Proceedings of the ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2013
August 12-15, 2013, Portland, OR, USA

DETC2013-13364

A NOVEL COMBINATORIAL ALGORITHM FOR DETERMINING THE GENERIC/TOPOLOGICAL MOBILITY OF PLANAR AND SPHERICAL MECHANISMS

Offer Shai
Mechanical Engineering School,
Tel-Aviv University
Tel-Aviv, Israel

Andreas Müller
Institute of Mechatronics, Chemnitz, Germany,
andreas.mueller@ifm.chemnitz.de

ABSTRACT

Structural mobility criteria, such as the well-known Chebychev-Kutzbach-Grübler (CKG) formula, give the correct generic mobility of a linkage (possibly of a certain class, e.g. planar, spherical, spatial) provided that it is not topologically overconstrained. As a matter of fact all known structural mobility criteria are prone to topological redundancies.

In this paper a combinatorial algorithm is introduced that determines the correct generic/topological mobility of any planar and spherical mechanism. The algorithm also yields a set of independent links that can be used as input, as well as the redundantly constrained sub-linkages. A mathematical proof of the algorithm and the underlying mathematical concept is presented. The proposed method relies on an established algorithm developed within combinatorial rigidity theory, called pebble game, originally developed for checking the rigidity/immobility of constraint graphs. A novel theorem is introduced and later proved in the paper which in turn enables applying the algorithm to any holonomic planar or spherical mechanism with higher and lower kinematic pairs and multiple joints. A further important result of applying this algorithm is that it gives rise to a decomposition into Assur graphs mentioned, which is briefly discussed in this paper.

1. INTRODUCTION

The mobility as the essential property of a mechanism has been a major matter of interest in mechanism theory. The approaches can be broadly classified as those that deal with the mobility of a given mechanism, with a particular geometry, and those that aim on the generic mobility of a class of mechanisms with certain topology [1]. Methods of the first class attempt explicit solution of the constraint equations or the approximation of the solution variety [13, 14, 15], possibly using tools from numerical algebraic geometry [16, 17]. Instead of considering a particular geometry, the second class approaches the problem from a structural point of view. These attempts have a long tradition and only need topological information about the existence of links and joints. The CKG formula is a well-known topological mobility criterion. It is assumed that they generally yield the generic mobility [6], i.e. the mobility of almost all realizations of a particular topology. Although they are independent of any geometric data (geometric overconstraints) all such methods are prone to topological redundancy since these criteria only take into account the existence of joints and links but not their particular arrangement.

The identification of topological redundancies requires graph-theoretic considerations and algorithms. Such an algorithm is presented in this paper. The basis for this algorithm is a graph representation of the constraints inherited from rigidity theory. This differs from the topological graph [3] often used in that it does not merely represent the arrangement of links and joints, but rather the links and the constraints they are subject to. This is presented in section 2, where the two established types (body-bar, bar-joint) are recalled, and a novel type of constraint graph is introduced. The actual combinatorial algorithm is introduced in section 3 together with the mathematical foundation. The algorithm is proved to converge to the unique generic mobility. In order to motivate the application of this algorithm an engineering interpretation of the steps and output of the algorithm is given. The application of the method is shown in section 4 for a simple example, and further interpretations of the output are discussed. The paper concludes with a brief outline of future work in section 5.

The algorithm used in this paper, called pebble game, was developed in 1997 [2] for checking whether a set of points subject to geometric constraints form a rigid structure. The use of this algorithm was also extended to check whether a graph consisting of rigid bodies is rigid or mobile as reported in [9]. In engineering, pebble game was applied to check the mobility of planar mechanisms consisting of only binary links and limited to lower kinematic pairs [8]. It was proved that pebble game can decompose any mechanism with only binary links to Assur graphs in 2d and 3d [7]. It should be noted that the algorithm reported in this paper is applicable to any type of planar mechanisms with holonomic higher and lower kinematic pairs and multiple joints. At first read the reader may skip section 3 and go directly to the example in section 4.

2. CONSTRAINT GRAPHS

The kinematic functionality of a mechanism is indeed due to the geometric and topological constraints imposed on its bodies. The topological graph already relates bodies and joints but it does not explicitly represent the imposed constraints. To

this end a *constraint graph* G is introduced. In the following δ denotes the generic mobility, $G(E,V)$ the constraint graph (undirected or direct), $e(G) = |E|$ the number, and $v(G) = |V|$ the number of vertices of G.

The idea behind constraint graphs is to represent a mechanism as an abstract relation of 'objects' representing certain degrees of freedom (DOFs). These objects constitute vertices of the constraint graph, and are chosen so to uniquely represent the mechanism's configuration. They can stand for rigid bodies or points. The constraints between them are represented by edges. In this sense the graph represents abstract constraint relations that possibly have different physical meanings (rotation, translation constraints).

Presented here next are the two established ways of describing the constraints in a mechanism.

2.1 Body-Bar (BB) Constraint Graph

A vertex of the body-bar graph stands for a rigid body. In its generalized form used in this paper the edges represent general scalar constraints between bodies. In particular an edge can stand for a distance constraint or a rotation constraint. For instance, in the BB graph in Figure 1 there are two edges between bodies 1 and 2, where both account for translational constraints. On the other hand, the bar between vertices 2 and 3 stands for the gear connection

A vertex represents a body that can (if considered unconstrained) move in the plane. It is therefore assigned the mobility 3.

The term 'body-bar' stems from rigidity theory of structures where each edge represents a mass-less bar imposing a scalar distance constraint between two bodies

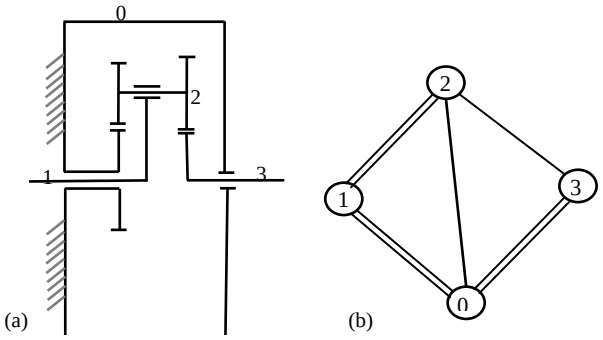

Figure 1: A gear train (a) and its corresponding body-bar graph (b).

2.2 Bar-Joint Constraint Graph

In this graph representation a vertex represents a point, which is permanently coinciding with a point on all the links attached to it. A point in space is assigned 3 DOF, and a point in the plane 2 DOF.

