ELSEVIER

Artificial Intelligence in Engineering 13 (1999) 273–285

ARTIFICIAL INTELLIGENCE *in* ENGINEERING

www.elsevier.com/locate/aieng

Graph theory representations of engineering systems and their embedded knowledge

O. Shai[a,*], K. Preiss[b]

[a]*Department of Solid Mechanics, Materials and Structures, Tel Aviv University, Tel Aviv 69978, Israel*
[b]*Department of Mechanical Engineering and School of Management, Ben Gurion University, Beer Sheva 84105, Israel*

Received 10 July 1998; received in revised form 25 November 1998; accepted 12 December 1998

Abstract

The discrete mathematical representations of graph theory, augmented by theorems of matroid theory, were found to have elements and structures isomorphic with those of many different engineering systems. The properties of the mathematical elements of those graphs and the relations between them are then equivalent to knowledge about the engineering system, and are hence termed “embedded knowledge”. The use of this embedded knowledge is illustrated by several examples: a structural truss, a gear wheel system, a mass-spring-dashpot system and a mechanism. Using various graph representations and the theorems and algorithms embedded within them, provides a fruitful source of representations which can form a basis upon which to extend formal theories of reformulation. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Graph theory; Embedded knowledge; Isomorphic structures; Knowledge representation

1. Introduction

When a human analyses or synthesizes an engineering system by using the mathematical representation governing its behavior, he or she creates a mathematical model of the engineering system, then manipulates the equations using knowledge about them and their relation with the physical reality. In usual engineering practice, one uses a model that is known to be suitable for the system at hand and the aim of the computation. Reformulation of the problem into another formally understood mathematical system [1,2] to the extent that it is done for engineering analysis, usually uses continuum mathematics. This paper shows that representations of graph theory for engineering problems can be useful as a basis upon which to extend formal theories of reformulation.

Research in engineering analysis usually starts with an understanding of the physical system, then the adoption of a suitable mathematical model for the system. In the work reported here a different approach was adopted. Rather than starting with the physical system itself or the mathematical representations historically used for the behavior of engineering systems, many other mathematical approaches were investigated to find those which can be useful representations of engineering systems. Representations were sought for which knowledge of the mathematical properties of those representations and the relations between them can be used to provide augmented understanding of the physical engineering system. For instance, if two engineering systems can be represented by graphs which are known to be dual, then the physical systems are dual, and this in turn leads to new insights regarding analogies between the systems. Section 7 shows how one can infer the behavior of a mechanism from properties of its dual truss. The Embedded Engineering Knowledge project was devoted to searching for representations isomorphic with one or more engineering systems, and finding common properties, if they exist, between them. After investigating many mathematical alternatives, attention was focused on graph theory, matroid theory and discrete linear programming. This paper illustrates the approach using results only from the graph theory representation, augmented by theorems from matroid theory.

Graph theory is a useful representation because on the one hand the elements of the graph can be defined so as to have a one-to-one correspondence with the elements of many kinds of engineering systems. On the other hand, the theorems and algorithms of graph theory allow one also to represent behavioral properties of the system, such as deformations and forces, or velocities and movements, as properties of the vertices or edges of the graph. This paper illustrates how engineering problems, for example: truss

* Corresponding author.

structures, planetary gear systems and dynamic mass-spring-dashpot systems are mapped into graphs, and then analyzed by using the theorems and algorithms of graph theory; these theorems and algorithms constitute knowledge that is embedded in the graph theory. These representations have been sporadically used for solving engineering problems, but to the best of our knowledge have never been brought together as a whole for dealing with engineering systems.

This approach enables one to apply efficient algorithms. An example is the derivation, from the embedded knowledge, of an efficient algorithm for analyzing indeterminate trusses. Sometimes one deduces implicit knowledge that was not known even to human experts. This will be explained by showing the dual relation between determinate trusses and mechanisms, engineering problems that seem today to belong to different domains, but discovered to be dual by using this approach. Another example of such a previously unexplored relation, given in the paper, shows the analogy between analysis of indeterminate trusses and dynamic systems.

This paper is organized as follows. Section 2 discusses the importance of representations in general. Section 3 shows some of the general properties of graphs necessary to follow the work. Section 4 shows use of the graph representation for structural trusses, Section 5 for planetary gear systems and Section 6 shows, briefly, application to other engineering systems. Section 7 shows how, by understanding the duality property of a graph, a new analogy between a structural truss and a mechanism was discovered, with which one can reason. Section 8 gives some concluding remarks.

2. Representations

In order to reemphasize the importance of representations, we use the well-known terminology and explanations of Simon about representation for engineering design [3]. To clarify and demonstrate the influence of problem representation on design, Simon used the game called number scrabble.

Number scrabble is played with nine cards, valued from one to nine. The cards are placed in a row, face up, between the two players. The players select, alternately, any one of the cards that remain in the center. The aim of the game is for a player to make up a “book”, that is, a set of exactly three cards whose spots add to 15, before his opponent can do so. The first player who makes a book wins; if all nine cards have been drawn without either player making a book, the game is a draw.

4	9	2
3	5	7
8	1	6

Fig. 1. The magic square.

Simon [3] shows how a change in representation makes it easy to find the solution. He uses the magic square, which is made up of the numerals from one to nine (as shown in Fig. 1).

Each row, column or diagonal sums up to 15, and every winning triple of the game is a row, column, or diagonal of the magic square. From this, it is obvious that “making a book” in number scrabble is equivalent to getting “three in a row” in the game tic-tac-toe. As Simon [3] points out, as many people know how to play tic-tac-toe well, they can transfer their tic-tac-toe knowledge to number scrabble.

Korf [2] reviewed a number of alternative representations which had been proposed for various problems including the tic-tac-toe problem mentioned earlier, and showed how they may be generalized into a more formal system. This paper gives some previously unexplored representations for engineering systems, which can be usefully generalized.

As in the number scrabble example, when changing the representation enables people to transfer their tic-tac-toe strategy to number scrabble, in the examples in this paper the engineering problem is changed to a graph theory representation. One can then use the many known graph theory algorithms and theorems that have been developed by researchers in the field.

The formalization of a more general reformulation theory in Artificial Intelligence was preceded by a period in which many people suggested particular isomorphic representations of particular problems or puzzles [2]. Eventually, we expect that the representations shown here, and others, will be combined into a theory for more generalized reformulation of representations of engineering systems. In a summary of the Second Workshop on Change of Representation and Problem Reformulation [4] it was stated that “the field of representation change does not have a solid theoretical foundation yet”. We believe that the approach presented here will help to advance theory of reformulation for engineering systems. Moreover, proceeding in this direction presents the possibility of deriving new mathematically proven connections between different engineering fields, an example of which is shown in Section 7.

3. Graphs as representation of engineering systems

An engineering system is usually represented as a diagram, with nodes, lines, and words or numbers which assign values to some or all nodes or lines. For instance, it is common to use a diagram to show a truss, or a mechanism, or a gear system, or electrical circuit, or a mass-spring-dashpot oscillator. The elements of the diagram are syntax symbols, and the diagram itself is considered to be a sentence which describes the system. Following mathematical tradition, the work should be done in two parts.

