scheme of constructing an atomic AG in dimension d.

In accordance to the construction rule formulated above, Figure 6a-d show the atomic AGs corresponding respectively to dimensions 2-5.

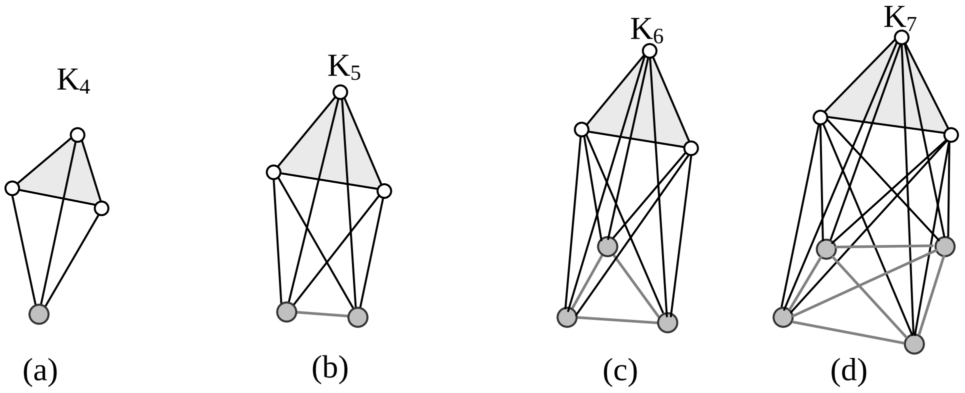


Figure 6– The atomic AGs in dimensions
a) d=2, plane. b) d=3, space. c) d=4. d) d=5

3.2 The derivation of the fundamental AGs (f-AGs) from the Atomic AG

Each fundamental Assur Graph, referred for brevity as f-AG, is a representative of an infinite number of AGs and subsequently of infinite number of isostatic frameworks, determinate trusses and linkages. Any feasible f-AG can be obtained through application of a sequence of operations, termed – fundamental extensions, starting from the atomic AG of the corresponding dimension, as is described in detail in the following section.

3.2.1 The Fundamental Extension Operation in 2D

The main property of the fundamental extension operation, for brevity referred here as f-extension, transfers f-AG into extended f-AG, which contains an additional basic triangle.
The f-extension is applied to the ground edges of the f-AGs, namely the edges incident to the ground vertices (gray vertices)**.**
The f-extension is done in three stages: first, one ground edge is removed; then a basic triangle is added with one vertex coinciding with a non-ground vertex of the removed edge; then the remaining two free vertices of the basic triangle, are both connected to the ground through one ground edge
Figure 7 schematically depicts the f-extension operation, while Figure 8 shows examples of f-AGs in different dimensions.

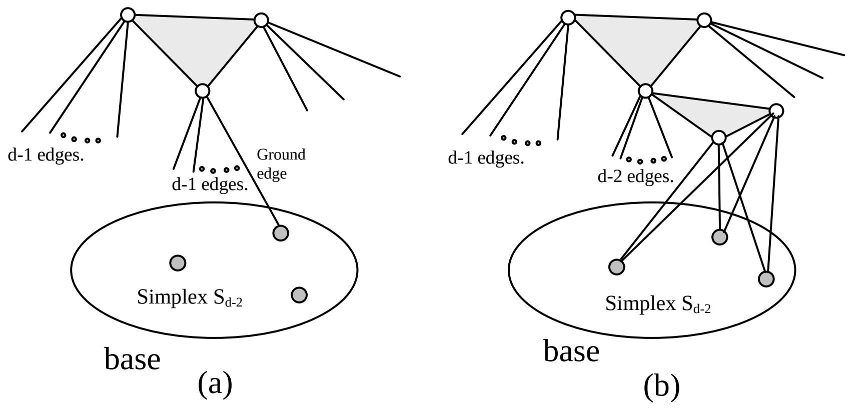


Figure 7 - The scheme of applying the f-extension operation on an f-AG in dimension d.

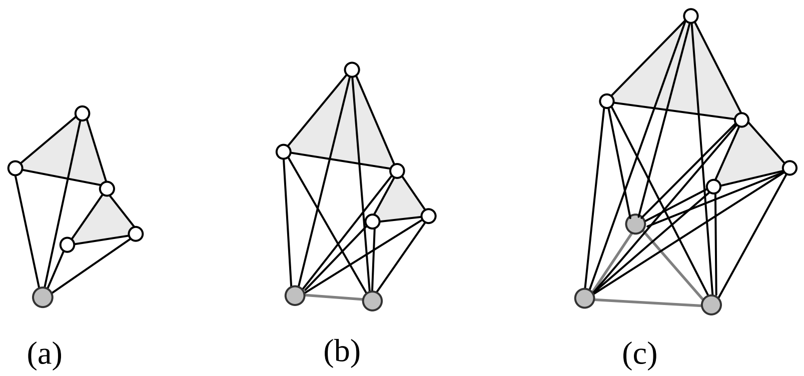


Figure 8 -The f-AGs obtained after applying the f-extension upon the atomic AGs in different domains.
a) in 2D. b) in 3D. c) in 4D.

3.2.2 Self-duality property of f-AGs in 2D

All the f-AGs in 2d are self-dual graphs, i.e. graphs that are isomorphic to their dual graphs (Swamy and Thulasiraman 1981). The fact that f-AGs are planar is straightforward since in each extension we replace a ground edge by a plane sub-graph built of two triangles, which can always be drawn in a manner such that no crossing edges exist. The proof that the dual graph is isomorphic to the original graph is made by observing that both the original edge and its corresponding dual edge are replaced by a self-dual sub-graph built of a two triangles.

3.3 The Canonical Forms of the Assur Graphs (AGs)

In the previous section, we have introduced the canonical forms of the fundamental AGs, all being derived from the atomic AG through repetition of the single operation of fundamental extension. In this section, it will be shown that each f-AG is the representative of an infinite class AGs, all derived through one operation - the d-1 extension applied sequentially to the corresponding f-AG. In set theory terminology, there are an infinite number of classes of AGs, each class is defined by a representative graph – the corresponding f-AG. The following section explains how all the AGs of the class are obtained by applying the d-1 extension to the representative AG. The table appearing in Figure 9 provides a general perspective on the canonical forms of all the AGs, as infinite classes (columns).

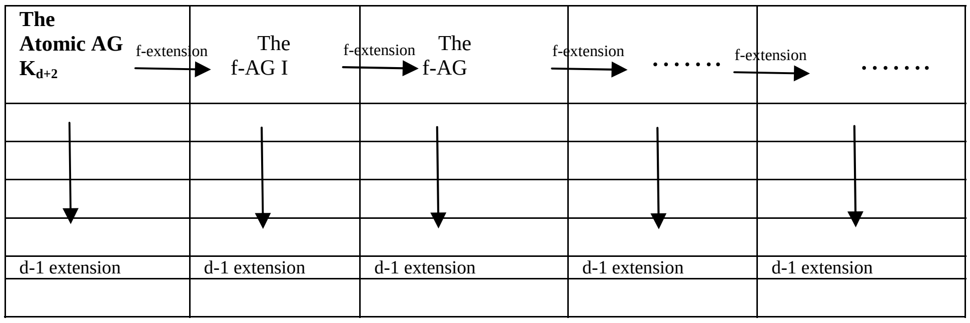


Figure 9 – General perspective on the class hierarchy of the AGs

3.3.1 The d-1 extension operation and its application rule.

In the previous section it was mentioned that each f-AG defines a class of an infinite number of AGs, all are derived through a single operation called the d-1 extension.

Definition. d-Cutset in dimension d is a set of vertices $x_1,..,x_d \in V$ in a d-connected graph G such that $V - \{x_1,..,x_d\}$ is disconnected. The d-cutset includes all the reference vertices (colored with gray) and one connecting inner vertex, colored with two colors.

For example, Figure 10a depicts an f-AG in dimension 2 thus the graph is 2-connected and the 2-cutset consists of vertex colored by both red and blue together with the only one ground vertex existing in the base, the gray vertex. The same explanation is applied to the other graphs in higher dimensions (Figure 10b and c).

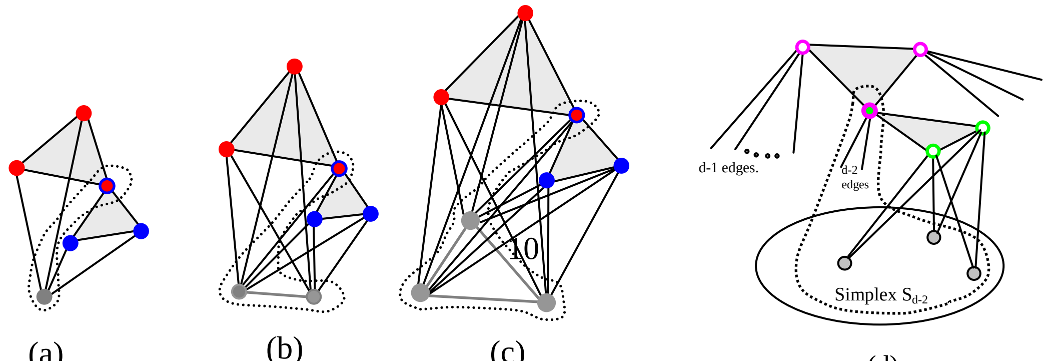


Figure 10 – Example of d-cutsets in f-AGs in dimensions:
a) 2-cutset in 2D. b) 3-cutset in 3D. c) 4-cutset in 4D. d) d-cutset in any dimension d.

Definition: d-1 extension. Choose edge XY. Add an additional vertex Z to the graph. Replace edge XY by two edges ZX, ZY and additional d-1 edges connecting vertex Z with d-1 other vertices of the graph (in order for the extension to be proper – the choice of these vertices should comply with a rule explained later in this section).

Figure 11 schematically depicts the application of the d-1 extension.

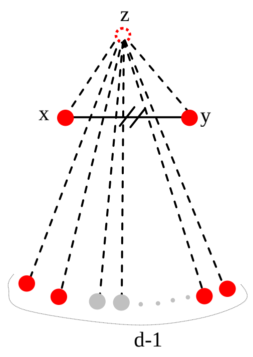


Figure 11 - The d-1 extension.

The special case of the d-1 extension in two dimensions is widely used and reported in the literature (Berg and Jordan, 2003).
As is mentioned above, in order to apply the d-1 extension operation to an AG and to remain in the same class, one has to preserve the following rule:

The color rule of the d-1 extension: Upon application of the f-extension, color the newly added vertices by a new color.
Upon application of the d-1 extension the added Z vertex should be colored by a color identical to the color of X and Y. If one of the vertices is colored with two colors (due to its inclusion in the d-cutset) then the other end vertex defines the color of Z. The application rule for the d-1 extension is that all the vertices which are connected to the Z vertex by this operation (except X and Y) should be chosen so that they are associated with the same color as Z.

In case above rule is not satisfied, one might obtain an AG belonging to a lower class, as is shown in the example of Figure 12, where the initial AG was f-AG of class 2, while after the d-1 extension has been applied to it, the resulting AG belongs to class 1.

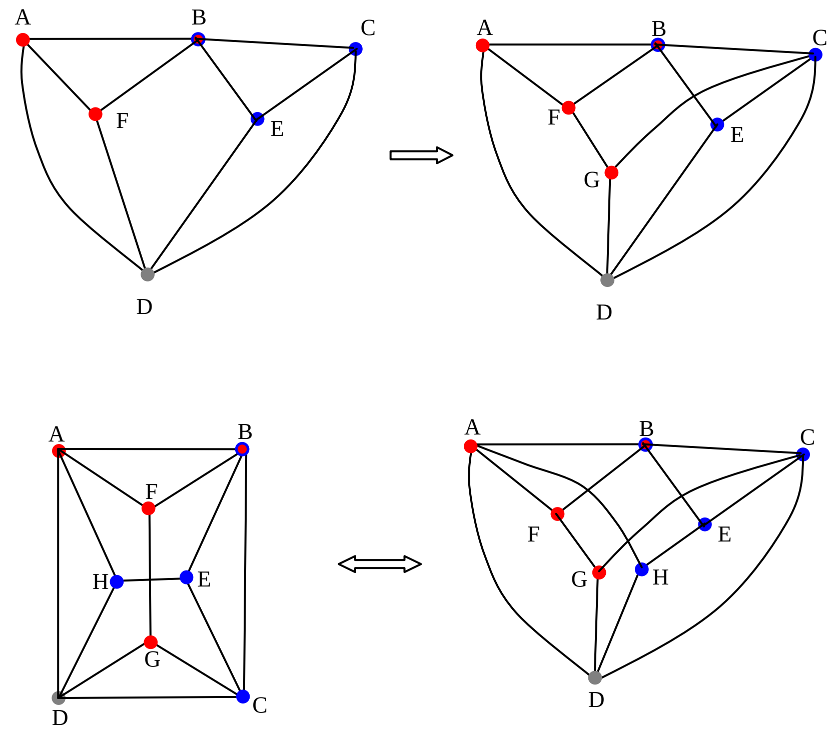


Figure 12- Example of applying the d-1 extension without obeying the d-1 extension color rule.