A peculiarity of Bar-joint graphs is the existence of so-called multiple joints. A multiple revolute joint in the plane is an aggregate of revolute joints that are geometrically placed at the same position. A multiple revolute joint connecting m bodies thus stands for $m-1$ revolute joints with collinear axes. For example, in Figure 4, joint B is a multiple revolute joint while the other two joints, A and C, are binary joints, i.e., connect between two bodies.

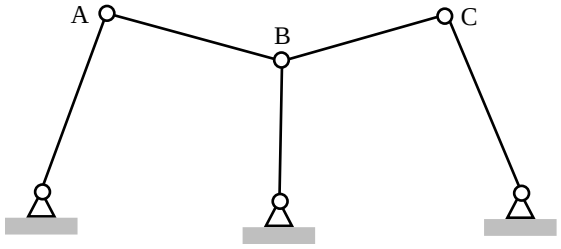

Figure 2: A bar-joint graph with a multiple joint at B.

2.3 Mixed Constraint Graph

As was explained in section 2.2, there is no unique assignment of a body-bar graph to a given mechanism containing multiple joints. To overcome this problem, we introduce, for the first time in the literature, a new type of graph, termed mixed constraint graph $G = (V_B \cup V_J, E)$. In this graph a vertex v can represent a rigid body, $v \in V_B$, as well as points, $v \in V_J$. That is, for a planar mechanism, each vertex of the mixed constraint graph embodies an object that can move in the plane, and its physical meaning follows from that of the body-bar and bar joint-graph. If a vertex represents a body (as in the body-bar graph), then it possesses three DOFs. If it represents a point (i.e. the location of a joint, as in the bar-joint graph), then it has two DOFs.

For example, in Figure 3 vertices A, B and C correspond to revolute joints while vertices 4, 8 and 10 correspond to bodies.

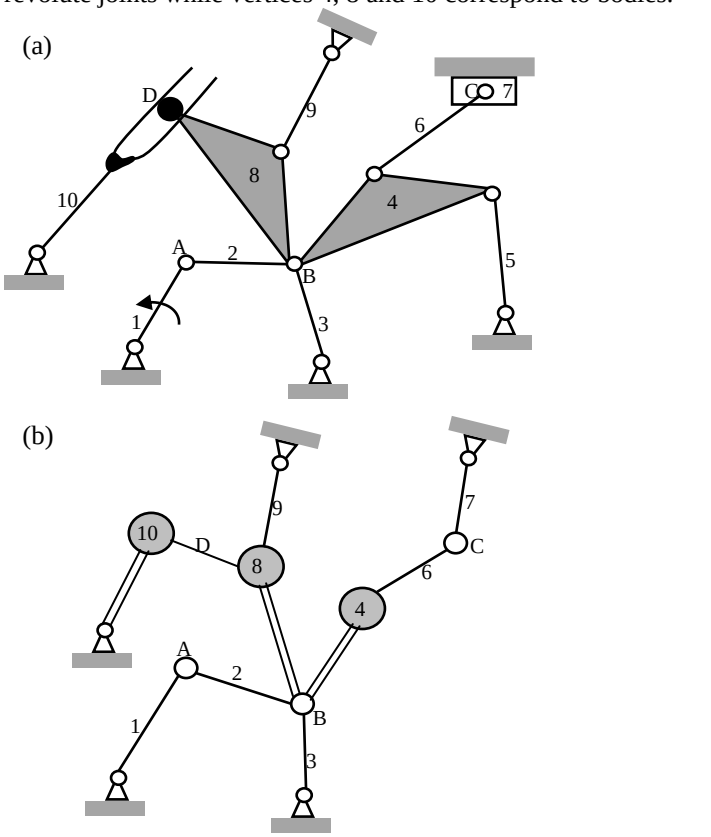

Figure 3: A mechanism (a) and its corresponding mixed constraint graph (b).

In the following $v_B(G)$ denotes the number of vertices of a mixed graph G representing bodies and $v_J(G)$ denotes the number of vertices representing points/joints.

3 A COMBINATORIAL ALGORITHM FOR GENERIC MOBILITY DETERMINATION

In this section the pebble game is introduced as a combinatorial algorithm for the determination of the generic mobility. Throughout this paper only floating planar linkages are considered. Aiming on the generic, i.e. topological, mobility the method operates exclusively upon the constraint graph, i.e. the topology, and a generic rather than a specific geometry is assumed. Redundancies due to the presence of special geometries are thus excluded. The basic idea behind the pebble game is to assign a set of pebbles to each vertex, where the number is equal to the DOF of the physical object represented by that vertex when considered unconstraint, and then to 'activate' the constraints, represented by edges, by coordinately relocating the pebbles.

3.1 Combinatorial Background

One of the main problems in checking the correct generic mobility of a mechanical system is to identify whether there is no sub-system having over-determinacy, redundant elements. A mathematical criterion for checking such non-existence of over-determinacy was established and proved in 1970 [4], which can be stated as follows:

Laman's theorem for planar Bar-joint graphs [4]: A floating planar bar-joint constraint graph $G = (V, E)$ with $e(G) = 2v(G) - 3$ determined if and only if $e(G') \leq 2v(G') - 3$ for every subgraph G' having v' vertices and e' edges.

The condition $e(G) = 2v(G) - 3$, together with the conditions on the subgraphs G', ensures that the bar-joint graph G is rigid. Clearly if $e' < 2v' - 3$, there are not sufficient constraints to make the graph rigid. On the other hand if $e(G) < 2v(G) - 3$ but the conditions on the subgraphs are satisfied, then there are no redundant constraints. Moreover the subgraph conditions imply that $e(G) \leq 2v(G) - 3$, and the constraint graph corresponds to a mobile linkage whose mobility is given by the following corollary.

Corollary: Let $G = (V, E)$ be a floating planar bar-joint constraint graph with e edges and v vertices. The constraint graph is non-redundant if and only if $e(G') \leq 2v(G') - 3$ for any subgraph G'. If this condition is satisfied, the linkage has generic mobility $\delta(G) = 2v(G) - e(G) \geq 3$.

For example, the graph in Figure 4.b is not a determined floating graph since the number of edges in the sub-graph spanned by the vertices $V' = \{A, B, D, E\}$ is 6 and is greater than $2 * 4 - 3 = 5$. The graph in Figure 4.a satisfies Laman's theorem thus it is a floating determined graph. Note, Grübler's equation determines the same DOF for both graphs in Figure 4 since it cannot distinguish overdetermined from uniquely determined graphs.