1. Check whether the problem is well defined and hence solvable. In the words of logic, check that the syntax of the engineering system being dealt with, in other words,

its diagrammatic representation, is a Well-Formed Formula (WFF). This question is usually dealt with in a cursory fashion in engineering work. Attention is immediately focused on the numerical mathematical formula to be applied to a problem, with no systematic attention paid to whether the system is correctly defined. For instance, as will be shown later, the rule often used to determine if a truss structure is just-stiff, is not a complete solution to that question. However, when viewing the truss as a graph, this question has a proven algorithmic solution, as shown in Section 4.

2. Before proceeding into the analysis effort, ensure that the solution will require as low a computational effort as possible. This is of less practical importance for small systems, but is very important for large systems with many components. This subject is usually dealt with by heuristic rules of thumb, known as expert domain knowledge. When using the graph theory representation of the engineering system, it can often be dealt with using mathematically proven algorithms of graph theory rather than man-made heuristic rules.

In this paper it is assumed that the reader has a basic knowledge of graph theory.

3.1. Network graphs

A graph is defined by the ordered pair $G = \langle V,E \rangle$, where V is the vertex set and E the edge set, and every edge is defined by its two end vertices. If each edge in the graph has a direction, the graph is known as a directed graph. If the directed graph is a network graph, each edge and vertex has properties of flow and potential, respectively.

As for logic, where the Conjunctive Normal Form (CNF) is a special representation of predicate calculus and Robinson [5] found an embedded algorithm for that special representation, in this paper three special graph theory representations are described and their embedded properties used. Central to understanding these graphs is a particular type of graph called a tree. A *tree* is a connected graph with no circuits. There are many properties and theorems embedded in this graph. A few that are used in this paper are shown:

Proposition 1. The relation between the number of vertices v and edges e of a tree is fixed, given by

$e(T) = v(T) - 1.$

Proposition 2. There is one and only one path between any two different vertices.

A spanning tree is a subgraph of graph G which is a tree and which includes all the vertices of G but only a subset of the edges. The edges of the tree are called branches, and the edges not in the spanning tree are called chords.

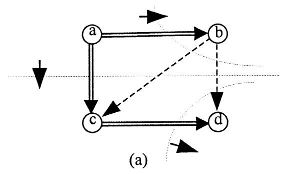
In this paper, we define three different network graphs, which have knowledge embedded in each. For convenience, this paper uses a line-type attribute. These are:

A *solid* line
Representing an edge with an unknown value of flow or potential difference at the current stage of the computation.
A *bold* line
Representing an edge for which the potential difference is known.
A *dashed* line
Representing a chord, which is an edge not included in the spanning tree, and if the flow value of the edge is known, then it is both dashed and bold.
A *double* line
Representing a branch of a spanning tree.

To deal with these representations we need first to develop the use of the cutset and circuit matrices. Given a connected network graph, we assign arbitrary directions to each edge, then find a spanning tree within it, thus defining branches and chords in the graph. There are obviously many spanning trees possible in the graph; the choice of a spanning tree does not effect the generality of this approach, but can affect the computational effort needed for the algorithms which later use that spanning tree.

When drawing an infinitely long curved line, which passes through at least one edge but not through any vertices, the graph is separated into two parts. Each such infinite cut defines a set of edges, known as a cutset. When the cutset includes only one branch of the spanning tree it is called a fundamental cutset. This paper deals only with fundamental cutsets, and for brevity they will be called cutsets. Each will be labeled with the name of the branch that defines it. The direction of the cutset is defined by its branch direction, as shown in Fig. 2(a).

The *cutset matrix* **Q** is a matrix of the graph obtained as follows. The matrix has $e(G)$ columns (corresponding to the edges of the graph) and has rows corresponding to the cutsets which in turn are defined by the branches. The index i refers to the row and j to the column. Because the quantity of cutsets is equal to the quantity of branches, the quantity of rows is equal to the quantity of branches. The value of the matrix element $[Q_{ij}]$ may be $+1$, 0, or -1. It will be $+1$ if edge j is included in the cutset which is defined by branch i and is with the same orientation as the cutset, -1 if it is

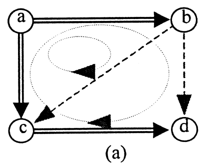


(a)

$$\mathbf{Q}=\begin{array}{c}\\ \text{ab}\\ \text{ac}\\ \text{cd}\end{array}\begin{array}{c}\begin{array}{ccccc}\text{ab} & \text{ac} & \text{cd} & \text{bc} & \text{bd}\end{array}\\ \begin{pmatrix}1 & 0 & 0 & -1 & -1\\ 0 & 1 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1\end{pmatrix}\end{array}$$

(b)

Fig. 2. The cutsets of a graph (a) and its cutset matrix (b).



$$\mathbf{B}=\begin{array}{c|ccccc} & ab & ac & cd & bc & bd \\ bc & 1 & -1 & 0 & 1 & 0 \\ bd & 1 & -1 & -1 & 0 & 1 \end{array}$$

(b)

Fig. 3. The circuits of a graph (a) and its circuit matrix (b).

with opposite orientation, and 0 if it is not included in the cutset as shown in Fig. 2(b).

A circuit is a closed path and is called a fundamental circuit if it includes only one chord and all the other edges are branches. This paper deals only with fundamental circuits, and for brevity they will be called circuits. A directed circuit is formed by traversing a chord in the direction of its arrow, then finding a path back to the initial vertex, but traversing any edge in that path only once, as shown for instance in Fig. 3. As is immediately obvious from the definition, circuits are defined by chords.

The *circuit matrix* **B**, demonstrated in Fig. 3, has $e(G)$ columns as for the cutset matrix, and its rows correspond to the circuits. The direction of each circuit is chosen to suit the direction of the chord which defines it. Because each chord defines a circuit, the number of rows is equal to the number of chords of the spanning tree. The element $[B_{ij}] = +1$ if edge j is included in the circuit which is defined by chord i, and is with the same orientation as the circuit, -1 if it is with opposite orientation, and 0 otherwise.

Every edge is assigned a value called the *flow*, which can be a force, flow of liquid, money, goods or the like.[1]

Every vertex is assigned a value called the *potential*.[2] The potential may represent a physical quantity such as displacement, pressure or voltage, but it can also be used for other attributes. For instance in the shortest path algorithm it represents the lower bound of the distance (or the combined weights of the edges) from the current vertex to the target vertex [6].

3.1.1. Flow graph representation

Definition of the flow graph representation: A directed graph G is a flow graph representation if the value of the flow in each edge is independent of the potential difference across that edge. The property of the flow graph representation that is used here is the Flow Law, which is stated as

The Flow Law The vector sum of the flows in every cutset of G is equal to zero.

This law may be recognized as a generalization of the well-known Kirchhoff's Current Law (KCL). Note that KCL is restricted only to one dimension which is appropriate for electrical circuits, while the flow law is multidimensional and can be used for two or three dimensions which are appropriate for trusses and other engineering systems.

A graph and its spanning tree is shown in Fig. 2 while Fig. 7 shows a truss, its graph with spanning tree and its cutset matrix.

