Proceedings of the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2011
August 29-31, 2011, Washington, DC, USA

DETC2011-48340

CHECKING MOBILITY AND DECOMPOSITION OF LINKAGES VIA PEBBLE GAME ALGORITHM

Adnan Sljoka
Department of Mathematics and Statistics
York University, 4700 Keele Street
Toronto, ON M3J1P3, Canada
Email: adnanslj@mathstat.yorku.ca
Offer Shai
Faculty of Engineering, Tel-Aviv University. Email: shai@eng.tau.ac.Il
and
Walter Whiteley*
Department of Mathematics and Statistics
York University, 4700 Keele Street
Toronto, ON M3J1P3, Canada
Email: whiteley@mathstat.yorku.ca

ABSTRACT

The decomposition of linkages into Assur graphs (Assur groups) was developed by Leonid Assur in 1914 - to decompose a linkage into fundamental minimal components for the analysis and synthesis of linkages. In the paper, some new results and new methods are introduced for solving problems in mechanisms, bringing in methods from the rigidity theory community. Using these techniques, an investigation of Assur graphs and the decomposition of linkages has reworked and extended the decomposition using the well developed mathematical concepts from theory of rigidity and directed graphs. We recall some vocabulary and provide an efficient algorithm for decomposing 2-dimensional linkages into Assur components using strongly connected decompositions of graphs and a fast combinatorial Pebble Game Algorithm, which has been recently used in the study of rigidity and flexibility of structures and in fast analysis of large biomolecular structures such as proteins. Working on a one degree of freedom mechanism, we apply our algorithm to give the Assur decomposition. The Pebble Game Algorithm such a mechanism is presented, along with an overview of the key properties and advantages of this elegant algorithm. We show how the pebble game algorithm can be used in the analysis and synthesis of linkages to mechanical engineering community. Core techniques and algorithms easily generalize to 3-dimensional structures, and can be further adapted to entire suite of other (body-bar) types of kinematic structures.

Key Words:linkages, mobility, decomposition, Assur graphs, directed graph, pebble game algorithm, strongly connected components.

1 Introduction

This paper introduces a novel algorithm for dealing with mobility and redundancy in mechanical systems. The main idea is new. It gives a physical interpretation, pebbles, for the meaning of degrees of freedom.

The problem of identifying the correct mobility of mechanisms attracts a lot of researchers, for example [26]. The most simple equation to compute the mobility of a mechanism is the

*Supported by a grant from NSERC (Canada).

Chybytchev-Grubler [8] formula, which for simplicity will be referred to in this paper as Grubler equation. The idea underlying Grubler equation and others is that each link has d DOF (degrees of freedom) and once we connect between two links, two rigid bodies, we add constraints thus the DOF is reduced. In the rigidity theory community, on which the pebble game relies on, the calculation of DOF is done differently. In this community we start with the free joints and begin to add links and constraints, that way we reduce the DOF till the actual DOF is obtained. This algorithm also treats the DOF in a very unique way. It provides a physical visualization to the DOF by physical entities, pebbles, that simplify the explanation and comprehension of this problem.

One of the advantages of this algorithm is that it resolves some of the instances where Grubler equation gives a wrong answer. The count of the formula must be applied not only once to the entire mechanism but to all subgraphs to ensure there is not redundancy buried in a piece (see §3). In the cases where there is redundancy it also indicates the passive links. The algorithm, as is shown in Section 3.1, indicates those links that are redundant.

The reader should note that the algorithm is topological, i.e., it gives the global mobility but can not relate to special DOFs due to the special geometrical configurations.

Most of the examples appearing in the paper are for mechanisms, but it should be clear that the algorithm can be equally applied to other mechanical systems, such as structures. In trusses, for example, it can indicate whether it is topologically rigid, the degree of indeterminacy of regions and more.

Surprisingly, this algorithm that was originally developed for checking the mobility and rigidity of structures, including mobility of large biomolecules, also provides the key ingredients for the decomposition into Assur Groups, now termed Assur Graphs. As it is shown in the paper, the algorithm gives directions to the elements of the mechanical systems that then define the decomposition as shown in Section 4.

The decomposition into Assur Graphs has recently been developed for three dimensions [17], which gives new insight onto the structures and topologies of Assur Graphs in 3D, previously unknown in the literature. In this paper all the joints are of revolute joints in 2d and spherical joints in 3D. There is an ongoing work to extend the applicability of the pebble game to other types of turning pairs, such as higher turning pairs and more.

2 Notation and Structural Preliminaries

This paper is transferring some techniques and algorithms from the mathematical work in rigidity theory (and related work in structural engineering) to mechanical engineering. The fields have distinct vocabularies, which we need to translate between in a consistent manner.

As an illustration, consider the linkage in Figure 1(a) or the topologically equivalent structural scheme (b). By designating one of the links to be a driver, then pinning the end of the driver to the ground or by adding an extra bar, this becomes a pinned statically determinate graph (Figure 1(c)). The focus of this paper is the study of specific decompositions of such associated pinned structures, in dimension 2, but all of the results extend to dimension 3 (see the last section).