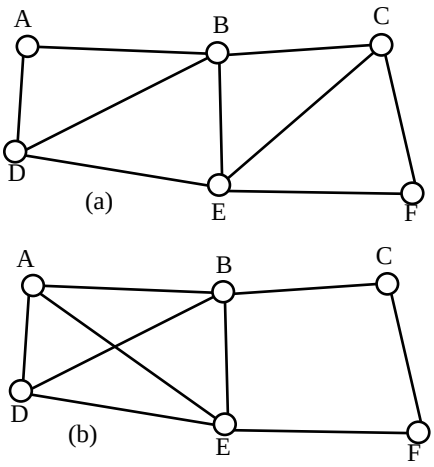

Figure 4: Floating graph that satisfies (a) and does not satisfy (b) Laman theorem.

In 1991 a theorem for determination of floating determined body-bar graph was reported [11] to which we refer in the paper as Laman's theorem for BB. In his paper Tay proved the theorem for any dimension.

Planar Body-bar Laman's theorem [11]: A floating planar body-bar constraint graph $G = (V, E)$ with $e(G) = 2v(G) - 3$ is determined if and only if $e(G') \leq 3v(G') - 3$ for every subgraph G'.

As for the bar-joint graph this gives rise to the following:

Corollary: A floating planar body-bar constraint graph $G = (V, E)$ is non-redundant if and only if $e(G') \leq 3v(G') - 3$ for any subgraph G'. If this condition is satisfied, the linkage has generic mobility $\delta(G) = 3v(G) - e(G) \geq 3$.

For example, the graph in Figure 5.b is not a determined BB graph since the number of edges in the sub-graph spanned by the vertices $V' = \{1,2,3,4\}$ is 10 and is greater than $3 * 4 - 3 = 9$. The graph in Figure 5.a satisfies BB Laman's theorem thus it is a BB determined graph.

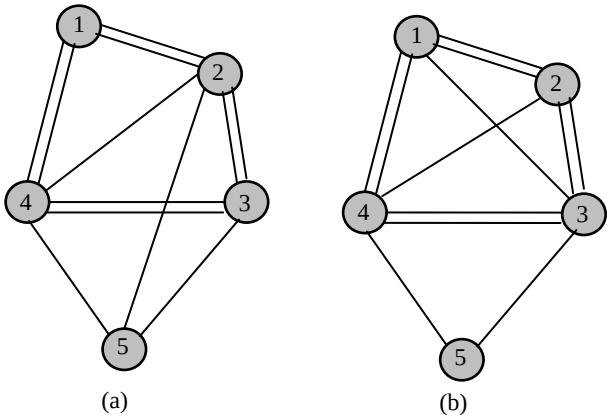

Figure 5. BB graphs that satisfy (a) and do not satisfy (b) the BB Laman's theorem.

Planar Mixed Laman's theorem (Shai and Müller, 2013): A floating planar mixed constraint graph $G = (V_B \cup V_J, E)$ with $e(G) = 3v_B(G) + 2v_J(G) - 3$ is determined if and only if $e(G') \leq 3v_B(G') + 2v_J(G') - 3$ for every subgraph G' of G, where $v_J(G) = |V_J|$ is the number of vertices corresponding to bodies and points/joints, respectively.

Corollary: A floating planar mixed constraint graph $G = (V_B \cup V_J, E)$ is non-redundant if and only if $e(G') \leq 3v_B(G') + 2v_J(G') - 3$ for every subgraph G'. If this condition is satisfied, the linkage has generic mobility $\delta(G) = 3v_B(G) + 2v_J(G) - e(G) \geq 3$.

For example, the floating system in Figure 6.a is not determined since the corresponding mixed graph in Figure 10.b does not satisfy Mixed Laman'n theorem. To prove that, let us choose the sub-graph spanned by the vertices: $V' = \{B_1, B_2, B_3, J_1\}$ having 9 edges which is greater than $3 \cdot 3 + 2 \cdot 1 - 3 = 8$, thus mixed Laman's theorem is not satisfied.

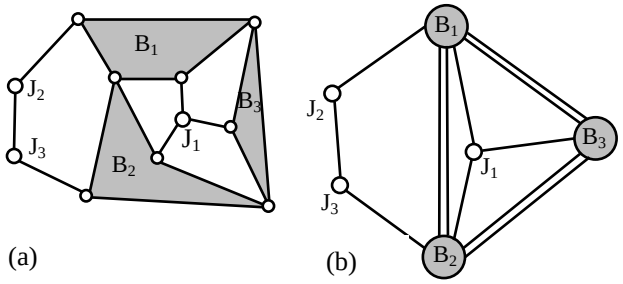

Figure 6. A structure (a) whose mixed constraint graph (b) does not satisfy the mixed Laman theorem.

3.2 A combinatorial Algorithm: The Pebble Game

Pebble game is a very efficient algorithm to check whether a graph satisfies Laman's theorem and thus to check whether there exists a sub-graph that is overdetermined. In the terminology of graph theory it checks whether there exists an over constrained subgraph $G' = (V', E')$, i.e. $(G') > 3v_B(G') + 2v_J(G') - 3$ for a general mixed graph, $e(G') > 3v(G') - 3$ for a BB graph, and $e(G') > 2v(G') - 3$ for a BJ graph. A naive algorithm requires checking all the possible subgraphs and therefore is bound to consume exponential time. The pebble game algorithm succeeds to perform this check in polynomial time, and even linear time for some examples [4].

The pebble game works as follows: for a planar linkage three pebbles are assigned to a vertex of its constraint graph if it represents a body and two pebbles if it stands for a point (joint location). Such an assignment of pebbles is a concept in graph theory referred to as a 'pebbling' of a graph.

The main concept of the algorithm is to assign 'pebbles' to any physical object in the kinematic model (bodies, points) that represents a certain DOF. These pebbles, i.e. DOFs, are reduced in the course of the algorithm, and the number of pebbles remaining after invoking the pebble game is equal to the generic mobility of the linkage.

The pebble game starts with an unconstrained system, in the sense that the number of pebbles assigned to a vertex is equal to the DOF as if its members were not subject to any constraint. Denote with $k(v)$ the DOF of the object represented by vertex v. For planar constraints graphs $k(v) = 2$ represents a point and $k(v) = 3$ if it is a body. The algorithm is initialized by assigning $k(v) = 2,3$ pebbles to each vertex v. That is, initially there are no constraints between the elements of a linkage, i.e. each element has $k(v)$ DOFs to move in the plane.

Each edge of G represents one constraint. Initially all constraints are inactive, i.e. all objects/vertices are unconstrained. An inactive constraint is represented by an undirected edge (constraint graph G is initially undirected). During the pebble game the constraints are successively activated by directing the edges. This indicates that the DOFs of one vertex are dependent on the DOFs of other vertices. In the algorithm this is achieved by moving a pebble from one of its end-vertices. The pebble game is summarized as follows:

Input to the Pebble Game algorithm:

The algorithm starts from the topological graph, i.e. an undirected graph as described in section 2. Each vertex represents a physical object that has k DOF.