The matrix form of the Flow Law is used in this paper, written as

$$\mathbf{Q}\cdot\vec{F} = \vec{0} \tag{1}$$

where $\vec{F}$ is the vector of the flows, or *Flow Vector*.

To solve the flow graph using the flow law we need to know the condition under which it can be solved.

Proposition 3 (The solvability condition). Let F represent the flows in the edges of G and let $\dim(F(G))$ be the dimension of the coordinate system for the flow in the elements. The dimension, which can in general be any integer value, is usually in practice one, two or three. One is for a scalar problem, two is for a plane problem, and three for a space problem.

If $\dim(F(G)) \times (v(G) - 1)$ is equal to the quantity of edges with unknown values of flow in $E(G)$ *then* G is solvable using only the flow law.

Proof. Each branch in the spanning tree defines a cutset in G, and the sum of flows in each cutset in each coordinate axis is equal to zero. Therefore, there are $\dim(F(G))$ equations for each cutset, and as there are $v(G) - 1$ cutsets, there are $\dim(F(G)) \times (v(G) - 1)$ equations. The equations are in general independent because in each cutset there is at least one edge (which is the branch that defines the cutset) that belongs generally to this cutset and not to any other cutset.

3.1.2. Potential graph representation

Definition of a potential graph: Let G be a graph in which for every vertex there will be associated a number which represents the potential at the vertex. In this representation the potential difference of the edge is independent of the value of the flow in that edge. In addition, every circuit satisfies the Potential Law, which is stated as

The Potential Law: For every circuit in the graph, the sum of the potential differences of the edges of the circuit is equal to zero. In matrix representation this is written as

$$\mathbf{B}\cdot\vec{\Delta} = \vec{0} \tag{2}$$

where $\vec{\Delta}$ is the vector of the potential differences, or *Potential Difference Vector*.

This law is a vectorial generalization to several dimensions of KVL which is stated for a one dimensional or scalar system.

[1] In control theory, this is called the "through variable", but the word "flow" is more suitable for the work reported here.

[2] The potential difference between the vertices defining an edge is known in control theory as the "across variable".

Proposition 4 (The solvability condition). Let dim(G) be the dimension of the graph and dr(G) be the number of edges with known values. The relation between the number of edges e and the number of vertices v of the graph must then satisfy

$$e - dr(G) = \dim(G) \times (e - v + 1). \tag{3}$$

Explanation: The quantity of independent circuits in a graph is equal to the number of chords in any spanning tree. As there are $v - 1$ branches in a spanning tree, the quantity of chords is $e - (v - 1)$, or $e - v + 1$. For each circuit there is one equation for each dimension, so the number of equations is then $\dim(G) \times (e - v + 1)$. The number of unknown variables (labeled as regular solid edges) is the number of graph edges minus the edges with known values, which is: $e - \mathrm{dr}(G)$. In order to have a unique solution for the analysis, Eq. (3) must be satisfied.

3.1.3. Resistance graph representation

Definition of a resistance graph: Let G be a graph in which

- for every vertex there will be associated a number which represents the potential at the vertex and
- for every edge there will be associated a number which represents the flow in the edge.

In this representation the potential difference of the edge is dependent on the value of the flow in that edge. The relation between the potential difference $\Delta(e)$ and the flow $F(e)$ in the edge is called the resistance of the edge, and is designated as $R(e)$; the inverse relation is the conductivity of the edge and designated as $G(e)$. In addition, the flows and potentials satisfy the Flow Law and Potential Law, respectively. A linear relationship is common between the flow and potential difference; it is termed Ohm's Law in electrical systems, or Hooke's Law in structural systems.

Proposition 5 (orthogonality property). The cutset and circuit subspaces of a graph are orthogonal and complementary to each other [7].

Explanation: In the terminology of matrices, this proposition is written as follows:

$$\mathbf{B}\cdot\mathbf{Q}^{t} = 0. \tag{4}$$

From Eq. (4) it can be proved [7] that the flows in the branches are dependent on the flows in the chords according to

$$\vec{F}_{T} = \mathbf{B}^{t}\cdot\vec{F}_{C} \tag{5}$$

and the potential differences in the chords are related to the potential differences in the branches by

$$\vec{\Delta}_{C} = \mathbf{Q}^{t}\cdot\Delta_{T}. \tag{6}$$

3.1.4. Line graph representation

Graph G is defined as a line graph of an engineering system if every vertex of G corresponds to an element, and every edge to the connection between the corresponding vertices.

In the special graphs described earlier, for every engineering element there is a corresponding edge. In a line graph, the element of the physical system is represented as a vertex and the edge represents the connection between the engineering elements.

This representation has no special properties as do the others previously discussed, but it enables one to find embedded properties of the engineering system which exist in the connection between the elements, in cases when these properties cannot be found by other representations. For example, in Section 5 the line graph represents a planetary system, the properties of which cannot usefully be represented by the three representations described earlier.

4. Checking the validity and analysis of a truss

One can conclude from the embedded knowledge of the representation whether the engineering problem is solvable. In other words, before the analysis process starts, check in the terminology of logic, whether the diagram representing the system has a well-formed syntax and hence is a WFF. Consider for example a truss, shown as a diagram which defines the topology and geometry of the elements. The well-formedness of the topology is a necessary condition for rigidity of the truss and hence stability of the structure with its supports, but is however not sufficient. The relative angles between the members, which define the geometry of the truss, must also be such that they do not lead to computational singularities. It is noteworthy that both checking the validity of and analyzing the truss are based on the embedded knowledge of the same representation, which in this case is graph theory.

4.1. Checking the validity of a truss

The common rule for a truss to be just-stiff [8] is that it should be all triangulated and there should be $2 \times v - 3$ elements, v being the number of vertices. Fig. 4 shows a

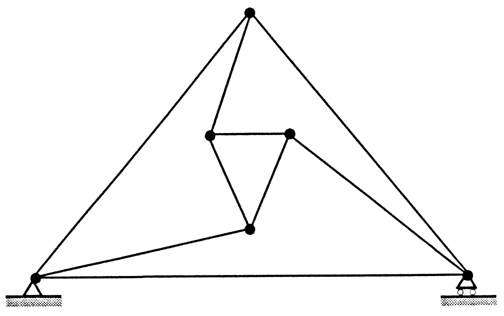


Fig. 4. A rigid truss for which the common well-formedness rule does not succeed.