It can be verified that the graph in Figure 12d belongs to class 1 since there is a sequence of 1-extensions from a corresponding f-AG. Figure 13 provides the reverse process to the d-1 extension operation, referred as the splitting-off process, which confirms that the AG of Figure 12d indeed belongs to the first class.

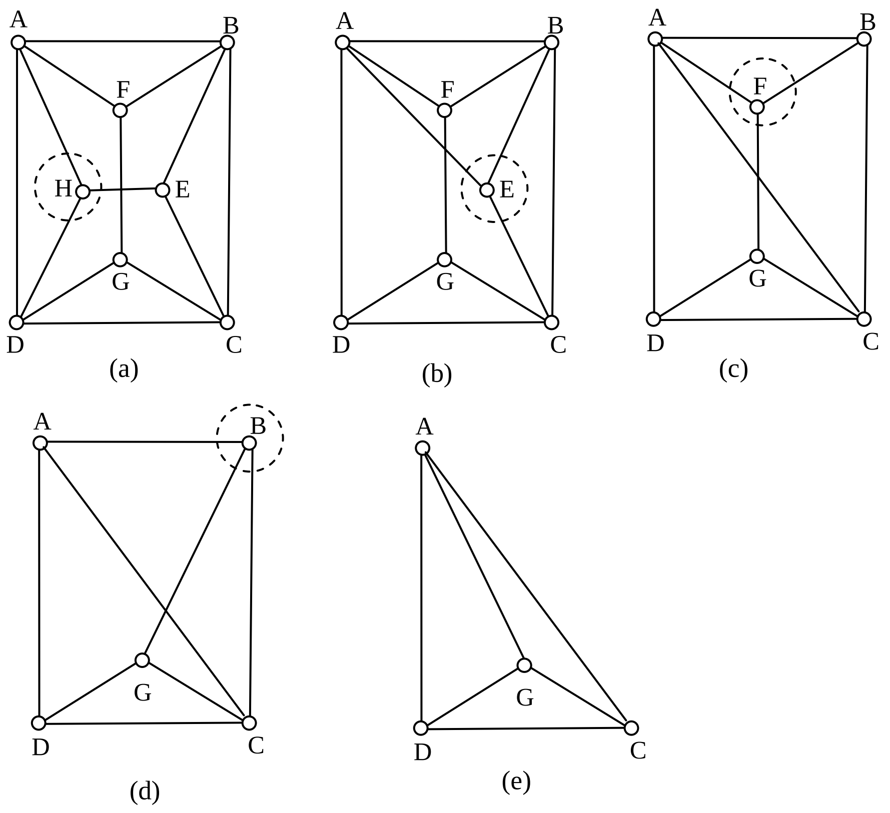


Figure 13 - Example of a splitting off sequence proving that the AG in Figure 12a belongs to class 1. The dashed circuits indicate those vertices on which the splitting off action is done.

Figure 14 depicts a valid sequence of AGs obtained through proper application of the d-1 extension (in this, 2D, case – a 1-extension) to an f-AG.

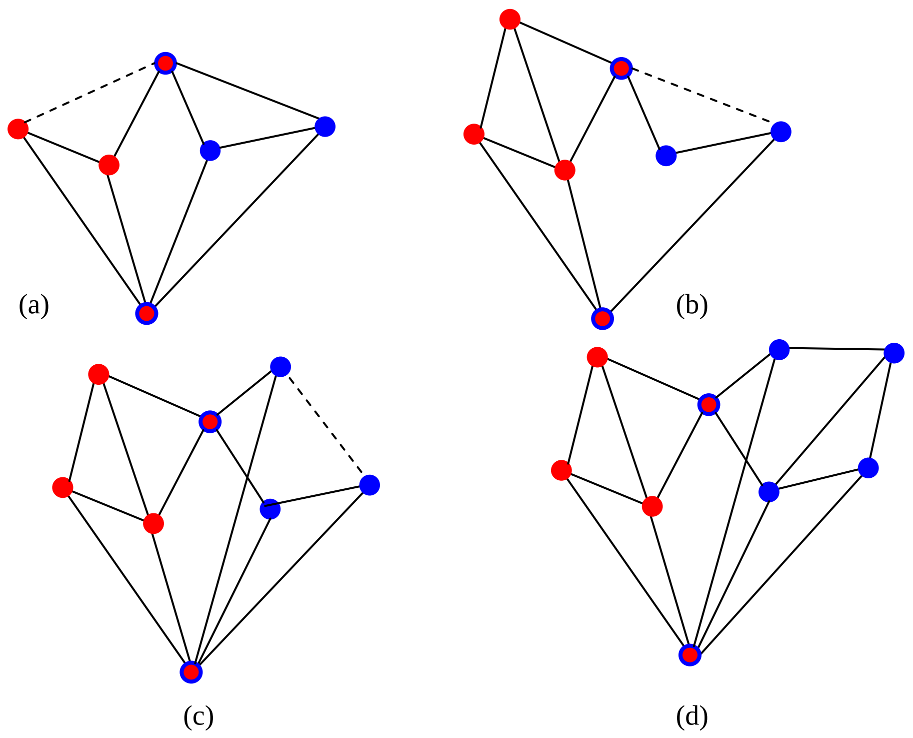
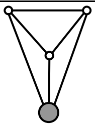
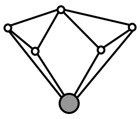
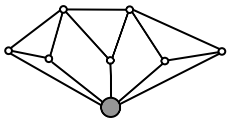
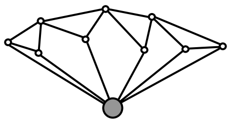
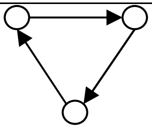
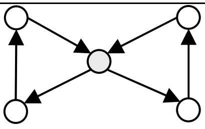
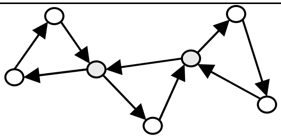
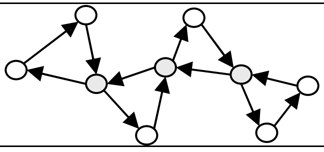
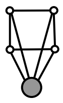
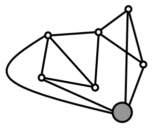
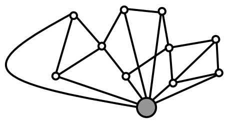
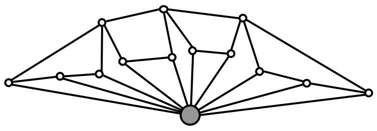
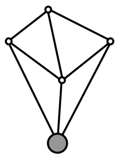
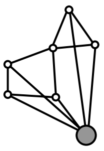
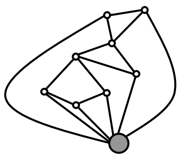
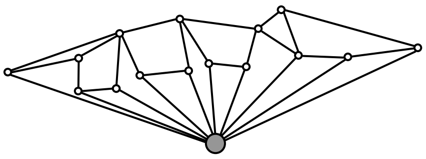
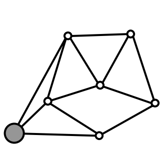
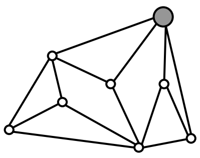
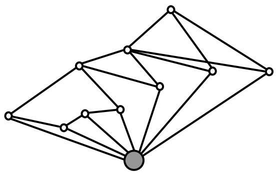
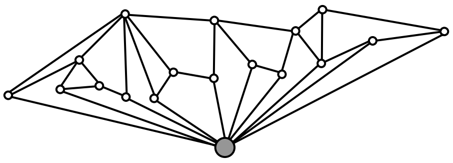
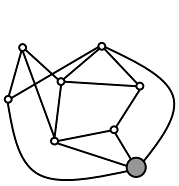
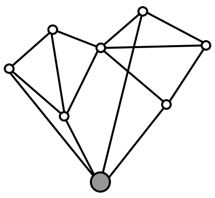
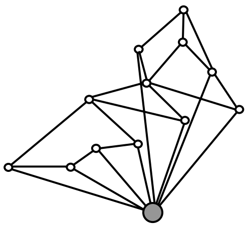
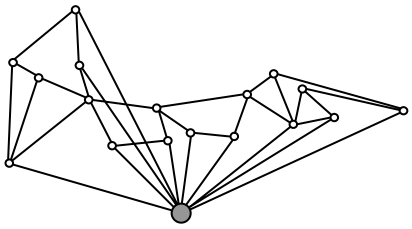
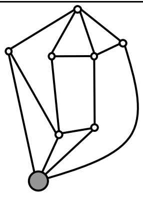
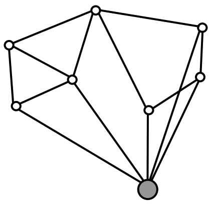
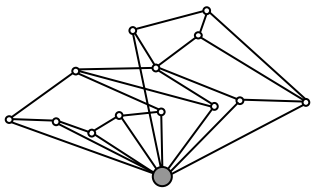
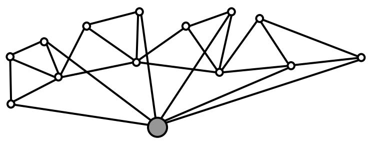


Figure 14. - Example of a valid sequence of 1-extensions of AGs belonging to class II.

3.4 The Canonical Form of the 2D AGs

Table 1 shows the canonical derivation of AGs in 2D, by starting from the atomic AG for 2D (K4) shown in Figure 3. Applying the sequence of f-extensions to it yields the derivation of the representatives of the classes, the f-AGs, and the elements in each class are obtained through applying the 1-extension upon the representative graph or any of its derivatives. As can be seen in Table 1, all the AGs are classified into classes and each AG belongs to one and only one class.

Class 1	**Class 2**	**Class 3**	**Class 4**
			
			
			
			
			
			
			

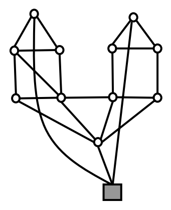
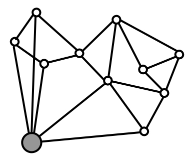
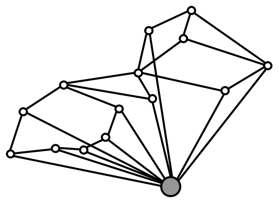
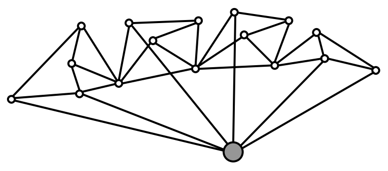
Class 1	Class 2	Class 3	Class 4
			

Table 1 - The canonical map of the 2D AGs.

3.5 The Canonical Form of the 3D AGs

The canonical form of the 3D AGs is obtained similarly to that in the 2D. All the representatives of the classes are constructed through applying the f-extension operation, and all the AGs belonging to the same class are obtained through applying the 2-extesion operation as described in the following section.

Theorem: Every 3D-AG has a unique self-stress and this self-stress is non-zero on all edges.

3.5.1 The f-extension operation in 3D

The f-extension operation replaces the ground edge with a basic triangle and connects the two new vertices to the two ground vertices, as shown in Figure 15.

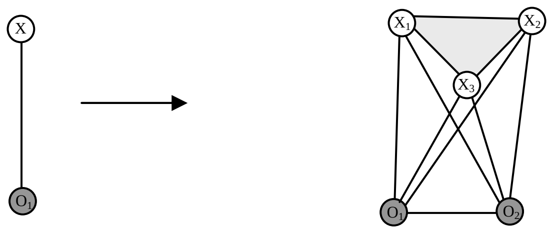


Figure 15– The resultant of applying the f-extension in 3D.
a) The ground edge. b) The replacement of the ground edge.

As can be seen, the process of constructing all the f-AGs in 3D is similar to the process of constructing the f-AGs in 2D, only this time there are two ground vertices thus each vertex of the basic triangle is connected to these two ground vertices. Example of f-AGs obtained through applying the f-operation in 3D appears in **Figure 16**, where the dashed lines indicate the edges upon which the f-extension is applied.

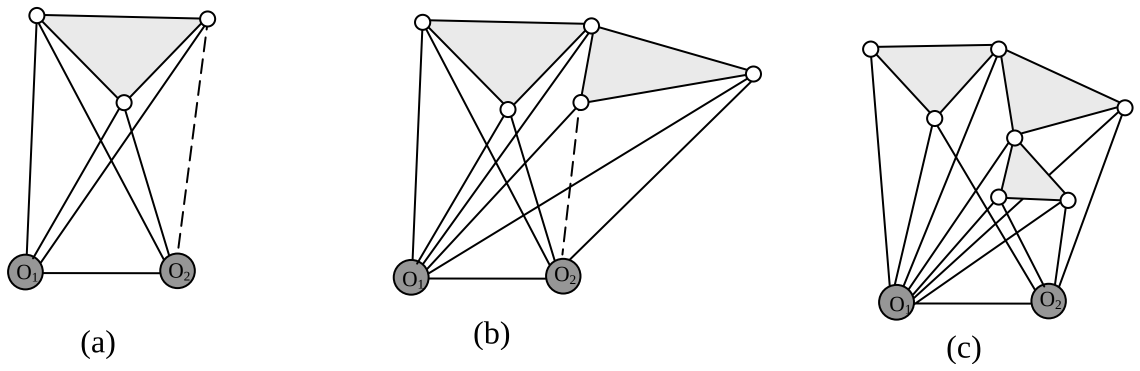


Figure 16 - Example of f-AGs resulting from applying a series of f-extension operations.

