Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXV, pp. 351-373
Alberto Cardona, Norberto Nigro, Victorio Sonzogni, Mario Storti. (Eds.)
Santa Fe, Argentina, Noviembre 2006

# A DECOMPOSITION METHOD FOR MODULAR DIMENSIONAL SYNTHESIS OF PLANAR MULTI-LOOP LINKAGE MECHANISMS

**Martín A. Pucheta and Alberto Cardona**

*Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), INTEC (Universidad Nacional del Litoral-CONICET), Güemes 3450, S3000GLN Santa Fe, Argentina,*
*e-mails: {mpucheta,acardona}@intec.unl.edu.ar, http://www.cimec.org.ar*

**Keywords:** Multiloop linkage mechanisms, dimensional synthesis, Graph Theory, combinatorial optimization, object-oriented programming.

**Abstract.** The essence of mechanism synthesis is to find the mechanism for a given motion or task. There are three customary tasks for kinematic synthesis: function generation, path generation and rigid-body guidance. The task is often defined by a number of prescribed displacements and orientations called *precision points*. Conceptual design of mechanisms has two main stages: (i) *Type Synthesis*, where the number, type and connectivity of links and joints are determined, and (ii) *Dimensional synthesis*, where the link lengths and pivot positions at the starting position are computed. From the first stage we already get a mechanism represented by a graph (Pucheta and Cardona, In *Mecánica Computacional*, volume XXVI, *proc. of MECOM 2005*, Buenos Aires, Argentina). To evaluate its feasibility to fulfill a given task it must necessarily have dimensions. To this purpose, we implement a strategy developed by Sandor and Erdman (*Advanced Mechanism Design: Analysis and Synthesis*, vol. 2, Prentice-Hall, 1984). This strategy consists in: (a) decomposing the complex mechanism topology into Single Open Chains (SOCs), (b) solving dimensionally each SOC using complex numbers and the analytical Precision Point Method, and (c) reassembling the solutions. Decomposition of complex multiloop linkages into single subsystems was deeply studied for automated kinematic and dynamic analysis. However, its use in automated synthesis applications is less addressed in the literature. The proposed SOCs Decomposition algorithm uses the graph structure, the geometry of the prescribed parts and the motion constraints data imposed on them. The resultant order of SOCs is not unique, there could be many valid orders. The optimal order will be a compromise between what best satisfies the solvability (number of equations for linearization required by analytical methods) and what best matches the number of prescribed motion constraints given by the precision points. In spite of the complexity of this method, it produces multiple good initial guesses for subsequent optimization stages based on gradient methods which often fail because of the bifurcating and highly non-linear nature of this inverse problem.

The method was programmed in *C++* language under the *Oofelie* environment (Cardona et al., *Engng Comp*, 11:365–381, 1994).

## 1 INTRODUCTION

The essence of mechanism synthesis is to find the mechanism suited for a given motion or task. Among the wide range of inverse problems in the field of mechanism synthesis we are focused on kinematic synthesis, i.e., to find the mechanism which satisfy exactly, or in its defect approximately, a set of displacements and/or orientations prescribed on one or more of their links or joints. In these problems masses and inertias of the constituting parts are ignored, so the skeleton representation is extensively used. Also, because of the discrete nature of the mechanism, Graph Theory is suitable to represent them. There are three customary tasks for kinematic synthesis: function generation (FG), path following (PF) (or path generation), and rigid-body guidance (RBG); but also, more complex tasks could be defined by combining them for dual-task or multiple-task purposes.

Conceptual design of mechanisms has two main stages: (i) *Type Synthesis*, where the number, type and connectivity of links and joints are determined for the required degree of freedom and structural constraints, and (ii) *Dimensional synthesis*, where the link lengths and pivot positions at the starting position are computed.

We look for a task-oriented synthesis where most or all of the requirements are satisfied while the enumeration of candidate mechanism solutions takes place. In previous works, we presented an automated task-oriented Type Synthesis method based on a subgraph search (Pucheta and Cardona, 2005a,b) where, from a FEM (Finite Element Method) description of the kinematic task including prescribed parts, we define a graph representation, and then we search this graph representing the problem inside an atlas of previously enumerated candidate mechanisms. As answer of the type synthesis software, we obtain the list of all non-isomorphic topologies for the given kinematics problem. Now we desire to evaluate and sort them in a ranking of optimality, and therefore, they must necessarily have dimensions.

There is not a general method to find the dimensions for any arbitrary closed-loop mechanism topology for the wide range of kinematics problems. The closed-form equations to solve kinematic synthesis problems are current areas of interest. *Analytical Synthesis* by means of closed-form equations is applicable when the topology is decomposed into single sub-systems called Single Open Chains (SOC) and the *task is simplified* by defining a number of finitely separated displacements and/or orientations called *precision points*[1]. The method for this simplified synthesis problem is known as the Precision Point Method (PPM)[2].

In order to apply analytical synthesis to solve complex-loop linkages, the most popular strategy is based on:

(i) decompose the topology into SOCs;

(ii) solve analytically each SOC in the order given by the previous step using complex-numbers for representing the links (Sandor, 1959; Sandor and Erdman, 1984; Lin et al., 1996), and

(iii) reassemble the sized SOCs to reconstruct the topology.

Although this strategy was successfully implemented in academical and commercial computer programs[3], and used to solve most of the linkages employed in industry and life, general

[1] Also known as “passing points”, “accuracy points”, “precise positions” or “finite positions”.

[2] Also known as “Finite Position Synthesis” and ‘Burmester Synthesis”.

[3] We can mention KINSYN (Kaufman, 1971), LINCAGES (Erdman and Gustafson, 1977), RECSYN (Waldron and Song, 1981), SAM (Rankers), TADSOL (Crone et al.), SYNTHETICA (McCarthy), WATT (Draijer and Kokkeler), and SYMECH (Cook), among others.

automated methods for dimensional synthesis using analytical synthesis are less addressed in the literature (Olson et al., 1987; Yannou and Vasiliu, 1995; Sardain, 1997; Yang et al., 1998).

The decomposition of the topology into SOCs is not unique, also the first two steps (i-ii) are strongly interrelated, because the solvability of one SOC may be dependent on a previous one. So all possible decompositions must be carefully analyzed. We shall remark that the decomposition method is the key step between the stages of *Type Synthesis* and *Dimensional Synthesis by analytical methods*.

The aim of this paper is to develop a method for the step (i) which best consider the given task and the subsequent solvability of the resulting SOCs using modules programmed by analytical synthesis techniques. The problem is solved using a FEM like description for the topology and the SOCs.

Although it is a well-known fact that the PPM is merely geometric and does not coincide, in general, with the kinematic requirements, our aim will be, whenever we can, to use the PPM as a first evaluation tool of the topology. Then, we can make some considerations based on kinematics and dynamics to evaluate and filter *defects* in the sized mechanisms (Balli and Chand, 2002b,a). And finally, we can use each of the filtered results as an initial guess to use later an optimization software (i.e. SAMCEF BOSS) which minimizes the error between the generated and the truly desired task (Pucheta and Cardona, 2005a).

The paper organization is as follows. In Section 2 we review the representation of a mechanism topology using graphs and the type synthesis process. In Section 3 we present the proposed method for graph decomposition. In Subsection 3.2 we review the dimensional synthesis methods for opened-chains using analytical methods. Finally, in Section 4 we present the applications to kinematics problems in complex linkages.

## 2 A TYPE SYNTHESIS OUTPUT

We first model the kinematics problem using a CAD interface (SAMCEF Field) common with a program of mechanism analysis (SAMCEF Mecano). In a FEM description of a mechanism, the rigid bodies are constructed from nodes, and using these nodes the body can be constrained by joints of different kinds, either to other bodies or to nodes fixed to ground, see Geradin and Cardona (2001). Once we sketch some parts or elements of the sub-mechanism, we can define *motion constraints* (the task) simplified into three or four precision points (Subsection 2.1). Then, for this desired kinematics task we define a graph representation called *initial graph* modeling mathematically the synthesis problem (Sub-section 2.2). After the execution of a type synthesis software the initial graph allows to find potential mechanism alternatives.

### 2.1 Motion constraints

In planar problems, we can impose three constraints per rigid body or link: two translations and one rotation on the axis perpendicular to the work plane. The motion constraints may be defined on nodes, links or joints (as motorization or input of motion). Thus, the *motion constraints* could be: sets of node displacements $\mathcal{D}$, sets of link rotations $\mathcal{L}$, and sets of joint parameters $\mathcal{J}$, e.g., rotations for revolute joints and displacements over the joint axis for the prismatic ones.

To define a trajectory, we give a *set of node displacements*

$$\mathcal{D} = \{N_{ID}, t, (d_x, d_y); ...; \},$$

where $ID$ is the node identifier, $t$ is the number of passing points in the sequence of precise positions, and the displacements, $d_x$ and $d_y$, are expressed in relative coordinates from the initial node position.

The description for the *set of link rotations* is

$$\mathcal{L} = \{E_{ID}, t, \alpha_t; ...; \},$$

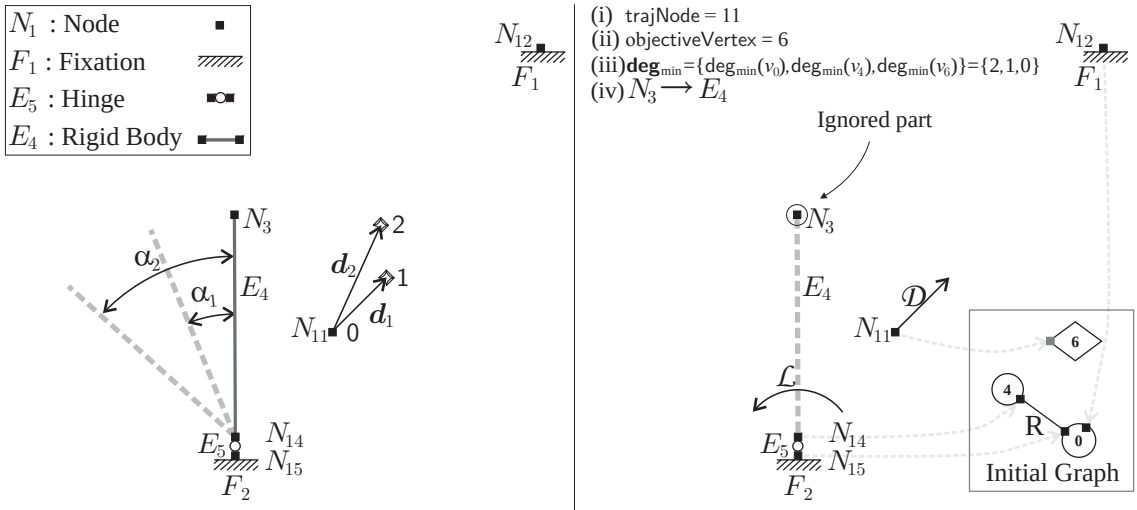
where $ID$ is the rigid body element identifier, and $\alpha_t$ is the rotation angle expressed in relative coordinates from the initial position.