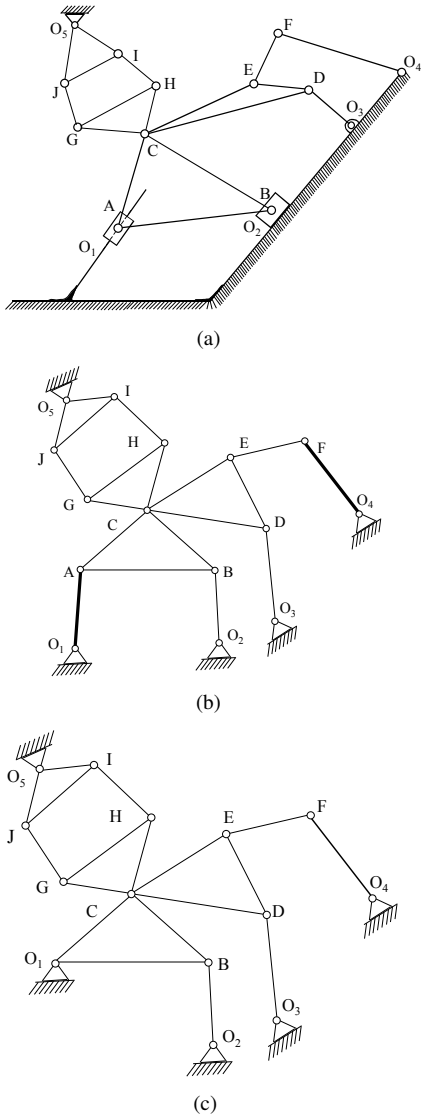


FIGURE 1. A mechanical engineering plane linkage (a) is translated into a flexible pinned structure (b). A mechanical engineer may pin one inner vertex, so that this becomes statically determinate (rigid) (c).

2.1 Pinned Structures, linkages and graphs

In rigidity theory, a *plane pinned bar and joint structure* is a graph, which we call a *pinned graph* $G = (I, P; E)$, with the vertices partitioned into *inner vertices* I, and *pinned vertices* P together with a *configuration* p locating the vertices I, P in the Euclidean plane. A *kinematic motion* of the structure is a displacement of the vertices which preserves the distance between adjacent vertices and does not move the pinned vertices. A pinned structure is *rigid* if the only motions are the zero motion. For more detailed mathematical definitions see [15, 17].

For the rest of this paper, we will focus just on the graph (the topology). We do not discuss any special (singular) geometry which may alter the structural and mechanical behavior, but on the 'generic' behavior (see [15] for details). We define a *pinned graph* by $G(I, P; E)$, where I is the set of inner vertices, P is the set of pinned vertices, and E is the set of edges, where each edge has at least one endpoint in I. Edges that have one *inner* vertex and one *pinned* vertex will be called *ground edges*, others will be called *non-grounded* or *inner edges*. As pinned vertices never move, edges between pinned vertices are not present in pinned graphs.

The refined counts for determining whether a graph can be realized as a minimal rigid structure (extending prior work of Laman [23] for unpinned structures) are the following:

A pinned graph $G = (I, P; E)$ satisfies the *Pinned Plane Structural Conditions* [15] if

1. $|E| = 2|I|$ and
2. for all subgraphs $G(I', P'; E')$ the following conditions hold:
   (i) $|E'| \leq 2|I'|$ if $|P'| \geq 2$,
   (ii) $|E'| \leq 2|I'| - 1$ if $|P'| = 1$, and
   (iii) $|E'| \leq 2|I'| - 3$ if $P' = \emptyset$.

The graphs with these counts are sometimes translated as *statically determinate* in mechanical and structural engineering. There is a unique set of solutions to the forces in the members to a given external loads. In structural engineering, these are also called isostatic [20]. Rigid structures are also described as kinematically determinate structures, with only the zero motion.

The key properties of associated graphs that we want to examine are:

1. *statically determinate graphs* - graphs realizable as statically determinate (minimally rigid) structures for generic configurations.
2. *mechanisms* (mobile structures) with various positive degrees of freedom (DOF): a linkage will be a mechanism with 1 DOF, and more generally, the structure is *mobile*.
3. *independent graphs* - graphs without redudance, so that removing any one edge results in a structure with an added DOF.
4. *redundant graphs* are graphs that are not independent. These may be rigid (kinematically determinate) or mechanisms.

Theorem 2.1 (Pinned Laman Theorem [15]). *A 2-dimensional pinned graph $G = (I, P; E)$ is pinned statically determinate for the plane if and only if G satisfies the Pinned Plane Structural Conditions.*

Corollary 2.2. *If a graph satisfies the modified count: $|E| = 2|I| - k$ for $k > 0$ along with the other inequalities above, then generic structures realizing the graph are k-DOF mechanisms, without redundance.*

The Pinned Laman Theorem is an elegant and remarkable result, and unlike Grubler equation it also gives us a counting criteria on subgraphs. In its original form it is obvious that it gives a poor algorithm, as it requires counting the number of edges in every subgraph, of which there are an exponential number. In the example in Figure 2 it was small enough that we can just visually inspect the counts. In the next section, we will present the Pebble Game Algorithm to efficiently check this count, and generate important additional information for the Assur decomposition.

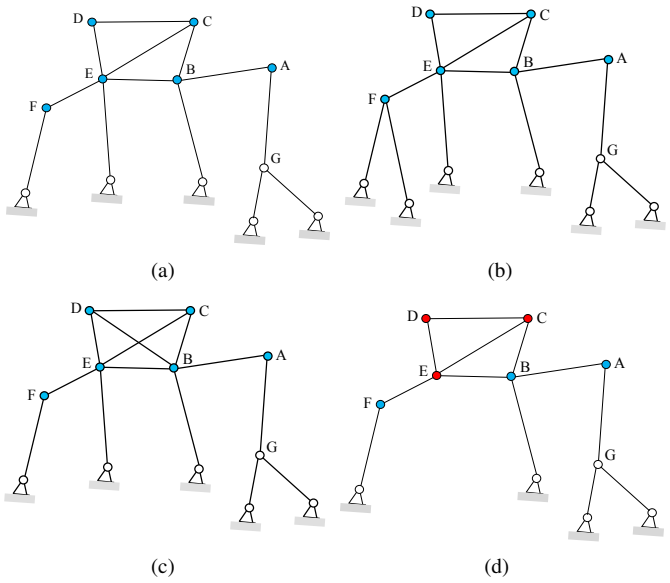


FIGURE 2. A linkage is a pinned statically determinate graph (a) which is one edge (link, constraint) short of minimum required to be rigid. (b) is a statically determinate graph. (c) is a one degree of freedom mechanism with a redundant region (B, C, D, E). (d) Is a two degree of freedom mechanism.