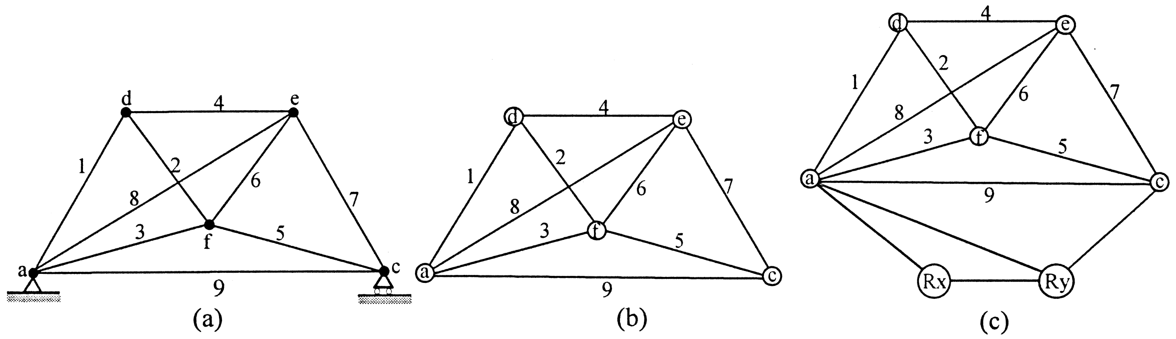


Fig. 5. (a) A truss; (b) its corresponding graph; and (c) the graph including the reactions.

just-stiff truss with $2 \times v - 3$ elements but is not all triangulated, and is considered a special case. A better rule, with no special cases, is needed, and is described later.

In the corresponding graph $G = (V,E)$ for the truss, every vertex corresponds to a pin-joint, and every edge to a rod, as shown for example in Fig. 5. The word topology refers to the data as to which vertices exist in the graph and which are neighbors, meaning that they are the end points of the same edge. The process of checking whether the topology of a truss satisfies the condition that the truss is rigid, is by use of the Laman theorem [9]. The truss will have a well-formed topology if in the corresponding graph, every subgraph G' satisfies

$$e' \leq 2 \times v' - 3. \tag{7}$$

Checking Eq. (7) requires exponential time, while the embedded knowledge leads us to use known algorithms which are efficient and of proved correctness. In this case, the truss has a well-formed topology, thus satisfying a necessary condition for its rigidity, if and only if (iff), when doubling each edge in turn, the graph contains two disjoint spanning trees [10,11]. For this problem there is a known greedy algorithm which has polynomial efficiency [7,12]. If the graph is of the truss only, this criterion determines whether or not the truss alone is rigid; if the graph includes the reactions, as described in Fig. 5, it will determine whether the whole system of truss and reactions is rigid.

4.2. Analysis of determinate and indeterminate trusses

A determinate truss can be solved by equilibrium at each vertex, whereas for an indeterminate truss information as to the relation between force and deformation in each element is needed. For analysis, the truss with its loadings and reactions is represented as a graph. In the language of network graph theory, force in the truss rod is flow in the graph edge, and deformation is potential difference. Table 1 summarizes the Flow Law which is used for statically determinate and indeterminate trusses and the Potential Law which is used for statically indeterminate trusses.

First the topology of the truss is checked. After it is found to have a well-formed topology, the process of analyzing the truss proceeds. In this process values are assigned to the variables which represent truss edges. For a determinate truss there is one variable associated with every edge, corresponding to the flow (or force) in the rod; for an indeterminate truss there is an additional variable, the potential difference (which is the deformation), for each dimension (two for a plane truss, three for a space truss). The two laws which the variables must satisfy are shown in Table 1.

4.2.1. The steps for building the corresponding graph of the truss

The network graph of a truss, for instance that shown in Fig. 6, is created as follows:

1. Create a vertex in the graph for every pinned joint in the truss.
2. Create an edge in the graph, called a “truss edge”, for every rod; its end vertices correspond to the joints that connect the rod to the truss. The direction of every truss edge will be arbitrarily assigned, because if the assignment is wrong the numerical result will be negative and

Table 1
The flow and potential laws and their details

	Flow Law	Potential Law
The Law	The sum of all the flows in every cutset is equal to zero	In every circuit, the sum of the potential differences of the circuit edges is equal to zero
The meaning of the variable	Flow → force	Potential → displacement
Engineering statement of the law	Static theorem	Compatibility constraint
Matrix representation	Cutset matrix	Circuit matrix
Domain of application	Determinate and indeterminate trusses	Indeterminate trusses

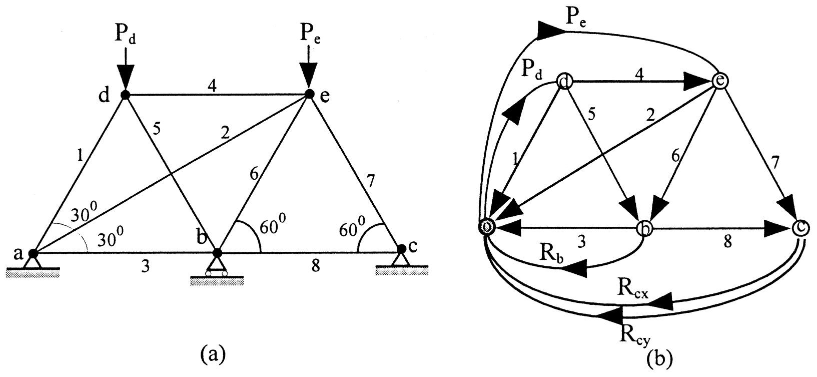


Fig. 6. Example of a truss (a) and its corresponding graph (b).

the solution will be correct. The properties of the edge in the graph include the cosine of the direction of the rod relative to the arbitrary datum direction chosen for the analysis in two dimensions (or two direction cosines in three dimensions).

3. One of the vertices which correspond to a pinned support will be chosen arbitrarily to be the "reference vertex", and will be labeled gray.
4. The following edges for the external forces and reactions are added – For each external applied force a *flow source edge* (or for short a "source edge") is added, from the reference vertex to the vertex corresponding to the joint upon which it acts. For each reaction a *reaction edge* is added from the vertex corresponding to the joint at which the reaction acts, to the reference vertex, one for each relevant principal direction. For a plane truss, for every mobile support there will be one corresponding edge, and for a pinned support there will be two corresponding edges.

4.2.2. Topological rules for deriving the equations

By using the graph representation, it is easy to automatically assemble the set of equations from its syntax [13, 14]. The central idea of deriving the equations is that each branch defines a cutset, so in a determinate truss, the corresponding row will contain the number $+1$, -1 or 0 (depending on the direction of the cutset) multiplied by the cosine of the rod direction for the *x*-coordinate, or the sine for the *y*-coordinate. The same approach is used for an indeterminate truss, only here we use the resistance graph representation where the conductivity of the edge which corresponds to the stiffness of the rod will be included, as explained in [15,16]. This is shown in Fig. 7, where in assembling the matrix it should be remembered that for the principal directions of a mobile reaction, if the force exists the displacement is zero, and if the displacement exists the force is zero. The square matrix which appears in Fig. 7(e) is the stiffness matrix, where in location *ij* appears the sum of the stiffnesses of the edges which are both in cutset *i* and cutset *j*. In location *ii* appears the sum of all the rod stiffnesses which are in cutset *i*. For example, in location '12' the element '$-\mathbf{G}_8$' appears because '8' is the only edge which is in both the first two cutsets. The minus sign shows that edge 8 is directed differently relative to each cutset. In this graph representation of trusses the conductivity (stiffness) of an edge is a 2 × 2 matrix for plane trusses, 3 × 3 for space trusses. The dimensions of the conductivity matrix in the resistance graph are derived from the Hooke's Law, which states: $F(e) = G(e) \times d(e)$, where $G(e) = ((A(e) \times E(e))/L(e))$ and $d(e)$ is the rod deformation. As the flows and the potential differences are two dimensional in plane trusses, the deformation can be written as a linear combination of the displacements: $d = \Delta_x \cos(\alpha) + \Delta_y \sin(\alpha)$ and after applying this to Hooke's Law, we get

$$F(e) = G(e)(\Delta_x \cos(\alpha) + \Delta_y \sin(\alpha)).$$

When writing this equation for each coordinate for a single rod *e*, we get

$$\begin{pmatrix} F(e)_x \\ F(e)_y \end{pmatrix} = G(e)\begin{pmatrix} \cos^2(\alpha) & \cos(\alpha)\sin(\alpha) \\ \cos(\alpha)\sin(\alpha) & \sin(\alpha) \end{pmatrix}\begin{pmatrix} \Delta(e)_x \\ \Delta(e)_y \end{pmatrix}$$