A similar description is given for the *set of joint parameters*

$$\mathcal{J} = \{E_{ID}, t, \alpha_t; ...; \},$$

but the entered parameter must be coherent with the joint type element, i.e. angles $\alpha_t$ for revolute joint elements, and displacements $\rho_t$ for the prismatic joint ones; both expressed in relative coordinates from the initial position. We shall remark that all prescriptions are expressed "in relative coordinates from the initial position" because this particular way to define the problem allows the user to leave some angles or displacements without being defined for some precise positions.

Different combinations of the sets $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{J}$, gives us the well-known classification for kinematics problems into FG, PF, and RBG. For instance, a function generation task could be defined by imposing two sets of rotations either on the joints $\mathcal{J}$ or on the links $\mathcal{L}$ of two grounded cranks.




**a)** F.E.M. description of the kinematic task

**b)** Graph representation

Figure 1: Translation from F.E.M to graph representation for a path following kinematics problem.

A path following problem could be defined by requiring that a point of the unknown mechanism satisfies a set of displacements $\mathcal{D}$. If we simultaneously require that a point of an unknown link satisfies a set of displacements $\mathcal{D}$, and we also prescribe a set of orientations $\mathcal{L}$ the task is a rigid body guidance problem. Other combinations of these sets would result in problems which do not match with the mentioned classification.

For example, in Figure 1-a we can see the problem of guiding one point of an unknown mechanism by three positions with prescribed timing: the set of displacements $\mathcal{D}$ defined on

node $N_{11}$ must be in coordination with the set of rotations $\mathcal{J}$ defined on the joint $E_5$ of the crank $E_4$. We declare three triplets for the two displacements desired on node $N_{11}$,

$$\mathcal{D} = \{N_{11}, 0, (0,0); N_{11}, 1, \boldsymbol{d}_1; N_{11}, 2, \boldsymbol{d}_2\}.$$

The crank rotations could be defined on joint $E_5$ as:

$$\mathcal{J} = \{E_5, 0, 0; E_5, 1, \alpha_1; E_5, 2, \alpha_2\}.$$

### 2.2 FEM to graph translation

From the kinematics information sketched in Figure 1-a, we construct an *initial graph* plus a set of additional variables to be used as input for the type synthesis stage (Figure 1-b, variables *i-iv*).

The graph $G(E, V)$ of a mechanism $\mathcal{M}$, is formed by a set of vertices $V$ representing the links, and a set of edges $E$ representing the joints connecting the links. The degree $d(v)$ of a vertex $v$ is the number of edges incident to it. Particularly, the initial graph that we define is a *labelled graph with attributes*. We take the IDs of the elements coming from CAD to use them as "labels" of the vertices and edges of the graph, and we also take the link and joint types defined by the user on the elements to give "attributes" of vertices and edges, respectively.

In a FEM description, the nodes can have different attributes: e.g. clamped node (with fixations), isolated node with prescribed movement, node taking part in the geometry of a rigid-body. A rigid-body can have $n_b$ nodes (with $n_b \geq 2$),

$$n_b = n_j + n_p + n_c,$$

where, $n_j$ nodes are used to assemble the body with other bodies nodes by means of *joints*, $n_p$ nodes are not assembled but have *prescribed* movements $\mathcal{D}$, and $n_c$ nodes only give the shape of the body. These latter nodes are ignored for the type synthesis purposes, however, they are stored in an auxiliary node-to-element integer map to be used later in the dimensional synthesis stage as *checking* points of the task, or to define the restricted area (or space) where the other links must hold for each configuration.

We define some rules to translate the FEM representation to a graph:

**Generation of Vertices:** The analysis of all nodes, fixations, and rigid-bodies and joint elements is required.

V1) **Isolated nodes:** For each isolated node with prescribed movements, an isolated vertex in the graph is assigned; the node does not belong to any element.

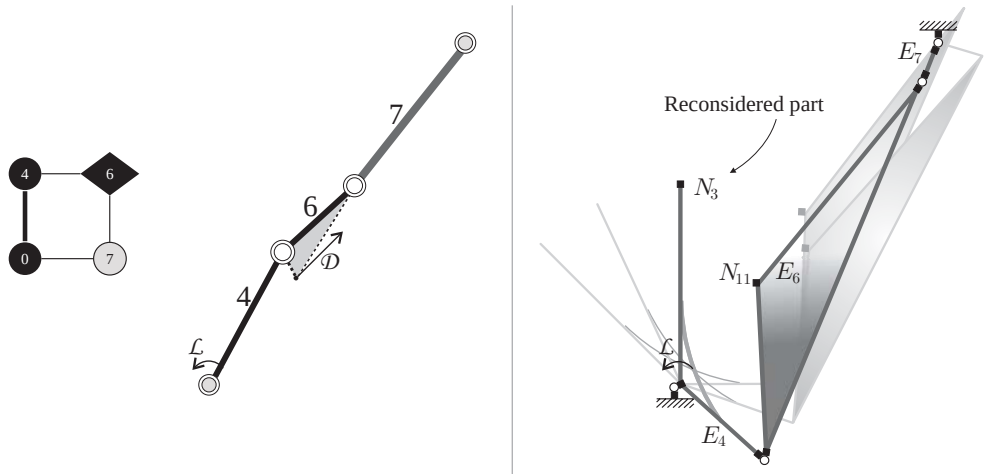
V2) **Rigid-bodies:** Free bodies with imposed movements, i.e., rotations $\mathcal{L}$ or displacements $\mathcal{D}$ on some of their nodes, will be isolated vertices of the initial graph. The remaining bodies, connected through joints, will be connected vertices of the graph. Each vertex has a *minimum degree constraint* $\mathsf{deg}_{\min}(v_i)$ equal to the number of nodes connected by joints $n_j$ in the corresponding rigid-body $i$.

V3) **Fixations:** Conventionally, the ground link will be the vertex zero . Depending on the number of grounded bodies, this vertex may be binary, ternary, etc. The isolated fixations represented by fixed nodes are also used as prescribing degree of the vertex zero, that is$\mathsf{deg}_{\min}(v_0) =$ #fixations.

**Generation of Edges:** All joints will be edges of the initial graph connecting two of the previous defined vertices, so that they are assumed to be binary (isolated joints are not allowed).

See, for example, in Figure 1 the element $E_4$ with $n_b = 2$, $n_p = 0$, $n_j = 1$ and $n_c = 1$. Following rule V1 the element is translated to a vertex $v_4$ with minimum degree 1. The node $N_3$ shown as "circled" in the element does not participate in the initial graph; however, it is stored in a map (iv) indicating $N_3 \rightarrow E_4$. Then, for the isolated node $N_{11}$ which will develop the required trajectory $\mathcal{D}$, we used rule V2 to create a new vertex $v_6$ with zero degree. Additionally, two integer variables are set and used later in the synthesis process. They are trajNode, the ID of the node which will develop the required trajectory, and the objectiveVertex defined as the vertex of the graph representing the mechanism which has the trajNode. Finally, by means of rule V3, the two fixed nodes $N_{12}$ and $N_{15}$, introduce two constraints to the degree of the ground vertex $v_0$. A list called *minimum degree of vertices* has the *minimum degree constraint* for each vertex. For example, it is filled as $\mathbf{deg}_{\min} = \{\mathsf{deg}_{\min}(v_0), \mathsf{deg}_{\min}(v_4), \mathsf{deg}_{\min}(v_6)\} = \{2, 1, 0\}$.

The constructed *initial graph* has labelled vertices with link type attributes, labelled edges with joint type attributes and the properties objectiveVertex, trajNode and $\mathsf{deg}_{\min}$. Using this information the initial graph is searched, as a pattern to match, inside an atlas of mechanisms. In the path following example, we set the search parameters as: (a) atlas of rigid one-degree of freedom mechanisms, (b) maximum distance from the objective vertex to ground equal to 2 (the minimum distance is always set to 2 when a trajNode is chosen by the user), and (c), avoidance of pseudo-mechanisms. Each matching is a feasible mechanism topology $\mathcal{M}_a$ which inherits the motion constraints $\mathcal{D}, \mathcal{J}, \mathcal{L}$ of the task.




**a)** Type Synthesis: graph and sketch. **b)** Dimensional Synthesis: FEM description.

Figure 2: The simplest solution for the two stages of synthesis for a path following problem.

Figure 2-a shows the graph and the sketch for the simplest first alternative found by the type synthesis execution. In the graph, the ground has the label zero, and the objective vertex has a diamond shape. The grey vertex (7) and their incident edges are the new type synthesized link and joints, respectively. In Figure 2-b we put the dimensional synthesis result only to remark that the ignored part is restored to its original element to complete and evaluate the mechanism. In these figures we can see that body $E_6$ is binary in terms of number of joints as considered in the graph; but it is physically ternary in terms of number of nodes. In the same mechanism, the body $E_4$ is binary in the graph, but it has three nodes after reconsidering the ignored part.