In Figure 2, we have some examples of pinned graphs with seven inner vertices. The vertices that are not labelled with small

triangles are pinned joints. The Pinned Laman Theorem and its Corollary tells us that we need at least $2 \times (7) = 14$ edges to be statically determinate. In (a) we have 13 edges, and by inspection it satisfies the counts on the subgraphs (there are not too many edges in any subgraph), so as it is one edge (constraint) short of minimum required, it should have one degree of freedom. The graph in (b) has 14 edges and satisfies the counts on the subgraphs so it is statically determinate. The graph in (c) has 14 edges, but in the region (B, C, D, E) with four inner vertices it should have at most 2x(4) − 3 = 5 edges, but it has six edges, so that region is *redundant* (overconstrained), and one edge is *redundant* and being wasted bringing the count down to 13, so it should have one degree of freedom. The graph in (d) has 12 edges, so it should have two degrees of freedom.

3 Pebble Game and Assur Graphs

The pebble game algorithm presented here was developed for the theory of statically determinate structures or minimal rigid frameworks [20,23], and builds on counts above, which extend the standard Grubler's [8] counting equations for mechanical linkages to subgraphs in an efficient way. As now developed by authors such as [5, 19], the algorithm will detect the exact degrees of freedom of any plane structure in 'generic realizations' - that is ones without any geometric singularity. A 3D version was more recently utilized in the program FIRST for fast predictions of rigidity and flexibility of large biomolecular structures such as proteins [9]. In this paper, it is applied to an undirected pinned *graph* $G = (I, P; E)$, and will be adapted here to output a directed graph which can be further used to give the Assur Decomposition of the linkage.

3.1 Pebble Game Algorithm

There are many algorithmically precise ways of presenting the Pebble Game Algorithm, with mathematically rigorous statements and speed ups which can be implemented [5]. Here, we give the basic intuitive form that can be understood by wider Mechanical Engineering community.

The pebble game presented here is structured to check the constraint counting conditions given by the Pinned Laman's Theorem 2.1, specifically that is the counts in the *Pinned Structure Conditions*. The reason we need a modified version of the pebble game is that we now have two types of vertices, pinned and inner. For this reason we will call this pebble game the *Pinned Pebble Game Algorithm*, but we also say just the pebble game. Note that we treat the ground edges with different rules, because the pinned vertex is never mobile.

Algorithm 3.1. – ***Pinned 2-dimensional Pebble game algorithm:***

Input: *A pinned graph* $G(I, P; E)$

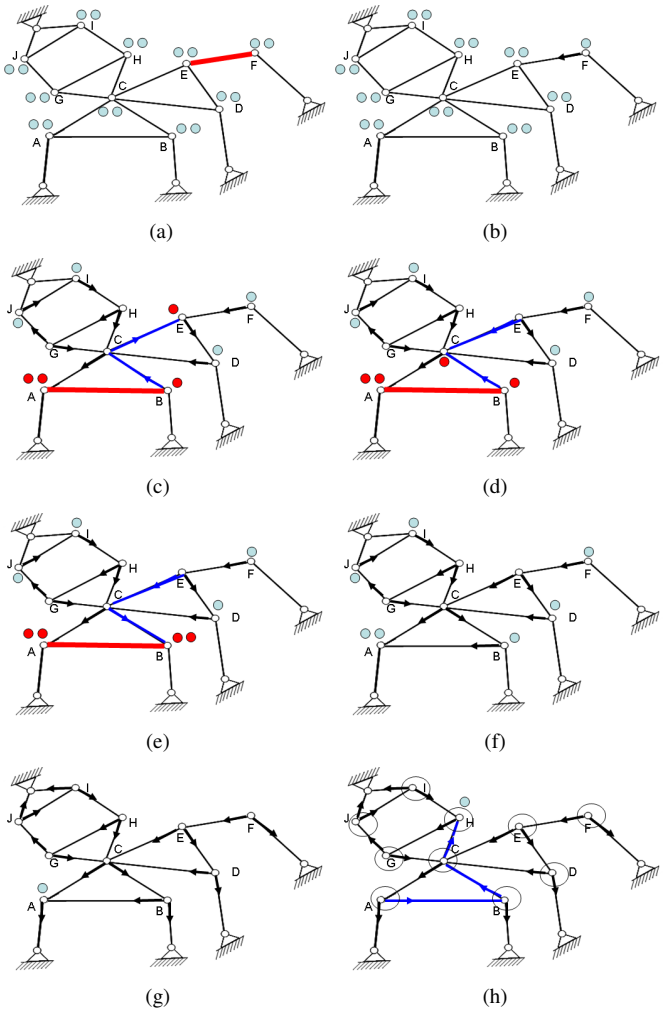


FIGURE 3. Pinned 2-dimensional pebble game (a-g). Inner edges are covered with pebbles when there are four free pebbles on its ends (b). We can search for free pebbles along the directed paths, drawing back the pebble (c, d, e). There is one remaining free pebble, indicating that the corresponding mechanism has one degree of freedom, with the pebble placed at the inner vertex of the driver (g). The remaining free pebble can be moved to all the inner vertices along the directed paths, indicating that all joints are mobile.

Output: *$D(G)$ (set of directed - independent edges) and $R(G)$ (set of redundant edges)*

Initialize $D(G)$ and $R(G)$ to empty sets of edges. Place 2 pebbles on each inner vertex of G.

1. Test all the non-grounded (inner) edges by playing the $2|V| - 3$ pebble game.

While there exists an untested inner edge e in G:

If endvertices of e (u and v) have 4 free pebbles:

Place a pebble from either u or v onto e, directing e out from that vertex. Place e into $D(G)$.

Else: Search for a free pebble from u and v, by following the directed edges in the partially constructed directed graph $D(G)$.

If a free pebble is found on some vertex w at the end of the directed path P: Perform a sequence of swaps, reversing the entire path P, until a free pebble appears on the initial vertex (u or v) of the path P (w looses one free pebble, and u or v gains one free pebble). Check again for 4 free pebbles.

Else: Place e into $R(G)$.

