Mechanics Based Design of Structures and Machines, 36: 364–391, 2008

ISSN 1539-7734 print/1539-7742 online
DOI: 10.1080/15397730802405929

Synthesis of Planar Multiloop Linkages Starting from Existing Parts or Mechanisms: Enumeration and Initial Sizing[#]

Martín A. Pucheta and Alberto Cardona

Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC),
INTEC (Universidad Nacional del Litoral-CONICET),
Güemes, Santa Fe, Argentina

Abstract: We present a methodology for dealing with mechanism synthesis problems in which the initial parts to move can have the form of either open-, closed-, or mixed-chain mechanisms, isolated points, or bodies to guide. We use a finite element method-like description of the initial kinematics problem, a graph representation to solve the number synthesis, and the precision-point method to solve the initial dimensions of the linkages. A preliminary analysis of the structural parts in conjunction with the imposed motion constraints is considered just after the initial description to distinguish those analyzable parts from the unknowns of the synthesis problem. A nonstandard example for function generation is illustrated throughout the paper.

Keywords: Genetic algorithms; Graph theory; Linkage mechanisms; Precision-point method; Type synthesis.

INTRODUCTION

Linkage mechanisms have the capability of developing linear, nonlinear, circular, and/or complex motions while transferring force and/or torque.

Received May 12, 2008; Accepted July 20, 2008
[#]Communicated by M. Ceccarelli.
Correspondence: Martín A. Pucheta, Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), INTEC (Universidad Nacional del Litoral-CONICET), Güemes 3450, S3000GLN, Santa Fe, Argentina; E-mail: mpucheta@intec.unl.edu.ar

They are extensively used as mechanical devices in a wide variety of fields including industrial, automotive, agricultural, biomechanical, and micro-electromechanical applications, among others.

Mechanism design consists in finding the mechanism for given requirements. Kinematic synthesis consists in finding the mechanism for a given motion. The earliest stage of the mechanism design process is the conceptual design stage, which can be divided into three stages: (1) problem requirements specification, (2) type synthesis, and (3) dimensional synthesis.

The goal of the *type synthesis* stage is to enumerate all mechanisms topologies (up to a certain complexity) that satisfy the structural requirements. At the type synthesis stage a discrete problem is solved, whereas at the *dimensional synthesis* stage we deal with a continuous problem that consists of computing the significant dimensions and the starting position of the linkages that satisfy the input and output motion constraints and the prescribed performance. The conjoint use of both stages enables the designer to find the optimal—often the simplest—mechanism for a given task (Sardain, 1997).

The first part of the type synthesis stage consists in selecting and combining various mechanism types: gears, cams, pulley and belts, linkages, and so on (Chen et al., 2006; Chiou and Kota, 1999). The scope of this paper is limited to the linkage type. In the field of linkage mechanisms, the *structural* requirements for the desired mechanisms are: nature of motion (planar or spatial), number of degrees of freedom (DOF), the topological description of the parts to move with their connections, types of links (rigid or flexible), types of kinematic pairs (revolute, prismatic, etc.), number of prescribed pivots, etc. The *functional* requirements are the motion constraints (rotations and translations) imposed over the input and output parts, allowed space, obstacles, required performances such as mechanical and geometrical advantages, etc. More complex requirements can be identified and reserved for the evaluation of the alternatives in the detailed design stage.

Starting from the problem requirements, Freudenstein and Maki (1979) proposed a type synthesis strategy based on the "separation" of the *structural requirements* to *generate* or search alternatives, from the *functional requirements* to *evaluate* the generated alternatives. This strategy was successfully used by Tsai (2001) for the enumeration of gear-trains, parallel manipulators, and linkage mechanisms. Chen and Pai (2005) formulated a new classification of the requirements to systematize the type synthesis process and validated Freudenstein and Maki's problem in an exhaustive way. They classified the requirements into structural, functional and design constraints. By also using an exhaustive enumeration, Pucheta and Cardona (2005, 2007a) developed a type synthesis method using the graph theory formalism and

implemented it in a computer program. They used a graph theory formalism to solve the number synthesis problem by means of a subgraph search of the so-called *initial graph* representing the initial parts inside a graph of a previously enumerated atlas of mechanisms used as design space. Some design constraints were tested to reduce the search and an identifier of mechanism isomorphisms was extensively used to detect all the functionally different occurrences of the initial parts inside the atlas. A third exhaustive but more direct approach for the enumeration of topologies for given requirements was proposed by Yan (1998) and was recently improved by Yan and Hung (2006). By using a method based on group theory concepts they completely eliminated not only the need for isomorphisms testing but also for constraints testing.

Concerning the *dimensional synthesis* of linkages, there are several efficient computational tools presented in the bibliography. All of them start the process from a topology selected by the user. In the last ten years, there has been a renewed interest on computer-aided linkage design. There is a jump between the very active mid-1970s when software like KINSYN, LINCAGES, and RECSYN were developed (Erdman and Sandor, 1997), and the beginning of the current decade when we can see more industrialization efforts on user-friendly software developments like WATT, SAM, LINCAGES 2000, SYNTHETICA, and SYMECH, which use modern computer software technologies. Currently, there is a constant emphasis on the communication and integration of old and new *synthesis theories* inside the most important CAD/CAE tools (Kinzel et al., 2006).

There are some *exhaustive* attempts to *integrate type and dimensional synthesis*, such as the proposals of Sardain (1997), Hansen (1993), Hayes and Zsombor-Murray (2004), and Pucheta and Cardona (2005), but none of them is fully automatic. However, there exist *heuristic* approaches to the automated type and dimensional synthesis problem, such as those proposed by Sedlaczek et al. (2005) or Liu and McPhee (2007). In these works, the topologies are generated using a genetic representation, and then each feasible alternative is evaluated by computing a kinematic analysis; the mechanism with best performance among all evaluations—that with the smallest error between the desired and generated trajectories—is retained as the optimal mechanism. These ones are some of the few fully automated design methods that deal the type and dimensional synthesis of linkages in an integrated way, although they have the disadvantage of requiring a very high computational cost and make use of a *cluster* of computers to solve this problem in reasonable time. Additionally, other approaches that use artificial intelligence techniques have been proposed for selecting the optimal mechanism (Campbell et al., 2003; Tian et al., 2006).

We should remark that the goals always pursued at any stage of the design process are to *avoid missing feasible alternatives* and also

to *avoid analyzing repeated solutions*. These objectives can be achieved only by exhaustive approaches, which are complex to develop (Tsai, 2001; Yan, 1998).

As we reviewed above, the planar linkage design process has been thoroughly studied in a large number of publications; however, few of them put emphasis on treating the class of difficulties that the designer may face in practice when defining the kinematic task. Customarily, single tasks of kinematic synthesis are classified into function generation (FG), path following (PF), and rigid-body guidance (RBG); see Fig. 1. However, in practice a combination of these single tasks often appears as a problem requirement.

We propose a *general* statement of a kinematic task, which can be precisely expressed in terms of three steps:

1. Define the existing *parts to move* (mechanisms or submechanisms) (Fig. 2).
2. Impose, on some or all of these parts, the input and output *motion constraints*.
3. Define other functional and design constraints, e.g., allowed space and obstacles.

These steps can be intuitively defined using a graphic interface for mechanism design; for example, we use the *Samcef-Field* CAD for finite elements (Samtech, 2006). The representation of mechanisms through the finite element method (FEM) (Cardona, 1989; Géradin and Cardona, 2001) that implicitly includes the topological representation provided by Graph Theory, proved to be an adequate tool to unify the description of the initial parts, the task specification, the computational implementation of solvers for both stages of synthesis (Pucheta, 2008), and moreover,

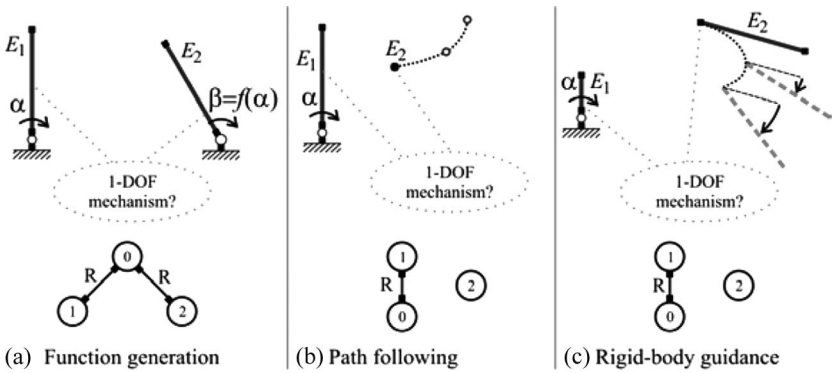

Figure 1. Examples of single kinematic tasks and their *initial graph* representation.