3.5.2 The d-1 extension operation in 3D

The d-1 extension, in this case the 2-extension, can be applied to any f-AG edge by splitting the edge, adding a new vertex between its two end vertices, and connecting the new vertex to the other two vertices. A special care, as explained in section 3.3.1, has to be taken to avoid changing the class to which the AG belongs. This is obtained by coloring the vertices throughout the process, and connecting the new vertex to another two vertices with the same color, or to the ground vertices. Example for such extension appears in Figure 17.

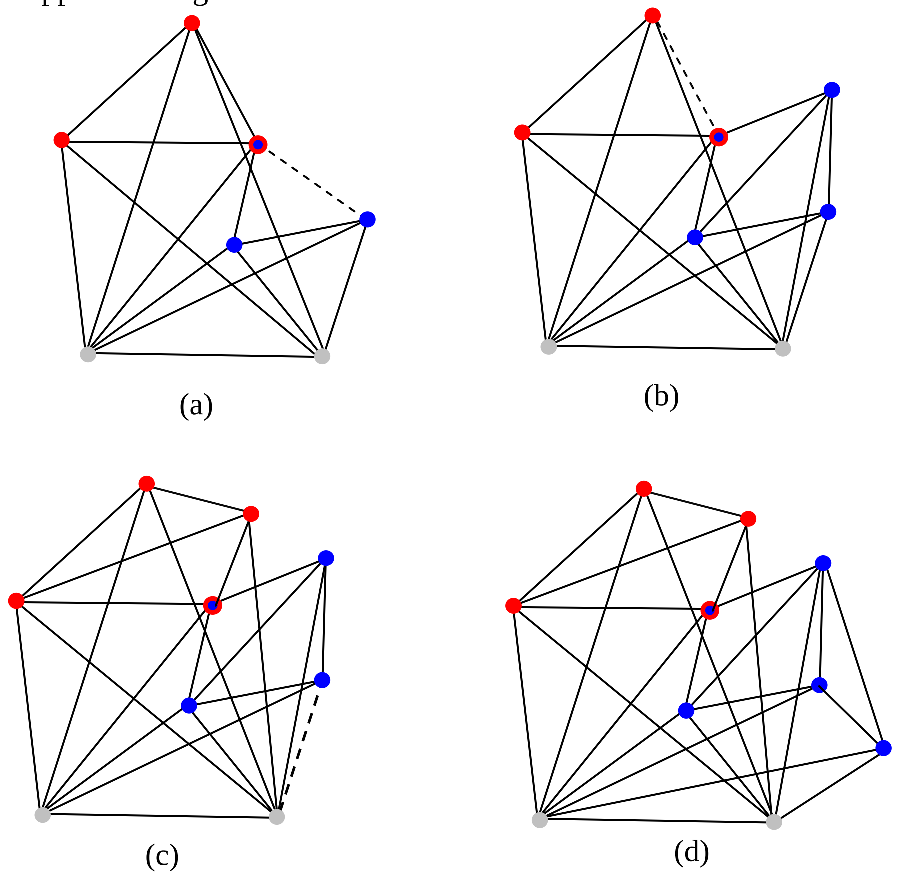


Figure 17 - Example of a valid sequence of 2-extensions of AGs belonging to class II in 3D.

3.5.3 Obtaining the canonical map of the AGs in 3D

Similarly to 2D, the following table gives a brief description on the canonical form of all the 3D AGs, where the first row contains the 3D f-AGs, each representing a different class of 3D AGs.

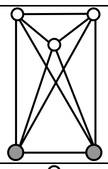
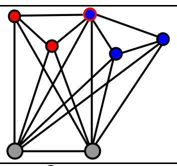
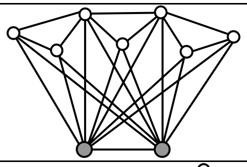
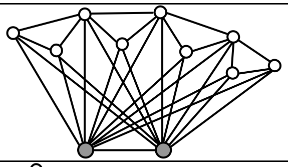
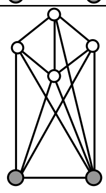
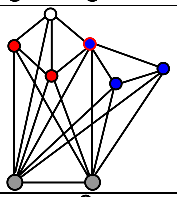
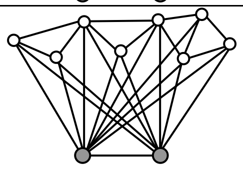
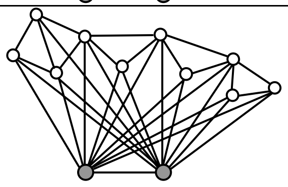
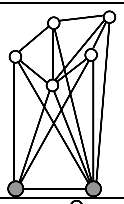
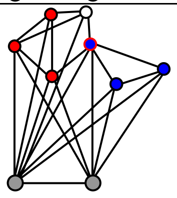
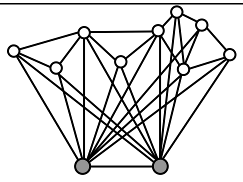
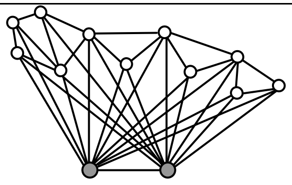
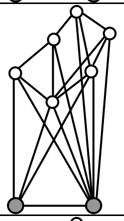
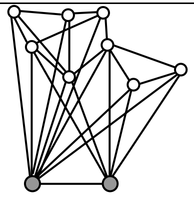
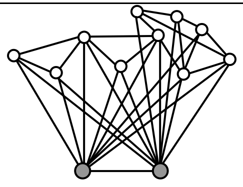
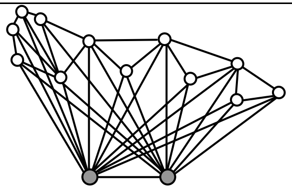
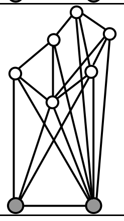
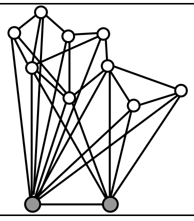
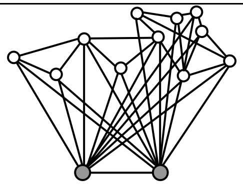
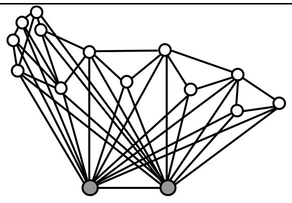
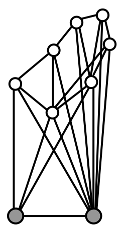
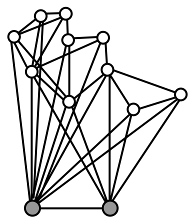
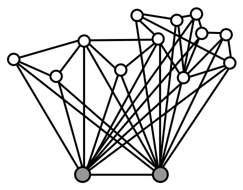
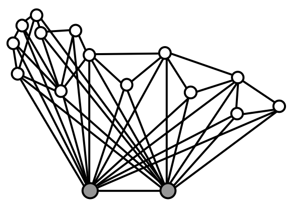
Class 1	Class 2	Class 3	Class 4
			
			
			
			
			
			

Table 2 - the table describing the classes and representatives of 3D AGs.

3.5.4 Special properties of 3D AG

It should be noticed that every 3D AG remains rigid when one removes the ground edge. In Figure 18 there appears the 3D triad AG. One can verify that when the ground edge, whose end vertices are colored with gray is removed, the remaining graph (Figure 18b) is rigid in 3D. This property is important and will assist us in understanding the problem of checking graph rigidity in 3D, as is discussed in section 4.

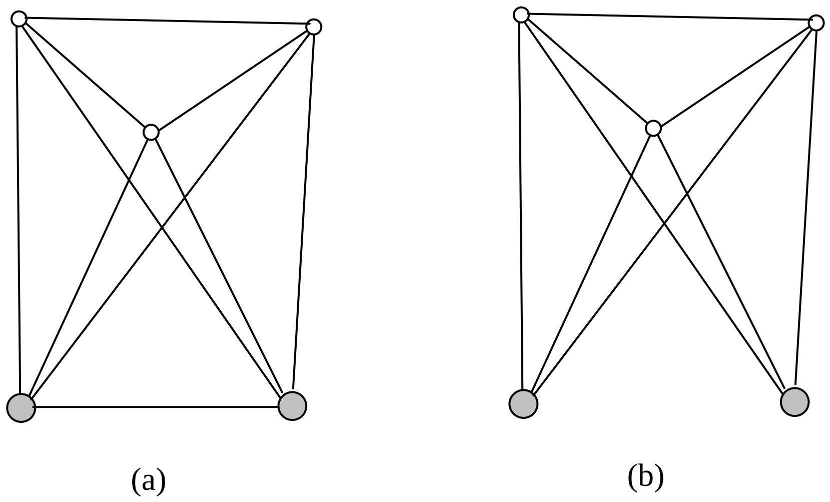


Figure 18 – Example of obtaining a 3D isostatic graph (b) After removing the ground edge from the 3D AG (a).

3.6 The Canonical form of the pinned AGs in 2D

As is explained in the above, the usage of the pinned AGs is the prerequisite for employing our canonical structure in generating real engineering systems, including determinate trusses, mechanisms and tensegrity systems. In general, for any regular AG there exists a pinned AG counterpart, which can be obtained from the regular AG by removing the simplex edges and splitting and pinning the simplex vertices. The opposite correspondence from any pinned AG to a regular AG also holds, despite a single exception, as follows:

In pinned AG there exists a special type of graphs for which there is no correspondence in AG, referred the dyad. This type of graph does not contain any triangle, and is the atomic pinned AG, i.e. all the f-pinned AGs can be derived from it, but it is not the representative of any class. Examples of dyads in two and three dimensions are depicted in Figure 19.

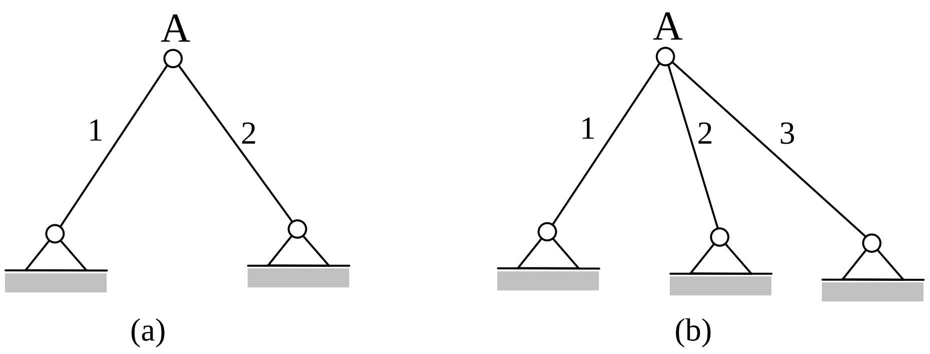


Figure 19– The special type of pinned Assur Graph – the dyad.
a) The dyad in 2D.   b) The dyad in 3D.

There is an additional way to obtain pinned AGs, besides deriving them from the regular AG. The second way, used in the engineering community, is to start from the simplest pinned AG – the dyad, and from it to derive all the pinned f-AG by applying the f-extension: replacing a ground edge by a triangle with two ground edges, as explained in section 3.2.1, and then applying the d-1 extension as is explained in section 3.3.1.
Example of deriving the pinned triad in two ways is shown in Figure 20.

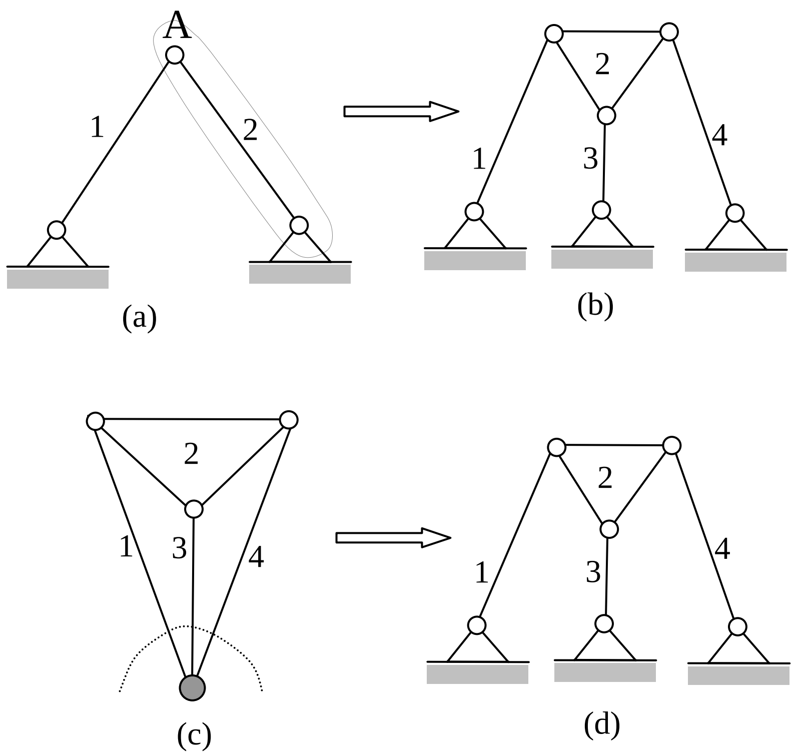


Figure 20– Deriving the pinned triad in two different ways.
a,b – from the pinned dyad. c,d- from the f-AG: the triad.