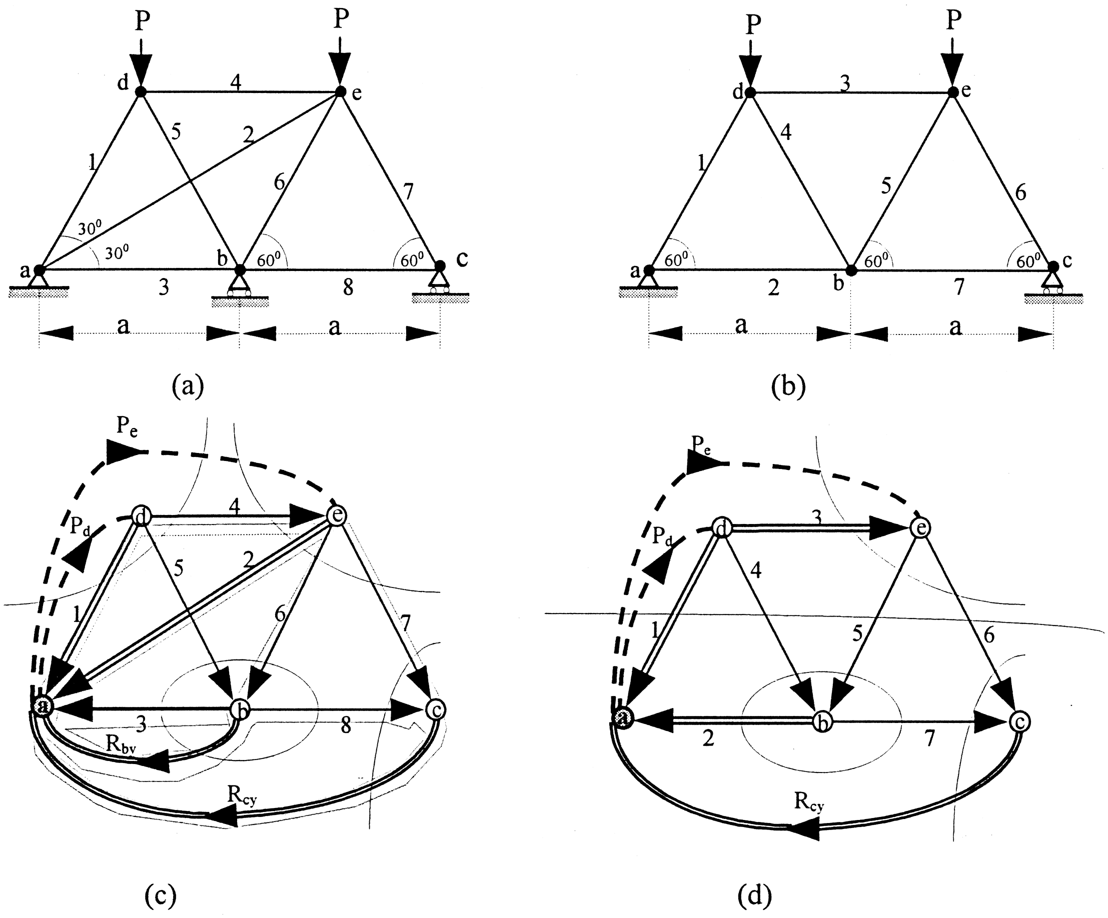
$$= G(e)\begin{pmatrix} \Delta(e)_x \\ \Delta(e)_y \end{pmatrix}$$

where $G(e)$ is a 2 × 2 matrix which appears in Fig. 7(e).

5. Checking the validity, and analysis of, a planetary gear system

It may appear a trivial task to know if a gear-box has a well-defined topology of which gears mesh with which others, but for a complex gearbox this can be a difficult problem. A case is known of a half-million dollar complex gearbox for a large power plant, in which the gear wheels stripped when first powered up, because its topology was not well defined, or in other words the syntax of the diagram that described it was not a WFF (K. Preiss, unpublished).

The same general process explained earlier with trusses, is now used when dealing with planetary gear systems. First,



$$\begin{pmatrix} 0 \\ Rb_y \\ 0 \\ Rc_y \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} G3+G5+G6+G8 & -G8 & -G5 & -G6 \\ -G8 & G7+G8 & 0 & -G7 \\ -G5 & 0 & G1+G4+G5 & -G4 \\ -G7 & -G7 & -G4 & G2+G4+G6+G7 \end{pmatrix} \begin{pmatrix} \Delta bx \\ 0 \\ \Delta cx \\ 0 \\ \Delta 1x \\ \Delta 1y \\ \Delta 2x \\ \Delta 2y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ P \\ 0 \\ P \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(e)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -c(\alpha_6) & -c(\alpha_7) \\ -1 & 0 & 0 & 0 & 0 & 0 & -s(\alpha_6) & -s(\alpha_7) \\ 0 & c(\alpha_1) & 0 & 0 & c(\alpha_4) & c(\alpha_{15}) & c(\alpha_6) & 0 \\ 0 & s(\alpha_1) & 0 & 0 & s(\alpha_4) & s(\alpha_5) & s(\alpha_6) & 0 \\ 0 & 0 & c(\alpha_2) & 0 & -c(\alpha_4) & -c(\alpha_5) & 0 & c(\alpha_7) \\ 0 & 0 & s(\alpha_2) & 0 & -s(\alpha_4) & -s(\alpha_5) & 0 & s(\alpha_7) \\ 0 & 0 & 0 & c(\alpha_3) & 0 & -c(\alpha_5) & -c(\alpha_6) & 0 \\ 0 & 0 & 0 & s(\alpha_3) & 0 & -s(\alpha_5) & -s(\alpha_6) & 0 \end{pmatrix} \begin{pmatrix} Rc \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} P_e \\ P_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(f)

Fig. 7. Example of analysis of determinate and indeterminate truss by using the graph representation: (a) is an indeterminate truss, (c) its graph and (e) its equations derived from the cutset matrix. (b) is a determinate truss, (d) its graph and (f) its equations derived from the cutset matrix.

check if the gear system is valid, by checking if the system has a well-formed topology. In the truss problem, an efficient method was embedded in the representation for checking the necessary and sufficient conditions for deciding the validity, or rigidity, of the truss. For planetary systems, the necessary conditions can be found in the book by Erdman [17]. Based on the information in that book, the conditions are given later as rules to check if the graph representation has a well-formed topology. For this, and for the analysis which follows, the embedded theorems in the graph representing the gear box, which make use of the spanning tree in the graph, are used.