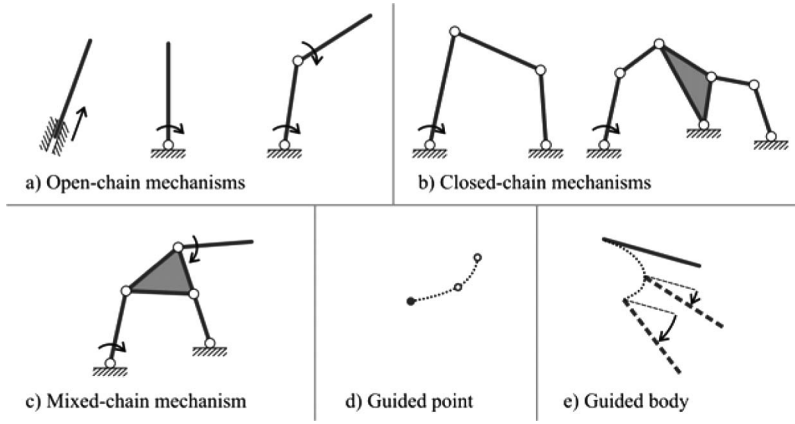

Figure 2. Examples of parts that can be combined, with repetition allowed, to define kinematic tasks.

to facilitate the optimization of the mechanisms (Cugnon et al., 2007). Using some predefined rules (that will be reviewed in this paper), an initial graph describing the synthesis problem can be automatically built from the FEM description of steps 1 and 2. This graph is then combined with the data provided in step 3, for solving the type and dimensional synthesis.

We consider some hypotheses with influence on the task: (1) we will work with rigid bodies so that all points of a body share the same rotation and also preserve distance between them; (2) we assume that the parts to move defined by the user have *mobility* and therefore the inclusion of rigid subchains is avoided; (3) the motion constraints are given in a discrete form as precision positions.

In this paper, we particularly consider an initial situation that very often appears as a problem requirement in the task: the existence of previously known significant dimensions of the existing parts. Some of these kinds of tasks were not addressed in previous works and probably cannot be solved by using the methods and available software mentioned above, so this motivates the present study. In this paper, we present a type and initial sizing solver able to deal with these tasks automatically. The focus of attention will be the procedure for automatically analyzing the existing parts to compute displacements and rotations of those nodes which are not unknowns of the synthesis problem and are required by the initial sizing solver to work automatically. The difficulty of this analysis is that the DOFs of the parts is often different from the desired DOFs of the mechanism to synthesize, so that the existing submechanisms and their inputs need to be determined and the kinematic description of their parts need to be analyzed, also without user intervention. The objective in mind is to solve automatically synthesis

problems for any kinematic task, giving as a result *the list of all initially sized solutions passing through some precision points*, i.e., valid initial guesses for initiating further optimization analyses and detailed design. The enumeration of alternatives and the initial sizing of each solution for an example of task with a previously known significant dimension are shown to illustrate the concepts.

The organization of the paper is as follows. In the following section we review the details of the developed solver for mechanism synthesis. In the section "Problem Description," we review the possible initial situations or tasks for the synthesis problem. Then, the algorithm for preprocessing and computing the kinematic analysis of some properly selected prescribed parts is detailed in the section "The Proposed Analysis for Known Parts." In the section "Subgraph Search of Parts in Atlas," we illustrate the output of the type synthesis process for a test problem. Finally, in the section "Decompositions into Single Open Chains," we show the influence of the previous kinematic analysis on the single-open chains decomposition stage used for sizing the mechanism by means of analytical equations.

MECHANISM SYNTHESIS SOLVER

The authors developed a new solver for mechanism synthesis which is integrated into a general software for mechanism analysis composed by a pre- and postprocessing module (*Samcef Field*), a nonlinear mechanisms analysis module (*Samcef Mecano*), and an optimization module (*Samcef BOSS-Quattro*) of parametric models (Samtech, 2006). The synthesis solver was implemented in the general finite elements code *Oofelie* (Cardona et al., 1994, Open Engineering, 2001). The function of this solver is to generate *the list of all initially sized solutions passing through some precision points and satisfying space requirements.* It is based on the exploitation of an atlas of mechanisms and on the exploration of alternatives with optimization techniques for nondifferentiable objective function, like genetic algorithms.

Figure 3 gives a simplified block diagram of the mechanisms synthesis solver. In the following paragraphs, each block of the blocks diagram will be referenced as (•). Figure 4 displays details of the synthesis solver and its input and output blocks.

Specifications Phase

From the interpretation of the customer's design requirements (1), the designer classifies the design requirements into structural (2), functional (3), and other requirements (4).

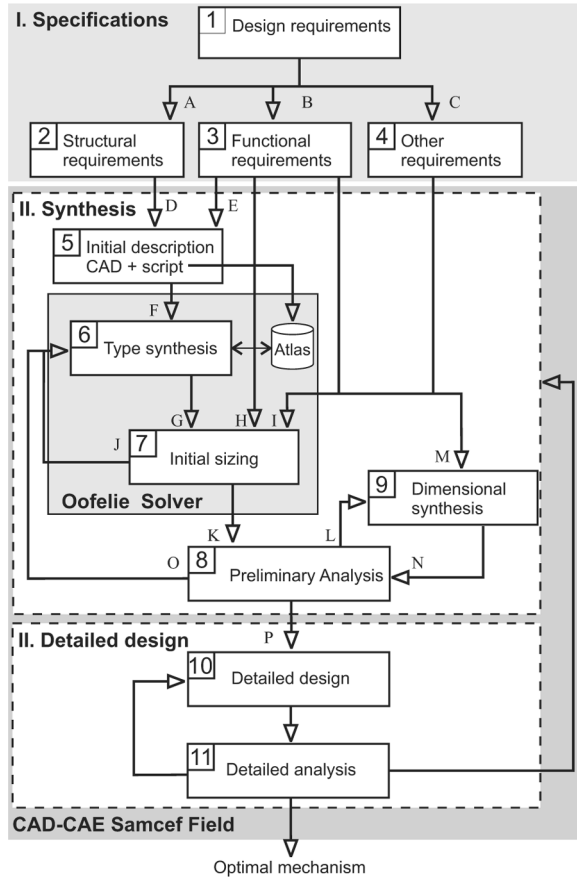

Figure 3. A method for optimal mechanism design: simplified block diagram.

Synthesis Phase

In order to define the problem (5), the designer draws the initial description of the existing bodies and mechanism parts using the *SAMCEF-Field* graphics interface, and then defines assemblies between them and constraints (fixations and motion constraints). The allowed space can also be defined geometrically. In this sense, some structural and functional constraints are loaded in a comprehensive way. In order to define the motion constraints, the kinematic task is discretized into a number of precision points, where initial, final, and some intermediate states are defined for the prescribed parts. Some nongeometric instructions are given in the form of a *script* to select the atlas of desired mechanisms, discrete constraints for the search, and other structural constraints.

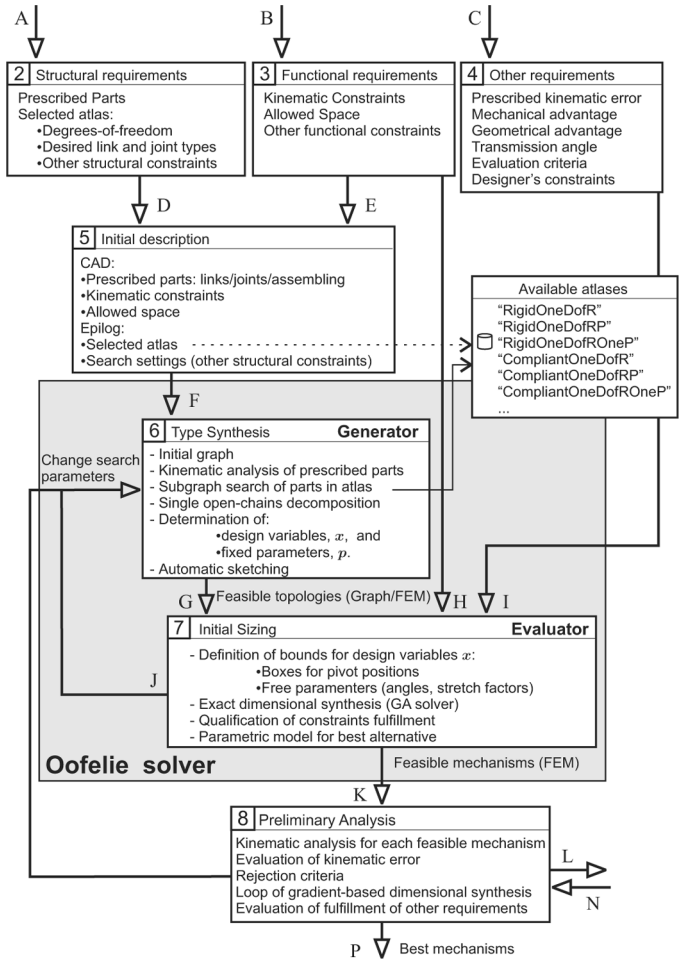

Figure 4. A method for optimal mechanism design: details of the synthesis solver.