The Pebble Game algorithm:

The pebble game converts a given (undirected) constraint graph into a directed graph where the direction assignments are determined by the directions in which pebbles are moved in course of the algorithm. An undirected edge is termed an *admissible edge* if the total of free pebbles next to its end vertices is at least four. Only admissible edges can be directed and can thus become active constraints. The pebble game can be summarized by three main tasks as follows:

1. INITIALIZATION: Assign $k(v)$ pebbles to each vertex of the undirected graph, thus all edges are admissible. This is equivalent to regarding all mechanical objects, corresponding to the vertices, as unconstrained, i.e. having $k(v)$ DOF.

2. WHILE there exist admissible edges DO the following **Orientation Move (Vertex - Edge move)**:

   Let (u, v) be an admissible edge, i.e., the total sum of pebbles next to the two end vertices is at least four. Remove one pebble from one of its end vertices, let it be vertex u, and replace the edge by a directed edge $\langle u, v \rangle$, i.e., u becomes the tail and v the head vertex of $\langle u, v \rangle$.

   END WHILE

   After this loop there are no admissible edges left. This move corresponds to activating the constraint corresponding to the pebble removed from vertex u. The direction of the edge introduces a causality in the sense that one DOF of the tail vertex u is assumed to be dependent on one DOF of the head vertex v. Note that this is an abstract assignment, i.e. it is not said that a certain DOF of u is made dependant on a certain DOF of v.

3. WHILE there are free pebbles left DO the following **Reorientation move (Vertex - Vertex Move):**

   Choose an undirected edge (u, v) and make it admissible by bringing free pebbles to its end vertices by applying the following steps ($\text{peb}(v)$ denotes the number of pebbles at vertex v):

   Suppose $\text{peb}(v) < 2$, if v stands for a point or $\text{peb}(v) < 3$, if v stands for a body. Then search for a vertex, say 'z', with free pebbles, i.e., $\text{peb}(v) > 0$, for which there is a directed path from v to z consisting of directed edges. Therefore this move consists of redirecting all edges of the path from v to z. Then, move one pebble from vertex z to v by reversing/swapping the direction of all the edges in the directed path, and assign $\text{peb}(z) := \text{peb}(z) - 1$ and $\text{peb}(v) := \text{peb}(v) + 1$.

   END WHILE

   This move can be interpreted as redefining the previously introduced causality of constraints along the path v to z.

Output of the Pebble Game algorithm:

The algorithm ends when all the edges that can be made admissible are directed. The edges that are left undirected are those for which it is impossible to move 4 pebbles next to their end vertices indicating the existence of redundant constraints.

Example: In Figure 7 a detailed explanation of the iterations performed by pebble the game for checking whether the BJ graph in Figure 7.a is well-determined. Initially, two pebbles are placed next to each vertex as shown in Figure 7.a thus all the edges are admissible. Without loss of generality, let us start with edge (A, B). Since the sum of the pebbles on its two end vertices is equal to four, i.e., it is an admissible edge so that the orientation move can be applied. Let us choose to move a pebble from A, thus edge (A, B) is now directed from A to B, i.e., $\langle A, B\rangle$ as shown in Figure 7.b. Now we search for another admissible edge, found to be (B, C), consequently orientation move can be applied causing edge (B, C) to be directed as shown in Figure 7.c. At this state, there are no admissible edges left and there is an undirected edge. Thus the reorientation move should be applied. At this stage, the only edge that is undirected is edge (A, C) and $\text{peb}(A) = 1 < 2$ thus we search for a directed path from A to a vertex with free pebbles. In this case vertex B is found, and the path is $\{A, B\}$, as shown in Figure 7.c. Consequently, a pebble is moved from vertex B to A, so that $\text{peb}(B) = 1 - 1 = 0$ and $\text{peb}(A) = 1 + 1 = 2$, and the direction of $\langle A, B\rangle$ is reversed to $\langle B, A\rangle$ and now two pebbles are placed next to vertex A, as shown in Figure 7.d. Edge (A, C) is now admissible thus the orientation move can be applied to edge (A, C) causing it to be directed as shown in Figure 7.e. When the algorithm terminates, shown in Figure 7.e, the following could be concluded: Since all the edges are directed it indicates that there is no redundancy. From the fact that three free pebbles are left next to vertices it can be concluded that the graph is well-determined (it has 3 DOF as a floating planar structure).

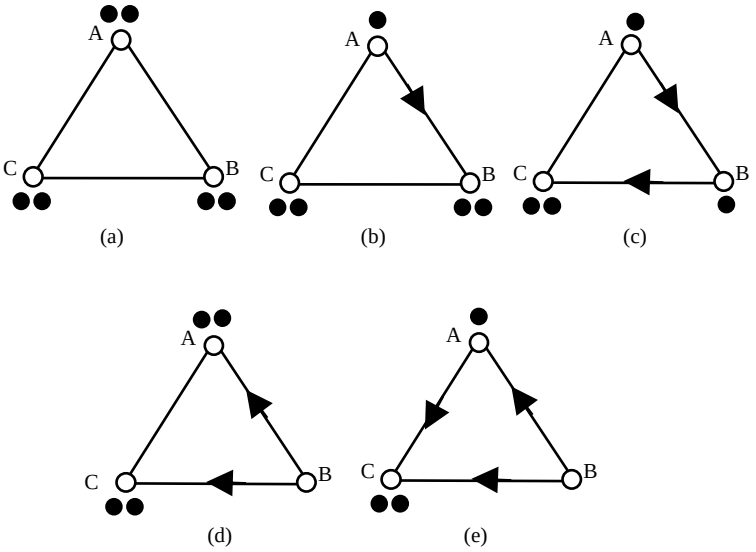

Figure 7. Example of tracing the operation of pebble game.

3.3 Engineering Interpretation of the Algorithm

Orientation move: moving a pebble from vertex to edge

Initially all edges of G are undirected and all vertices have an initial number of pebbles corresponding to the DOF the associated physical objects (bodies for BB and points for BJ graphs) thus all the edges are admissible. Undirected edges represent inactive constraints. With the orientation move we activate a constraint/edge by defining its direction. By assigning this direction we introduce a causality in the sense that one DOF (embodied by one pebble) of the tail vertex is now considered dependent on one DOF of the head vertex. That is, the motion of one DOF of the tail vertex is determined by one DOF of the head vertex.

Once an edge was directed it remains so throughout the algorithm, meaning that one DOF of the tail vertex determines one DOF of the head vertex. However, the direction of an edge can be changed, meaning that the constraint between the two elements is imposed by the other element.