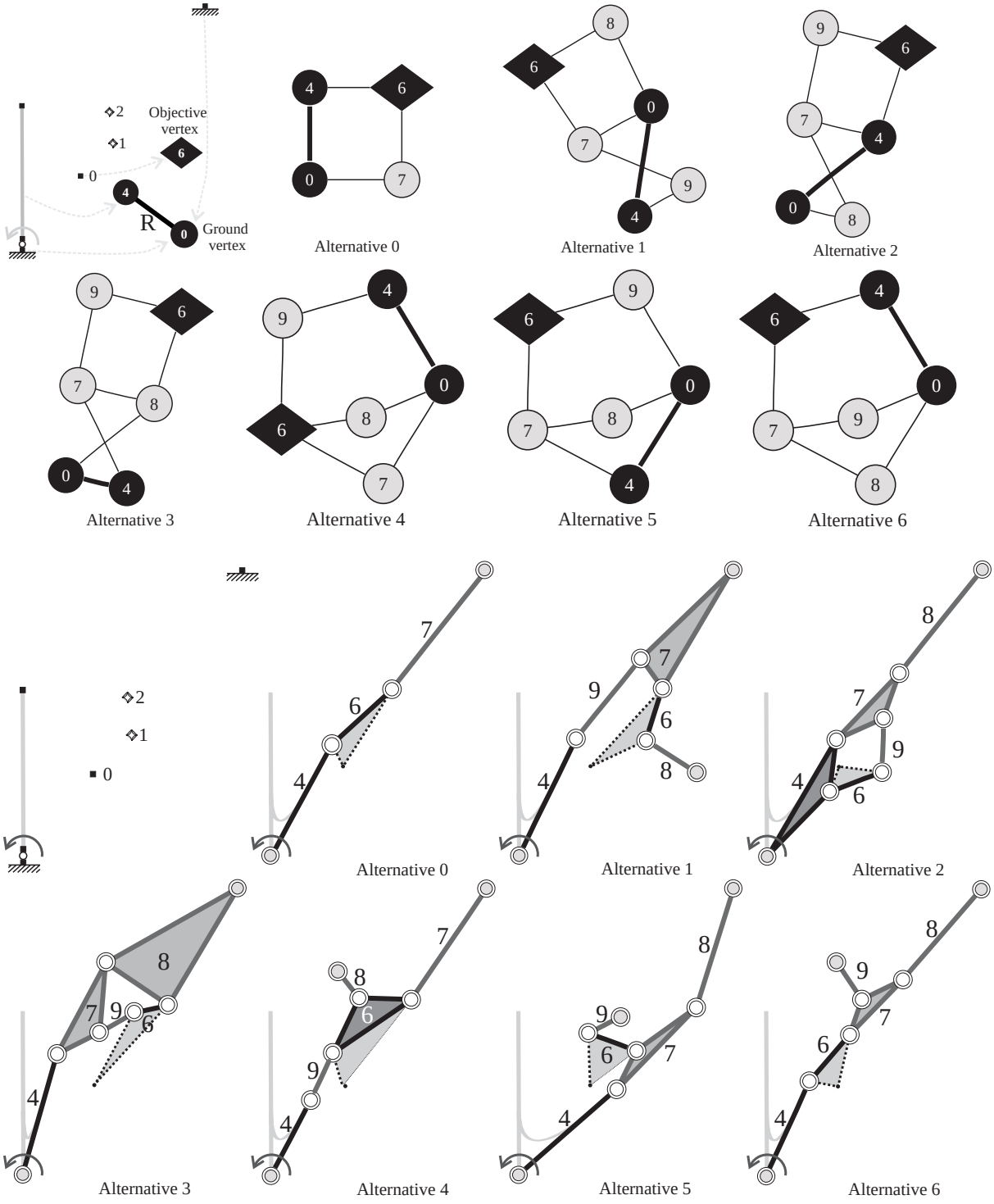



Figure 3: Type Synthesis outputs for the path following problem. The graphs are shown above and the corresponding sketches below.

In Figure 3 we can see more type synthesis results for this example problem where alternatives $\mathcal{M}_0$ to $\mathcal{M}_6$ are the simplest non-isomorphic mechanisms.

## 3 THE PROPOSED METHOD

The decomposition method consist in the following steps:

S1) **Topology decomposition:** The kinematic chain (closed-loops chain mechanism) is decomposed into a set of separated closed-loops[4] of minimal length.

S2) **SOCs decomposition:** For each set of closed-loops, each closed-loop is selected in a given order to be decomposed into SOCs, i.e. dyads, triads, quadriads, etc., using the node displacement constraints.

S3) **SOCs evaluation:** After analyzing data (geometry and synthesis data definitions), the SOCs solvability is evaluated in the resultant order.

S4) **Retained ordered SOCs:** The best valuated combination/s of open-chains is/are stored for dimensional synthesis.

### 3.1 Topology decomposition

In the *Type Synthesis* stage we obtain a mechanism structure represented by a *connected graph* $G(V, E)$, where the vertex set $V$ has cardinality $v$, and the edge set $E$ has cardinality $e$. We know from Graph Theory that a planar graph has $\nu = v - e + 1$ *independent closed loops*, and particularly, we can find the basis of minimal length loops or *minimal independent loops*. Using these loops we can efficiently explore all the *significant dimensions* of the links in a systematic way.

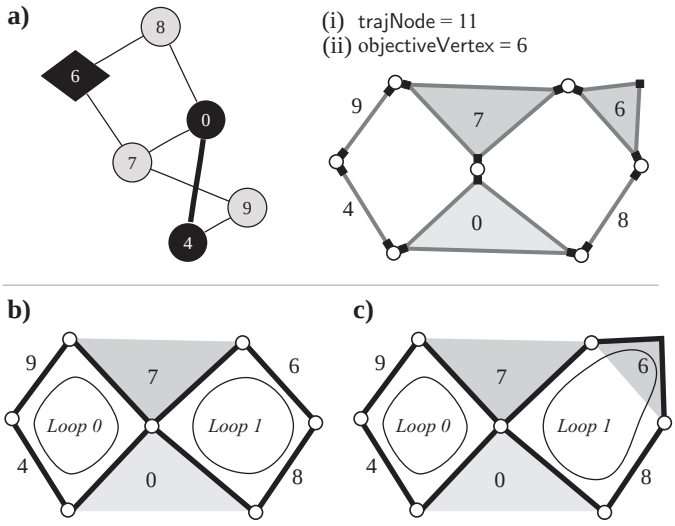



Figure 4: The set of independent loops of minimal length allows to find the significant dimensions of links.

The Watt-II kinematic chain has $\nu = 7 - 6 + 1 = 2$ independent closed loops (Figure 4-a). One line per non-binary link is left without being explored by the independent loops, e.g. links 7 and 0 in Figure 4-b, but their end points are visited by the loops and consequently taken into account. Thus, the loops, traditionally given in terms of edges in Graph Theory visit all edges of the graph and therefore all nodes of the FEM description. In order to consider the link with the trajNode, we extend the loop to pass through this node (see Figure 4-c).

[4]Know as *Cycles* in Graph Theory (Harary, 1969) or *Circuits* (Tsai, 2001), and also called *Kinematic Loops* by Kecskeméthy et al. (1997).

## Closed-loops determination

A *spanning tree* $T$, is a tree containing all the vertices of a connected graph $G$. Therefore, $T$ is a subgraph of $G$. In general, the spanning tree of a connected graph is not unique. For a given spanning tree $T$, the edge set $E$ of $G$ can be decomposed into two disjoint subsets, called the arcs and chords. The arcs of $G$ consist of all the elements of $E$ that form the spanning tree $T$, whereas the chords consist of all the elements of $E$ that are not in $T$. The union of the arcs and chords constitutes the edge set $E$. The addition of a chord to a spanning tree forms one and only one circuit.

The collection of all the circuits with respect to a spanning tree forms a set of independent loops or fundamental circuits. The fundamental circuits constitute a basis for the circuit space. Any arbitrary circuit of the graph can be expressed as a linear combination of the fundamental circuits using the operation of *modulo* 2, i.e., $1 + 1 = 0$ (Harary, 1969; Tsai, 2001).

Based on the previous definitions, a possible set of $\nu$ independent loops can be computed by the following algorithm:

S1 Take one spanning tree $T$ of $G$.

S2 Compute the complement of $T$ ($C_T = G - T$). The graph $C_T$ is composed by as many components as independent loops the graph $G$ has. Also, since it is the complement of a tree, each component is an isolated edge (also called chord) connecting two vertices.

S3 Make a copy of $T$, i.e., $T_{\text{Aux}} := T$. Take an edge of the complement $C_T$ and add it to the spanning tree $T_{\text{Aux}}$. Then, delete this edge from $C_T$. This results in a loop with branches and leaves.

S4 Prune recursively all leaves of $T_{\text{Aux}}$. This leads to a single loop. Save the loop in other data structure.

S5 Repeat steps S3 and S4 $\nu$ times (that is to say, until all edges in $C_T$ disappear).

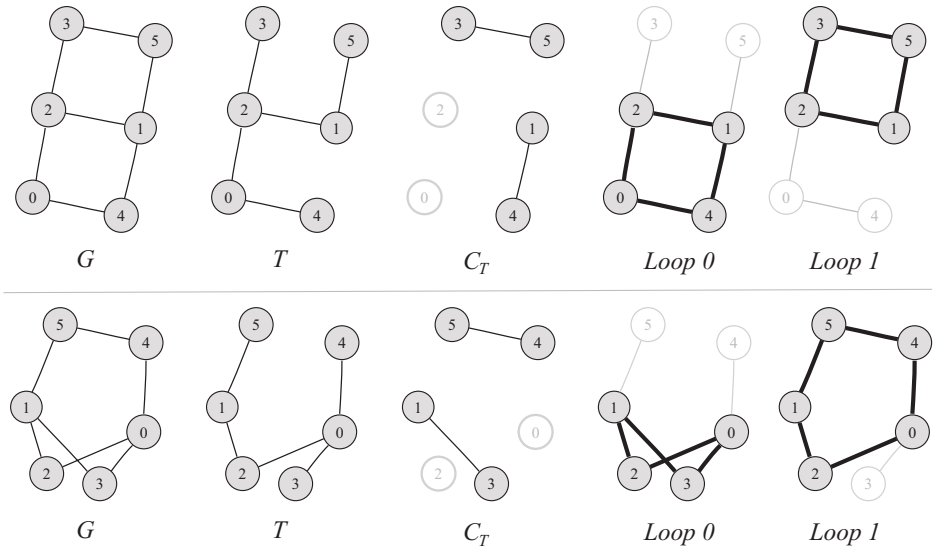



Figure 5: Stages of the algorithm to find a set of independent loops in the graphs of a Watt-I topology (above) and for a Stephenson one (below).

In Figure 5, we illustrated the stages produced when we apply the algorithm.

This algorithm does not necessarily lead to loops with minimal lengths, i.e. with minimal number of edges per loop. Therefore, we post-process the obtained loops making all possible sums of *modulo* 2 between them, and saving those with minimal lengths to form the minimal loop basis.

### 3.2 Dimensional synthesis review

Some introductory topics about dimensional synthesis are needed to justify some rules implemented in the SOCs decomposition procedures to be explained in the next subsection.

The data necessary to solve dimensionally an open-chain by means of analytical methods using the complex-number approach are:

- **Number of links** $n_L$ in the open chain. The SOCs were traditionally called *dyad* ($n_L = 2$), *triad* ($n_L = 3$), *quadriad* ($n_L = 4$), etc. We numbered the links in zero base as $l = 0, \ldots, n_L - 1$.
- **Number of precision points** $n_{pp}$ defined in the task. The notion of finite *precise position* for a SOC is used to describe the states of the chain members. The configuration of the SOC at time $t$ is relative to time 0, and $t = 0, \ldots, n_{pp} - 1$.
- **End-points displacements**. Two sets of displacements on the *tail* $\boldsymbol{h}^t$ of the SOC and on the *tip* or effector point $\boldsymbol{g}^t$ must be defined.
- **End-points positions**. The position of at least one of the end-point nodes, $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$, must be known.
- **Type of joints preceding each link**. In the complex-number approach, joints are the tails of each complex-number representing each link.
- **Prescribed motion constrains**. Rotations and translations may be applied either on joints or links. Some of them could be prescribed by the user or by a previously computed SOC which shares the link.