The *Oofelie* synthesis solver consists of two substages:

- In the type synthesis (6) substage, the solver produces a graph representation of the kinematic problem called the initial graph. Then, the number synthesis is solved by running a subgraph search of such initial graph into an atlas of mechanisms also represented by graphs. Nonisomorphic subgraph matchings are saved as feasible alternatives (up to a number of solutions defined by the user). Each mechanism alternative is afterward analyzed and decomposed into several single-open chains (SOCs): dyads and triads. The kind and number of the free parameters needed for each SOC solver module are next

determined (e.g., bounds on boxes for pivot locations, missing angles, or stretching factors). The sketch of the mechanism as well as the default bounds of the involved variables are presented to the user for the subsequent substage. We remark that this substage is completely achieved in an automatic way and runs very fast on any PC.

- In the second substage, the initial sizing (7) for each alternative is made using analytical equations based on the precision-point method. The design space is defined by the set of free parameters, if any. The user can set the values of their bounds, and a genetic algorithm is used to sweep the design space. The fitness function consists in the minimization of the size of the mechanisms together with three weighted constraints: minimal length of link dimensions, noninversion of transmission angle, and allowed space violation. Eventually, instead of considering noninversion of transmission angle as a constraint, a full kinematic analysis is made for each individual to compute the fitness function. The parametric model of the obtained solution (an alternative inserted in the environment, with some dimensions and properties) matches the user's requirements first approximation (trajectories, obstacles and allowable space given by points or planes) and is the starting point for the next step, the dimensional synthesis.

For each alternative, the designer makes a preliminary analysis (8) using the nonlinear mechanisms analysis solver *Samcef Mecano*, for evaluating the generated task and some added design criteria: for example, path and function error, envelope and interferences (requiring knowledge of the CAD geometry), forces at connection points, and other functional constraints. Based on the results of this preliminary evaluation, three steps are undertaken: (1) the detailed design stage can be accessed directly; (2) a dimensional synthesis stage (9) is executed to make fine adjustments of the parameters values; and (3) a new execution of the synthesis solver with reformulated constraints or search parameters must be made.

The dimensional synthesis (9) consists in the fine adjustment of parameters employing gradient-based optimization techniques (managed by *Samcef BOSS-Quattro*). These methods require an initial guess of the solution. If the initial guess is not close enough to a solution, the iterative process may either converge very slowly, converge to an unacceptable solution, or diverge. The Genetic Algorithm used in the initial sizing solver helps to obtain a starting mechanism which is near to the global optimum.

Detailed Design Phase

A detailed design (10) of the full nonlinear mechanical system is modeled using the same software to perform a detailed analysis (11) of the system,

taking into account additional CAD details, loadings, and physical effects. The user is thus able to check the system behavior under real conditions and to perform stress analysis with realistic dynamic loads. The designer evaluates and simulates the performance of the mechanism variants and can perform further optimizations on some selected mechanical or structural parameters. If the requirements are not satisfied, a new execution of the synthesis phase may be reformulated.

PROBLEM DESCRIPTION

The more general statement of the kinematic synthesis problem is to define a set of connected/disconnected bodies with motion constraints on some of their points: prescribed displacements and/or prescribed rotations. To this end, a FEM description is used as follows:

1. Initial parts or the (sub)mechanism $M_{\text{ini}} = \{\mathbf{N}_{\text{ini}}, \mathbf{F}_{\text{ini}}, \mathbf{E}_{\text{ini}}\}$ can be conceived as a multiple set of data composed by sets of nodes $\mathbf{N}_{\text{ini}}$, fixations $\mathbf{F}_{\text{ini}}$, and elements $\mathbf{E}_{\text{ini}}$. The nodes serve to define elements such as rigid bodies, flexible bodies, and lumped masses (one-node elements). Two bodies can be assembled by other elements such as kinematic pairs (revolute, prismatic, slider, etc.), fixed joints, boundary conditions (assemblies with ground), and fixations, basing the description on the nodes of the two bodies to link. Nodes are also used to define other geometrical constraints, such as allowed space, obstacles, and the desired continuous trajectories.
2. Motion constraints can be defined by imposing a finite number of *prescribed displacements* and *prescribed rotations* on nodes or elements. Three possible constraints can be defined: sets of node displacements D, sets of link rotations L, and sets of joint parameters J (angles for revolute or displacements for prismatic joints).

The combination of these tools—parts and motion definition—gives freedom to the user to state different types of kinematics problems or tasks (Pucheta and Cardona, 2007a). The *significant dimensions* of a linkage determine its kinematic behavior. These dimensions are identified as the link dimensions between nodes connected by joints and nodes that develop a trajectory (commonly present in PF and RBG tasks).

On top of Fig. 5, we can see the FEM description of a coordination problem: an unknown 1-DOF-mechanism must guide the rotations of two bodies, E_{12} and E_{15}, passing through three prescribed angular positions. This example is developed throughout this paper, and represents a task with previously known significant dimensions in the initial parts description. This type of mechanism is used in practice to control the change of section of a convergent–divergent nozzle in

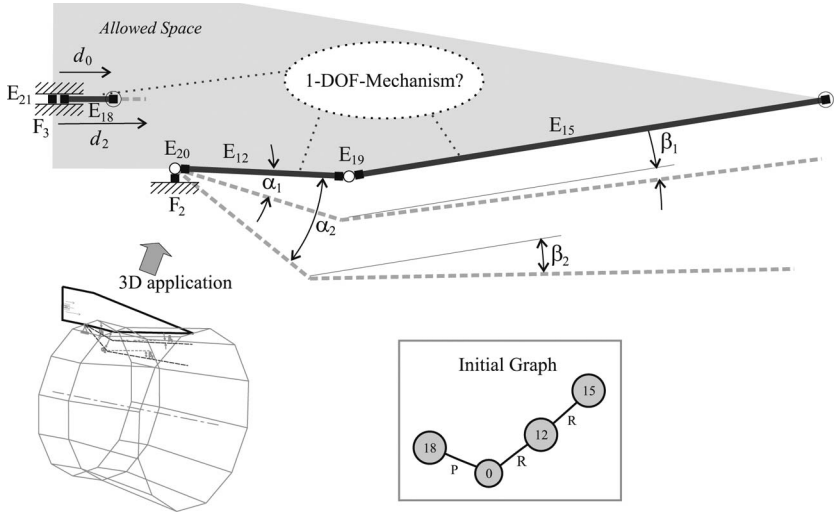

Figure 5. A three positions coordination problem between the rotations of two articulated bodies and the displacement of a prismatic actuator.

a turbine engine. The mechanism must be actuated by means of a grounded prismatic actuator for which the initial and final positions are prescribed. The area where the new bodies and new pivots must hold is also prescribed by a *face* defined by a closed polygon. Using this face, an *allowed space* restriction is defined.

Because we use graph theory to solve the number synthesis, we give a graph representation to the prescribed parts, M_{ini}, which we call *initial graph*, G_{ini}. In order to build this graph, we follow these rules:

- Vertices: Free bodies with imposed movements will be isolated vertices of the initial graph. The remaining bodies, connected through joints, will be connected vertices of the graph. Conventionally, the ground link will be the vertex zero. Depending on the number of grounded bodies, this vertex may be binary, ternary, etc. The number of isolated fixations (represented by fixed nodes in M_{ini}) is used to prescribe the degree of vertex zero (ground). For each isolated node with prescribed movement in M_{ini}, we assign an isolated vertex in the graph, although this node is not attached to any element.
- Edges: Joints will be edges of the initial graph connecting two of the previously defined vertices; all edges are assumed to be binary (isolated joints are not allowed).

Vertices and edges of the graph are labeled using the identifiers (IDs) assigned by the CAD interface. In the example, the set of vertices $V_{\text{ini}} = \{0, 1, 2, 3\}$ is labeled using a label function as

$V(V_{\text{ini}}) = \{0, 15, 12, 18\}$, whereas the set of edges E_{ini} is conveniently defined and labeled using pairs of labeled vertices, e.g., $E(E_{\text{ini}}) = \{(0, 2), (0, 18), (12, 15)\}$. Thus, the graph built from this set of integer labels is a labeled graph $G_{\text{ini}}(V(V_{\text{ini}}), E(E_{\text{ini}}))$. Then, to characterize the (sub)mechanism completely, the types of links and joints are assigned on the labeled vertices and edges of the graph, respectively (Pucheta and Cardona, 2007a; Yan, 1998). The graphic representation obtained is a colored labeled graph, which means that the graph has attributes or colors on each vertex and each edge.