3.6.1 Obtaining the canonical map of the pinned AGs in 2D

The canonical form of the pinned AG is similar to that of the AG as appears in section 3.4, i.e. it consists of f-pinned AGs that are the representatives of an infinite number of pinned AGs, such that each pinned AG belongs to one and only one class. The only difference, which has already been outlined above, is with the existence of the first column, whose representative pinned AG is the dyad (as is shown in Table 3 there are no elements in its class except for the dyad itself).

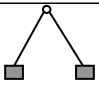
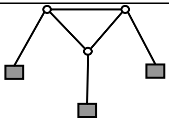
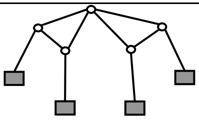
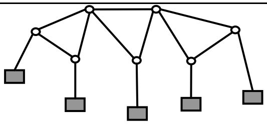
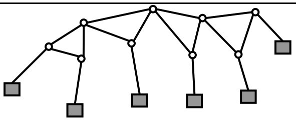
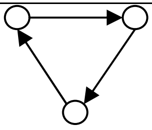
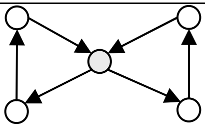
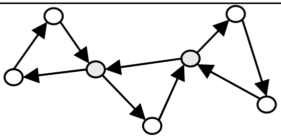
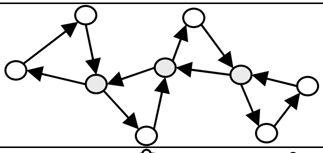
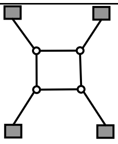
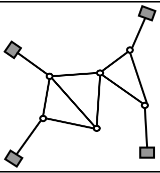
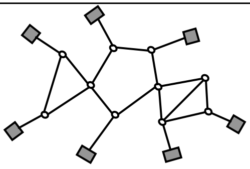
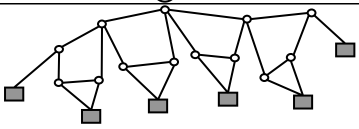
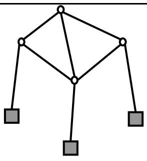
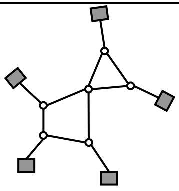
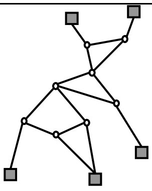
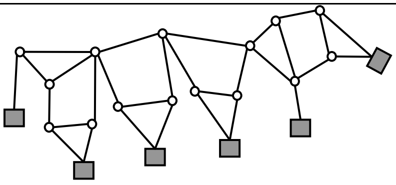
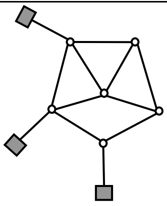
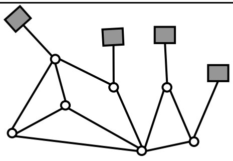
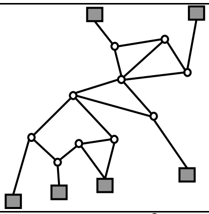
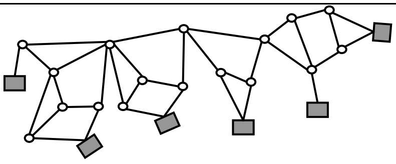
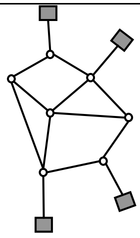
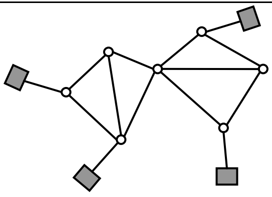
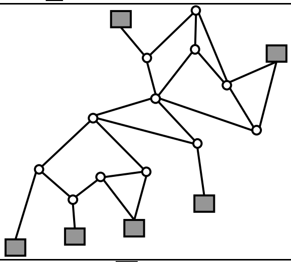
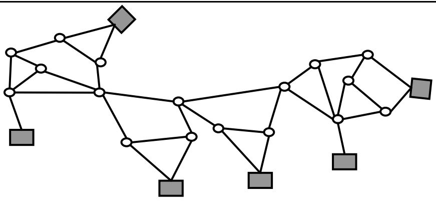
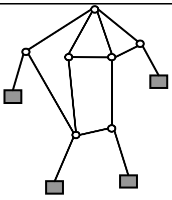
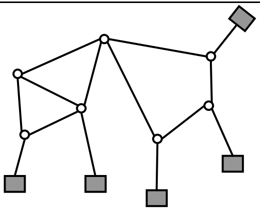
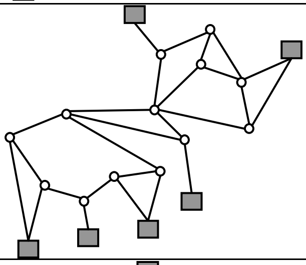
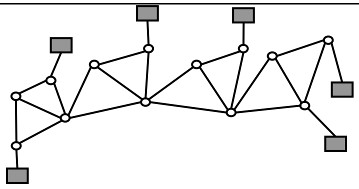
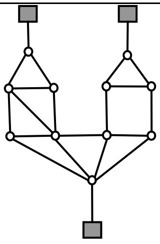
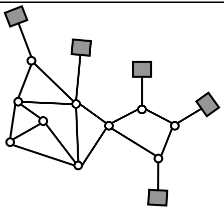
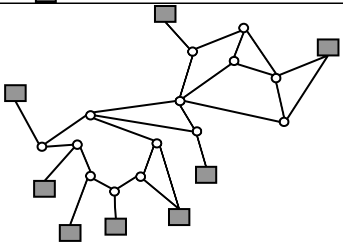
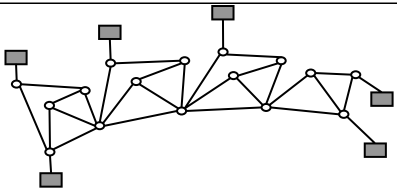
Class 0	**Class 1**	**Class 2**	**Class 3**	**Class 4**
				
				
				
				
				
				
				
				

Table 3 - Some classes of pinned AG.

3.7 Characterization and identification of pinned f-AGs

It is possible to assign an id to each f-AG that can be used to identify it and also, from which it is also possible to construct it. All the f-AGs can be sorted according to the lexicographic order, using the flowing rule:

1. Find the longest chain in the pinned-fAG, which will define the number representing the main chain. The ordering in the main chain is defined so that the number obtained will be the largest in the lexicographic order.
2. This numbering is defined for every sub-chain, recursively.

For example, there are two possibilities for numbering the f-AG in Figure 21a, as appears in Figure 21b and Figure 21c. The corresponding number for the framework in Figure 21b is 123_1456 and for the second framework is 1234_156, and since 123_1456 $>1234_156$ in lexicographic order, thus the numbering in Fig. b will be the one chosen.

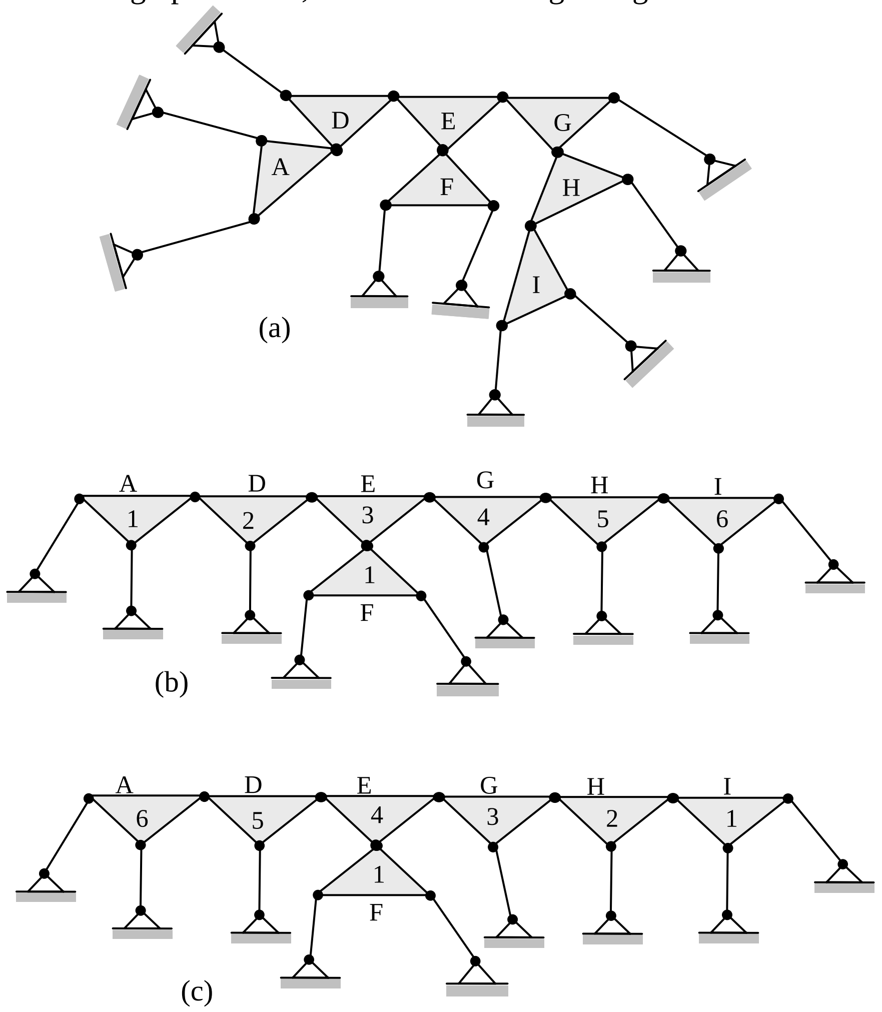


Figure 21 – Example of different numbering of f-AG.

Additional example is shown in Figure 22, with the f-AS associates with the unique numbering of: $123_145_16_17_189(10)$.

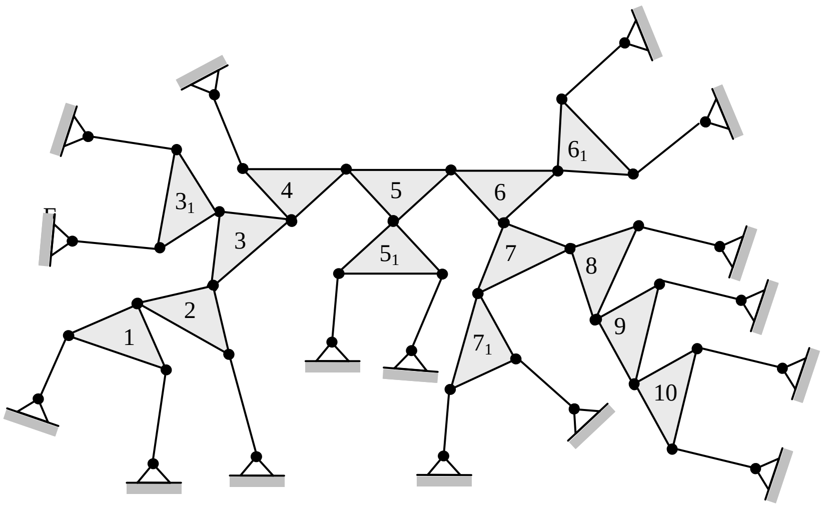


Figure 22 - Example for a f-AG, whose unique numbering is $123_145_16_17_189(10)$.

3.8 The canonical form of the pinned-AGs in 3D

The transformation from pinned-AG in 2D into 3D, is straightforward. In the fundamental extension we replace each ground edge with a triangle, but this time two edges come out from each of the two new vertices.

3.8.1 Transformation of a 3D AG into pinned-AG

The transformation from AG into a pinned-AG is done, this time, by choosing arbitrarily an edge and pinning down all edges incident to that edge, as shown in Figure 23.