When the chains move to the $t$-th precise position, the obtained configuration is characterized by the nature of each joint, that is, revolute joints permit link rotations $\alpha_l^t$ (see for example, the first link of Figure 6-a where $\boldsymbol{L}_0^t = \boldsymbol{L}_0 e^{i\alpha_0^t}$), and prismatic joints permit stretch factors $\rho_l^t$ through joint direction of the subsequent link but preserving a fixed angle with the previous link/complex-number, e.g. the second link of Figure 6-b where $\boldsymbol{L}_1^t = \rho_1^t \boldsymbol{L}_1 e^{i\alpha_0^t}$. Using this notation we can write the *Loop-Closure Equations* to solve, for example, the RR-dyad

$$\underbrace{\boldsymbol{L}_0 + \boldsymbol{L}_1}_{\text{initial}} + \underbrace{\boldsymbol{g}^t - \boldsymbol{L}_0 e^{i\alpha_0^t} - \boldsymbol{L}_1 e^{i\alpha_1^t} - \boldsymbol{h}^t}_{t-\text{th position}} = \boldsymbol{O}. \tag{1}$$

Calling $\boldsymbol{\delta}^t = \boldsymbol{h}^t - \boldsymbol{g}^t$, it can be rearranged as

$$\boldsymbol{L}_0(e^{i\alpha_0^t} - 1) + \boldsymbol{L}_1(e^{i\alpha_1^t} - 1) = \boldsymbol{\delta}^t. \tag{2}$$

Expression (2) is known as the *Standard-Form Equation* for a Dyad. When $n_{pp} = n_L + 1$ this complex-number system is linear and can be easily solved. For instance, if three positions ($t = 0, \ldots, 2$) are prescribed for a dyad, the resultant system is written as,

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_0 \\ \boldsymbol{L}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \end{bmatrix} \tag{3}$$

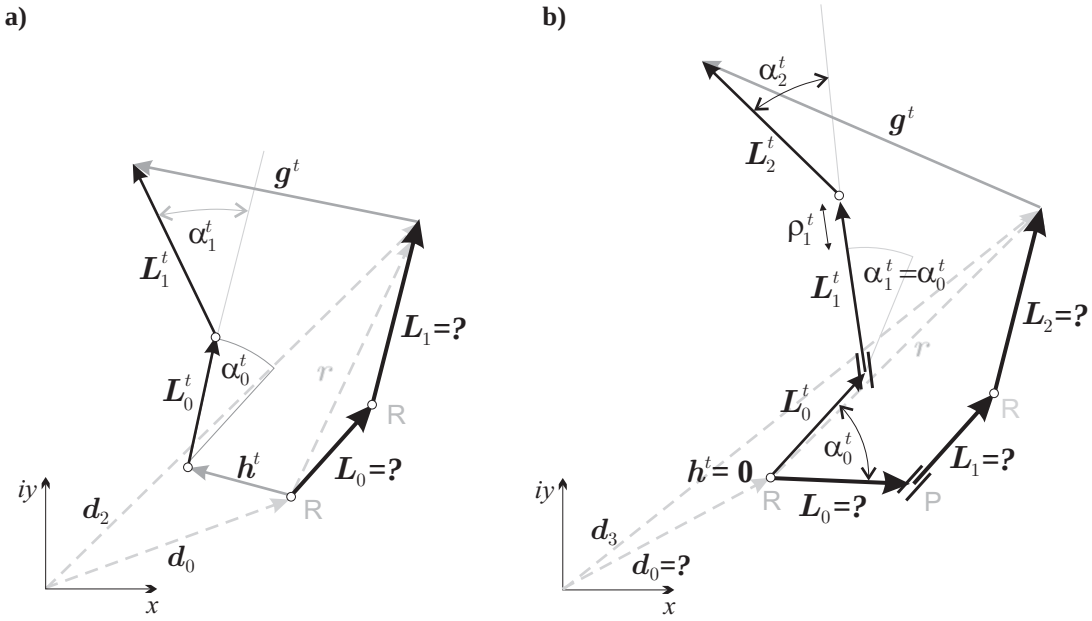



Figure 6: Examples of data for a RR-Dyad and a RPR-Triad modeled by complex-numbers. The initial and $t$-th precise position are shown.

or in a more compact way as

$$\mathbb{CL} = \mathbb{D}. \tag{4}$$

Note that $\boldsymbol{h}^t$ and $\boldsymbol{g}^t$ can be given independently of the two end-point positions $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$, but at least one of them, $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$, must be known to locate the synthesized $n_L$ links. If the two end-point positions are known, an additional equation is considered for the initial situation augmenting the system (2) with the equation:

$$\boldsymbol{L}_0 + \boldsymbol{L}_1 = \boldsymbol{r}, \tag{5}$$

where $\boldsymbol{r} = \boldsymbol{d}_{n_L} - \boldsymbol{d}_0$ is the complex-number which closes the SOC at the starting position, and it is often known as *offset*. Then, we have a system modified as

$$\begin{cases} \boldsymbol{L}_0(e^{i\alpha_0^t} - 1) + \boldsymbol{L}_1(e^{i\alpha_1^t} - 1) = \boldsymbol{\delta}^t \\ \boldsymbol{L}_0 + \boldsymbol{L}_1 = \boldsymbol{r}, \end{cases} \tag{6}$$

it also can be written as

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_0 \\ \boldsymbol{L}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \boldsymbol{r} \end{bmatrix}, \tag{7}$$

or briefly as

$$\mathbb{C}^{\text{off}}\mathbb{L} = \mathbb{D}^{\text{off}}. \tag{8}$$

The system (8) has $\text{rank}(\mathbb{C}^{\text{off}}) = n_L$, this means that two rows or columns of $\mathbb{C}^{\text{off}}$ are linearly dependent. Thus, from the condition $\det(\mathbb{C}^{\text{off}}) = 0$, the coupled motion constraints can be found by means of geometrical constructions called *Compatibility Linkages*. Note that the trivial solution is found by proposing $\alpha_1^1 = \alpha_0^1$ and $\alpha_1^2 = \alpha_0^2$ in $\mathbb{C}^{\text{off}}$, while the non-trivial

one is deduced by augmenting the matrix $\mathbb{C}^{\text{off}}$ with the column of the independent term $\mathbb{D}^{\text{off}}$ and computing

$$\det([\mathbb{C}^{\text{off}}|\mathbb{D}^{\text{off}}]) = 0 \tag{9}$$

to find the geometrical relationships between the parameters (Erdman and Sandor, 1997). Given $\alpha_1^1$ and $\alpha_1^2$, two pair of sets of $\alpha_0^1$ and $\alpha_0^2$ can be found to fulfil (9). Therefore, the problem (7) is known as synthesis with imposed offset when the two end-point positions $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$, and the sets of end-point displacements $\boldsymbol{h}^t$ and $\boldsymbol{g}^t$ of the SOC are known. Also, if one of the sets $\boldsymbol{h}^t$ or $\boldsymbol{g}^t$ are null, it is said that the pivot is imposed[5].

The necessary condition to dimensionally solve a SOC, either in the form of Equations (2) or (6), is that there must be at least one link with completely unknown rotations. Under this condition, the resultant loop closure equations can be solved by using the structure of *Compatibility Linkages* proposed by Sandor (1959), Hartenberg and Denavit (1980), Erdman and Sandor (1997), and Lin et al. (1996). They gave methods to linearize the non-linear systems when the number of prescribed positions is higher than the number of equations needed to have a linear solution ($n_{pp} > n_L + 1$). These cases are dyads in 4 to 5 positions, triads in 5 to 7 positions, quadriads and 5 to 9 positions. When $n_{pp} > n_L + 1$, some parameters must be proposed by the user, and the remaining ones are coupled and computed by means of geometrical constructions, often resulting in sets of parameters with multiplicity of solutions. The proposed parameters are the so-called *free choices*. The exploration of their ranges (for example, $[0, 2\pi]$ for a rotational parameter) gives infinities of open-chains from which the user or the computer program may find the optimal for some criteria (we have experimented with genetic algorithms to find optimal free parameters in Pucheta and Cardona (2005a)).

If one of the two end-point positions $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$ is unknown, but the sets of end-point displacements $\boldsymbol{h}^t$ and $\boldsymbol{g}^t$ of the SOC are known, all links may have imposed rotations, but this does not assure the existence of a solution. It depends on the determinant of the system $\det(\mathbb{C})$. Then, the unknown position $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$ may be computed later. For example, in Figure 6-b a free-pivot problem is shown for a RPR-triad. Here, $\boldsymbol{g}^t = 0$ and $\boldsymbol{d}_0$ is unknown. After computing the links, the pivot position can be found by

$$\boldsymbol{d}_0 = \boldsymbol{d}_{n_L} - \sum_{l=0}^{n_L-1} \boldsymbol{L}_l.$$

In Figure 7 we can see the available modules to solve the SOCs. Each SOC has a directed graph representation or *path* because it is oriented to the complex-number method. Using the FEM description, the path begins with a node and ends in other node and between two links there is always a pair of nodes constrained by a joint. So we have particular subcases. For instance, the subcases for a dyad could be LJL or JLJL (JLJLJ will never occur for the proposed decomposition), where J means joint and L denotes link. The second subcase also solves the symmetric one, i.e, the LJLJ.

### 3.3 SOCs decomposition

Once the loops are computed and stored, we will decompose them into SOCs based on some necessary conditions for dimensional synthesis.

The method uses the FEM description for each loop and the initial motion constraints coming from the topology. Given a loop, we go through the nodes chained by the loop and divide it

[5]Note that, both sets, $\boldsymbol{h}^t$ and $\boldsymbol{g}^t$, can be simultaneously null for a triad, quadriad, and so on. But it configures a structure for a dyad.