5.1. The representation of the planetary system

The most important property to be emphasized in this representation is the connection between the system elements, showing how element *i* is connected to element *j*. The line graph representation is suitable for this purpose, and therefore, every rotation rod or gear wheel will be represented by a vertex, and the connection between a pair of links by an edge. There is a special type of vertex which is colored gray that corresponds to a link or planet carrier which maintains the distance between a pair of gear wheels. In the literature [18] this vertex is called a ‘transfer vertex’.

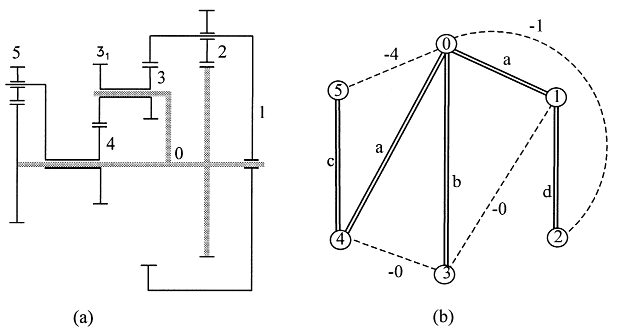


Fig. 8. The planetary mechanism (a) and its line graph representation (b) (*Note*: (a) is a standard representation in engineering drawing for a gear system).

In the terminology of this paper, the name 'local reference vertex' is more suitable. In this representation, all the turning edges on one side of the local reference vertex are at the same level, and those on the opposite side of the local reference vertex are at a different level (The word "level" refers to the radial distance from the centerline of the whole gear system.) (see Fig. 8):

A dashed line
An edge which represents a gear connection.
A double line
An edge which represents a turning connection, and because the turning edges form a spanning tree they are shown as double lines. Every double line has a label which represents the level.
Other information about the labeled edges and the vertices is added to the representation as follows:
Gray vertices
The distance between each pair of connected gear wheels must be constant throughout, there being a link or planet carrier, which maintains this distance. In this representation, all the turning edges on one side of the local reference vertex are at the same level, and those on the opposite side of the local reference vertex are at a different level.

5.2. Checking the validity of the planetary gear system

As explained earlier, the process of checking the validity of a planetary system becomes a process of checking whether it has a well-formed topology. The properties of the graph representation given for this problem are based on Erdman [17], who published a set of necessary conditions that are used to check whether the system is physically infeasible. Erdman's conditions are here rephrased in graph terms.

Proposition 6. There is no circuit formed exclusively by turning edges.

Explanation: Suppose in contradiction to the rule, that a circuit of turning edges exists. There would then be, in the chain, a set of pin-connected links. A circuit of sizes 1 or 2 is not feasible. A circuit of size 3 is a triangle which is a locked structure. In a circuit of size 4 or more the rotatability of the links would not be proportional. This contradicts the hypothesis that the system is a proportional kinematic chain.

Proposition 7. All the vertices must incident to at least one turning edge.

Explanation: Every link, which is represented as a vertex, has at least one element which rotates around it. Between these two elements, there will be a turning edge in the graph representation. There may be elements, such as a planet carrier, for which the vertex that represents them is incident to at least two turning edges.

Proposition 8. The subgraph of the turning edges forms a connected subgraph.

Explanation: Each connected gear pair should operate with a constant radius or center distance. This distance is maintained by the planet carrier, which is either directly paired to ground or connected to ground through a sequence of turning edges.

Proposition 9. In each fundamental circuit, there is one local reference vertex, and all the edges on one side of the local reference vertex are at the same level, while all the edges on the opposite side of the local reference vertex are at a different level.

Table 2
Embedded properties of the line graphs which correspond to planetary gear systems

No.	Embedded property	Derived from	Graph theory formulation
1	The subgraph formed by turning edges is a spanning tree	Propositions 6–8	A subgraph that is connected, with no circuits, and contains all the vertices, is a spanning tree
2	Every gear edge forms a fundamental circuit with the spanning tree	Embedded property 1	Adding an edge to the spanning tree forms one and only one circuit
3	$e(T) = v(G) - 1$	Embedded property 1	The number of spanning tree vertices is one more than the number of its edges
4	$g(G) = v(G) - 2$	From embedded property 1 and Gruebler theory	
5	$g(G) = e(T) - 1$	From embedded properties 3 and 4	

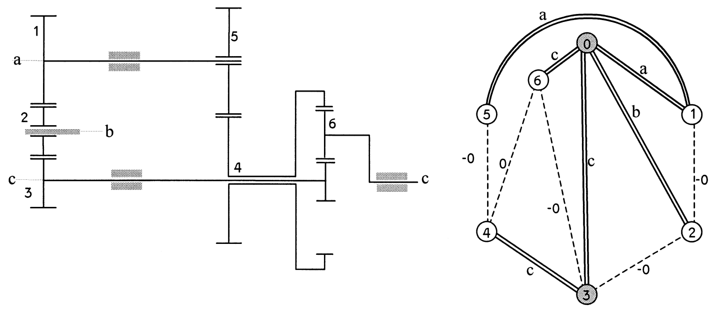


Fig. 9. Example of topological analysis of a planetary gear system, with the computer program output shown. (*Note*: The system is not valid because there is contradiction with Proposition 9, because in circuits {6,0,3} and {6,0,3,4} there is no local reference vertex. The explanation to the user is: *the connection between wheels 6 and 3 is not legal because the distance between their centers is zero. The same problem occurs with the connection between wheels 6 and 4.*)

Explanation: Each gear pair is located on a different turning edge level. Because the distance between the centers of these two gears must be constant, there is one and only one planet carrier (local reference vertex) in the fundamental circuit defined by this gear pair.

In addition, in the graph representation syntax, there are embedded properties, part of which are listed in Table 2. The Gruebler theory referred to in the table is well known in theory of machines and can be found for instance in Erdman [17].

5.2.1. *The diagnostic system for the planetary system*

With this representation, checking the validity of the system becomes a problem of checking whether there is a contradiction between the domain knowledge (in this case the embedded properties and propositions) and the graph representation of the given system. For example, the computer program using this representation with the rules given before to analyze a gear system [6,19] found that the system in Fig. 9 is not valid. By knowing which rule was contradicted, but stating the cause of conflict not in graph terms, but in "user" terms, the program explained why it was not valid. Following the same approach it is possible to arrange that the computer program advise the designer what to change in the gear kinematic chain, in order that it would be valid.