Other complementary data identified from the CAD description contain:

1. Map of Trajectory Nodes and their associated Links, $\mathbf{n}_{\text{traj}}$: nodes which develop the prescribed trajectories in PF and RBG problems.
2. Map of Ignored Nodes to Links, $\mathbf{m}_{\text{ign}}$: note that nodes which neither pertain to any joint nor have prescribed motion are filtered out (these nodes are shown as "circled" in Fig. 5). Although they do not participate in the initial graph construction, they are stored to be used later in the dimensional synthesis stage. They serve either as checking points of the task or to define the restricted area (or space), where all links must hold for each precise position.
3. The degree of links connections is stored in a separated vector called minimum degree of vertices, $\mathbf{deg}_{\text{min}}$. It will be used later to accelerate or customize the subgraph search.
4. Map of Link Types (rigid, flexible), $\mathbf{T}_{\text{L}}$.
5. Map of Joint Types (revolute, prismatic, flexible or clamped), $\mathbf{T}_{\text{J}}$.

Note that the existing flexibility at the problem definition (Fig. 2) may lead to a very general *initial graph*. The initial graph can contain loops, branches, leaves, disconnected vertices, etc. For instance, for the specification of a function generation between two cranks, the initial graph is a tree where the ground can be considered as the root (see Fig. 1(a)). PF and RBG problems have disconnected vertices representing the guided bodies, which, in principle, are not connected to any other body (see Fig. 1(b) and (c)). In other words, the initial graph must correspond to a substructure of a mechanism, with the constraint coming from the FEM convention that defined joints must always have their two nodes defined and belonging to an existing body. In this example, the prescribed parts lead to an initial graph of tree type. In Fig. 5, the depicted *initial graph* is acyclic and has a tree form in which the ground can be taken as the root.

The initial graph is represented by its *type adjacency matrix*, $\mathbf{T}_{\text{ini}}$. It is defined using the symmetric adjacency matrix A_{ini} of the labeled graph G_{ini}, with integer entries on the diagonal representing the link types, and integer entries on the outer diagonal representing the joint types,

as follows:

$$T_{ii}(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is the ground,} \\ 1 & \text{if } v_i \text{ is a rigid link,} \\ 2 & \text{if } v_i \text{ is a flexible link,} \end{cases} \quad ,$$
$$T_{ij}(e_{ij}) = \begin{cases} 0 & \text{no connection} \\ 1 & \text{if } e_{ij} \text{ is a revolute joint,} \\ 2 & \text{if } e_{ij} \text{ is a prismatic joint,} \\ 3 & \text{if } e_{ij} \text{ is a flexible joint,} \\ 4 & \text{if } e_{ij} \text{ is a clamped joint.} \end{cases} \tag{1}$$

Thus, for the given example we have:

$$V(V_{\text{ini}}) = [0 \quad 15 \quad 12 \quad 18]$$
$$A_{\text{ini}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \mathbf{T}_{\text{ini}} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The complementary data for the given example are:

1. $\mathbf{n}_{\text{traj}} = \square$;
2. $\mathbf{m}_{\text{ign}} = \{L_{15} \rightarrow N_4, L_{18} \rightarrow N_9\}$;
3. $\mathbf{deg}_{\min} = \{\deg_{\min}(v_0), \deg_{\min}(v_{15}), \deg_{\min}(v_{12}), \deg_{\min}(v_{18})\} = \{2, 1, 2, 1\}$
4. $\mathbf{T}_{\text{L}}(V(V_{\text{ini}})) = \{0 \rightarrow 0, 15 \rightarrow 1, 12 \rightarrow 1, 18 \rightarrow 1\}$; and
5. $\mathbf{T}_{\text{J}}(E(E_{\text{ini}})) = \{(0, 12) \rightarrow 1, (0, 18) \rightarrow 2, (12, 15) \rightarrow 1\}$.

THE PROPOSED ANALYSIS FOR KNOWN PARTS

The procedure to identify the already known dimensions and consequently those parts with already known kinematic behavior, involves some manipulation on the initial graph in conjunction with the well-known Grüebler equation for the planar case:

$$f = 3(n - 1) - 2j, \tag{3}$$

where f is the number of degrees of freedom, n is the number of bodies, and j is the number of lower-pair joints (revolute or prismatic).

In Fig. 3, we can see the location of the type synthesis solver inside the process for mechanism design. In Fig. 4, block (6), we can see

that a second step is added for running a kinematic analysis, just after the initial graph construction and before the subgraph search. So, this analysis will be made only once and may result in the interruption of the whole process if any data are either badly defined or well stated but not supported by the solvers.

We need to make use of some basic graph theory definitions. A graph is *connected* if there exists a path for every two pairs of vertices. A *component* of a graph is a maximal connected subgraph. An isolated vertex forms a *trivial component*, consisting of one vertex and no edge. Then, the procedure called *initial kinematic analysis* is the following:

I. Build the initial graph G_{ini} of the prescribed parts.
II. Identify submechanisms by preprocessing data in the following steps:

- Disconnect and delete the ground of the initial graph.
- Compute the resultant connected *components*, H_i's.
- For each component H_i, connect the ground again so that a submechanism M_i is obtained.

III. For each submechanism M_i, *consider all motion constraints as inputs*:

- Define parts of submechanisms taking progressively joints and bodies starting from the ground.
- For each part:

1. Compute the number of links and joints, n^i and j^i, respectively.
2. For each precision position (configuration):

   i. Compute the number of motion constraints I^i for the configuration.
   ii. Compute the Grüebler mobility: $f^i = 3(n^i - 1) - 2j^i$.

3. If $I^i = f^i$, i.e., the number of inputs equals the mobility, then analyze and compute components of displacements and rotations of links, joints, and nodes of the part for this configuration.

IV. Classify nodes into three categories:

- With unknown position and all unknown displacements.
- With known position and all known displacements.
- With known position and some unknown displacement.

In Fig. 6, we can see simple illustrations of the submechanism formation for the problem presented in the introductory section. We may also note

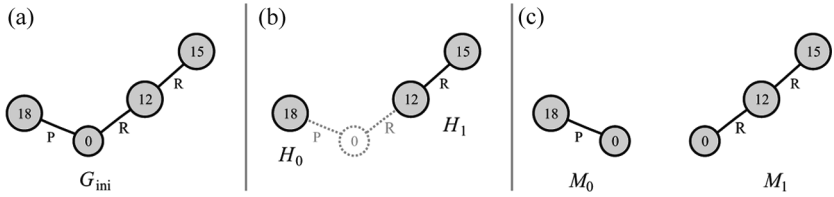

Figure 6. Algorithm for submechanism identification: (a) given graph; (b) ground disconnection and components computing; (c) submechanisms formation.

that the submechanism M_0 has incomplete motion constraints definition, because the user only specified the initial and final positions of the actuator (displacements d_0 and d_2). Therefore, the nodes of the prismatic joint must participate in the dimensional synthesis process for calculating the intermediate displacement d_1. The other submechanism M_0 satisfies the Grüebler equation for all the precision positions; then its parts are analyzed at step III-3, and thus all nodes are classified as data in step IV. This classification has no influence on the type synthesis stage (see section "Subgraph Search of Parts in Atlas"), so we will use it later where the unknowns of the problem are identified (see section "Decompositions into Single Open Chains").

We should also remark that the condition $I^i = f^i$ may result false for many well-posed problems for which dimensions are given but motion constraints are incomplete. Therefore, step III requires a complex iterative algorithm for navigating through the vertices and edges of a given submechanism in increasing order, retaining the last set of vertices and edges for which the number of DOFs equals the number of inputs. The example developed in this paper does not have this complication.

SUBGRAPH SEARCH OF PARTS IN ATLAS

A list of available atlases (see Fig. 4) was enumerated and stored using the matrix representation given by Eqs. (1) and (2). From the selection of an atlas, the user configures the discrete design space for the mechanism solutions. We must mention that the *atlas generation* involves two steps:

1. *Enumeration of all nonisomorphic kinematic chains for the required degrees of freedom.*
2. *Specialization of links and joints types in all nonisomorphic ways*, i.e., the assignment of links and joints types over the kinematic chains without repetition.

Because these enumerations are already available, the CPU time consumption can be considerably reduced. In this way, in order to obtain

feasible topologies, we only need to find those mechanisms of the atlas which match the initial graph. This is however a very tedious task to perform by hand, and may lead easily to errors. Therefore, this subgraph search is done automatically.