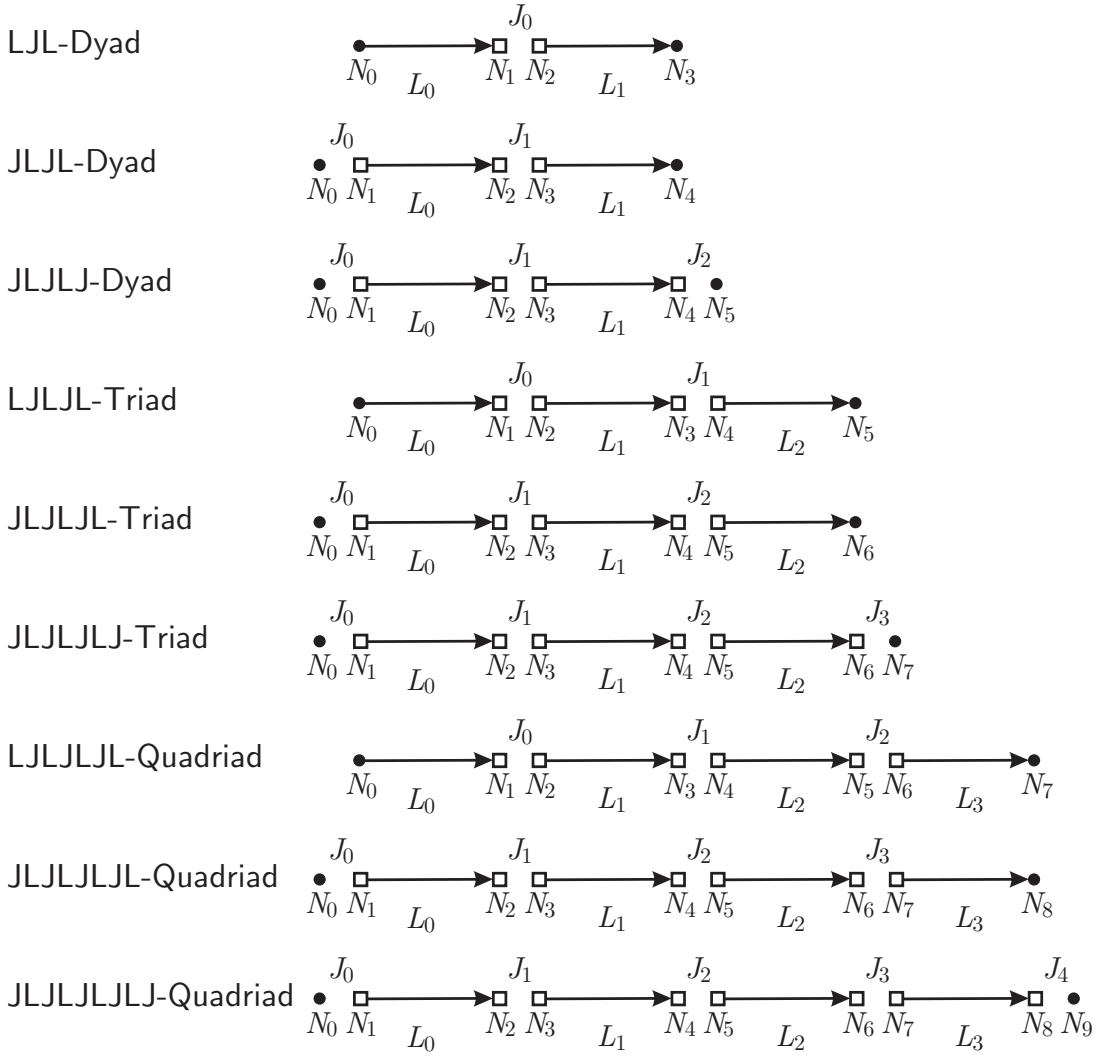



Figure 7: Modules for solving SOCs. For the internal nodes, the □ symbol means that position and displacements are unknowns. The • symbol means that the sets displacements are known for both end-nodes, and positions must be known at least for one of them.

into SOCs *starting from a node with prescribed displacement and ending in another node with prescribed displacement*. These nodes will be the tail and the tip of each SOC.

This decomposition is "loop order dependent", so we analyze the $\nu!$ *Loop-Orderings* in lexicographical order. For a SOC existence, we require that *there must be at least two nodes with prescribed displacements per loop*, and the position of at least one of these end-point nodes must be known, otherwise, the decomposition of the given loop, and also the loop-ordering is abandoned.

Additionally, since the decomposition is also "loop orientation dependent" when there are more than one SOC per loop, the opposite orientation must also be explored. For example, in the presence of the objectiveVertex the number of loop-orderings are multiplied, i.e., we need to explore all possible orderings for each orientation of the loops containing such vertex. In this case, we can predicts that we will obtain at least $\nu + 1$ SOCs.

We decompose an individual closed-loop into SOCs with the aid of a *circular table* which have the information only relative to such loop. To simulate the states of the sets of displace-

ments we use a Boolean variable stdispl for each node. Initially, the fixed nodes have null set of displacements, so they are known. The trajNode is the second source of prescribed displacements. If we suppose that once a SOC is identified it is solved, the initial positions and the sets of displacements for all of their nodes are computed, so they will impose new constraints for decomposing/solving the SOCs in the following loop. Therefore, after one SOC is identified the Boolean variable stdispl is updated as "true" for all the involved nodes.

We will illustrate the algorithm for SOCs decomposition using the path following example, which is shown in Figure 8. Part of the mechanism is prescribed by data (the circled nodes filled in black have known positions and known sets of displacements), and the remaining part was synthesized (open squares nodes).

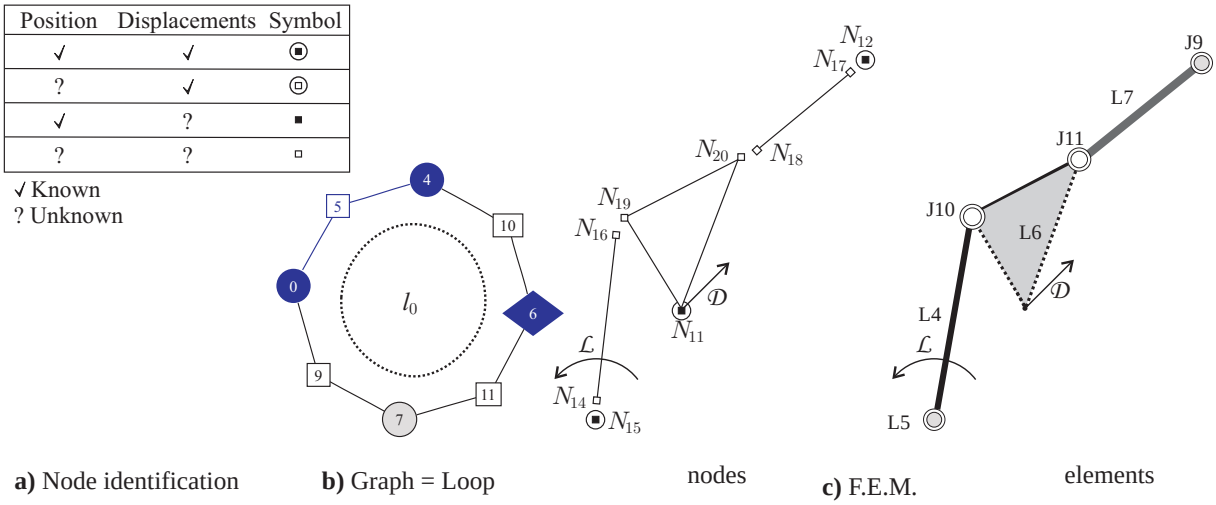



Figure 8: A first feasible graph for a path following problem.

For the unique loop, the auxiliary *circular table* has four rows and as many columns as nodes in the loop ("circular" denotes that the last column is connected to the first). When the loop contains an objective vertex the table has an additional column for the trajectory node.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 12 | 15 | 14 | 16 | 19 | 11 | 20 | 18 | 17 | ⟳ |
| linkID | ⟼ | 0 | 0 | 4 | 4 | 6 | 6 | 6 | 7 | 7 | ⟳ |
| jointID | ⟼ | -9 | -5 | -5 | -10 | -10 | 0 | -11 | -11 | -9 | ⟳ |
| stdispl | ⟼ | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ⟳ |
| SOC 0 | ⟼ | | ▲. | . . . | . . . | . . . | .▼ | | | | ⟳ |
| SOC 1 | ⟼ | .▼ | | | | | ▲. | . . . | . . . | . . . | ⟳ |

Table 1: Example of circular table for open chains identification.

In order to fill the table, a cursor starts pointing to a node of the first vertex, and writes (see Table 1 and Figures 8-b and 8-c) :

1. the node ID in the first row;

2. the ID of the link to which the node belongs in the second row;

3. the ID of the joint to which the node belongs in the third row;

4. a boolean state variable, indicating that at this node has a prescribed displacement.

Two SOCs are identified by analyzing the fourth row content. In the example, we have one SOC going clockwise from node 15 to node 11, and a second SOC continuing clockwise from node 11 to node 12. For the running of the loop with the inverted orientation we obtain the decomposition shown in Table 2.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 15 | 12 | 17 | 18 | 20 | 11 | 19 | 16 | 14 | ⟳ |
| linkID | ⟼ | 0 | 0 | 7 | 7 | 6 | 6 | 6 | 4 | 4 | ⟳ |
| jointID | ⟼ | -5 | -9 | -9 | -11 | -11 | 0 | -10 | -10 | -5 | ⟳ |
| stdispl | ⟼ | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ⟳ |
| SOC 0 | ⟼ | | ▲. | . . . | . . . | . . . | .▼ | | | | ⟳ |
| SOC 1 | ⟼ | .▼ | | | | | ▲. | . . . | . . . | . . . | ⟳ |

Table 2: Circular table for the counter-clockwise orientation.

The decomposition gives us two SOCs with the same type, that is the JLJL-Dyad module. The resultant complex-number models can be seen in the illustration of Figure 9.

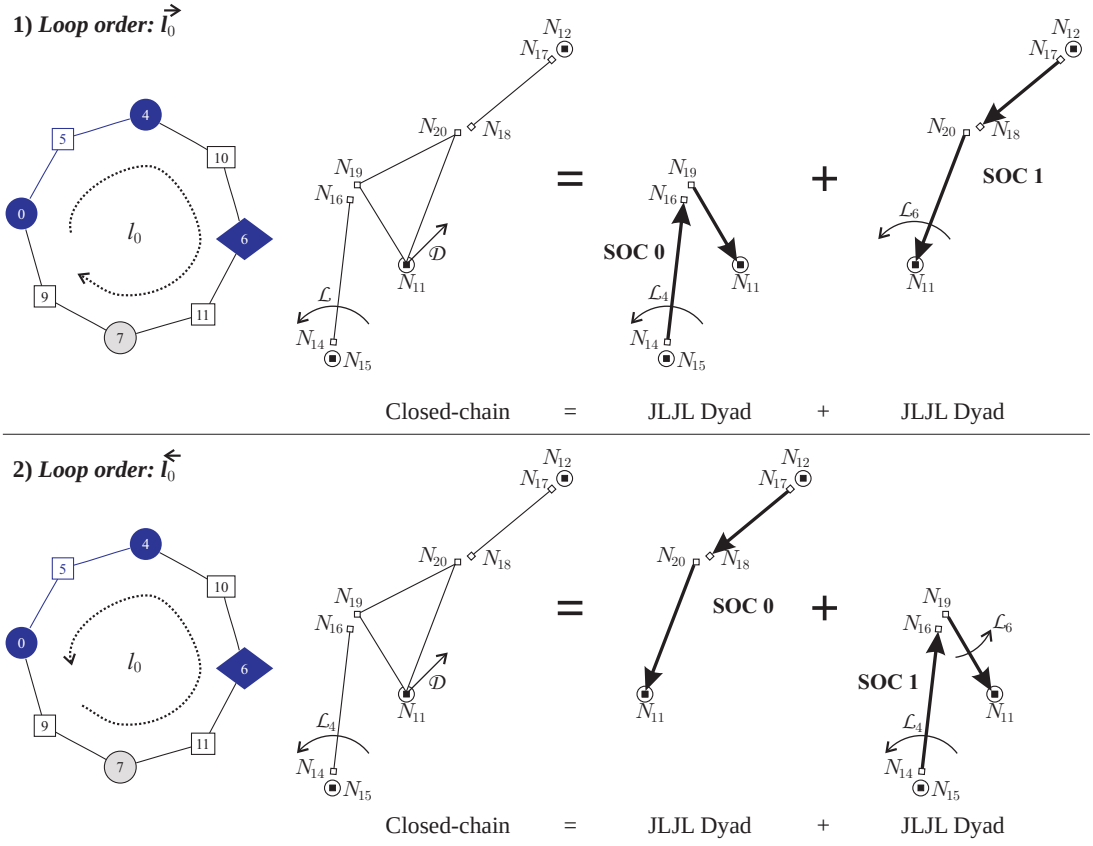



Figure 9: Complex-numbers built for both decompositions.