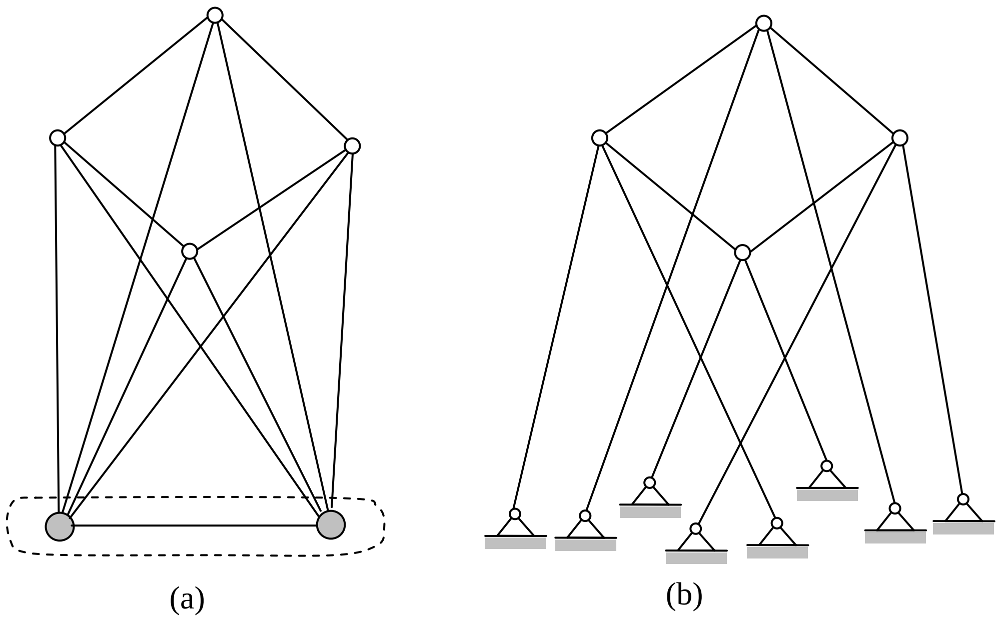
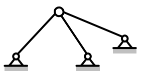
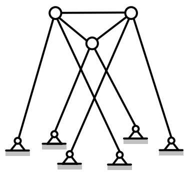
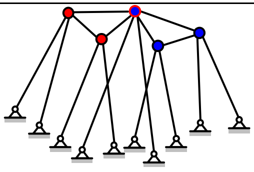
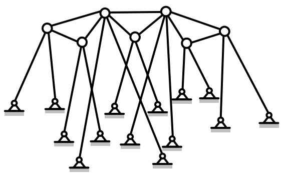
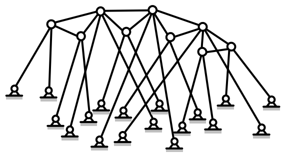
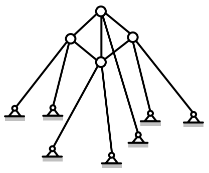
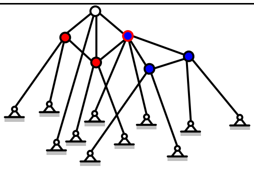
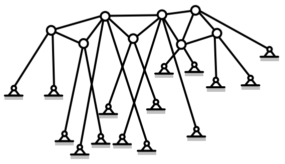
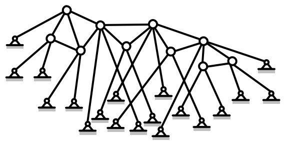
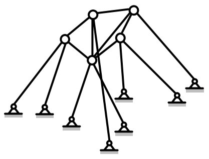
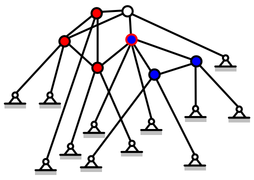
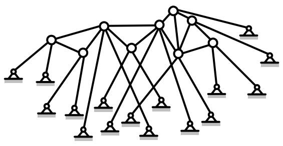
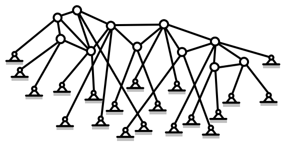
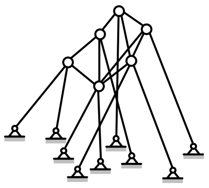
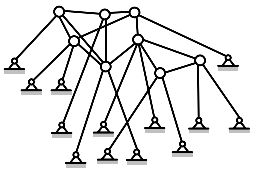
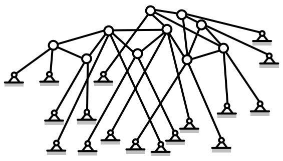
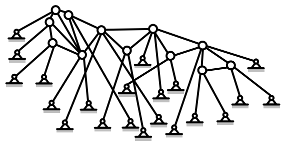
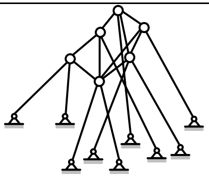
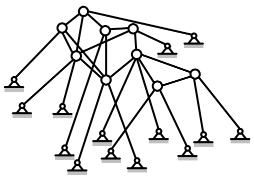
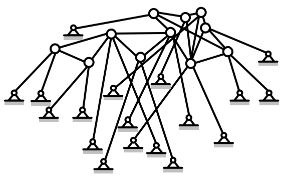
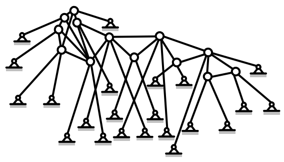


Figure 23 – Transforming a 3D AG into a pinned AG.
a) The 3D AG – Tetrad. b) The corresponding pinned-AG.

3.8.2 The table of the canonical form of 3D pinned AG

The structure of the canonical form is the same as in the 2D pinned-AG. The first column consists of the 3D dyad, which does not have a correspondence in the 3D AG, thus there are no derivations in that column. The first row consists of the f-pinned AGs, and each column contains all the derivations from that representative using 2-extension operation.

Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
				
				
				
				
				

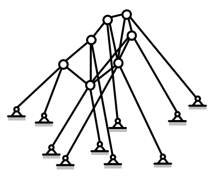
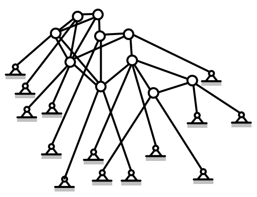
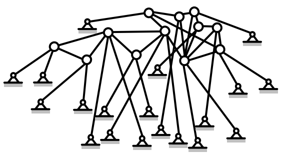
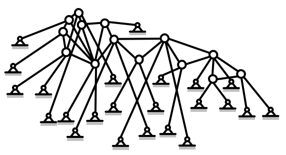
Class 1	Class 2	Class 3	Class 4	Class 5
				

Table 4. The table of 3D pinned-AGs.

4 Characterization of pinned AGs through pinned pebble game

4.1 The pinned pebble game

The current section presents a modification of the pebble game algorithm coming to treat pinned Assur graphs in contrast to the regular graphs treated by the original algorithm. The need for such a modification stems from the fact that pinned AGs are different from regular graphs due to their pinned vertices. The slight modification of the algorithm introduced here takes into consideration the pinned vertices and the ground edges. Therefore, we call this algorithm a "pinned pebble game". In this section we introduce the pinned pebble game for 2D, however, it can be easily extended to 3D by assigning three pebbles to each vertex instead of two.
The steps for the pinned pebble game can be formulated as follows:

1. Assign two pebbles to each inner vertex (not pinned), thus N(v)=2 where N is the number of the pebbles associated with the inner vertex 'v'.
2. Search for an edge whose two end vertices have two free pebbles.
   Then, assign a pebble from one of the end vertices to the edges. Accordingly assign the edge with orientation, such that the vertex from which the pebble has been taken is the tail vertex of the directed edge.
4. Suppose there are two adjacent vertices 'x' and 'y' and one of them, say 'x', has N(x) < 2. Search for a directed path from 'x' to a vertex, say ‘z’, with free pebbles. Then, move one pebble from vertex z to x along the directed path, assign: N(z) = N(z)-1 and N(x) = N(x) + 1, and reverse the directions of all the edges along this path.
5. After the pebbles have been assigned to the inner edges, assign one pebble to each of the ground edges. If needed, since all the inner edges have been assigned, it is possible to transfer pebbles through the inner edges by changing their directions.

Figure 24 depicts the process of applying the pinned pebble game on a pinned AG. At first, the inner edges <CA>, <AB>, <DE>, <EC> are assigned with pebbles since both of their end vertices have two pebbles. Then, pebbles are transferred through the directed edges, as done in Figure 24c where a pebble is moved from vertex A to C (thus the direction of edge AC has been changed) so that edge BC can be assigned, and so forth. At the last stage, it can be easily verified that all the pinned edges are directed towards the ground and the inner edges of the Assur graph induce a well connected graph as shown in Figure 24f. Theorem explained below argues that this is the characteristic of any pinned AG (and only a pinned AG), to which the pebble game algorithm has been applied.

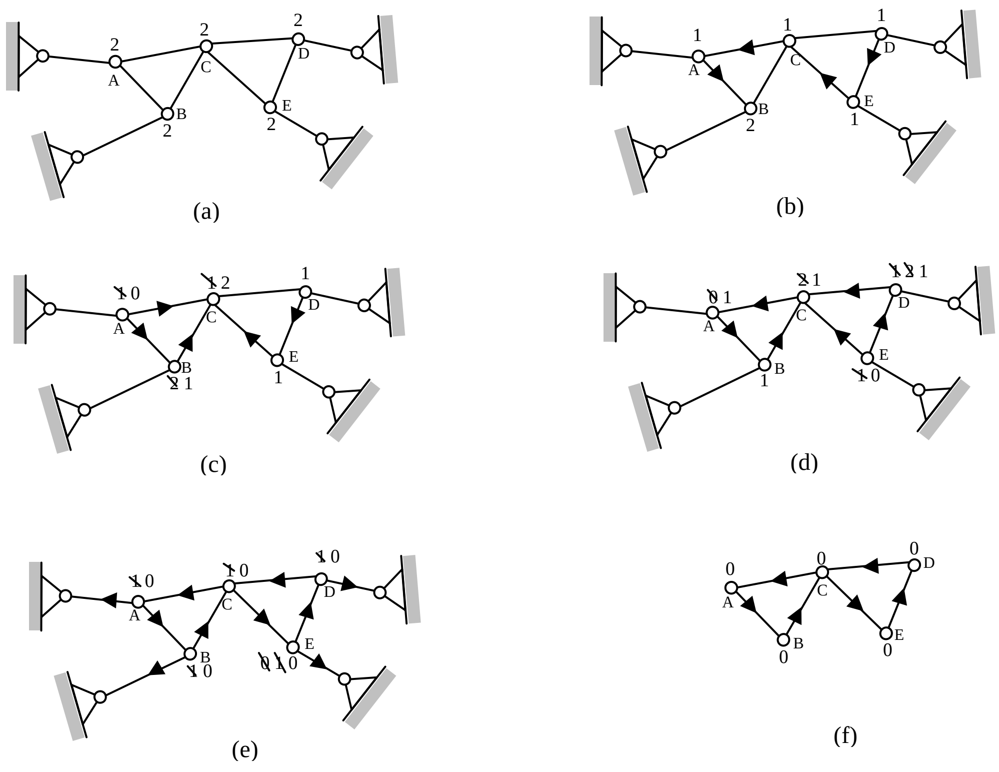


Figure 24 - The stages of applying the pinned pebble game on a given pinned AG

4.2 The physical interpretation of the directed graph resulted from the pebble game

As was mentioned in the previous section, after applying the pinned pebble game on a pinned AG, all the edges become directed and all the vertices have zero pebbles. The physical interpretation of the direction of an edge is that there is one constraint related to the location of the tail vertex, while the head vertex is assumed to be known or fixed. In 2D (3D) we need two (three) edges to define a joint (point) and three (six) edges for a body, respectively.

For example, in Figure 25, the three edges 1,5, 6 define the location of the body {7,8,9} and since there are no edges entering the vertices {A,B,C}, it means that there are no vertices in the graph whose locations are dependent upon these three vertices. In other words, removing this set of vertices will not affect the locations of the other vertices, i.e., the rigidity of the other vertices is not dependent on the existence of this sub-graph. The latter property fully correlates with the property of decomposing a graph into Assur graphs. Now, let us see why this graph is a pinned Assur graph. First, the sub-graph is rigid since all the inner vertices have zero pebbles. Since the sub-graph {A,B,C} is well-connected, and in this case it is even a directed circuit, it means that the location of each vertex is dependent on the location of all other vertices, i.e. all the vertices in this sub-graph are dependent on the locations of the other vertices, namely, we cannot remove any vertex from this sub-graph without affecting the locations of the other vertices. This property confirms that this sub-graph is a pinned AG.

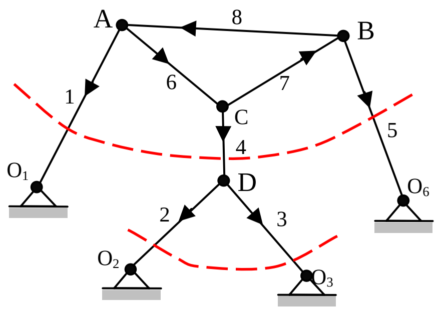


Figure 25 - Example of the edge orientation resulted from applying the pinned pebble game.

Theorem: Suppose G is a pinned isostatic graph and $\vec{G}$ is the directed graph whose edge directions were obtained due to the pinned pebble game. $\vec{G}$ is a pinned AG IFF it does not contain a directed cutest consisting of inner edges.

Proof: G is strongly connected graph iff it does not contain a directed cutset. From the physical interpretation of the pebble game explained in above, it follows that there is no set of vertices such that the location of all the rest of the vertices in the graph is dependent on them (the cutset is directed). Since all the vertices have zero pebbles associated with them (due to the completion of the pebble game) this means that there is no set of vertices whose removal results in a rigid graph, which is equivalent to the definition of a pinned AG.

5 The canonical form of pinned isostatic graphs (Deteminate trusses), Laman graphs and Linkages

In the previous sections it was shown that in the canonical form, the Assur graphs are the **modular components** of all other types of graphs introduced in this paper. Moreover, since many types of the graphs represent engineering systems, we introduce the canonical form of engineering systems, such as: linkages; determinate trusses; tensegrity systems and more.