Stop (all inner edges have been tested).

2. Test all the grounded edges by playing the $2|V|$ pebble game.

While there is an untested grounded edge e: Follow the same procedure as in step 1, placing a pebble onto e whenever we have at least 1 pebble on the inner vertex end of e. Place e into $D(G)$, otherwise e is redundant and placed in $R(G)$.

Stop (all ground edges have been tested).

Figure 3 depicts the process of applying the Pinned 2-dimensional Pebble Game on the pinned graph of the linkage we initially introduced. The first step is to assign 2 pebbles to each vertex (a), which tracks the $2|I|$ count. Pebbles are either free on an inner vertex, or they are being used to cover edges. The pebbles represent the degrees of freedom associated to a vertex. We pick any inner edge, and as long as its ends have four free pebbles we can pebble that edge, directing it out of that vertex (b). When we have four pebbles on the ends, it ensures we respect the critical subtraction $2|I|-3$ on the subsets of pebbled edges. With this rule the pebble game ensures we are checking the counting criteria in the Pinned Laman's Theorem 2.1 on all the subgraphs. When we place a pebble onto an edge, it means that the edge is constraining the overall motion of the structure with this graph and it has reduced the overall degree of freedom count by one. At some later stage, when we do not have four free pebbles on the ends of the inner edge, we search for a free pebble along the partially constructed directed graph (c). When the free pebble is found (red pebble on vertex E (c)), we perform a sequence of swaps [19], reversing the paths and the free pebble (d, e). We now have four free pebbles, so we can place a pebble onto that edge (f). Once all the inner vertices have been tested, we place pebbles onto ground edges, making sure we have at least one free pebble on its inner vertex end (g). At this point each edge is directed towards the ground. We never take an edge $e \in D(G)$ that is covered by a pebble and leave it without a pebble (make it undirected). Note that there is no distinction among pebbles, we have just coloured some red in order to depict available free pebbles at the end of the edge being tested or a free pebble that is being reversed back.

The output has several key properties which are immediate.

1. The remaining free pebbles tell us how many degrees of freedom the pinned graph has;
2. A generically rigid pinned graph will be identified by having no remaining free pebbles;
3. A statically determinate graph will have no free pebbles and $D(G) = E$;
4. A mechanism with 1-DOF will have one free pebble.
5. The pebble game is greedy in the sense of computer science, meaning that we will always end up with the same remaining number of free pebbles, regardless of the order the edges are tested.
6. This algorithm is fast: well implemented, it is worst case $O(|V||E|)$ [5], and in practice it is often near linear.
7. the residual set $D(G)$ is a maximal independent set of edges - adding any other edges from the original graph G would not change the degrees of freedom of any part of the structure.

In our graph in Figure 3 there is one free pebble remaining at the end, indicating correctly that this is a one-degree of freedom mechanism. We need to draw the pebble back to the inner vertex of that driver at the end of the game, which indicates that vertex corresponding to the driver is mobile (Figure 3 (g)).

Where any free pebbles come to rest after a play of the game does depend on the order of testing the edges. They can be moved around, by reversing paths. The distribution of free pebbles on the graph, in another words, how free pebbles can be moved around on the graph can provide us a lot of useful information [19]. We recall that at the end of the pebble game on our original graph, we have drawn the pebbles around and placed the free pebble on the end-vertex of the edge corresponding to the driver. If we were not able to draw the pebble back to the end of the driver, then that indicates that the structural topology we are given is not good, as each driver has to have at least one free pebble associated with it. So, knowing where free pebbles can move can help in the the verification and design of mechanisms, including the appropriate placement of drivers.

Furthermore, in (Figure 3 (h)) we see that we can move the one remaining free pebble onto any vertex, as there is a directed path from every vertex to the vertex holding the free pebble, indicating that all the joints are mobile. On the other hand in Figure 4 (a, d, e), the free pebble can never be reversed to joint G, so it is not mobile. In Figure 4 (e) we verify that the pinned graph has 2 degrees of freedom as there are 2 remaining free pebbles. Moreover, one can see that the two degree of freedom are only accessible to joints C, D and E. That is we can shuffle the pebbles around and place the 2 pebbles on any one of these three vertices. Vertices A, B and F can have a maximum of one free pebble.

3.2 Directed Graphs from the Pebble Game

The pebble game also gives directions to the edges of $D(G)$ - going out from a vertex when a pebble from the vertex is placed

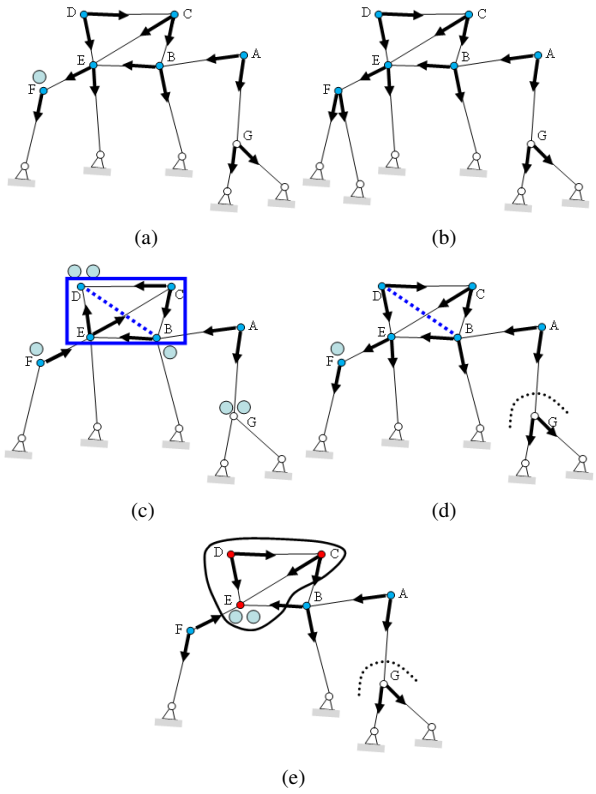


FIGURE 4. Output of the pebble game on 1 DOF pinned graph has one free pebble remaining (a), statically determinate graph with no pebbles remaining and all edges covered by pebbles (b). When we cannot get the fourth free pebble on the ends of an inner edge (c), that edge is redundant, and the region we can search over (B, C, D, E) in the partially constructed directed graph, called failed search region is redundant. Any edge could be removed there without affecting the degrees of freedom. The pinned graph in (c, d) has local redundance, but it still has 1 DOF indicated by a remaining free pebble (d). Joint G is immobile as we cannot get a free pebble there. The graph in (e) has 2 DOF as it has 2 remaining free pebbles. The 2 pebbles are only accessible to joints C, D and E, and maximum one free pebble to A, B and F.