Figure 8 clarifies this idea. Suppose we have two bodies connected by a higher kinematic pair as shown in Figure 8.a, and denote with $(VAix', VAiy')$ the planar velocity vector of body i expressed in the given frame $\{A, x', y'\}$. The two bodies are presented by two vertices each having three pebbles standing for the 3 DOF (three independent motion parameters) of a body and one edge between them corresponding to the higher kinematic pair as shown in Figure 8.b. Initially edge $(1,2)$ is not directed meaning that the constraint has not been activated yet and each of the two bodies has 3 DOF thus it is an admissible edge. Moving a pebble from vertex 1 (body 1) to the edge makes it directed/activated and oriented from 1 to 2, as shown in Figure 8.c. The higher kinematic pair imposes the constraint that the normal velocities of both bodies are equal. In the reference frame $\{A, x', y'\}$ this means that $VA1y' = VA2y'$. The interpretation of the orientation in figure 8.c is that the

normal motion of body 1 is prescribed by that of body 2. Hence, from an algorithmic point of view, the constraint is resolved by the assignment $VA1y' := VA2y'$. This is an important aspect to be noticed, and it should be clear that once the pebble game is completed it eventually yields a solution flow of the kinematics since the remaining DOFs (indicated by the free pebbles) determine the motion of the mechanism.

If for an edge there is a sufficient number of pebbles on either vertex, i.e., it is an admissible edge, the definition of direction (dependence) is arbitrary. Once the edge is directed, its direction can be reversed by changing the vertex from which the pebble is consumed. In Figure 8.d the pebble on the edge is now consumed from vertex 2 meaning that one of the parameters of body 2 is imposed by the parameters of body 1.

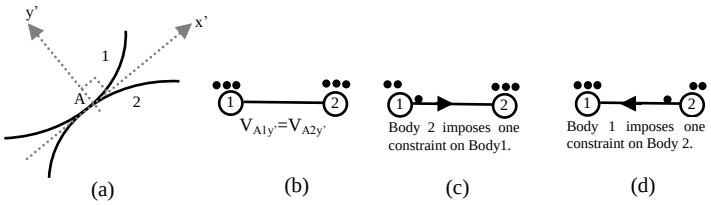

Figure 8. The meaning of the edge directions in pebble game.

Reorientation move - Moving a pebble from vertex to vertex to make edges admissible

The condition to orient an edge is that the edge is admissible, i.e., in total at least four pebbles should be next to the two end-vertices of the edge. In case that there are less than four pebbles, it may still be possible to collect additional pebbles. This requires rearranging the pebbles that are already assigned to some vertices, achieved by the following move:

Let $e = (x, y)$ be a constraint edge and x and y its two end vertices and suppose that $\mathrm{peb}(x) + \mathrm{peb}(y) < 4$, i.e. the edge is inadmissible. Assume that $\mathrm{peb}(x) < 2$, if x stands for a point in BJ or mixed graph, or $\mathrm{peb}(x) < 3$, if x stands for a body in BB or mixed graph. Since the number of free pebbles next to vertex 'x' is less than its DOF, we search for a directed path from vertex 'x' to the first vertex, let us term it 'z', whose $\mathrm{peb}(z) > 0$. To move a pebble from z to x the directed path from x to z is reversed, $\mathrm{peb}(z) := \mathrm{peb}(z) + 1$, and the pebble is moved along the path until it reaches x,where $\mathrm{peb}(x) := \mathrm{peb}(x) + 1$.

In Figure 9 we give an example of applying this type of move on the graph. The sum of the pebbles on x and y is three, i.e. less than four, consequently an inadmissible edge, thus one pebble should be moved to vertex x. As can be seen in Figure 9.a a directed path from vertex x to vertex z with one pebble was found and is designated by a dashed line. This path is now reversed and the pebble is moved from vertex z to x enabling now to direct the edge (x, y).

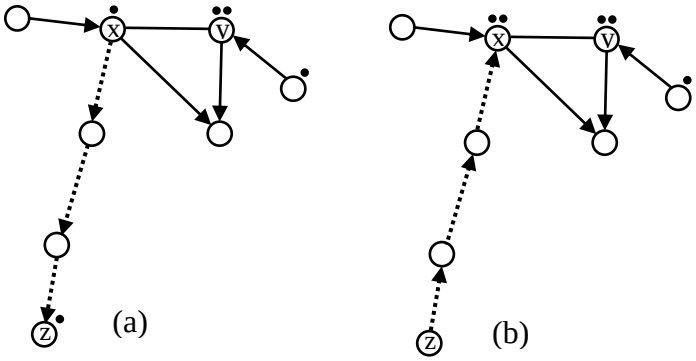

Figure 9. Example of applying the reorientation move to move a free pebble from vertex to another vertex. a) The directed path from x to z. b) The graph after reversing the directed path from x to z and moving the pebble from z to x.

The output of the Pebble Game algorithm:

1. If **all** the edges are directed it indicates that the given mechanism does not possess redundant constraints. If there are undirected edges the mechanisms is generically overconstrained.
2. The total number of free pebbles indicates the DOF of the mechanism as a whole. Notice that the vertices where free pebbles are left are not unique in general. If a particular set of inputs (with number equal to the overall DOF) is specified for a given mechanism it should be possible to locate the free pebbles at these input links. If this is not possible the mechanism can generically not be actuated by the selected inputs (drivers). The pebble game thus provides a check whether or not a selected set of input links are feasible.
3. The remaining pebbles indicate (generically) independent DOFs. The number of free pebbles that can be moved to a vertex (physical object) indicates that this object (link, point) can be moved independently, with a mobility equal to the number of pebbles next to it. The overall motion of the mechanism is thus uniquely determined by the motion of those vertices that have free pebbles left. How the motion of these vertices affects the overal motion of the mechanism is indicated by the following result 4.
4. After locating the left free pebbles next to the input links/drivers, the directed cutsets indicate the decomposition into Assur Graphs.

3.4 Proof of the Algorithm

The pebble game is a combinatorial algorithm that iteratively produces a solution flow (a directed dependency graph) for a system of constraints. In general, there is no unique solution, and any solution graph is admissible as long as it is causal, i.e. does not comprise over-determined subgraphs. The latter is ensured by the algorithm as it will be shown in this section. It will be proved that once an admissible edge is being directed it cannot produce an over-determined sub-graph with other directed edges. This is expressed by the following theorem:

Theorem 1: At any time all the sub-sets of edges that are oriented by the algorithm do not possess an over-determined sub-graph.

Proof: The theorem is proved for BJ, BB and mixed graphs by showing that all the oriented sub-graphs $G' = (V', E')$ satisfy Laman's theorems, i.e.

$$e(G') \leq 3 * v'B(G') + 2 * v'J(G') - 3 \quad (1)$$

where $e(G')$ is the number of edges in G' and $v_B(G')$ and $v_J(G')$ are the number of vertices in G' that stand for bodies and points, respectively.