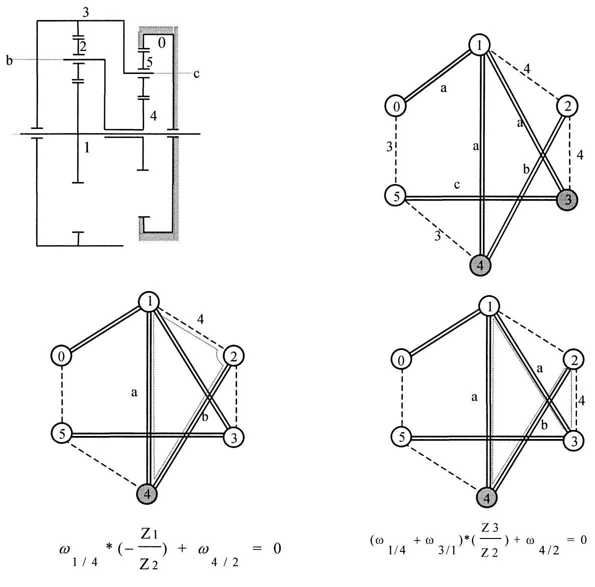
5.3. *Analysis of the velocities in the planetary systems*

The variables that represent the angular velocities of the links must satisfy the Potential Law. So, by using the potential graph representation, embedded property 2 implies that every circuit is defined by a gear edge, and that circuit yields an equation. To solve the system it is necessary that the number of variables equals the number of equations. This is so because the number of variables is equal to $e(T)$ which according to embedded property 5 of Table 2 is equal to number of equations plus the data for the driving link. Fig. 10 shows an example of this analysis where from each circuit an equation is derived. For example, Fig.10(c) shows the equation derived from the circuit with vertices '1,4,2', which is defined by chord '12'. In this circuit, the reference vertex is '4', so going along this circuit from vertex '4', applying the Potential Law, we get: $\omega_{1/4} \times i_{12} + \omega_{4/2} = 0$, where i_{12} is the gear ratio between gear wheels 1 and 2. As the transmission between these two gear wheels is external, the sign of the ratio is negative, and we get: $\omega_{1/4}(-(Z_1/Z_2)) + \omega_{4/2} = 0$, where Z_i is the number of teeth in gear wheel i. The number of gear teeth is proportional to the radius or a circumference of the wheel.

6. Application to other engineering systems

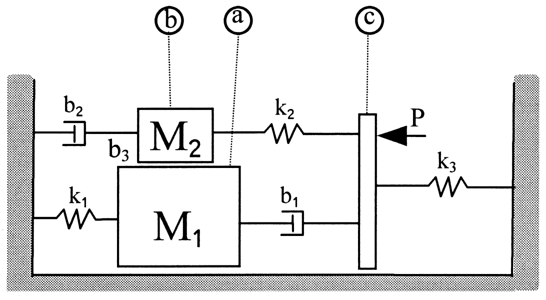
In this paper, a method was explained while solving two engineering problems, using the knowledge embedded in the graph theory representation. This representation can be used in other fields of engineering which today seem to belong to disjoint engineering domains. For example, analysis of dynamic systems, as shown in Fig. 11, is based on the cutset and circuit properties which were explained in Table 1 for trusses. As a result analysis of indeterminate trusses and dynamic systems become similar.

This paper showed how embedded engineering

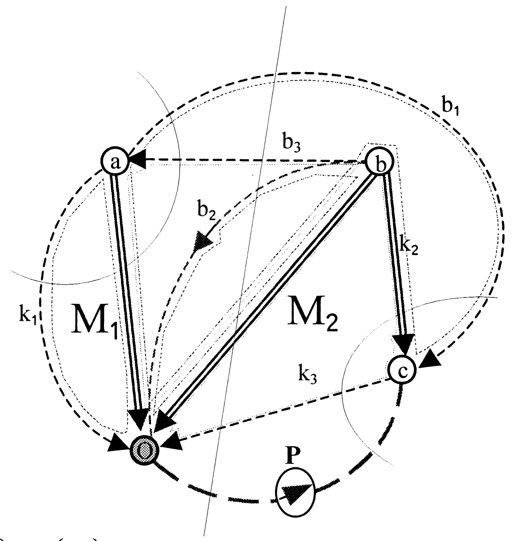


$$\omega_{1/4} * \left(-\frac{Z1}{Z2}\right) + \omega_{4/2} = 0 \qquad (\omega_{1/4} + \omega_{3/1}) * \left(\frac{Z3}{Z2}\right) + \omega_{4/2} = 0$$

Fig. 10. Example of analysis of a planetary gear system.



(a)



(b)

$$\begin{pmatrix} m_1 s+\frac{k_1}{s}+b_1+b_3 & -b_1-b_3 & b_1 \\ -b_1-b_3 & m_2 s+\frac{k_3}{s}+b_1+b_2+b_3 & -b_1-\frac{k_3}{s} \\ b_1 & -b_1-\frac{k_3}{s} & \frac{k_2}{s}+\frac{k_3}{s}+b_1 \end{pmatrix}\begin{pmatrix}\Delta_{ao}\\ \Delta_{bo}\\ \Delta_{bc}\end{pmatrix}=-\begin{pmatrix}0\\ -1\\ 1\end{pmatrix}P$$

(c)

Fig. 11. Example of analysis of a dynamic system using the graph representation: (a) a dynamic system, (b) the graph, (c) equations derived from the representation.

knowledge is derived from the properties embedded in the graph representation, and can be used to derive the engineering equations of the systems. By using this method, new algorithms can also be derived for many problems from the embedded properties of the representations. For example, Shai [6] shows how the known best first search algorithm used in artificial intelligence [1] is derivable from the embedded properties in the Discrete Linear Programming (DLP) representation. Moreover, in the LP representation, by using the embedded "primal dual" algorithm, a new algorithm was derived [6] to find the maximum external force that can be applied to a general truss constructed from ideal elastoplastic elements.

7. Reasoning by analogy based on the embedded knowledge

There are many other applications in this direction of research. This section will introduce briefly how one can reason by analogy based on the connection between the syntaxes of the representations. As an example, the equations for analyzing a determinate truss are derived from the flow graph representation and the velocity equations for mechanisms are derived from the potential graph representation [6]. Because the cutset and circuit are dual [12], one can derive the relation that the flow and potential graphs are dual. The dual to a plane graph (a graph embedded in a plane such that no edge crosses another) has a vertex for each face of the original graph, and an edge which crosses each edge of the original graph at right angles. Given the plane graph of a mechanism one can therefore draw its plane truss dual, and vice versa. Fig. 12 shows a mechanism (a) that is dual to a truss (b), and graphs (c) and (d) show the process of deriving the mechanism from the truss using the dual connection between their corresponding graphs.

The duality between trusses and mechanisms can be used in many directions. For example, one can deduce the solvability property of a mechanism by checking its dual truss. An instance is Fig.13(b) that shows a mechanism which satisfies the Gruebler theorem [17] and can be divided into legal kinematic chains. But, in the dual truss, it is transparent that it is not rigid, so one can conclude that the corresponding mechanism is not legal. Indeed, to the surprise of three human experts it was found that this mechanism in the geometry of Fig. 13 is actually locked in that geometry. In the terminology of Simon [3], looking for a proper representation not only makes the solution transparent but also makes transparent whether or not the diagram showing the engineering system is well defined.