on the edge. More generally, a *direction assignment* $\vec{G}$ to graph $G = (I, P, E)$ is a pair of maps init:$E \to V$ and ter:$E \to V$ assigning to every edge e an *initial vertex* init(e) and a *terminal vertex* ter(e). The edge e is said to be *directed* out of init(e) and into ter(e). In another words, *Direction Assignment* $\vec{G}$ assigns directions to edges of G. We will also refer to $\vec{G}$ as a *directed graph* associated with G. For a given vertex in the directed graph $\vec{G}$ the *out-degree* is the number of edges directed out of that vertex.

The output of the pebble game is a set of directed edges $\vec{D}(G)$, with all edges to the ground directed towards the ground. We will shortly see that such a directed graph is the ideal form for a further decomposition of the mechanism into minimal Assur components. As preparation for this, we highlight a few key properties of the directed graph $\vec{G} = (I, P; \vec{D}(G))$:

1. at each inner vertex, the out-degree is at most 2;
2. the out-degree of each pinned vertex is 0;
3. two plays of the pebble game with the free pebbles (if any) at the same vertices will produce two directed graphs which differ by at most a reversal of cycles [17].

More generally, it is possible that the synthesis of a mechanism used methods that already generated a directed graph, without reference to the pebble game. Will that matter? There are some basic results that confirm that in that situation, once the graph tests to be independent (e.g. by the pebble game), and the given directions (from any source) have the same out-degree at each vertex (the degrees of freedom are the same, and are focused on the same vertices) then the key properties needed for our upcoming decomposition are the same.

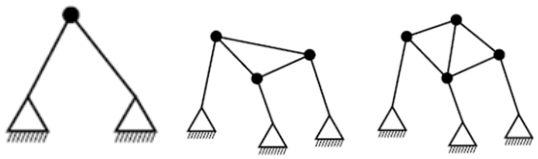
Theorem 3.2 (2-Directed Graphs [17]). *Given two pinned directed graphs $\vec{G}_1$, $\vec{G}_2$ on the same underlying undirected graph G, such that for each vertex in G, the out-degree in $\vec{G}_1$ is the same as the out-degree in $\vec{G}_2$, then the two directed graphs differ only by a set of cycle reversals.*

3.3 Assur Graphs

The concept of Assur graphs (previously named Assur graphs) was originally developed by Leonid Assur in 1914 and is widely used in the kinematical community. A central concept of Assur‘s method is the decomposition of a linkage into fundamental minimal components which serves as an important mechanical engineering tool of analysis and synthesis of linkages. In Mechanical engineering Assur graphs (groups) are viewed as pinned structures with zero mobility that does not contain a simpler substructure of the same mobility [13]. Another definition states that ”An Assur group is obtained from a kinematic chain of zero mobility by suppressing one or more links, at the condition that there is no simpler group inside” [24]. A recent and ongoing investigation of Assur graphs and decomposition of linkages was reformulated using the well developed mathematical concepts from the rigidity theory and directed graphs [15, 18].

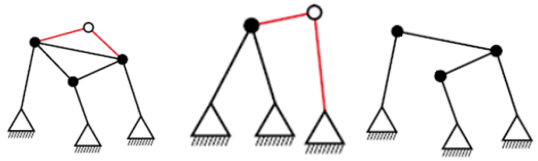
Given a mechanism and its associate pinned graph, we recall that our goal is to turn such a graph into a statically determinate pinned graph and decompose the graph into Assur components. Using the rigidity vocabulary we define an Assur graph as a ‘minimal’ pinned statically determinate (isostatic) structure where the deletion of any vertex and its incident edges (or removal of any edge) makes it non-rigid. For a series of equivalent more rigorous mathematical characterization of Assur graphs see [15].

In Figure 5 we have some examples of Assur graphs. If we delete any inner vertex it will result in a flexible structure. In



(a) (b) (c)

FIGURE 5. Examples of simple Assur graphs. In (a) we have a dyad and (b) a triad.



(a) (b) (c)

FIGURE 6. Examples of pinned structures that are not Assur. In (a) and (b) removing a two-valent vertex results in a rigid structure. In (c) we have a mobile structure.

literature, the pinned structure in (a) is called a dyad, and in (b) a triad. These are some of the most basic Assur graphs and serve as building blocks of kinematic linkages [14]. In Figure 6 we see examples of pinned graphs that are not Assur. In (a) and (b) we have two examples that are not Assur graphs, they can be decomposed into further Assur components. We can remove the two-valent vertex with its two edges and remain with a triad and dyad, respectively, which are pinned statically determinate graphs. The pinned structure in (c) is not statically determinate, it is mobile (non-rigid), so it is not Assur.

In the next section we use the directed graph from the pebble game algorithm to decompose the pinned statically determinate graph into Assur components, and give useful pebble game characterizations of the Assur graphs.

4 Directed Graphs and Assur Decompositions

We have demonstrated that the pebble game algorithm created a directed graph associated with pinned structure. Such a directed graph will now be decomposed into strongly connected components (components in directed cycles in the directed graph), which we then use to obtain the Assur components. The Assur decomposition algorithm is outlined in Section 4.2. We will also explore a few additional properties of these directed graphs.