To prove equation 1 let us choose an arbitrary set of vertices, V', and examine the distribution of the pebbles that were assigned to the vertices in V'. It is easy to verify that the pebbles can only be in one of the following three classes: remain on the vertices, i.e., were not used, yet, to orient edges; used to orient edges whose both two end vertices, tails and heads, belong to the set V'; used to orient edges whose only tail vertices belong to the set V'. Note, those edges whose only head vertices are in V' are not counted since the orientation of edges is defined by moving a pebble from the tail vertex to the edge, i.e. orientation move. Let us formulate the latter claim in a mathematical form:

$$\text{peb}(V') + e(G') + \text{out}(V') = 3v'B(G') + 2v'J(G') \quad (2)$$

where $\text{peb}(V')$ is the number of free pebbles on the vertices V'; e(G') number of oriented edges whose both end vertices are in V'; $\text{out}(V')$ number of directed edges going out of V', i.e. whose head vertices do not belong to V' ; $3v_B(G') + 2v_J(G')$ is the sum of pebbles that were initially assigned to the graph G' before the pebble game is applied.

We now prove that for any set of vertices V' with more than two vertices the sum of the first and third terms in equation 2 is greater than 3, i.e.,

$$\text{peb}(V') + \text{out}(V') \geq 3 \quad \text{for } |V'| \geq 2 \quad (3)$$

It will be shown that this inequality remains satisfied from the initial state and that applying any move on edges incident with V' does not violate this inequality.

At the initial state (by definition of the input to the pebble game) $\text{peb}(V') = 3v_B(G') + 2v_J(G') \geq 3$ since for any sub-graph with at least two vertices there are at list 4 or 6 free pebbles in BJ and BB, respectively. For the sake of clarity we distinguish in the proof between two types of edges: *inner edge*- whose two end vertices are in V'; outer edge – whose one end vertex is in V' and the other in the complement $V - V'$. Now we examine all possible cases of applying the two moves to the inner and outer edges. As can be seen below there is a difference between BJ, BB and mixed, thus we prove the correction of pebble game for these three types of graphs.

Case 1: Orientation move on an inner edge – for orienting an edge the edge should be admissible, meaning, there should be at least four pebbles on its two end vertices both in BJ, BB and mixed. Consequently, after the orientation there are at least three pebbles in V' in all the three types of graphs, thus the inequality 3 is valid, as shown in Figure 10.a.

Case 2: Orientation move on an outer edge directing it away from and towards V' – in case the outer edge is being oriented out of V' one pebble is consumed, so that $\text{peb}(V')$ decreases by 1 but $\text{out}(V')$ is increased by 1, as shown in Figure 10.b. In case the edge is orientated towards V' then there is no change of $\text{peb}(V')$ since no pebble is consumed or added, and no change in the number of outgoing edges, as shown in Figure 10.c.

Case 3: Reorientation move on an inner edge – the sum of the free pebbles and outgoing edges on the two end vertices of the inner edge remain the same, as shown in Figure 10.d.

Case 4: Reorientation move on an outer edge – in case an outgoing edge is reoriented toward V', one pebble is brought from the outside to the inside of V' and the number of outgoing edges is decreased by 1, as shown in Figure 10.e. In the case that an edge going toward V' is reoriented outward 1 free pebble is moved from the inside to the outside, so $\text{peb}(V')$ is decreased by 1, while the number of outgoing edges in V' is increased by 1.

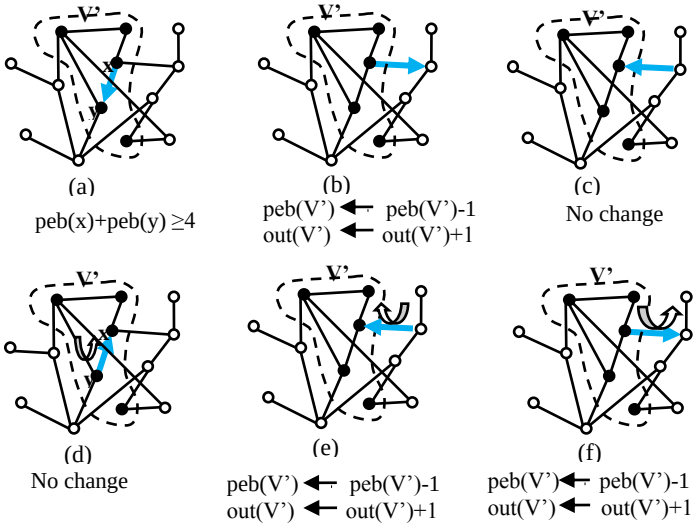

Figure 10. Schematic description of all the cases applying the two moves on an edge and how it relates to equation 3.

Let us move the summation of the two terms to the RHS, resulting in equation 4:

$$e(G') = 3v_B(G') + 2v_J(G') - (\text{peb}(V') + \text{out}(V')) \quad (4)$$

Since we proved above that the sum of the last two terms is always greater or equal to 3 we proved that equation (3) remains fulfilled when the pebble game is applied.

ENDPROOF

The above theorem asserts that the pebble game does not introduce redundant sub-graphs. It is left to prove that the algorithm converges to the correct topological mobility.

Theorem 2: Let G be the constraint graph of the mechanism. The generic/topological mobility of G is determined as

$$\delta = K - e(G')$$

where G' is the directed subgraph, and $K = 3v_B(G) + 2v_J(G)$ is the number of pebbles initially assigned to the constraint graph G. That is, after the pebble game terminates, the mobility is determined by the number of oriented edges in G' (representing non-redundant constraints).

The idea underlying theorem 2 is that the algorithm starts with K DOFs, i.e., the DOF of the unconstrained objects. Each edge corresponds to a constraint thus reduces the overall DOF by one. In theorem 1 we proved that if an edge is oriented (the constraint is activated) it does not introduce redundancy. It is thus ensured that orienting an edge reduces the DOF by one. We will prove that when pebble game ends it finds the maximum set of active constraints independently of which edge is oriented first and if there is an unoriented edge it is proved that it is a redundant constraint.

Lemma 3: When the algorithm ends there are at least 3 free pebbles that can be moved to any edge.