This is a simple example with a unique loop but in Section 4 we will show results for multi-loop mechanisms.

### 3.4 SOCs evaluation

In the previous example, we did not appreciate any direct advantage from the change of orientation in the loop decomposition. For both decompositions, we obtained two SOCs of JLJL-Dyad type. However, the SOCs ordering will be very important in the dimensional synthesis stage. Following the same example, to solve the **SOC 0** of Figure 9 (above), we have the first link (L4) with its motion completely defined, that is the prescribed timing transmitted by the input joint (J5). Then, we need to propose two free-parameters $\alpha_1^1, \alpha_1^2$ to solve the SOC. Then, the sets of rotations of the second link (L6) are known. To solve the **SOC 1**, the set of rotations of link 6 are imposed by the results of **SOC 0**; proposing two free choices $\alpha_0^1, \alpha_0^2$ the problem is solved. For the inverted orientation decomposition, Figure 9 (below), we must define four free choices to solve the **SOC 0**, and then, we have the **SOC 1** with completely defined movements on both links (L4 and L6). As the offset is imposed, only if $\det([\mathbb{C}^{\text{off}}|\mathbb{D}^{\text{off}}]) = 0$ it will have a solution, which is a very odd case. Thus, the first decomposition is preferred for design.

The second decomposition violates the important design condition **E1**: "when the system becomes non-linear or has rank defect, at least one link must have completely unknown rotations". In the example, the imposed offset produces rank defect.

Additionally, to choose a SOC decomposition **E2**: "we will give preference to that decomposition in which the first SOC solves more initial motion constraints". For example, in the first decomposition, the **SOC 0** solves three motion constraints prescribed by the user (timing) and the **SOC 0** of the second decomposition none. So, the first one is also preferred for this reason.

More than one ordering could be well-posed in terms of these two evaluation criteria, **E1** and **E2**, so we retain all of them to pass through dimensional synthesis stage to give a valid judgement. Note that we cannot predict if a SOC has solution until the free paramerters are proposed for their whole ranges.

The evaluation takes place in the order in which the SOCs where decomposed and stored. This will be the *computing* and thereby the *assembling order*. For this given order the SOCs are evaluated simulating, either the initial constraints (node and link imposed movements) as well as the data transference between link rotations, and node positions and displacements.

For each SOC, we count the *number of links with completely undefined constraints* $n_U(\mathbf{SOC})$. As we said, it must be

$$n_U(\mathbf{SOC}) \begin{cases} \geq 1 & \text{if } n_{pp} \geq n_L + 1 \text{ and, } d_0 \text{ and } d_{n_L} \text{ are known (imposed offset),} \\ \geq 1 & \text{if } n_{pp} > n_L + 1 \text{ and there is one free end-point position, } d_0 \text{ or } d_{n_L} \text{ is unknown,} \\ \geq 0 & \text{if } n_{pp} = n_L + 1 \text{ and, } d_0 \text{ or } d_{n_L} \text{ is unknown.} \end{cases} \tag{10}$$

Also we count the number of constraints $n_C(\mathbf{SOC})$ solved by each SOC.

Finally, an index for ranking each decomposition is given in terms of the best satisfaction of two rules:

**R1** Number of SOCs satisfying the constraint (10). It would be equal to the number of SOCs.

**R2** Number of constraints solved by the first SOC, the second, the third, and so on, are compared between SOCs in assembling order. That is, the lists

$$\{n_C(\mathbf{SOC\ 0}), n_C(\mathbf{SOC\ 1}), \ldots, n_C(\mathbf{SOC}\ \boldsymbol{\nu}), \ldots\}$$

for each SOC.

For example, the evaluation for the decompositions shown in Figure 9 gives:

1. **R1** $= \{\text{true}, \text{true}\} = 2$, **R2** $= \{3, 3\}$;
2. **R1** $= \{\text{false}, \text{false}\} = 0$, **R2** $= \{0, 6\}$.

So that the first one is considered at the first position in the ranking.

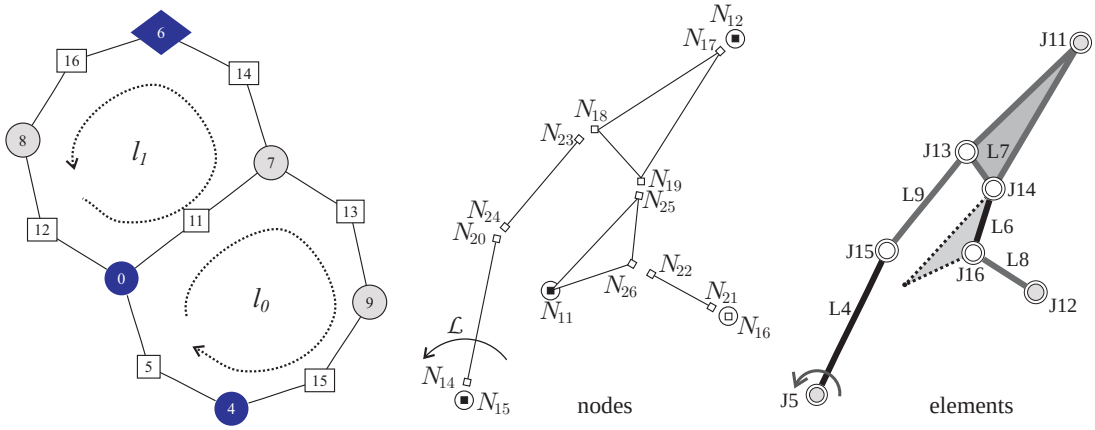
## 4 RESULTS

Several kinematics problems were tested and used as feedback to develop the method.

### 4.1 A multi-loop curve path generator

We will continue with the path following example of Figure 3 whose minimal independent loops were shown in Figure 4-c.

The **Alternative 1** of the type synthesis output is a two-loop chain (see Figure 10), where a new pivot was synthesized. Also, since there is an objective vertex, the number of possible decompositions is duplicated. There are $2\nu!$ loop orderings decomposed into $\nu + 1$ SOCs each one.




**a)** Graph and minimal independent loops

**b)** F.E.M. description

Figure 10: Initial situation for the decomposition of the second alternative of the path following problem.

First we identify those loops where the objective vertex is located. Then, the possible orderings are: $l_0 - \overrightarrow{l_1}$, $\overrightarrow{l_1} - l_0$, and the inverted cases $l_0 - \overleftarrow{l_1}$, and $\overleftarrow{l_1} - l_0$. The resultant decompositions are shown in Figure 11.

The obtained evaluations are:

1. **R1** $= \{\text{true}, \text{true}, \text{true}\} = 3$, **R2** $= \{3, 3, 3\}$;
2. **R1** $= \{\text{false}, \text{true}, \text{true}\} = 2$, **R2** $= \{0, 3, 6\}$;
3. **R1** $= \{\text{true}, \text{true}, \text{false}\} = 2$, **R2** $= \{3, 0, 6\}$;
4. **R1** $= \{\text{false}, \text{true}, \text{true}\} = 2$, **R2** $= \{0, 3, 6\}$;

The final ranking is 1, 3, 2, 4.