5.1 Constructing pinned isostatic graphs from Assur graphs

As was introduced in section 3.6, every pinned Assur graph consists of two types of vertices: inner and pinned vertices. We start with an Assur graph and add another Assur graph to it by joining the pinned vertices of the latter to either the inner vertices of the former or to the ground.
The composition of two Assur graphs G1 and G2 yields a pinned isostatic graph, where X is the set of the pinned vertices of G2 that are connected to the inner joints of G1 or in mathematical form:

Composition rule of Assur Graphs: Let $G_1=(I_1,P_1;\ E_1)$ and $G_2=(I_2,P_2;\ E_2)$, then the composition of $G_2 \overset{X}{\mapsto} G_1$ yields:
$G_3 = (I_1 \cup I_2, P_1 \cup P_2 - X; E_1 \cup E_2)$, and accordingly: $| E_1 \cup E_2 | = 2|I_1 \cup I_2|$.

Example of a composition of two tryads appears in Figure 26, where different pinned vertices of G2 are connected to the inner joints of the tryad G1.

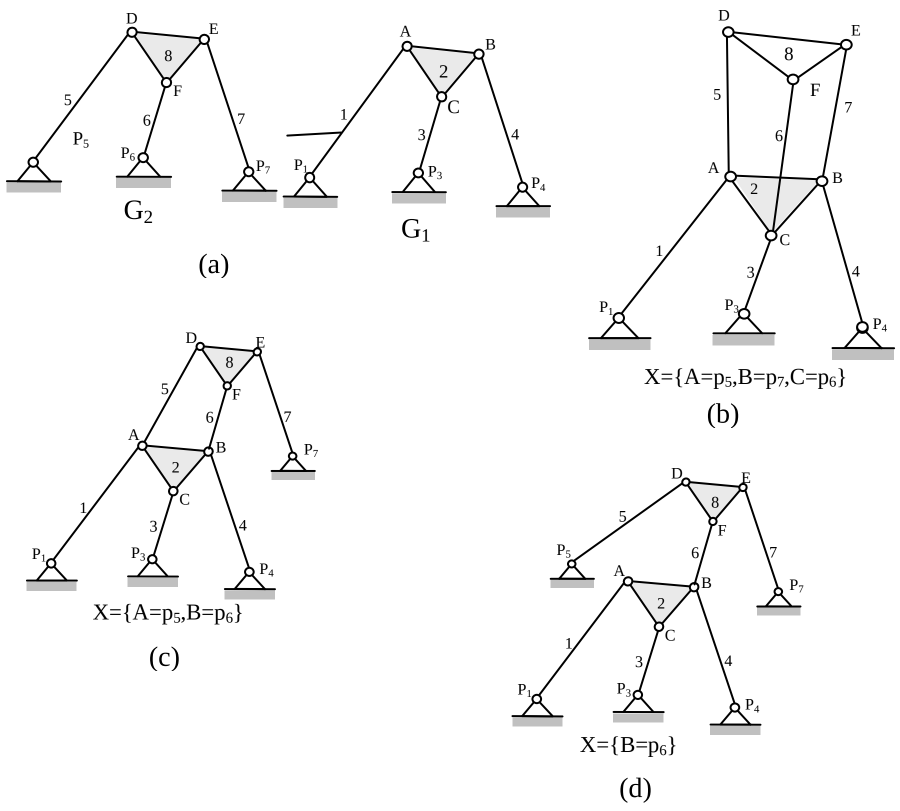


Figure 26 - Example of a composition of two tryads.

In general, pinned isostatic graphs can be constructed from AGs in various ways, composing simultaneously and serially several AGs, under the following theorem.

Theorem: composition theorem – let G be a pinned isostatic framework and G_1 be an AG. Then the connection of G_1 to G will be rigid if and only if these two counterparts are connected through at least d vertices, where d stands for the number of dimensions.
The poof of this theorem becomes transparent, once considering the following lemma from mechanical engineering.

Lemma: for any two rigid bodies in dimension d there is need of at least d connections so that the whole two bodies will be rigid.

Proof: Suppose there are two bodies with k connections in dimension d, and let p be an arbitrary point in one of the bodies (Figure 27). For each connection we can express the square distance from the connection to the point, as follows:

$$(P - O_k)^2 = \text{const.}$$

The derivation of the equation yields d variables, corresponding to the velocity component of the point in each coordinate axis. To assure that the all the velocity components are equal to zero, we should have at least d equations, i.e., at least d connections.

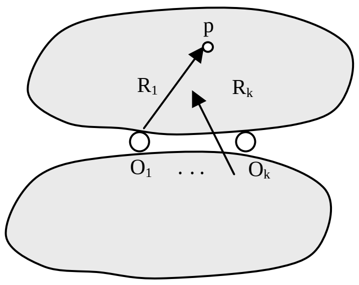


Figure 27. Number of connections needed between two bodies in rigid structure.

5.2 Decomposition of pinned isostatic graph into Assur graphs

Theorem – The canonical form of pinned Isostatic Graph: Every pinned isostatic graph is a composition of Assur graphs.

Proof: the proof is based on the fact that every pinned isostatic graph is decomposed into Assur graphs. This can be shown by applying the pinned pebble game, and showing that each directed cut-set defines another Assur component. Another proof based on rigidity circuits appears in (Servatius et al., 2006).
Figure 28 depicts an example of decomposition of a pinned isostatic graph. Figure 28b indicates on the directions of the edges after applying the pinned pebble game. There are three directed cut-sets indicating that there is a decomposition into three Assur graphs, which can be thought also as consisting of three modular components. The first component is the Tryad (Figure 28c), the second is another Tryad (Figure 28d) and the last one is the dyad (Figure 28e).

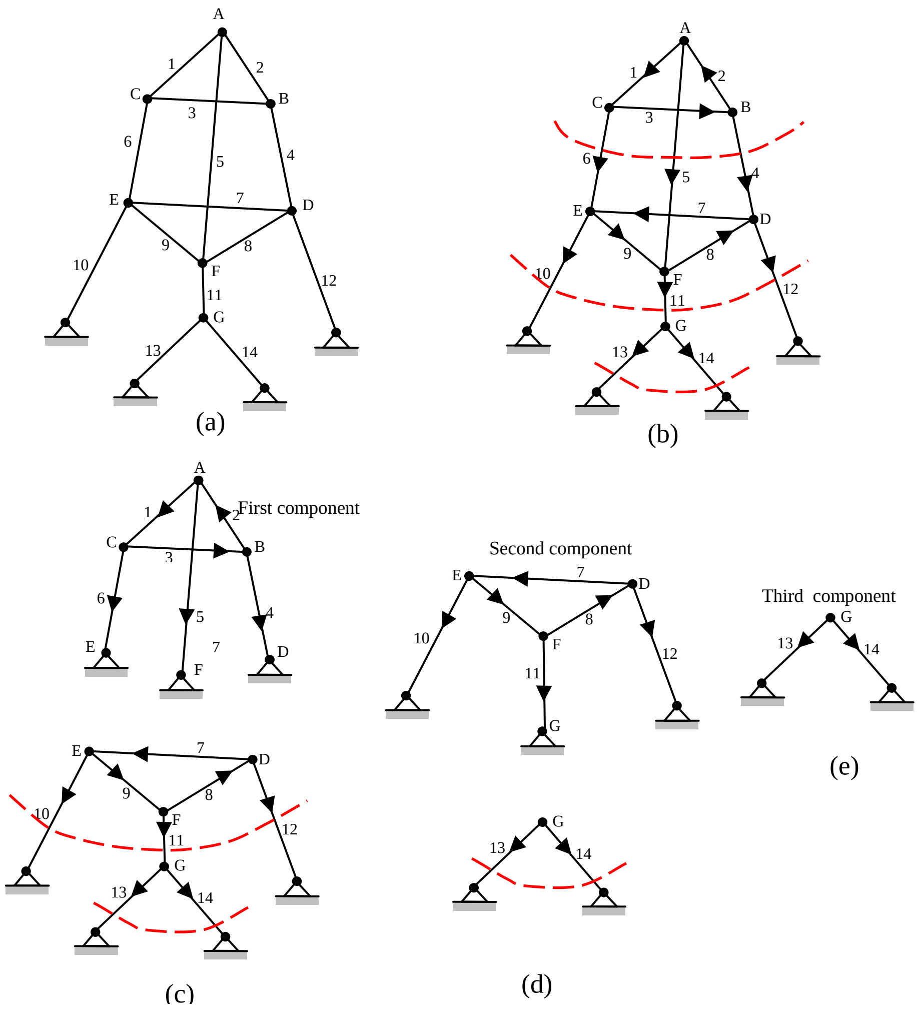


Figure 28 – Example of a decomposition of a pinned isostatic graph into Assur graphs.

The idea of decomposing isostatic frameworks into pinned-AGs may assist in revealing rigidity problems in the framework. For example, when we decompose the known double banana structure into pinned-AG as shown in Figure 29, one can perceive that the connection between the pinned-AG and the rest contradicts the composition theorem (5.1), thus the whole graph is not rigid.
In general, if you have a triangle then all the six free pebbles are recommended to be associated to the vertices of the triangle and will be: 3,2 and 1. If there is no triangle, then other considerations should be taken when we check the decomposition process.

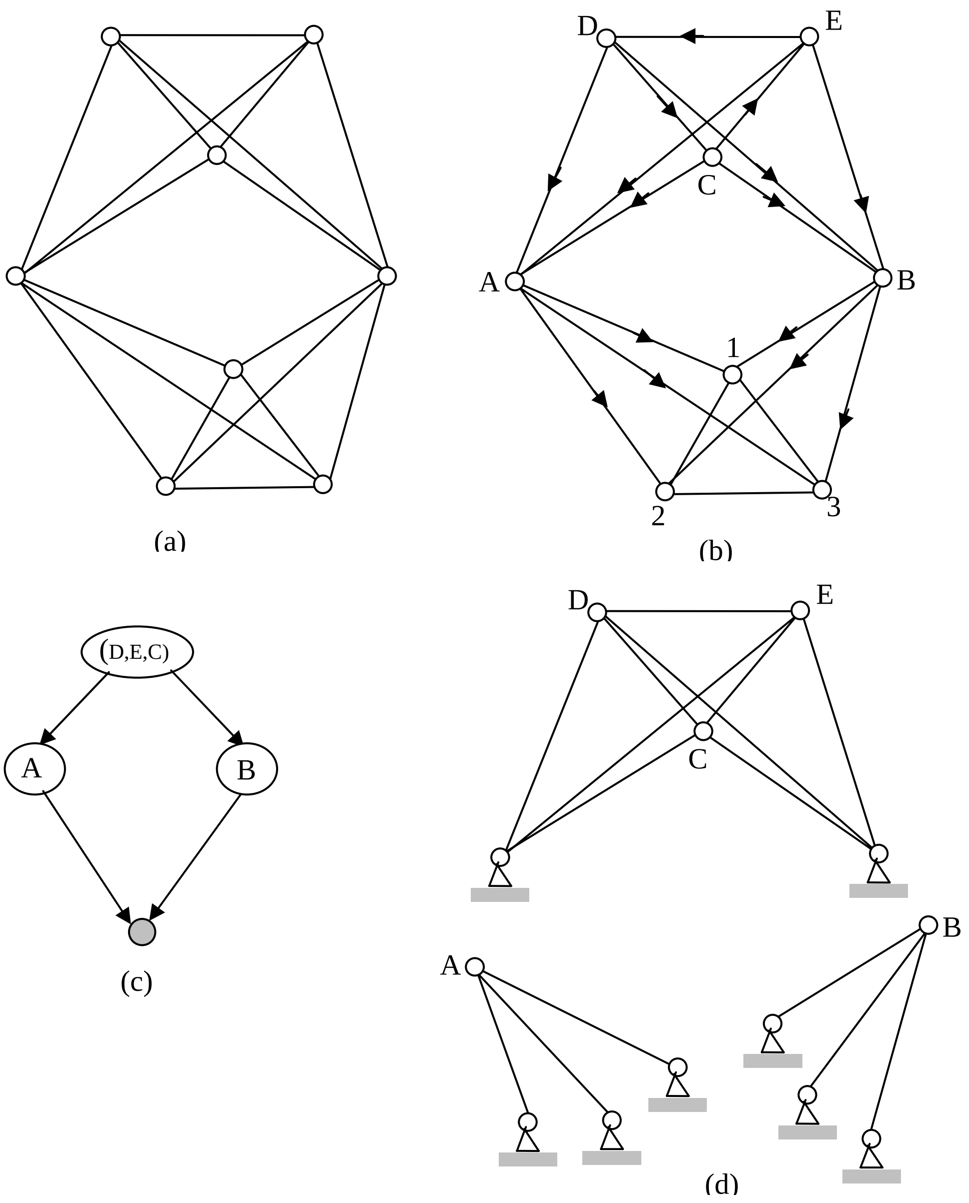


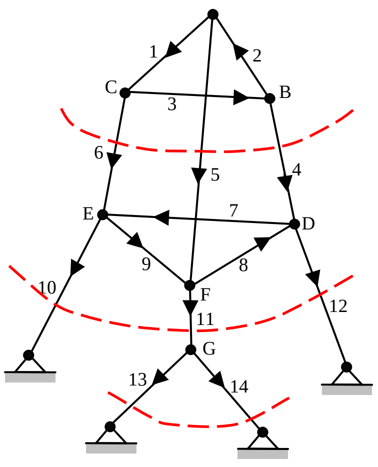
Figure 29 - The decomposition of the double banana into pinned-AGs.
a) the double banana. b) the graph after applying the pebble game. C) the decomposition graph into pinned-AG. D) The pinned-AGs of the constituting the double banana.