4.1 Directed Graphs and Strongly Connected Decompositions

We need a few additional general terms for directed graphs.

A *cycle* in a graph G is a subset of the edge set of G which forms a path such that the start vertex and end vertex are the same. A *directed cycle* is an oriented cycle such that all directed edges are oriented in the same direction. A directed graph $\vec{G}$ is *acyclic* if it does not contain any directed cycle.

A directed graph is called *strongly connected* if and only if for any two vertices i and j in $\vec{G}$, there is a directed path from i to j and from j to i. The *strongly connected components* of a graph are its maximal strongly connected subgraphs. That is, strongly connected components cannot be enlarged to another strongly connected subgraph by including additional vertices and its associated edges. Each vertex can belong to only one strongly connected component (which may consist of only a single vertex), so the strongly connected components form a partition of the set of vertices V [7].

A directed graph is acyclic if and only if it has no strongly connected subgraphs except single vertices, since any cycle is strongly connected. If each strongly connected component is contracted to a single vertex, commonly called the *condensation* of the directed graph, the resulting contracted graph forms a directed acyclic graph. Other graph theoretic definitions that we will use can be found in any introductory book to graph theory (for instance [2]).

4.2 Assur Decomposition Algorithm

In this section we give a fast, efficient algorithm that decomposes a pinned statically determinate graph into Assur components. We illustrate this on our original one-degree of freedom linkage and demonstrate this algorithm, detailed steps are given below. We represent such an engineering system as a pinned graph (Figure 7 (a, b)). As we have seen in the introduction, to start the analysis we can make such a structure statically determinate by deleting the edge corresponding to the driver and pinning the inner vertex A to the ground (Figure 7 (c)) (equivalently we could have instead added an extra edge from A to the ground.) Of course these two techniques would be equivalent under the counting conditions of Pinned Laman Theorem and would both give a statically determinate graph. We should point out that if we did not make the structure statically determinate and instead left it as a 1 DOF linkage, we would get almost identical results, with some minor differences (see Remark below).

The first step in our Algorithm is to play the Pinned Pebble game to generate the directions (Figure 7 (d)). The second important step is to find the strongly connected components among the inner vertices. There are various fast algorithms for computing the strongly connected components such as Tarjan's algorithm [21] whose complexity is $O|E|$, or we can use an alternative

version based on an additional extension of the pebble game algorithm which is also confirmed with mathematical proofs [18]. The strongly connected decomposition of any directed graph is also included in the code of current Computer Algebra Systems such as Maple, Mathematica and SAGE. The overall computational cost of our algorithm is on the order of the complexity of the pebble game algorithm $O(|V||E|)$.

Algorithm 4.1. – ***Assur Decomposition Algorithm in 2-dimensions:***

Given Pinned statically determinate graph and corresponding pinned graph $G = (I, P; E)$

1. Generate directions on $G(I, P; E)$ towards the ground using the 2-dimensional Pinned Pebble Game Algorithm.

2. Find strongly connected components of the directed graph (using Tarjan's algorithm or an equivalent)

3. Each strongly connected component and its outgoing edges is an Assur component. For the component Make every outgoing edge out of the strongly connected component a grounded edge, pinning the end-vertex.

In Figure 7 (d) we have encircled all of the strongly connected components, one component containing vertices G, H, I and J and the other components being single vertices. Each strongly connected component and its outgoing edges, where we pin down the ends of its outgoing edges, forms an Assur component. The pinned structure is now decomposed into 6 Assur components, of which 5 are dyads (a dyad contains one inner vertex and two pinned vertices).

We have several alternative characterizations of Assur graphs (Assur components of structures) [15–17] . We summarize a few here.

Theorem 4.2. *Given a pinned statically determinate graph G and a 2-drected assignment $\vec{G} = (I, P; E)$ the following are equivalent:*

1. *the graph G is an Assur graph;*
2. *the pebble game applied to G has no free pebbles and has all inner vertices in a single strongly connected component;*
3. *$\vec{G}$ has one strongly connected component on the inner vertices;*
4. *$\vec{G}$ does not contain a directed cutset separating the inner vertices;*
5. *removing any edge of G leaves 1 DOF at every inner vertex;*
6. *removing any vertex of G leaves 1 DOF at every other inner vertex.*

As noted above, different outputs of the pebble game may differ by reversal of a cycle. This now translates as having no impact on the strongly connected decomposition - or equivalently, on the Assur decomposition.