Proof: Let $e = (x, y)$ be an arbitrary edge and x and y its end vertices. Let us define $\text{reach}(w)$ as the set of all vertices that are reachable from w, i.e., there exists a directed path from w to them. Define the set $V' = \text{reach(x)} \cup \text{reach(y)}$. Since V' is the set of all reachable vertices, it is $\text{out}(V') = 0$.
According to equation 3 it follows that: $\text{peb}(V') + 0 \geq 3$, meaning, there are at least three pebbles in V' that, if not located next to x and y, can be moved to this edge by applying reorientation moves.
So far it was not mentioned whether edge (x, y) is directed or not. In case it is undirected and we can't bring a fourth pebble to it, the edge is inadmissible, and it can be concluded that the sub-graph induced by the set V' is well-determined and the edge (x, y) is redundant.
ENDPROOF

We will extend lemma 3 and show that it is invariant, i.e., it is true for any edge throughout applying the algorithm.

Lemma 4: For any edge and at any step of the algorithm the number of free pebbles in the sub-graph induced by the reachability of the two end vertices of the edge is at least 3.

From lemma 4 it follows that every edge can be grounded, and if the algorithm does not succeed to direct an edge it is proved to be redundant. This proves theorem 2.

Since each orientation move, i.e. activation of a non-redundant constraint, reduces the number of free pebbles by one, theorem 2 implies that the number of free pebbles remaining after the pebble game is equal to the generic DOF of the graph. This gives rise to the following statement.

Corollary 5: The pebble game returns a set of free pebbles whose number is equal to the unique generic mobility of the linkage. All objects corresponding to the vertices with free pebbles can be independently moved with a DOF equal to the number of free pebbles assigned to them. The algorithm yields a non-redundant directed subgraph G', which represents a feasible set of non-redundant constraints, and a non-directed subgraph, which represents the corresponding redundant constraints. The final result for the generic mobility does not depend on the edge where the algorithm starts.

Remark: It is known [4] that the complexity of the pebble game is of polynomial order in the number of vertices, $O(|V|)^2$ and the required memory also grows quadratically, i.e. with $O(|V|)^2$.

4. AN EXAMPLE

In Figure 11 we apply the mixed pebble game to the mixed graph representing the linkage in Figure 6.a.

Initially, all the bodies and joints have three and two pebbles, respectively, as shown in Figure 11.a.

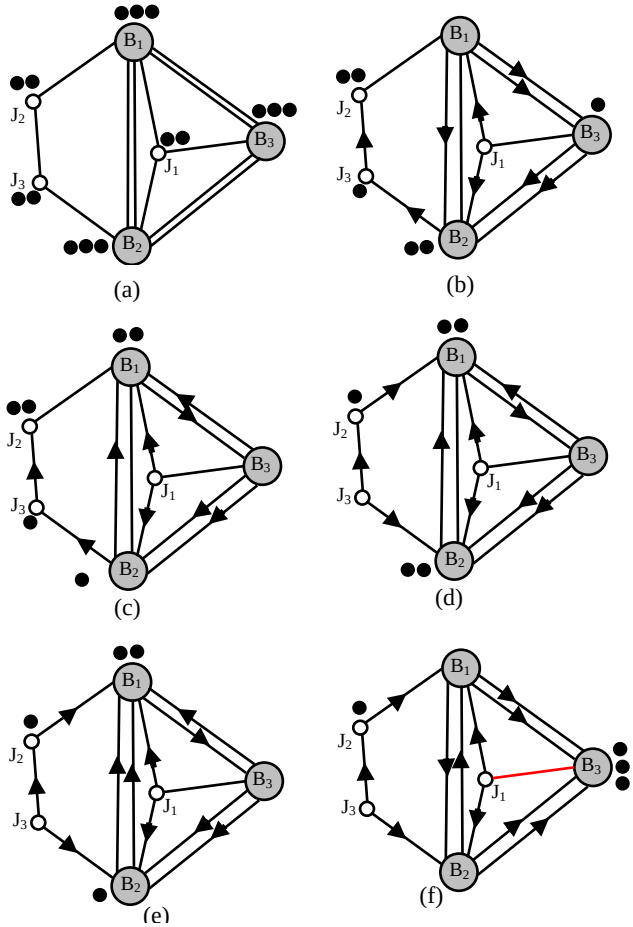

Figure 11. Example of applying mixed Pebble game on mixed constraint graph.

The orientation move is first applied and all the admissible edges are directed. For example, the two edges (B_1, B_3) and (B_1, J_1) are admissible, thus can be oriented, since there are 6 respectively 5 pebbles next to the two end vertices. Figure 11.b

shows all edges that could be directed by applying the orientation move. Since there are no more admissible edges the reorientation move is being applied next. For example, in Figure 11.c edge (J_2,B_1) becomes admissible by moving one pebble from vertex B3 and one from B_2 thus it can be oriented as shown in Figure 11.d.

Applying reorientation move on edge $\langle B_2, J_3 \rangle$ brings a free pebble to vertex B_2 thus edge $\langle B_2, B_1 \rangle$ is now directed as shown in Figure 11.e.

Now we are left with four free pebbles and one edge, (J_1, B_3) unoriented. According to Lemma 3, it is possible to move 3 pebbles next to any two end vertices, and we move them to the end vertices of edge (J_1, B_3). For sake of consistency, we move them to vertex B_3 as shown in Figure 11.f.

In figure 11.f there are no edges that can be made admissible by applying the reorientation move and the algorithm terminates. The output of the algorithm allows for the following interpretations:

Result 1: The most obvious result is the generic mobility of the associated linkage (corollary 5). Since the algorithm terminates with 4 free pebbles the linkage generically possesses 4 DOFs.

Result 2: Beside the generic mobility the particular location of the pebbles indicates which links can be moved independently, hence can be used as control inputs. As we deal with floating planar linkages there are always three DOFs that correspond to the relocation of the linkage as a whole. In this example there are 4 free pebbles. Each of the pebbles represents one DOF that can be independently controlled. The specific allocation of pebbles in figure 11.f together with the original mechanism in figure 6.a allows for an apparent interpretation: the 3 DOF of B_3 describe the location and orientation of the linkage in the plane, and the one pebble at J_2 is a translation DOF of the location point of J_2 that controls the internal shape. Notice that

1. The pebble at J_2 is not the joint angle but one component of the location vector.
2. There is no specific assignment of coordinates to the DOFs so that ANY generalized coordinate can be used to represent the DOF of J_2. The pebble game algorithm operates on an abstract level and does not need specific selection of coordinates.
3. The particular allocation of pebbles is not unique and can be controlled in course of the algorithm. Also the algorithm's result can be changed by application of the reorientation move (which does not change the number of free pebbles). For instance, in figure 11.f a free pebble is now assigned to vertex J_2. With a reorientation of $\langle J_2, J_3 \rangle$, as shown in figure 12, this pebble can be moved to J_3. Now the one independent DOF is assigned to J_3.