6 The canonical form of the rigidity matrix of pinned isostatic graphs

Since the Assur graph has special property related to rigidity – minimal rigid related to vertices, thus this property is also expressed in its rigidity matrix, as appears in the following theorem:

Theorem: A rigidity matrix of a pinned isostatic graph is block upper (lower)-diagonal IFF it is the rigidity matrix of a graph that is not Assur graph, i.e., it is decomposable.

Proof: Necessity. By induction. Let us choose the first component, which is a square matrix. Let us delete its rows and columns, yielding the deletion of the component itself and the deletion of the edges connecting it to the other parts. We are left with a graph G' that satisfies |E'|=2*|V'|. The process continues until the last component remains.
Sufficiency. Suppose the graph is decomposable. Let us choose the component that all of its pinned joints are connected to the ground, thus it will be the last component on the diagonal. Then, choose a component that its pinned joints are connected to the inner joints of the last component and to the ground. This will be the n-1 component on the diagonal and so forth.
Example of the rigidity matrix corresponding to the pinned isostatic graph appearing in Figure 30a is depicted in Figure 30b. Since the graph consists of three Assur graphs, we can see three components on the diagonal of the matrix.



	Ax	Ay	Bx	By	Cx	Cy	Dx	Dy	Ex	Ey	Fx	Fy	Gx	Gy
1	▨	▨			▨	▨								
2	▨	▨	▨	▨										
3			▨	▨	▨	▨								
4			▨	▨			▨	▨						
5	▨	▨									▨	▨		
6					▨	▨			▨	▨				
7							▨	▨	▨	▨				
8							▨	▨			▨	▨		
9									▨	▨	▨	▨		
10				0					▨	▨				
11											▨	▨	▨	▨
12							▨	▨						
13										0			▨	▨
14													▨	▨

Figure 30 – The canonical structure of the rigidity matrix (b) corresponding to the pinned isostatic graph (a).

6.1 The Canonical form of Laman graphs

The common definition of Laman graphs is that it satisfies rigidity conditions. Many works have been done for checking whether a given graph satisfies the required criteria, such as: Two edge-disjoint spanning trees (Rescki, 1989) and more.
By adopting the approach of using Assur graphs as is done in the paper, it is also possible to find a way to construct them or to decompose them into Assur graphs.

The first step is to transform the Laman graph into a pinned isostatic graph, which can be done by choosing arbitrarily an edge and pinning its two end vertices. Then, all the process described in this paper is applicable to the transformed pinned graph.
Example of such transformation appears in Figure 31, where the pinned graph is decomposed into four Dyads (Figure 31b).

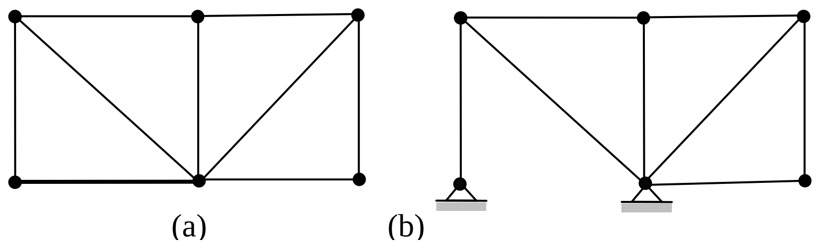


Figure 31 – Example of transforming a Laman graph into pinned isostatic graph.

This transformation enables to use Assur graphs for checking rigidity of graphs, as follows.

Theorem: Let G be a graph with e(G)=2*v(G) – 3. Then G is rigid IFF after pinning one of its edges it is decomposable into Assur graphs.

Proof: If G is rigid then the corresponding pinned graph is rigid. Thus according to theorem from section 6 it is decomposable into Assur graphs.
If the pinned is decomposable into Assur graphs thus its rigidity matrix corresponds to a pinned rigid graph thus the G is rigid.

It follows from this theorem, that when the graph is not rigid, regardless of which edge is pinned, there is no decomposition into Assur graphs. Figure 32 depicts such a case, where no matter which edge is pinned, whether it is in the part that is under-constrained (Figure 32b) or in the over-constrained (Figure 32c, d), there is no decomposition into Assur graphs.

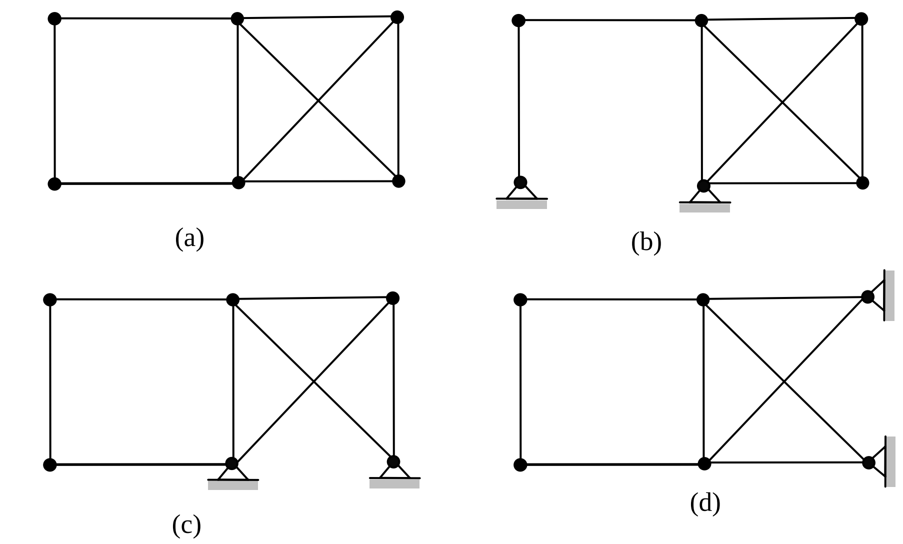


Figure 32 – Example of transforming a graph that is not rigid into a pinned graph.

7 Canonical form of tensegrity systems

In this section we introduce only those tensegrity structures that have only one independent self-stress on all the edges, termed in the literature also: rigid circuit, r-tensegrity, one time indeterminate and more.
The topology of the r-tensegrity is actually the same as the topology of an AG, thus the canonical form for r-tensegrities in 2D and 3D is also described in Table 1 and Table 2, respectively

To characterize the geometry of the tensegrity system we use the parallel drawing concept augmented with the virtual work principle, the main idea for which is described in the following section.

7.1 Parallel drawing augmented with virtual power.

Virtual power is a scalar product between the force acting on a vertex and its linear velocity vector, i.e., multiplying the magnitude of the force by the projection of the velocity onto the direction of the force, $V\cdot \mathrm{Cos}(\alpha)$, as appears in the following equation:

$$P = \vec{F}\circ\vec{V} = F\cdot V\cdot \mathrm{Cos}(\alpha).$$

Suppose we rotate the velocity vector by 90 degrees, let us designate it by $\vec{V}^{\perp}$, then the projected part is opposite to the angle between the force and the rotated velocity, yielding a vector product, as follows:

$$P = V^{\perp}\cdot \mathrm{Sin}(90-\alpha)\cdot F = \vec{V}^{\perp}\times\vec{F}.$$

The latter equation resembles the equation of a moment. Thus, if we treat the vector $\vec{V}^{\perp}$ as a location vector we derive a new physical interpretation for this equation which is the moment exerted by the force around a point defined by the radius vector $\vec{V}^{\perp}$.
Since parallel drawing, by its definition, yields the perpendicular velocities of the vertices thus when forces are applied in parallel drawing, the sum of the virtual power becomes the sum of the moments exerted by the forces acting on the vertices.

7.2 Geometric considerations using parallel drawings with virtual power in tensegrity structures

In r-tensegrity structures, there is at least one vertex with degree three, for which we can apply the following two rules:
The Arrow rule – if there exists an angle between two edges whose degree is greater than 180, then the edges between this edge are of the same type (cable vs. strut) and the remaining edge (the one between them) is of the other type.

The Y rule – if the angle between any two edges is less than 180, then all of the edges are of the same type (either cables or struts).

Thus, for any tensegrity system it is easy to verify whether an edge is assigned with a proper type if one of its end vertices is with degree three. But, when both of its end vertices are with degree four it is difficult to verify the type of the element. Adopting the canonical decomposition idea as appears below might assist in this matter.

1. Let (x,y) be the edge whose two end vertices are with degree greater than three. Remove this edge from the graph and replace it with two external forces acting on vertices x and y according to the type of the element.

2. Search for an edge one of whose end vertices is with degree three. Let us designate it with (z,w) where d(w)=3. Replace the edge by two forces acting on the two vertices according to the type of the element. It should be noticed that the replaced forces are proved to be in the correct direction.

3. Now, we are left with a linkage. Apply pebble game or choose a proper driving link and ground edges including one of the end vertices of each of the removed edges. If needed, go to step 2 and choose another edge with an end vertex with degree three. Let the vertices w and x be the vertices not included in the ground.

4. Decompose the linkage into pinned-AGs.

5. Apply the parallel drawing method to the linkage resulting in the velocities of the vertices x and w.

6. According to the sign of the moments exerted by the forces Fx and Fw verify the type of the edge (x,y).

Example: Following is the demonstration of the idea explained above through an example. In Figure 33 we have a tensegrity structure where the type of edge 7 is not known, and since the degrees of vertices C and O are four and five, respectively, we cannot employ neither the arrow nor the Y rules.

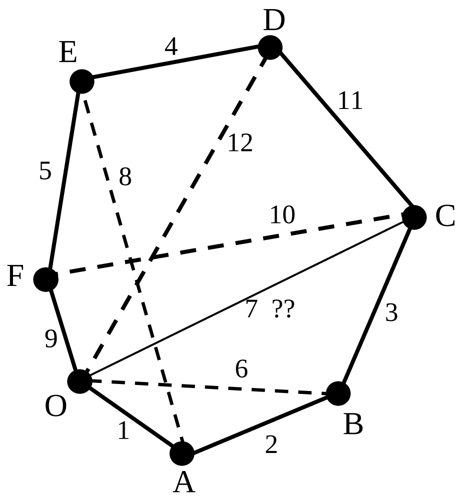


Figure 33 - Example of a tensegirty structure where the type of one of its edges is not known.

Figure 34a depicts the structure after replacing edges 12 and 7 by forces, where the type of edge 7 is not known and assumed, arbitrarily, to be a cable. The driving link was chosen to be 9 and the ground edges {1,2,8}. The graph is decomposed into dyads, as can be seen in Fig… c, thus parallel drawing is constructed easily. The signs of the two moments: $\vec{V}d\times\vec{F}_{12}$ and $\vec{V}c\times\vec{F}_{7}$ are of the same type indicating that their sum cannot be equal to zero, thus it can be concluded that the type of edge 7 is strut.