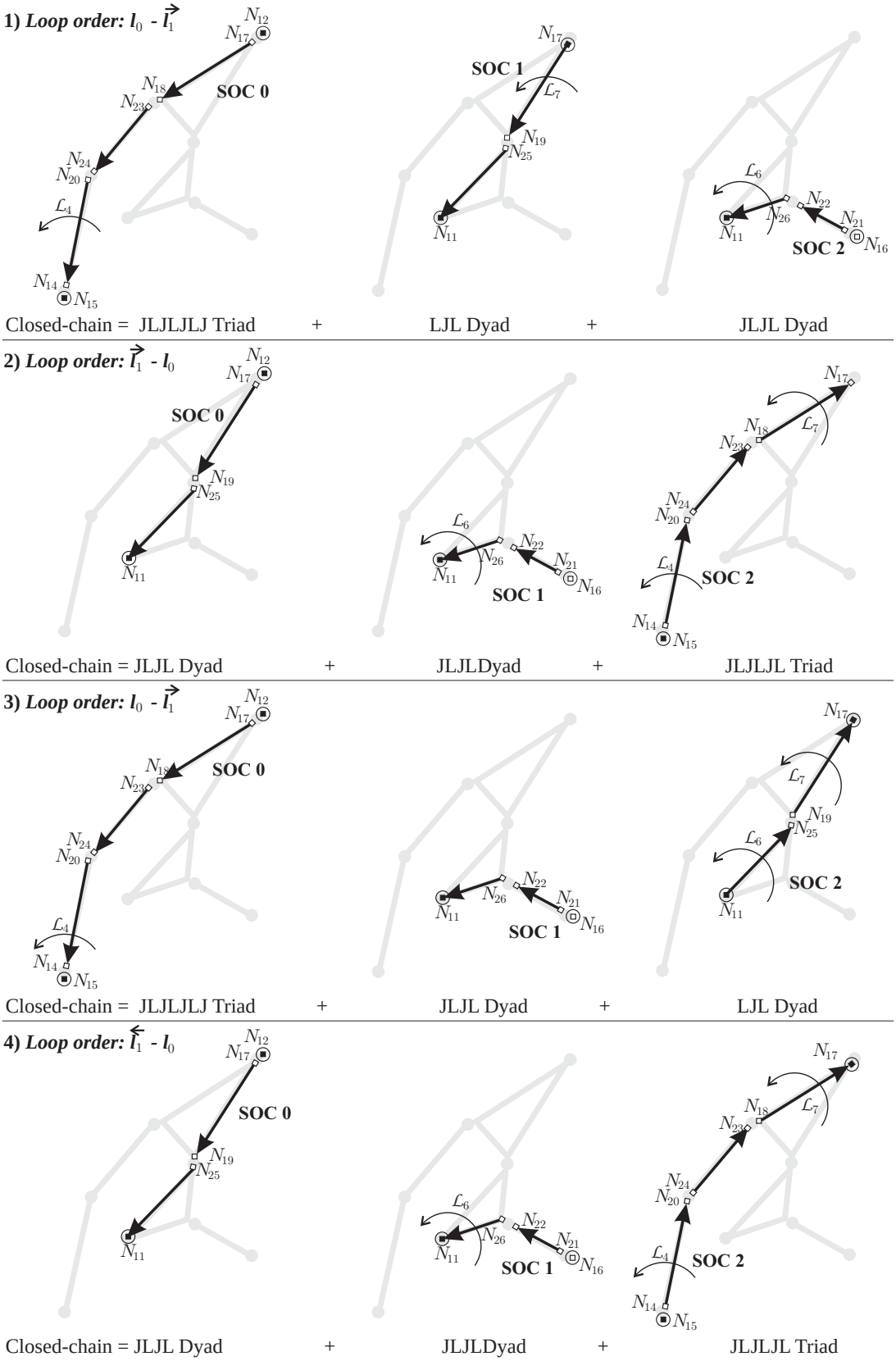



Figure 11: All decompositions for the second alternative of the path following problem.

## 4.2 Nozzle of a turbine engine

The problem shown in Figure 12 schematizes the prescribed coordination between two flaps of a turbine engine and the horizontal movement of an hydraulic cylinder. The physical meaning of the initial graph vertices is: 8 (primary flap), 10 (secondary flap), and 12 (hydraulic cylinder). Four positions are given for each flap while only the starting and ending positions are prescribed for the cylinder (2 positions).

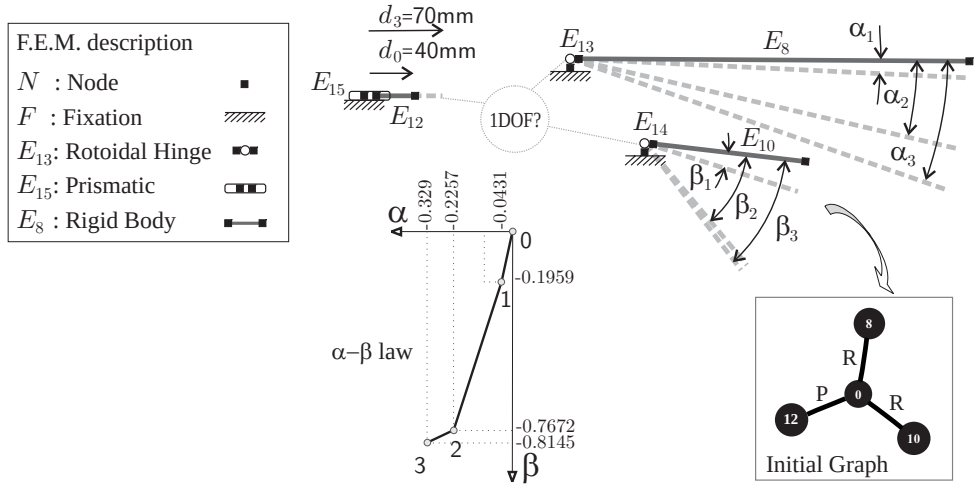



Figure 12: Graph representation for a kinematics problem of double function generation.

The first six solutions available from the type synthesis stage are shown in Figure 13.

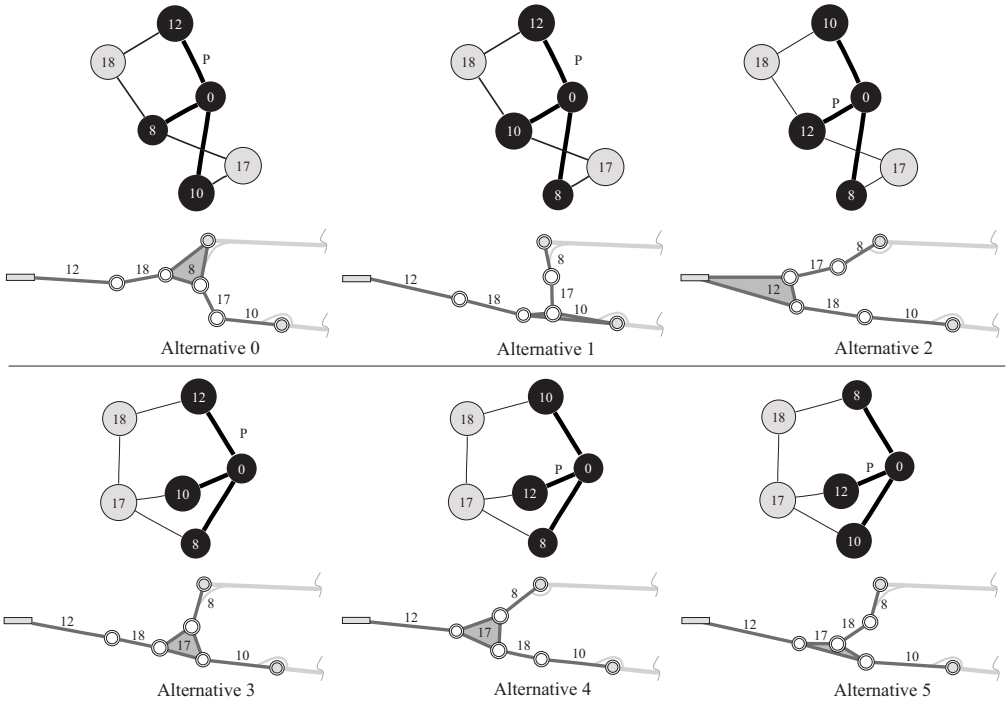
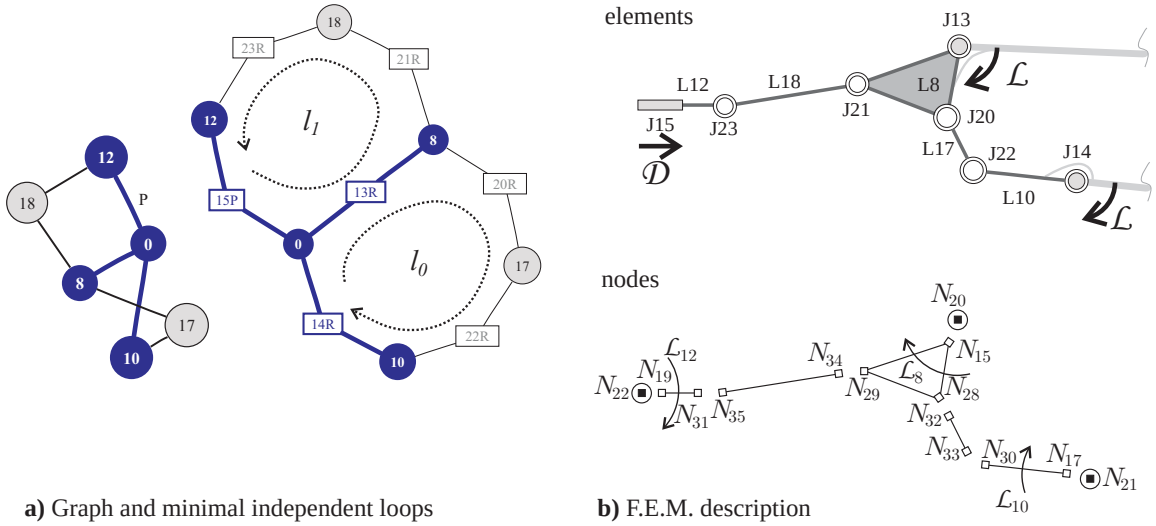



Figure 13: Type Synthesis outputs. For clarity, an automatic sketch is drawn below each solution.

The synthesized nodes and elements for the **Alternative 0** are shown in Figure 14.




**a)** Graph and minimal independent loops **b)** F.E.M. description

Figure 14: Initial situation for the decomposition of the simplest alternative for double function generation.

The decomposition process results in two SOCs for each loop-ordering. Now, we will describe the circular tables obtained for the first loop-ordering $l_0 - l_1$.

The circular Table 3 for SOC identification in loop $l_0$ is filled as follows.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 21 | 22 | 20 | 15 | 28 | 32 | 33 | 30 | 17 | ↻ |
| linkID | ⟼ | 0 | 0 | 0 | 8 | 8 | 17 | 17 | 10 | 10 | ↻ |
| jointID | ⟼ | -14 | 0 | -13 | -13 | -20 | -20 | -22 | -22 | -14 | ↻ |
| stdispl | ⟼ | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ↻ |
| SOC 0 | ⟼ | .▼ | | ▲. | ... | ... | ... | ... | ... | ... | ↻ |

Table 3: Circular table for loop $l_0$ in the nozzle problem.

Since in this example there is not a trajectory node breaking the loop, an unique SOC is obtained and the running of the loop with the inverted orientation is not necessary.

The next loop that we obtained, $l_1$, is filled as it is shown in Table 4.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 22 | 21 | 20 | 15 | 29 | 34 | 35 | 31 | 19 | ↻ |
| linkID | ⟼ | 0 | 0 | 0 | 8 | 8 | 18 | 18 | 12 | 12 | ↻ |
| jointID | ⟼ | -15 | 0 | -13 | -13 | -21 | -21 | -23 | -23 | -15 | ↻ |
| stdispl | ⟼ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | ↻ |
| SOC 1 | ⟼ | .▼ | | | ▲. | ... | ... | ... | ... | ... | ↻ |

Table 4: Circular table for loop $l_1$.

The corresponding decomposition is illustrated in Figure 15-1. The orientations of the complex-numbers in the **SOC 1** are inverted to make them compatible with the available solver

module JLJLJL-Triad. The program automatically inverts the orientations of the complex-number chain whenever it finds that the SOC is started by a link. The decomposition for the loop-ordering $l_1 - l_0$ is treated in the same form, and it is shown in Figure 15-2.