The algorithm for subgraph search that we implemented is very basic and can be found fully detailed in Pucheta and Cardona (2007b). Before giving a brief description of the algorithm, we define two tools:

1. *Diagonally extended degree code computed by rows*, DC_b^r: This procedure for detecting isomorphisms in colored graphs is inspired on the degree code presented by Tang and Liu (1993). The upper-triangular elements of the adjacency matrix including the main diagonal are concatenated row-by-row to form a code $C_b^r(A)$. Every row is converted from an adequate basis b to a decimal system. Comparisons between two codes are made in lexicographical order. Among all permutations inside the groups of vertices of equal degree, we compute the code retaining that one with the maximum value. We denote it as $DC_b^r(A)$. An example of code computing is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 \end{bmatrix} \rightarrow C_3^r(A) = \begin{bmatrix} (0110)_3 \\ (101)_3 \\ (12)_3 \\ (1)_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 5 \\ 1 \end{bmatrix}.$$

   For this matrix, the degree code occurs for permutation $p = \{2, 0, 1, 0\}$, then

$$A^p = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \therefore DC_3^r(A) = C_3^r(A^p) = \begin{bmatrix} (1210)_3 \\ (101)_3 \\ (11)_3 \\ (0)_3 \end{bmatrix} = \begin{bmatrix} 48 \\ 10 \\ 4 \\ 0 \end{bmatrix}_{10}.$$

   The degree code is *unique* and *decodable*. Note that b is the number of different entries in A. In order to enumerate a given atlas, to compare and retrieve the colored graph, we must take the same basis b for computing and storing the codes of every mechanism of such atlas.
2. *Synthesis adjacency matrix*, $S(G_A^p)$: It is used to codify the locations of the subgraph occurrences $G_{\text{ini}}(V(V_{\text{ini}}), E(E_{\text{ini}}))$ inside a permuted graph taken from the atlas $G_A^p(V_A^p(V_A^p), E_A^p(E_A^p))$. The construction is similar to the type adjacency matrix definition, but we make a difference on the prescribed parts in such a way to consider every prescribed part as a *different functional part*, independently of the link type assigned by the user. Thus, the definition is:

$$S_{ii}(v_i) = \begin{cases} T_{ii} & \text{if } v_i \text{ is a synthesized vertex,} \\ k & \text{if } v_i \text{ is a prescribed vertex,} \end{cases}$$

$$S_{ij}(e_{ij}) = T_{ij} \quad \forall e_{ij}$$

where $k = b + j$; $j = 0, 1, \ldots, (n_{\text{ini}} - 1)$, with $n_{\text{ini}} = |V_{\text{ini}}|$ the cardinality of the prescribed vertices set, and b the number of colors in the selected atlas.

Using these two definitions the algorithm for detection of all subgraph occurrences of the initial graph, G_{ini}, inside a graph of the atlas named as G_A can be briefly described. Depending on the structure of both graphs there could be more than one subgraph occurrence.

We take G_{ini} as reference for comparison and take convenient permutations of the vertices of G_A to build graphs that we call G_A^p. Because graphs are colored, we consider that a given subgraph occurrence $G_{\text{ini}} \subseteq G_A^p$, if vertices, edges, links and joint types match; then, the labels of vertices and edges of G_{ini}, are copied into G_A^p; for the remaining vertices and edges of G_A^p, any label not already used is assigned. The type adjacency matrix T_A^p and synthesis adjacency matrix S_A^p are constructed for G_A^p. Then, the degree code of S_A^p is computed, $DC_b^d(S_A^p)$. If this code is not already stored, the code and the alternative are saved $M_a \leftarrow [C_b^d(T_A^p), V_A^p(V_A^p), E_A^p(E_A^p)]$; otherwise a new permutation is taken.

Starting from this algorithm, we implemented some improvements oriented to the type synthesis process:

- The *rejection of pseudo-isomorphisms*: it consists in saving a mechanism only if it does not have a subgraph that coincides with a previously retained mechanism.
- The constraint *degree of prescribed links connection* is used to force the connectivity of the prescribed parts to be smaller or bigger than a prescribed integer value. This is easily implemented by checking the degree of the vertices of the permuted graph taken from the atlas G_A^p, which matches with G_{ini}. By default, the initial graph has a vector with minimum degree $\mathbf{deg_{min}}$. For example, if the user prescribes three pivots, $\deg_{\min}(v_0) = 3$, and those mechanisms with binary ground will be rejected early in the algorithm. To better restrict this search, the user can prescribe either a minimum degree $\deg_{\min}$ or a maximum degree $\deg_{\max}$.
- A third constraint to be imposed is the *distance of ground to the objective vertices*. When the initial graph has floating links, a distance constraint from the ground vertex v_0 to these links is forced to have a minimum value of 2. This constraint is also computed after detecting the subgraph occurrence $G_{\text{ini}} \subseteq G_A^p$.

The initial graph for the example problem was shown in Fig. 5. In Fig. 7, we show the first 10 occurrences of the solutions for a subgraph search

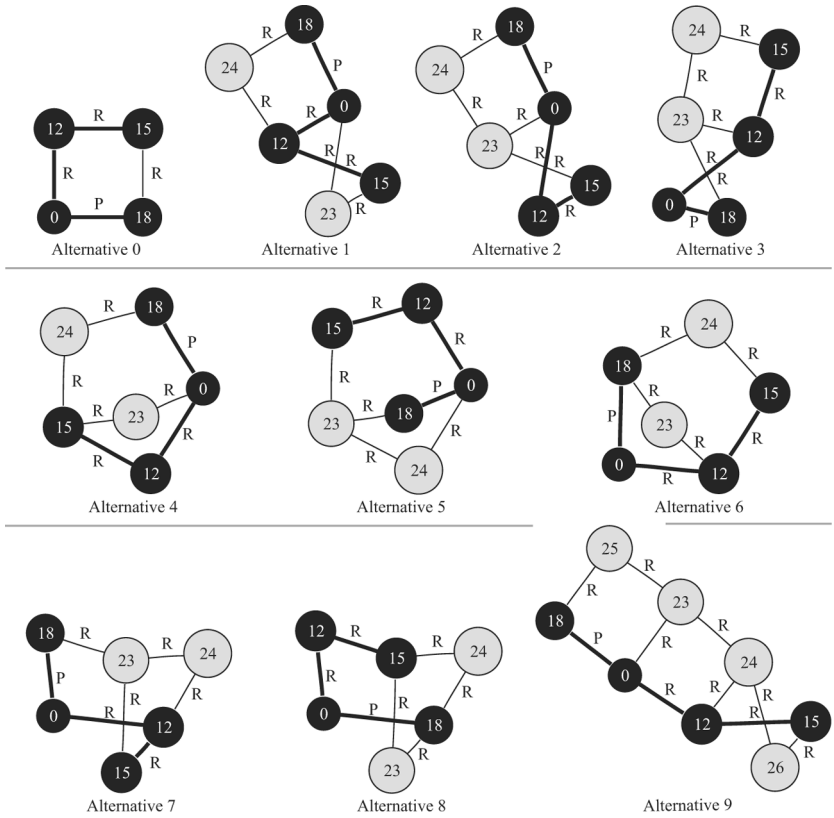

Figure 7. Nonisomorphic graphs occurrences of the initial graph inside mechanisms of the atlas, obtained in the number synthesis stage. For clarity, edges are only labeled with their joint types (R: revolute, P: prismatic).

of the initial graph inside a selected atlas of one-DOF mechanisms with revolute joints and one prismatic joint (RigidOneDofOneP). Note that if we do not consider the difference in prescribed parts, Alternatives 1 and 2 would be isomorphic. In this case the detected difference in the degree codes of their synthesis adjacency matrices $DC_b^d(S_A^p)$ enables their acceptance as feasible and different mechanisms.