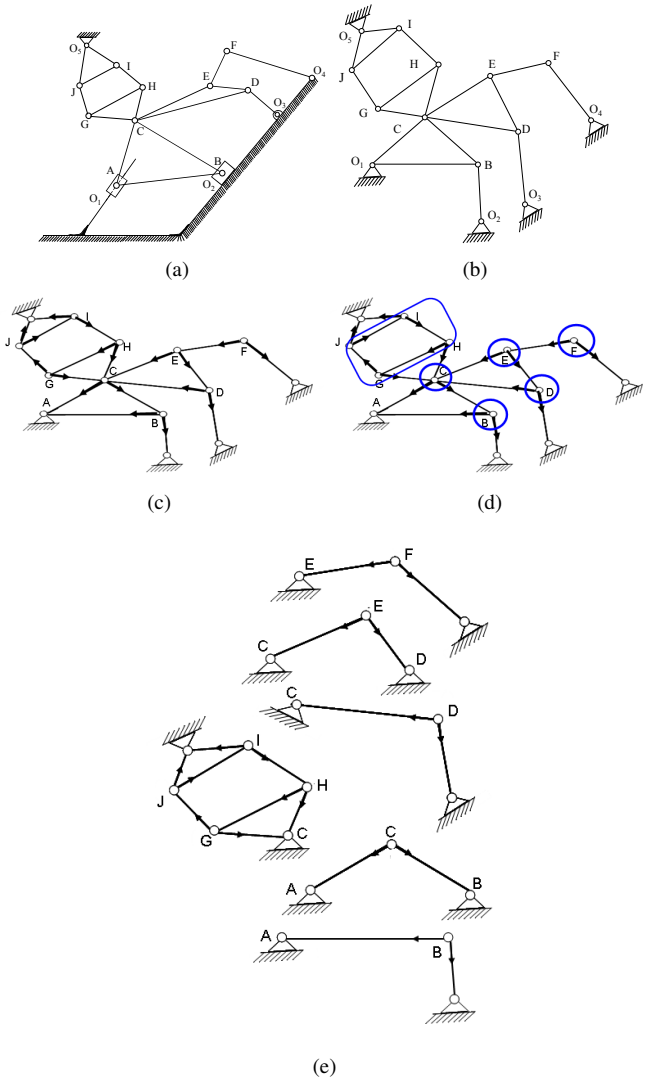


FIGURE 7. A plane linkage (a) is translated into a flexible pinned structure (b). We pin the inner vertex of the drive, so the structure becomes a pinned statically determinate (c). Pinned pebble game generated appropriate directions (d). Strongly connected components are circled (e). Each strongly connected component and its outgoing edges gives an assur component (f).

Remark. We should illustrate an important remark here about the choice of which decomposition we are considering. When we add an extra edge to the ground from the inner vertex of a driver (or equivalently we can shift the inner vertex and make it pinned [17]) to make it statically determinate, the remaining graph will have exactly the same set of directed edges (up to cycle reversals) as the graph with a free pebble remaining at the end of the driver, so the decomposition would not change whether we add the edge or not. This follows basically, because whether we

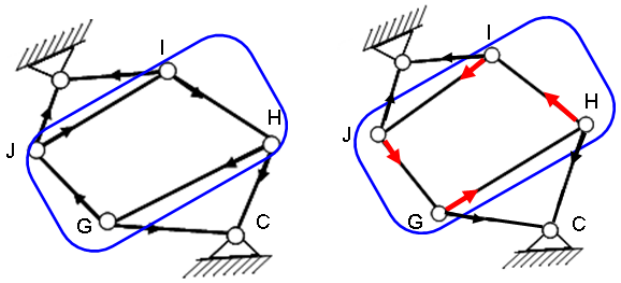


FIGURE 8. Different output of the pebble game can produce a different orientation in the graph. All inner vertices are still 2-directed, but different orientation always leaves the strongly connected components the same, and the subsequent Assur decomposition.

keep the pebble on that inner vertex of the driver or not does not impact the strongly connected components. On the other hand, moving that pebble around and placing it on other inner vertex (i.e. choosing a different driver) will change the directed graph and consequently the decomposition. So, decomposition is relative to what is a movable piece, i.e. what the chosen driver is. However, in mechanical engineering, it is usually known where the driver is. We are not giving the proof here but it follows this same idea. For a mathematical proof that a pinned statically determinate structure has a unique Assur decomposition see [17]).

Another reason we chose to make the graph statically determinate, is that in the final decomposition, the strongly connected component with the free pebble on it and its outgoing edges will not be statically determinate, it will have 1 DOF, hence it will not be an Assur component in our definition. However, we should note that if we left the free pebble on the driver, then we would be left with the same overall decomposition with this one component not being Assur. □

5 3D Structures

The basic process of decomposing a 2D mechanism into Assur Graphs, based on a directed graph, extends directly to mechanisms in 3D [17, 18].

Specifically, if we are given a graph in 3D which is already known to be Pinned statically determinate (or at least not redundant), an analogous version of the pebble game will give a directed graph and lead to a corresponding decomposition [17, 18]. For this game, we need the modified counts:

A pinned graph $G = (I, P; E)$ satisfies the *Pinned 3D Structural Conditions* [15] if

1. $|E| = 3|I|$ and
2. for all subgraphs $G' = (I', P'; E')$, $|E'| \leq 3|I'|$.

These are always necessary conditions, and are sufficient for tracking DOF in many critical situations.

The simplified pebble game corresponding to these counts will place 3 pebbles on each vertex, and place a pebble on the edge if a free pebble can be located at one of the vertices. Because we assumed the graph was statically determinate, we do not need to play on inner edges first. Below is a simple example. Given the directed graph $\vec{G}$ from the pebble game (or some other source which makes the out-degree of each inner vertex 3), the strongly connected decomposition described above will decompose the graph in 3D Assur Graphs, with fully analogous properties to the 2D analysis. More generally, if we know the graph was independent (perhaps a mechanism), then we can compute the DOF, derive the directed graph and complete the strongly connected decomposition as outlined above. Moreover, all the advantages of the Assur decomposition still apply to this decomposition, in terms of the analysis and synthesis of the mechanisms. Here is an example on a very simple 3D 1 DOF linkage.

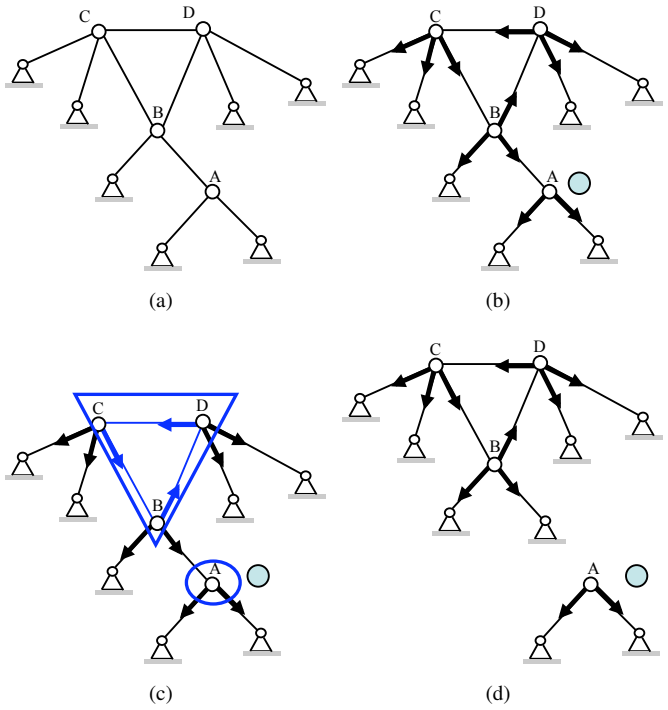


FIGURE 9. A 3D linkage (a), with directed graph from the 3D pebble game with one free pebble at the intended driver A. (C) shows the corresponding strongly connected components and (d) shows the Assur decomposition in 3D.