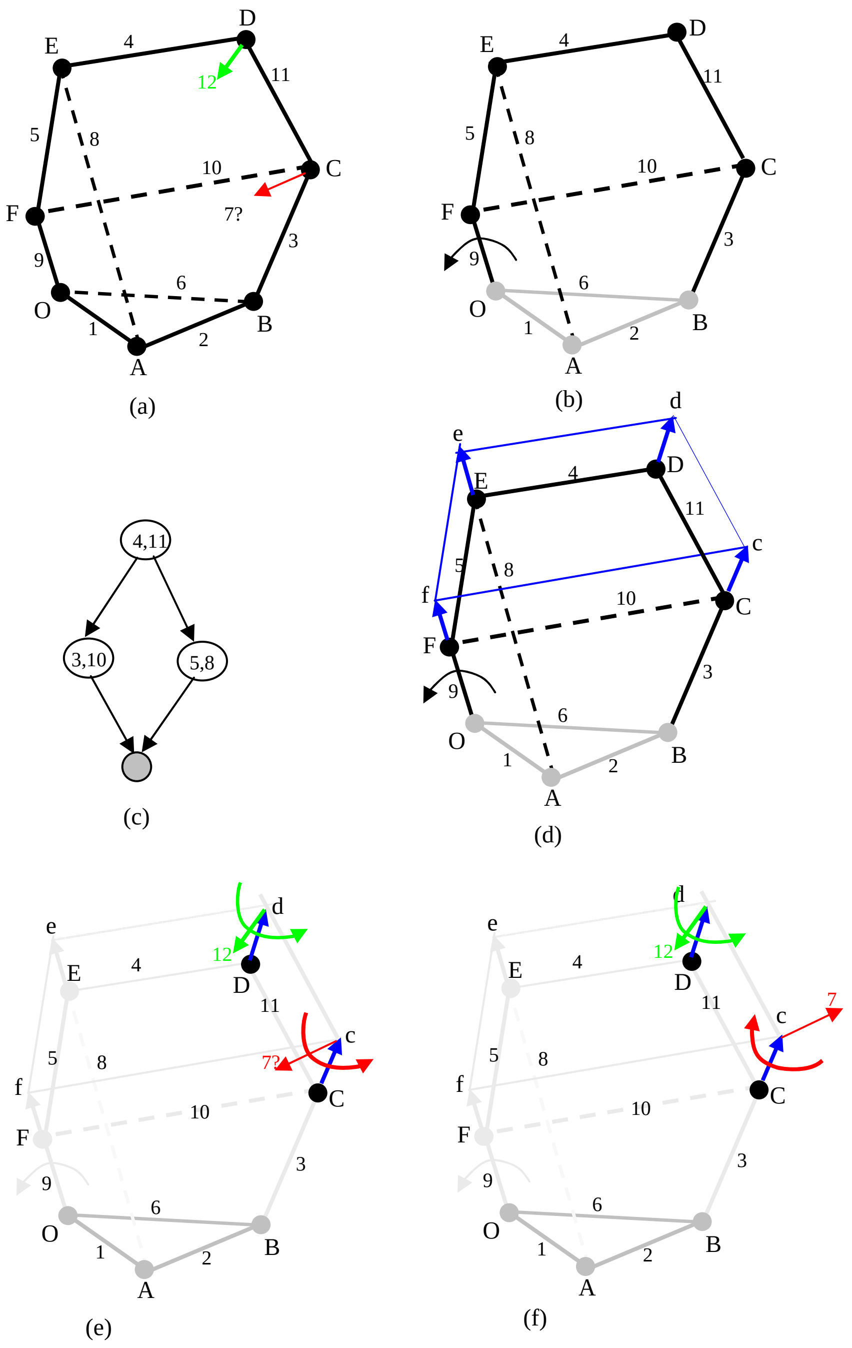


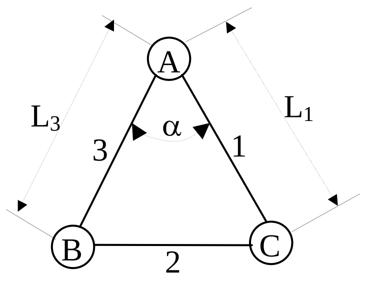
Figure 34 – The steps of applying the parallel drawing with the virtual power on the given tensegrity structures.

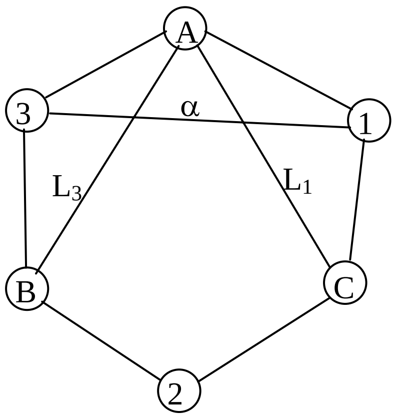
8 Application of the canonical approach for geometric constraint systems

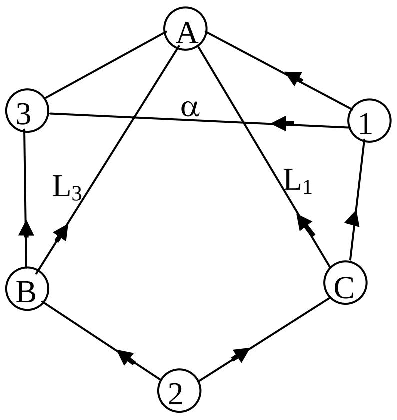
One of the known problems in CAD is checking whether the given geometric data is sufficiently constrained to be able constructing the corresponding geometric objects. In this respect, one may adopt the approach proposed in this paper for facilitating the analysis of the geometric constraint system, through execution of the following steps:

1. Construct the constraint graph   (reference).
2. Apply pebble game yielding a directed graph.
3. Construct the decomposition into pinned AGs.
4. Following the hierarchic order of the decomposition, construct the geometric entity.

Example of applying the proposed method is depicted in Error! Reference source not found. where the constrained graph (Figure 35b) is decomposed into pinned-AGs according to the result of applying the pebble game.







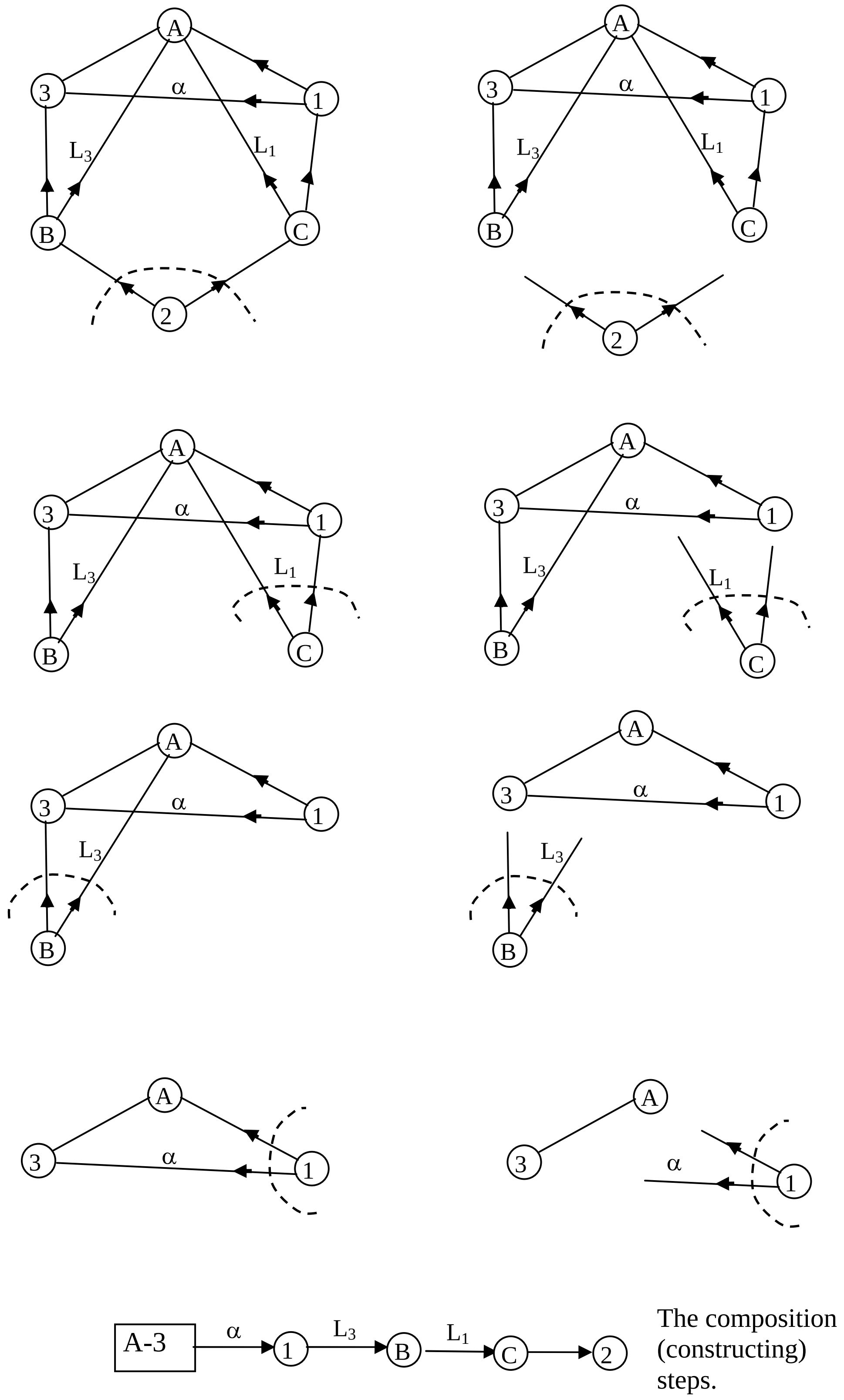


Figure 35 - The decomposition steps of the geometric constraint graph following the order of the decomposition into pinned AGs.

It should be noticed, as was explained in section 5, that there is a unique decomposition into pinned AGs, however, the order of the decomposition is not unique.
In this example, from the hierarchic order of the decomposition graph, it can be easily seen that there are two sets of steps, namely: <1,B,C,2> or <1,C,B,2>.

Once we have the decomposition graph, which describes the order of the decomposition into the pinned AGs, it is possible to follow the decomposition graph in the reverse order and thus construct the geometric system as is shown in Figure 36.

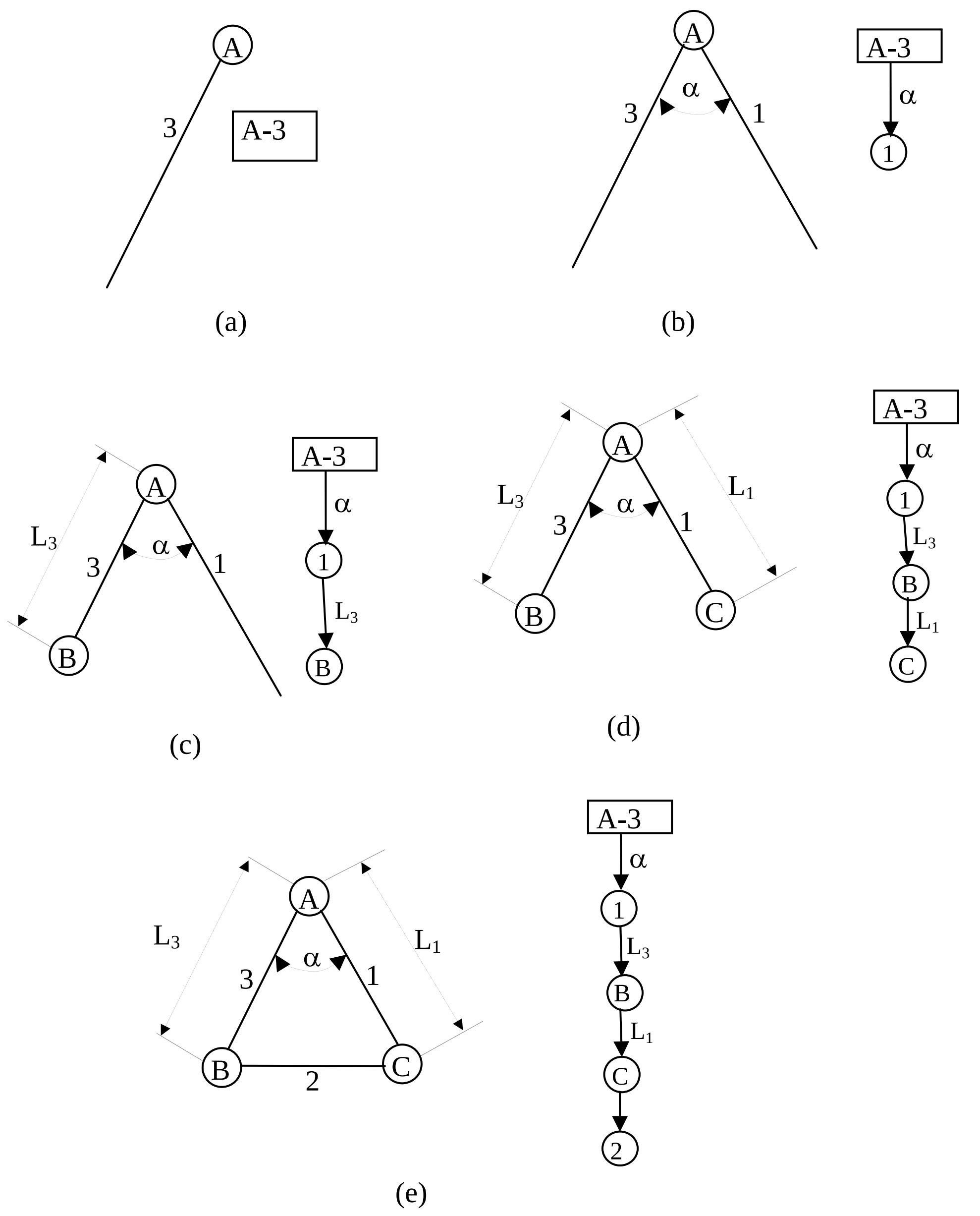


Figure 36 – The steps for constructing the geometric system from the decomposition graph.

9 References

Assur, L.V., Issledovanie ploskih sterzhnevyh mehanizmov s nizkimi parami s tochki zrenija ih struktury i klassifikacii. Akad. Nauk SSSR, 1952. Edited by I.I. Arotobolevskii

Berg A. R. and Jordan T., A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid, *J. Combin. Theory Ser. B,* 88(1), pp. 77-97, 2003.

Graver J., Servatius B. and Servatius H., Combinatorial rigidity, volume 2 of Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, 1993.

Recski A. Matroid theory and its applications in electric network theory and in statics. Berlin: Springer-Verlag, 1989

Servatius B., Shai O. and Whiteley W., Combinatorial characterization of the Assur graph from engineering, Preprint, 2008.

Servatius B., Shai O. and Whiteley W., Geometric properties of Assur graphs, Preprint, 2008.

Swamy MN, Thulasiraman K. Graphs: networks and algorithms. New York: Wiley, 1981.

More relevant are going to be added.

10 Acknowledgements

This work has been supported by the IDF (Israel Defense Forces – Mafat), grant number 0605614102. The authors would like to express his gratitude for the support.