DECOMPOSITIONS INTO SINGLE OPEN CHAINS

We implemented computational methods to find the significant dimensions of multiloop linkages and to decompose a topology into Single-Open Chains (SOC) using methods based on the classical work of Erdman and Sandor (1997). Figure 8 details the decomposition

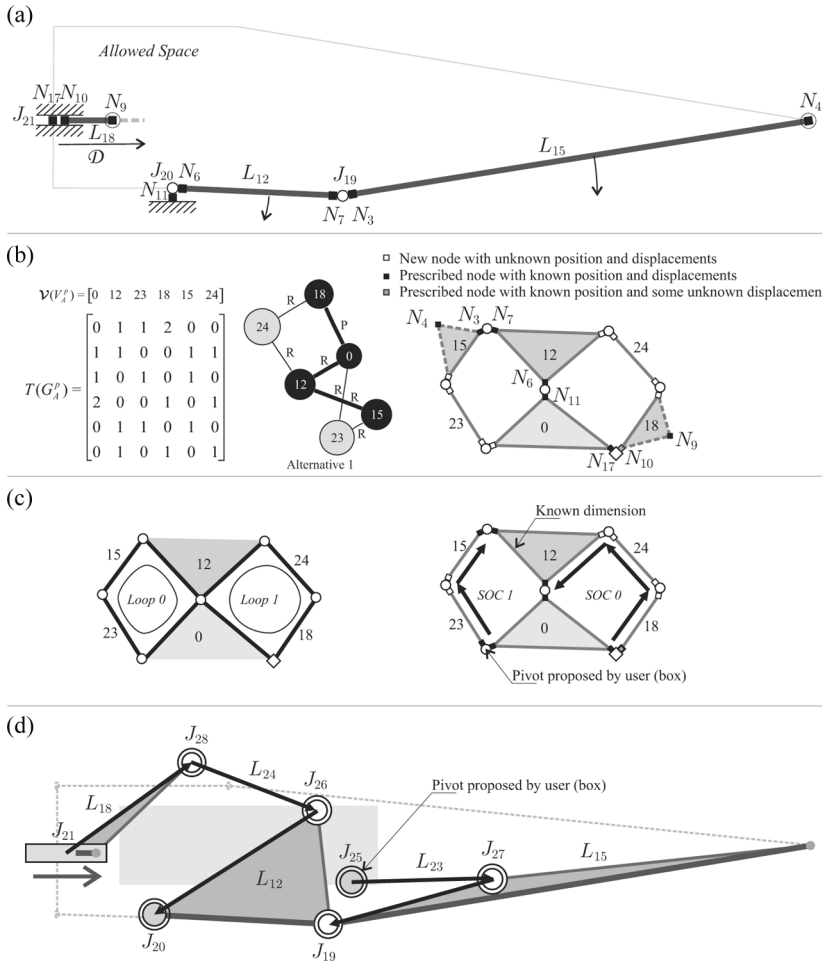

Figure 8. Decomposition process: (a) data, (b) type synthesis and nodes classification, (c) loops computation and identification of significant dimensions, (d) single-open chains decomposition with true and sketched coordinates.

process (Pucheta, 2008, Chap. 5). In Fig. 8(a) the initial description of the prescribed parts is shown up the node level, whereas Figs. 8(b)–(d) show the initial parts integrated with the first type synthesized alternative.

Note that for synthesis purposes, nodes N_4 and N_9, which belong to links L_{15} and L_{18}, respectively, are ignored because they are not connected by joints and have not prescribed displacements (see Figs. 8(a) and (b), right), therefore they are stored in the auxiliary map $\boldsymbol{m}_{\mathbf{ign}}$ to be restored in the final result.

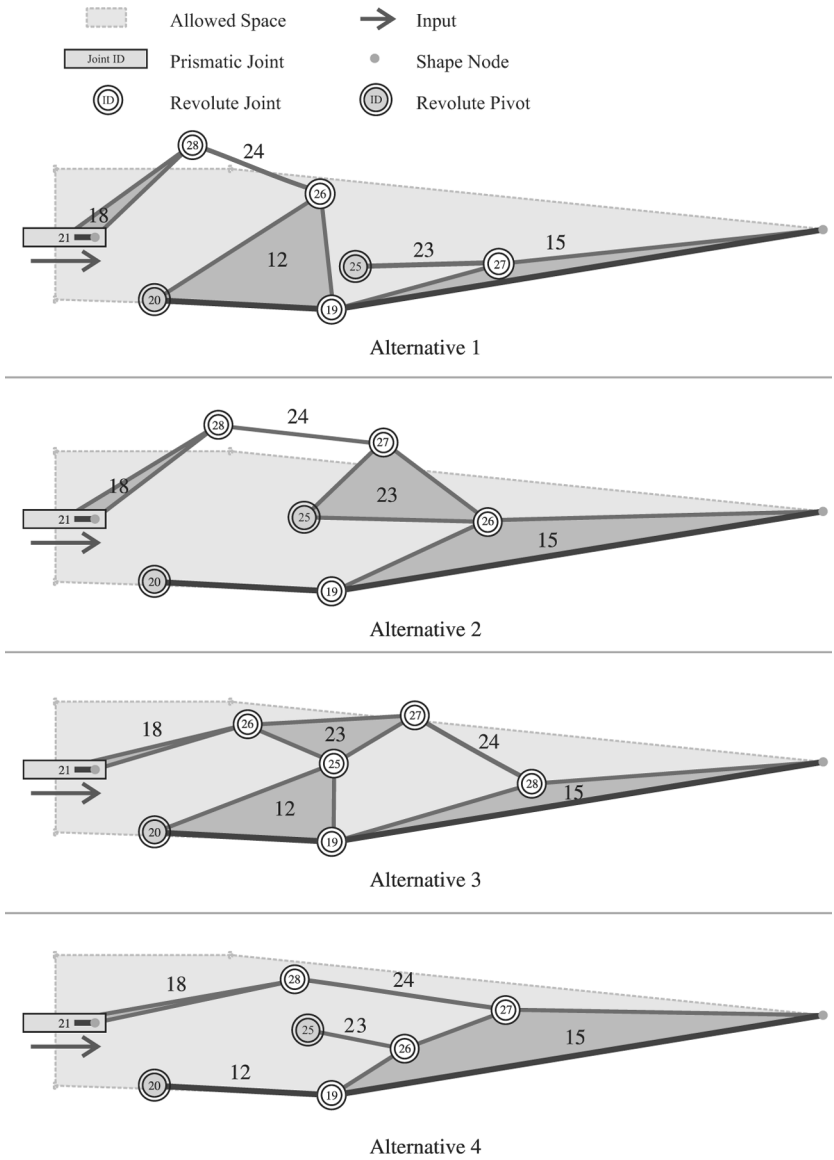

Figure 9. Sketches of the first nine alternatives found (continued).

Figure 8(b) shows the node classification obtained by computing the *kinematic analysis of the initial parts*. Because node N_{10} has an unknown displacement component, it is shown in gray. This missing component will be computed in the dimensional synthesis stage (see Fig. 8(c), right). This figure also shows nodes N_3, N_7, and N_6 with a

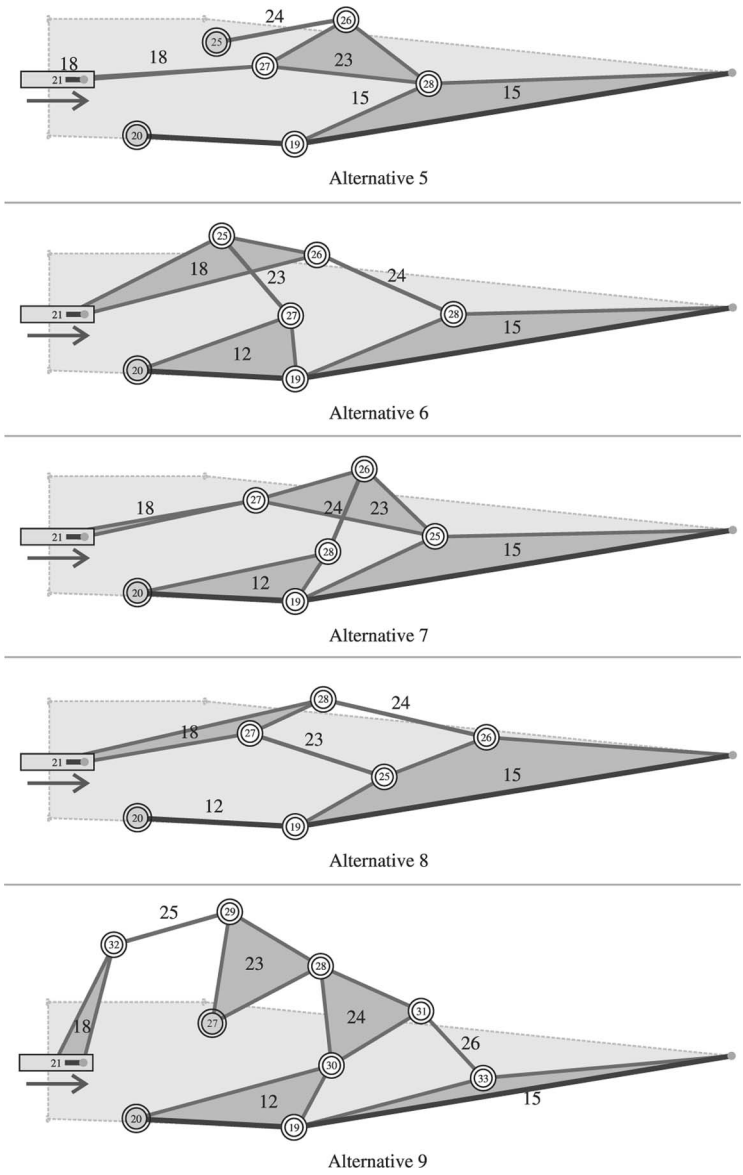

Figure 10. Sketches of the first nine alternatives found.