There are further extensions to more other mechanisms in 3D built with bodies, hinges, and isolated bars (e.g. extensions of the Stewart Platform, robotic arms). These are also the structural

models used for protein analysis [9, 19]. In 3D the counts for body bar, body hinge, and molecular structures all start with core $6|B|-6$ [22]. For this count there is an efficient pebble game which fits the count and leads to e full analogs of the results in the previous two sections.

There is a gap in 3D because the rigidity theory lacks a Laman type theorem for very general structures [18, 23]. However, it is important to emphasize that the original results here, and related investigations already provide key insights for 3D mechanisms, and this work can lead to additional new methods for the analysis and synthesis of 3D kinematic systems both in practice and in pedagogical settings. This is an initial transfer of the key pebble game algorithm from rigidity theory to the study of mechansims. The ongoing connections between work on mechanisms and work in the theory of rigidity promise to be productive for both fields.

REFERENCES

[1] Assur L. V., 1952, Issledovanie ploskih sterznevyh mehanizmov s nizsimi parami s tocki zreniya ih struktury i klassikacii. Izdat. Akad. Nauk SSSR. Edited by I. I. Artobolevski.

[2] Diestel R., 2000, **Graph Theory**, Springer-Verlag, New York, Second Edition.

[3] Gogu G., 2005, Chebychev-Grubler-Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations, European Journal of Mechanics / A Solids, **24**, pg.427-441.

[4] Hoffman C., Lomonosov A., Sitharam M., 2001, Decomposition plans for geometric constraint systems, parts I : Performance measures for CAD. J. Symbolic Computation **31**, 367408.

[5] Lee A., and Streinu I., 2005. Pebble Game Algorihms and (k,l)-sparse graphs, 2005 European Conference on Combinatorics, Graph Theory and Applications (EuroComb 05), DMTCS Proceedings: 181186.

[6] Norton R.L., 2004, **Design of Machinery: An Introduction To The Synthesis and Analysis of Mechanisms and Machines**. McGraw Hill, New York.

[7] Fiedler M., Special Matrices and Their Applications in Numerical Mathematics; Springer, Aug 31 1986.

[8] Gogu G., Chebychev-Grubler-Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations, European Journal of Mechanics / A Solids (May 2005), 24 (3), pg. 427-441

[9] Kuhn L.A., Rader D.J. and Thorpe M.F., 2001, Protein flexibility predictions using graph theory, Proteins, 44:150-65.

[10] Mitsi S., Bouzakis K.-D., Mansour G., and Popescu I., 2003, Position analysis in polynomial form of planar mechanisms with Assur groups of class 3 including revolute and prismatic joints. Mech. Mach. Theory, 38(12):1325 1344.

[11] Lo S., Monagan M., Wittkopf A., 2006, Strongly Connected Graph Components and Computing Characteristic Polynomials of Integer Matrices in Maple , preprint, from http://www.cecm.sfu.ca/CAG/papers/CharPoly.pdf.

[12] Penne R., 2010, Relative Centers of Motion, Implicit Bars and Dead-Center Positions for Planar Mechanisms, European Journal of Combinatorics 31.

[13] Pennock G. R. and Kamthe G. M., 2006, A study of dead-center positions of single-degree-of-freedom planar linkages using assur kinematic chains,. IMechE J. Mech. Eng. Sci., Special Issue on Kinematics, Kinematic Geometry, and their Applications, An Invited Paper, in press

[14] Shai O., Topological Synthesis of All 2D Mechanisms through Assur Graphs, ASME Design Engineering Technical Conferences, August 15-18, Montreal, Quebec, Canada, (2010).

[15] Shai O., Servatius B., and Whiteley W., 2010, Combinatorial char- acterization of the assur graphs from engineering. European Journal of Combinatorics 31, 1091-1104.

[16] Shai O., Servatius B., and Whiteley W., 2010, Geometric Properties of Assur Graphs. European Journal of Combinatorics 31, 1105-1120.

[17] Shai O., Sljoka A. and Whiteley W. Directed graphs, Decompositions and Spatial Rigidity, arXiv:1010.5552.

[18] Shai O., Sljoka A. and Whiteley W., Algorithms for Assur Graphs in 2 and 3-D, to appear.

[19] Sljoka A., 2006, Counting for Rigidity, Flexibility and extensions vis the Pebble Game Algorithm, Masters Thesis, York University.

[20] Tay T-S. and Whiteley W., 1985, Generating isostatic frameworks. Structural Topology, (11), 2169.

[21] Tarjan R., 1972,Depth-first search and linear graph algorithms, SIAM Journal on Computing. Vol. 1, No. 2, P. 146-160.

[22] White N. and Whiteley W., 1987, The algebraic geometry of motions of bar-and-body frameworks. SIAM J. Algebraic Discrete Methods, 8(1):1 32.

[23] Whiteley W., 1996, Some Matroids from Discrete Applied Geometry, Contemporary Mathematics, AMS **197**, 171–311.

[24] Yannou B. and Vasiliu A, 1995, Design platform for planar mechanisms based on a qualitative kinematics, Proceedings of QR-95. pages 191-200.

[25] Zhang G.F and Gao X.S., 2006, Spatial Geometric Constraint Solving Based on k-connected Graph Decomposition, SAC06 April 23-27, Dijon, France, ACM.

[26] Zhao J.S., Feng Z.J. and Dong J.X., 2006, Computation of the configuration degree of freedom of a spatial parallel mechanism by using reciprocal screw theory, Mechanism and Machine Theory **41**, 1486-1504.