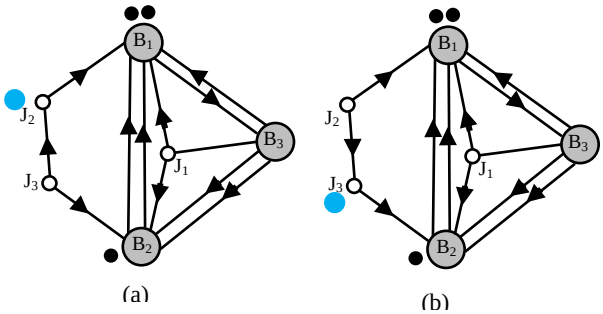

Figure 12. The free pebbles indicating that (a) joint J_2 and (b) J_3 have independent DOF.

Result 3: The algorithm further yields information about redundant constraints. An edge is redundant if in all vertices that belong to the reachability of its two end vertices there is no free pebble (lemma 3). For this example it is in particular $\text{reach}(J_1) \cup \text{reach}(B_3) = \{B_1, B_2)\}$, and there are no free pebbles on the latter two vertices, as shown in figure 13.a. Hence the edge/constraint (J_1, B_3) is redundant. Moreover the result allows for identification of over-determined regions. These are defined by the reachable vertices of the redundant edge including the end vertices of the redundant edge. In this case this is $\{B_1, J_1, B_3, B_2\}$ and the edges between them. These edges are colored with red in Figure 13.b. Indeed it is obvious already from figure 6 that the red sublinkage is rigid.

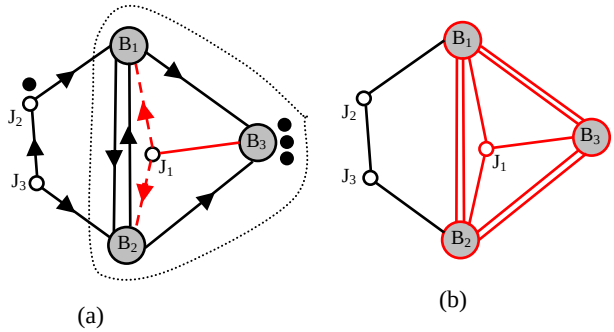

Figure 13. Example of over-determined region. a) The reachability of the end vertices of the redundant edge (J_1, B_3). b) The over-determined region, colored with red, defined by the redundant edge (J_1, B_3).

Remark: An important result, which cannot be discussed here, is that the oriented edges in the over-determined region have a unique decomposition into Assur graphs. Details of this decomposition will appear in the forthcoming paper of the authors.

5. CONCLUSIONS AND OUTLOOK

The paper introduces, for the first time in literature, an efficient combinatorial algorithm for determining the correct generic/topological mobility for any planar or spherical mechanism with higher and lower kinematic pairs, including multiple joints. As a prerequisite, the paper first introduces two types of constraint graphs for modeling the topology of linkages: body-bar and bar-joint graphs. It is explained that either type fails in modeling certain linkages. To overcome this

limitation the concept of a mixed constraint graph is introduced. One of the salient conclusions of this paper is that by using this graph representation it is possible to represent any planar mechanism, and consequently to invoke the corresponding mixed pebble game algorithm. The latter is the main contribution of the paper: it determines the correct generic mobility of the mechanism modeled by a mixed constraint graph. The planar mixed Laman theorem, which is an extension of the well-known Laman theorem for bar-joint graphs, is given as a mathematical foundation of the algorithm. Thereupon it is proved that the novel mixed pebble game converges to the correct generic mobility. Moreover it is discussed that this combinatorial algorithm allows for decomposing any mechanism into its building blocks, namely Assur graphs.

The reported algorithm applies to floating linkages, i.e. linkages that are not fixed to a ground. In a forthcoming publication, the mixed pebble game will be amended to include, so-called grounded mixed constraint graphs, a novel class of constraint graphs representing space-fixed mechanisms. To this end the algorithm needs to be qualified so to be able to treat immobile ground vertices.

REFERENCES

[1] G. Gogu, Mobility of mechanism: a critical review, Mech. Mach. Theory 40 (2005) 1068–1097.

[2] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. Journal of Computational Physics, 137 (1997) 346 – 365

[3] A. Jain, Graph theoretic foundations of multibody dynamics, Part I: structural properties, Multibody. Syst. Dyn. 26 (2011) 307–333.

[4] G. Laman. On graphs and rigidity of plane skeletal structures. J. Engineering Mathematics, 4 (1970) 331–340.

[5] A. Lee and I. Streinu, Pebble game algorithms and sparse graphs, Discrete Mathematics, 308: 8 (2008) 1425-1437.

[6] A. Müller: Generic Mobility of Rigid Body Mechanisms, Mech. Machine Theory, Vol. 44, no. 6, June 2009 1240-1255.

[7] O. Shai, "Rigidity of 2d and 3d Pinned Frameworks and the Pebble Game", London. Mathematical Society Workshop: Rigidity of Frameworks and Applications, July 12-15, Lancaster University , England, 2010.

[8] O. Shai, A. Sljoka and W. Whiteley, "Directed Graphs, Decompositions, and Spatial Rigidity" submitted to Discrete Applied Mathematics, 2010.

[9] A.Sljoka , 2006, Counting for Rigidity, Flexibility and extensions via the Pebble Game Algorithm, Master's Thesis, York University.

[10] A. Sljoka, O Shai and W. Whiteley, "Checking mobility and decomposition of linkages via Pebble Game Algorithm" ASME Design Engineering Technical Conferences, August 28-31, 2011, Washington, USA.

[11] T.S. Tay, Henneberg's method for bar and body frameworks, Structural Topology, vol. 17, pp. 53-58, 1991.

[12] T. Tiong-Seng, Henneberg's method for bar and body frameworks, Structural Topology 17 (1991), 53–58.

[13] K.J. Waldron, The constraint analysis of mechanisms, J. Mech. 1 (1966) 101–114.

[14] K.J. Waldron, A study of overconstrained linkage geometry by solution of closure equations – Part I. Method of study, Mech. Mach. Theory 8 (1973) 95–104.

[15] K.J. Waldron, A study of overconstrained linkage geometry by solution of closure equations – part II. Four-bar linkages with lower pairs other than screw joints, Mech. Mach. Theory 8 (1973) 233–24.

[16] C. Wampler, B.T. Larson, A.G. Erdman, A new mobility formula for spatial mechanisms, in: Proc. ASME International Design Engineering Technical Conferences (IDETC), September 4–7, 2007, Las Vegas.

[17] C.W. Wampler, J.D. Hauenstein, A.J. Sommese: Mechanism mobility and a local dimension test, Mech. Mach. Theory 46 (2011) 1193–1206.