black-filled square, meaning that their positions and displacements were computed by the kinematic analysis of the initial parts step: by actuating joints J_{20} and J_{19}, the kinematic description of the nodes in links L_{12} and L_{15} is analyzed. In effect, this calculation changes the behavior of the

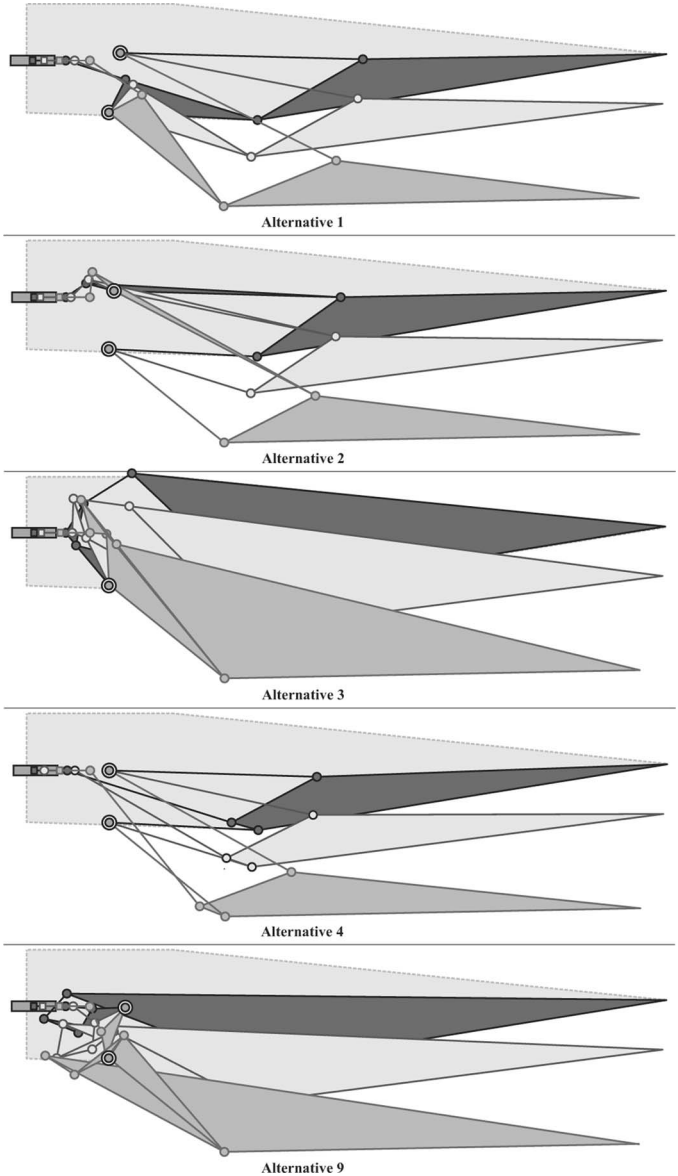

Figure 11. Mechanisms solution for Alternatives 1 (Watt II), 2 (Watt II), 3 (Watt I), 4 (Stephenson III), and 9 (three loops).

SOC's decomposition algorithm (shown in Fig. 8(c), right, and explained below). Nodes N_{17} and N_{11} with a black-filled square are fixations of the mechanism, and therefore their positions and displacements are known. The exception to the rule is constituted by the nodes of newly

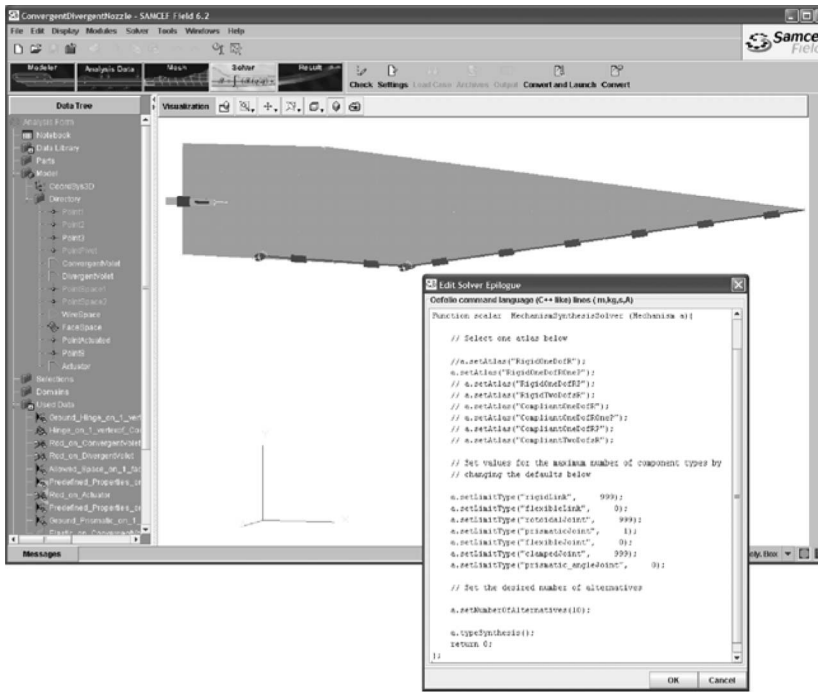

Figure 12. Initial description of the kinematic problem using the Samcef-Field CAD environment and basic settings for the type synthesis solver.

created pivots, which are considered as known for the decomposition purpose. The remaining nodes represented by white-filled squares are the unknowns of the problem.

The *basis of loops with minimal length* is used to determine the significant dimensions of the linkage Fig. 8(c). The number of loops in the basis, L, is equal to the number of independent loops which can be easily computed from graph theory by using equation $L = e - v + 1$, where v is the number of vertices and e is the number of edges of the graph. However, in some cases like those of Alternatives 4 to 6, there might be two or more loops with equal lengths, and therefore we must repeat the procedure for more than one single basis.

The criterion to break the loops and form single-open chains is to visit the loops, node by node, using node classification, *starting from a node with known position and displacements* and *ending in the first node with known position and displacements, passing through the unknown nodes*. One single link represented by the arrow of a complex number is added for every *significant dimension*; see Fig. 6(c), right.

Because rotations of links that belong to one single body are the same, SOCs have to be evaluated following an appropriate *order of*

computation (some particular cases are however order independent). Among all possible sets of SOC decompositions we retain that which:

- solves a maximum number of imposed constraints in the *order of computation*, and
- satisfies the necessary conditions for solving every SOC: dyads and triads passing through the prescribed number of positions.

Otherwise, the algorithm reports that the topology cannot be solved for the required task.

The 10 topologies shown in Fig. 7 were analyzed and decomposed into single open chains (dyads and triads passing through three positions) using this decomposition method. In this case, our decomposition algorithm rejected Alternative 0 because the decomposition into dyads resulted in an over-constrained case. The remaining alternatives are sketched in Figs. 9 and 10.

By analyzing the graphs or, more easily, the sketches, we can realize that Alternatives 3, 6, 7, and 8 have a binary ground so that these solutions used only the imposed fixations. The other feasible alternatives have a ternary ground, so a new pivot location must be computed. When a pivot location is unknown, the user must specify a box for the fixed node location; otherwise, a default box is computed as the bounding box which contains all nodes. Then the program looks for the optimum location inside the defined box.

The retained decomposition for Alternative 1 is shown in Fig. 8(d). We refer to Pucheta and Cardona (2007c) and Pucheta (2008, Chap. 5) for complete details of the decomposition method.

INITIAL SIZING

The design space for assigning dimensions to the mechanism is defined by the set of free parameters, if any. The user can change the values of bounds, and a genetic algorithm is used to sweep the design space (Pucheta, 2008, Chap. 6; Pucheta and Cardona, 2005). The fitness function consists in the minimization of the size of the mechanisms together with three weighted constraints:

- minimal length of link dimensions,
- noninversion of transmission angle,
- allowed space violation.

Eventually, instead of considering noninversion of transmission angle as a constraint, a full kinematics analysis is made for each individual to compute the fitness function.

The initial sizing solver was run for the nine feasible alternatives shown above.

Examples of initial sizing for Alternatives 1–4 and 9 are shown in Fig. 11 where the three constraints were taken into account.

From the obtained simulations, we observed that all of them fulfill the noninversion of transmission angle constraint whereas only Alternatives 1 and 2 satisfy the allowed space constraint for all the configurations. We also observed the following:

- Alternatives 1 and 2 are Watt II six-bar linkages because they have Watt's kinematic chains and ternary grounds; thus a new pivot in each one was synthesized. The second solution seems to be simpler than the first one because there is no bar connected to the grounded flap.
- Alternative 3 is a Watt I six-bar linkage. Only the two proposed pivots are used and therefore no new pivot is synthesized. This solution slightly violates the allowed space at the starting position.
- Alternative 4 is a Stephenson III six-bar linkage. It has no bar connected to the grounded flap. Note however, that there exists interference between the links of the coordinated flaps at the final position. The same interference occurs in the following solution, the eight-bar linkage of Alternative 9. This alternative is unnecessarily complex.