Other results of this project have included successful use of these reasoning methods in high school classes, where students have assimilated the experience of using several representations to solve, or reason about, an engineering system. In the last decade over 300 high school students have successfully attained a much better-than-usual grasp of both mathematics and physics by using a variety of representations. We postulate that this success is partially owing

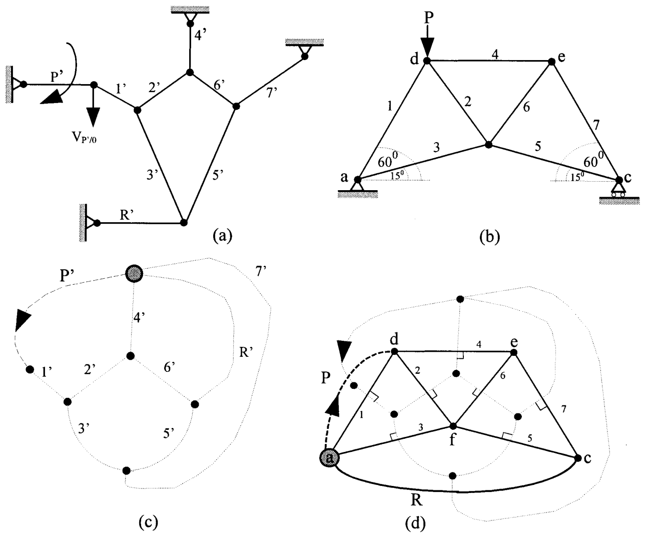


Fig. 12. Example of the duality between a mechanism and a determinate truss: (a) the mechanism, (b) the truss, (c) the graph of the mechanism, (d) the graphs of the truss and mechanism shown superimposed.

to the use of multiple approaches and the ability to switch seamlessly from one representation to its analogy. Confirmation of this hypothesis awaits experiments with trial groups to investigate the related cognitive process.

The representations developed and used, for which a series of papers is now in preparation, increase the available knowledge about the theory behind various engineering systems, both because of the added understanding available from any one representation, and the information generated by investigation of analogies between representations.

8. Concluding remarks

Simon's observation on the usefulness of a mathematical representation which is isomorphic to the elements of an engineering system was shown by the use of the discrete representations of graph theory, matroids and linear programming [6]. This paper showed results only for the network graph theory representation, where the cutset syntax or the flow graph representation was used to solve truss structures, and the circuit or potential graph representation syntax used to solve planetary gear systems. Using the graph theory representation enables application of graph theory which is a representation with properties and algorithms which are known and with proven properties. Among these are efficient, low complexity, and hence useful algorithms. Computational systems based on these will not only be provably correct, but will enable efficient computation on large systems with many elements.

Using mathematical representations with properties embedded in the representation which match physical properties of engineering systems, enables the development of computational reasoning and analysis systems for engineering analysis, based on mathematically proven properties of the representation, with algorithms which have proven properties. The approach produced interesting overall perspectives of the engineering systems, and when the same representation is applicable to different systems, opened new possibilities for reasoning by analogy.

The engineering knowledge embedded in the syntax of the graph representation enables one to explicitly and systematically determine if the diagram defining a given engineering system (in this paper a truss or planetary gear system) is a WFF. If it is a WFF, the given system is a valid initial state for analysis and reasoning, using algorithms of proven properties and complexities. If it is not, the system will either have an invalid solution or will be insolvable.

Acknowledgements

The work was partially supported by the Pearlstone Center for Aeronautical Engineering Sciences, and by the Paul Iwanir Center for Robotics Research and Production Management, at the Ben Gurion University of the Negev.

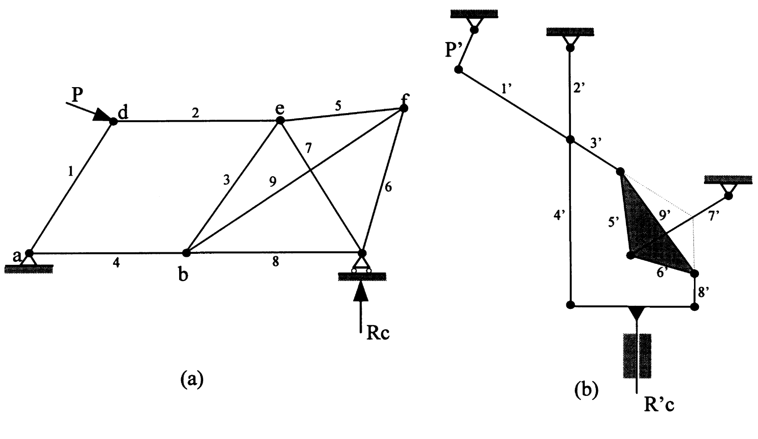


Fig. 13. Example of not-rigid truss and its corresponding dual mechanism: (a) the not-rigid truss, (b) the corresponding dual mechanism.

References

[1] Nilsson NJ. Problem-solving methods in artificial intelligence. New York: McGraw-Hill, 1971.
[2] Korf RE. Toward a model of representation changes. Artificial Intelligence 1980;14:41–78.
[3] Simon HA. The sciences of the artificial. 2. Cambridge, MA: MIT Press, 1981.
[4] Baalen J. Second Workshop on Change of Representation and Problem Reformulation, Menlo Park, CA, 1990.
[5] Robinson J. A machine oriented logic based on the resolution principle. J ACM 1965;12(1):23–41.
[6] Shai O. Representation of embedded engineering knowledge for artificial intelligence systems. Ph.D. thesis, Ben Gurion University of the Negev, 1997.
[7] Swamy MN, Thulasiraman K. Graphs: networks and algorithms. New York: Wiley, 1981.
[8] Timoshenko SP, Young DH. Theory of structures. Singapore: McGraw-Hill, 1965.
[9] Laman G. On graphs and rigidity of plane skeletal structures. J Eng Math 1970;4:331–340.
[10] Lovasz L, Yemini Y. On generic rigidity in the plane. SIAM J Algebraic Discrete Meth 1982;3(1):91–98.
[11] Nash-Williams C. Edge-disjoint spanning trees of finite graphs. J London Math Soc 1961;36:445–450.
[12] Recski A. Matroid theory and its applications in electric network theory and in statics. Berlin: Springer-Verlag, 1989.
[13] Balabanian N, Bickart TA. Electrical network theory. New York: Wiley, 1969.
[14] Preiss K, Shai O. Deep artificial intelligence knowledge for truss analysis. The 25th Israel Conference on Mechanical Engineering, Haifa, Israel, 1994:207–209.
[15] Fenves SJ, Branin FH. Network topological formulation of structural analysis. J Structural Division, ASCE 1963;89(ST4):485–514.
[16] Preiss K, Moisa R, Shai O. Truss analysis by graph theory. The 26th Israel Conference on Mechanical Engineering, Haifa, Israel, 1996:621–623.
[17] Erdman AG, editor. Modern kinematics – developments in the last forty years New York: Wiley, 1993.
[18] Freudenstein F. An application of boolean algebra to the motion of epicyclic drives. ASME J Engineering for Industry 1971;93:525–532.
[19] Polomodov B, Gershon T. Matriculation project: checking the validity and analysis of planetary gear system. High School Ort Rehovot, 1995.