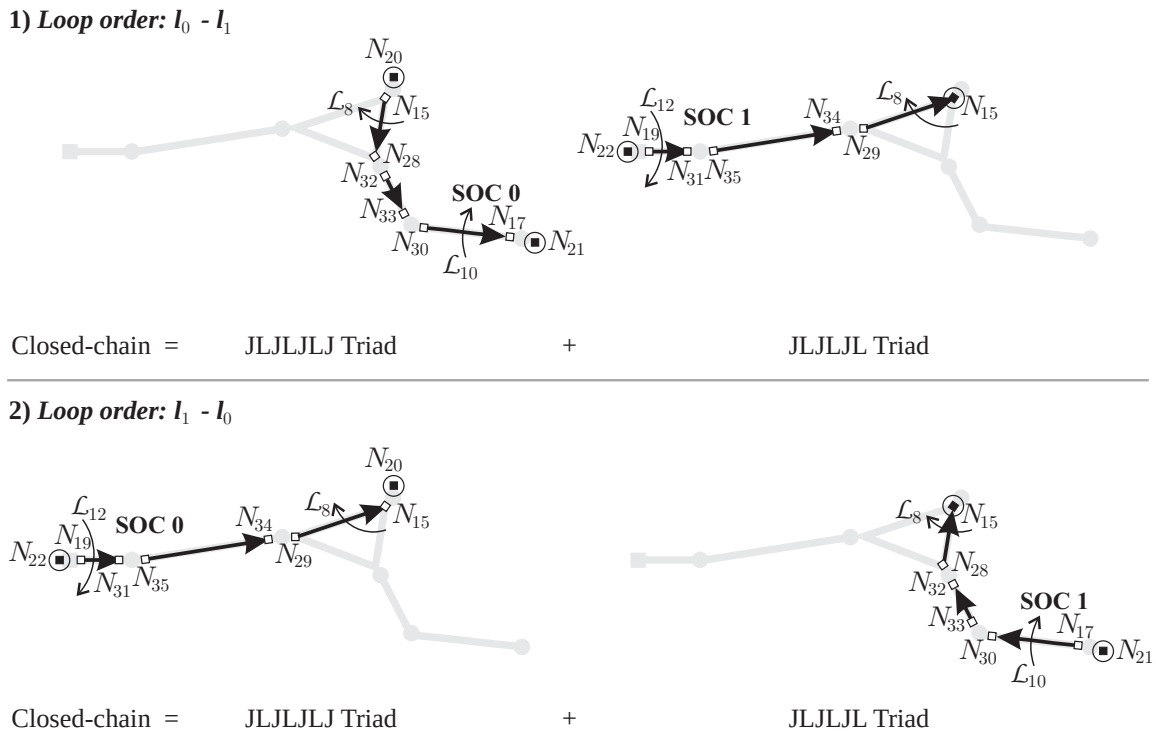



Figure 15: All decompositions for the double function generation problem.

The obtained evaluations for both decompositions were:

1. **R1** $= \{\text{true}, \text{true}\} = 2$, **R2** $= \{8, 6\}$;
2. **R1** $= \{\text{true}, \text{true}\} = 2$, **R2** $= \{6, 8\}$;

This means that the first one will be chosen for dimensional synthesis (Figure 16).

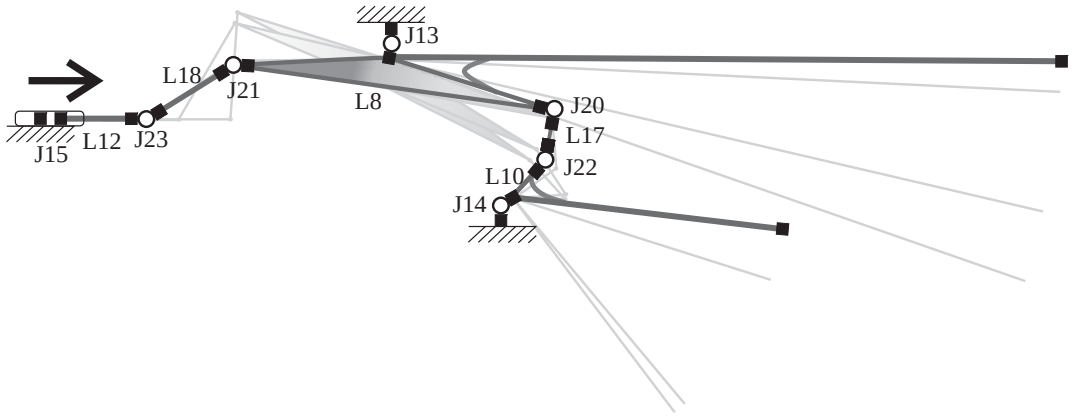



Figure 16: Dimensional synthesis of the nozzle problem using the first decomposition.

## 5 CONCLUSIONS

We proposed an automated method to generate, evaluate and rank all SOCs decompositions from the sets of minimal independent loops of the graph representation of a mechanism. The decomposition can be used for the synthesis of any kinematics problem based on the Precision Point Method. The rules for evaluation are heuristics, and the examples showed that they do not reject any feasible solution and offer an optimal sequence of SOCs ranked in the first place.

The presented algorithm for computing the set of *minimal independent loops* works well with graphs without attributes on their vertices and edges (link and joint types respectively). However, when we consider attributes, loops with equal lengths could be different. Therefore, we may obtain more than one set of minimal independent loops. Using an isomorphism identifier for coloured graphs we would obtain *all non-isomorphic sets* of minimal-length independent loops.

The rules for decomposition based on FEM description, i.e. the positions and displacements of nodes, and the successive transfert of data between SOCs, can be extended to tridimensional problems.

We think that it is not possible to build a data base of decompositions for all kinematics problems. However, the presented method can be useful to construct a data base for the most popular kinematics tasks, like PF, FG and RBG, for mechanisms up to eight links.

The next direction of research will be the study and development of this method for planar compliant mechanism applications using the pseudo-rigid body concept (Howell, 2001).

## 6 ACKNOWLEDGMENTS

This work received finantial support from *Consejo Nacional de Investigaciones Científicas y Técnicas*, *Agencia Nacional de Promoción Científica y Tecnológica*, *Universidad Nacional del Litoral* and from the *European Community* through grant *SYNCOMECS* (SYNthesis of COmpliant MEChanical Systems) project UE FP6-2003-AERO-1-516183.

The first author wants to thank for the very useful publications sent by the professor Ting-Li Yang from Nanjing, P.R. China.

## REFERENCES

S.S. Balli and S. Chand. Defects link mechanisms and solution rectification. *Mechanism and Machine Theory*, 37(9):851–876, 2002a.

S.S. Balli and S. Chand. Transmission angle in mechanisms (triangle in mech). *Mechanism and Machine Theory*, 37(2):175–195, 2002b.

J.B. Cook. *SyMech Synthesis Software for Pro/E*. URL *http://www.symech.com*.

H.A. Crone, A.J. Klein Breteler, and K. van der Werff. *TADSOL Type And Dimension Synthesis Of Linkages*. URL *http://www.ocp.tudelft.nl/tt/cadom*. Delft University of Technology, Netherlands.

H. Draijer and F. Kokkeler. *WATT Mechanism Synthesis*. URL *http://www.heron-technologies.com/watt*. Heron Technologies.

A.G. Erdman and J. Gustafson. LINCAGES: A Linkage Interactive Computer Analysis and Graphically Enhanced Synthesis Package. *ASME Paper No. 77-DTC-5*, 1977.

A.G. Erdman and G.N. Sandor. *Mechanism Design: Analysis and Synthesis*, volume 1. Prentice-Hall, New Jersey, 3rd edition, 1997.

M. Geradin and A. Cardona. *Flexible Multi-Body Dynamics. A Finite Element Approach*. John Wiley & Sons, 2001.

F. Harary. *Graph Theory*. Addison-Wesley Series in mathematics, 1969.

R.S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. McGraw-Hill, New York, 1980.

L.L. Howell. *Compliant Mechanisms*. John Wiley & Sons, New York, 2001.

R.E. Kaufman. KINSYN - An interactive system for the kinematic synthesis of mechanisms. In *Third World Congress on Theory of Machines and Mechanisms*, pages 13–20, Dubrovnik, Yugoslavia, september 1971.

A. Kecskeméthy, T. Krupp, and M. Hiller. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics*, 1(1):23–45, 1997.

C.-S. Lin, A.G. Erdman, and B.-P. Jia. Use of compatibility linkages and solution structures in the dimensional synthesis of mechanism components. *Mechanism and Machine Theory*, 31: 619–635, 1996.

J.M. McCarthy. *Synthetica*. URL *http://synthetica.eng.uci.edu/~mccarthy*. Laboratory for the Analysis and Synthesis of Spatial Movement. University of California, USA.

D.G. Olson, A.G. Erdman, and D.R. Riley. Formulation of dimensional synthesis procedures for complex planar linkages. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 109(3):322–328, 1987.

M.A. Pucheta and A. Cardona. Type synthesis and initial sizing of planar linkages using graph theory and classic genetic algorithms starting from parts prescribed by user. In *Multibody Dynamics 2005, ECCOMAS Thematic Conference*, Madrid, Spain, 2005a.

M.A. Pucheta and A. Cardona. Type synthesis of planar linkage mechanisms with rotoidal and prismatic joints. In *Mecánica Computacional*, volume XXVI of *VII Congreso Argentino de Mecánica Computacional, MECOM 2005*, pages 2703–2730, Buenos Aires, Argentina, 2005b.

A.M. Rankers. *SAM: Synthesis and Analysis of Mechanisms*. URL *http://www.artas.nl*. ARTAS-Engineering Software.

SAMCEF. *SAMCEF Field v5, SAMCEF BOSS/Quatro v5*. URL *http://www.samcef.com*. SAMTECH S.A.

G.N. Sandor. *A General Complex-Number Method for Plane Kinematic Synthesis with Applications*. PhD thesis, Columbia University, New York, 1959.

G.N. Sandor and A.G. Erdman. *Advanced Mechanism Design: Analysis and Synthesis*, volume 2. Prentice-Hall, New Jersey, 1984.

P. Sardain. Linkage synthesis: Topology selection fixed by dimensional constraints, study of an example. *Mechanism and Machine Theory*, 32:91–102, 1997.

L.-W. Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function*. CRC Press, Boca Raton, 2001.

K.J. Waldron and S.M. Song. Theoretical and numerical improvements to the interactive linkage design program, RECSYN. In *Seventh Applied Mechanisms Conference*, pages 8.1–8.8.7, Kansas City, Missouri, 1981.

T.-L. Yang, F.-H. Yao, and M. Zhang. A comparative study on some modular approaches for analysis and synthesis of planar linkages: Part 2 – Modular dynamic analysis, modular structural synthesis and modular kinematic synthesis. In *ASME Design Engineering Technical Conferences, DETC/MECH-6058*, Atlanta, Georgia, USA, 1998.

B. Yannou and A. Vasiliu. Design platform for planar mechanisms based on a qualitative kinematics. In *QR'95: Ninth International Workshop on Qualitative Reasoning about Physical Systems*, pages 191–200, Amsterdam, 1995.