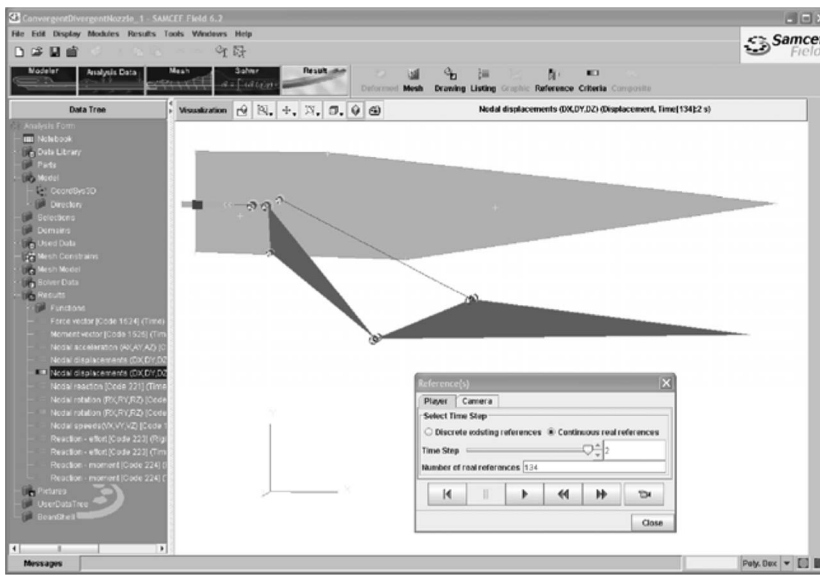

Figure 13. Example of simulation after the initial sizing execution (Alternative 1).

One (or several) of these alternatives can be selected and analyzed for evaluating further nonkinematic requirements, such as mechanical advantage, power consumption, and others. Interference avoidance as well as the automatic qualification of the alternatives will be considered in future research.

Finally, two screen snapshots of the developed software are displayed, illustrating the way in which the user communicates with the software. Figure 12 displays the user interface at the moment of giving the problem description and asking for the type synthesis execution. Figure 13 displays the results of simulation of the computed mechanism.

CONCLUSIONS

The kinematic analysis of the initial parts opens new possibilities for designing mechanisms. This aspect has not been taken into account previously in software for synthesis of mechanisms. The algorithms proposed in this work satisfied this requirement, allowing computing solutions for problems of synthesis in cases in which some parts are defined from the beginning.

The method was implemented in C++ language within the object-oriented code Oofelie. Although the method presented here deals only with planar linkages, it can be extended for synthesis of spatial (spherical) mechanisms. Additionally, an extension to the synthesis of partially compliant linkages has already been presented elsewhere (Cugnon et al., 2007; Pucheta and Cardona, 2007a).

In the current software implementation, the need of defining intermediate bounds for the variables for the initial sizing stage hinders making the procedure fully automatic. The computation has to be stopped after the type synthesis phase, and the user should introduce the information that is needed for the initial sizing phase. In a near future, the major aim is to automatically qualify each solution and sort them as function of requirements satisfaction. This would enable us to apply an optimal design strategy starting from scratch.

ACKNOWLEDGMENTS

This work has received financial support from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT), Universidad Nacional del Litoral (UNL) from Argentina, and the European Community through grant SYNCOMECS (SYNthesis of COmpliant MEChanical Systems) project UE FP6-2003-AERO-1-516183. The authors wish to acknowledge the kindness of Christian Paleczny, Ettore Baldassin, Aurelio Boscarino, Benoît Colson, and Frédéric Cugnon in the assistance and valuable feedback in this project.

REFERENCES

Campbell, M. I., Cagan, J., Kotovsky, K. (2003). The A-Design approach to managing automated design synthesis. *Research in Engineering Design* 14(1):12–24.

Cardona, A. (1989). An Integrated Approach to Mechanism Analysis. PhD thesis, University of Liège, Belgium.

Cardona, A., Klapka, I., Géradin, M. (1994). Design of a new finite element programming environment. *Engineering Computations* 11(4):365–381.

Chen, D.-Z., Pai, W.-M. (2005). A methodology for conceptual design of mechanisms by parsing design specifications. *ASME Journal of Mechanical Design* 127(6):1039–1044.

Chen, Y., Feng, P., He, B., Lin, Z., Xie, Y. (2006). Automated conceptual design of mechanisms using improved morphological matrix. *Journal of Mechanical Design* 128(3):516–526.

Chiou, S.-J., Kota, S. (1999). Automated conceptual design of mechanisms. *Mechanism and Machine Theory* 34(3):467–495.

Cugnon, F., Cardona, A., Selvi, A., Paleczny, C. (2007). Synthesis and optimization of flexible mechanisms. In: Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2007. Milan, Italy.

Erdman, A. G., Sandor, G. N. (1997). *Mechanism Design: Analysis and Synthesis* Vol. 1. 3rd ed., Upper Saddle River, New Jersey: Prentice-Hall.

Freudenstein, F., Maki, E. R. (1979). Creation of mechanisms according to kinematic structure and function. *Journal of Environmental and Planning B* 6:375–391.

Géradin, M., Cardona, A. (2001). *Flexible Multi-Body Dynamics: A Finite Element Approach*. New York: John Wiley & Sons.

Hansen, M. R. (1993). A multi level approach to synthesis of planar mechanisms. In: Proceedings of the NATO-Advanced Study Institute on Computer Aided Analysis of Rigid and Flexible Mechanical Systems. Troia: Portugal.

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

Hayes, M. J. D., Zsombor-Murray, P. J. (2004). Towards integrated type and dimensional synthesis of mechanisms for rigid body guidance. In: Proceedings of the CSME Forum 2004. University of Western Ontario: Canada, pp. 53–61.

Kinzel, E. C., Schmiedler, J. P., Pennock, G. R. (2006). Kinematic synthesis for finitely separated positions using geometric constraint progamming. *ASME Journal of Mechanical Design* 128(5):1070–1079.

Liu, Y., McPhee, J. (2007). Automated kinematic synthesis of planar mechanisms with revolute joints. *Mechanics Based Design of Structures and Machines* 35(4):405–445.

Open Engineering S. A. (2001). *OOFELIE: Object Oriented Finite Elements Led by Interactive Executor*. http://www.open-engineering.com. University of Liège, Belgium and INTEC: Argentina.

Pucheta, M. A. (2008). Computational Methods for Design and Synthesis of Planar Mechanisms. PhD thesis, Universidad Nacional del Litoral, Santa Fe, Argentina.

Pucheta, M. A., Cardona A. (2005). Type synthesis and initial sizing of planar linkages using graph theory and classic genetic algorithms starting from parts prescribed by user. In: Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2005. Universidad Politécnica de Madrid: Madrid, Spain.

Pucheta, M. A., Cardona, A. (2007a). Kinematics synthesis of compliant mechanisms using rigid-body replacement. In: Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2007. Politecnico di Milano: Milan, Italy.

Pucheta, M. A., Cardona, A. (2007b). An automated method for type synthesis of planar linkages based on a constrained subgraph isomorphism detection. *Multibody System Dynamics* 18(2):233–258.

Pucheta, M. A., Cardona, A. (2007c). Kinematics synthesis of planar multi-loop linkage mechanisms for multiple purposes. In: *Mecánica Computacional*. Vol. XXVI, Proceedings of XVI Congreso Sobre Mét. Numéricos y sus Aplicaciones, ENIEF 2007. Córdoba, Argentina: AMCA, pp. 3043–3055.

Samtech, S. A. (2006). *SAMCEF Users Manual*. http://www.samcef.com.

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

Sardain, P. (1997). Linkage synthesis: Topology selection fixed by dimensional constraints, study of an example. *Mechanism and Machine Theory* 32(1):91–102.

Sedlaczek, K., Gaugele, T., Eberhard, P. (2005). Topology optimized synthesis of planar kinematic rigid body mechanisms. In: Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2005. Universidad Politécnica de Madrid: Madrid, Spain.

Tang, C.-S., Liu, T. (1993). The degree code – a new mechanism identifier. *ASME Journal of Mechanical Design* 115:627–630.

Tian, Y.-L., Zou, H.-J., Guo, W.-Z. (2006). An integrated knowledge representation model for the computer-aided conceptual design of mechanisms. *International Journal of Advanced Manufacturing Technology* 28(5–6):435–444.

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

Yan, H.-S. (1998). *Creative Design of Mechanical Devices*. Singapore: Springer-Verlag.

Yan, H.-S., Hung, C.-C. (2006). Identifying and counting the number of mechanisms from kinematic chains subject to design constraints. *ASME Journal of Mechanical Design* 128(5):1177–1182.