Multibody Syst Dyn
DOI 10.1007/s11044-007-9087-x

An automated method for type synthesis of planar linkages based on a constrained subgraph isomorphism detection

Martín Pucheta · Alberto Cardona

Received: 12 February 2007 / Accepted: 24 June 2007

Abstract We present a method to enumerate and codify all non-isomorphic solutions, for the problem of synthesizing the type of single-DOF linkage mechanisms that satisfy structural requirements for a given kinematic problem. The method is based on the construction of an "initial graph" taking into account prescribed parts (such as fixations, moving bodies, joints and their interconnections) and the kinematic constraints imposed on them. This initial graph containing structural characteristics of the problem is used as a pattern to search inside a selected atlas of one-DOF mechanism also represented by graphs. A new graph-matrix representation of mechanisms and a mechanism identifier based on the degree code concept was developed to avoid isomorphic occurrences of the initial graph inside each mechanism of the atlas. The same tools were used to enumerate various atlases specialized in a non-isomorphic way from basic kinematic chains. This enumeration takes into account different types of links (rigid, flexible) and joints (revolute, prismatic, flexible, clamped), and proper restrictions were designed to avoid kinematically invalid topologies. The methodology is illustrated with examples for several kinematic tasks.

Keywords Linkage mechanisms · Type synthesis · Combinatorial analysis · Graph theory · Sub-graph search · Graph isomorphism · Degree code

1 Introduction

The essence of the mechanism synthesis problem is to find the mechanism for a given motion or *task*. The given motion could be exactly or approximately achieved by different *types* of candidate mechanisms. The *type* could be linkage, gear, cam, belt, etc., or a combination thereof. In this work, we focused our investigation on the *linkage type*, restricted to planar mechanisms that utilize only lower-pair joints also called *planar linkages*.

Linkage synthesis methods are usually decomposed into two phases: (i) *Type synthesis*, where the number, type and connectivity of links and joints are determined [1–3], and

M. Pucheta · A. Cardona (✉)
Centro Internacional de Métodos Computacionales en Ingeniería, CIMEC-INTEC, Universidad Nacional del Litoral-CONICET, Güemes 3450, S3000GLN, Santa Fe, Argentina
e-mail: acardona@intec.unl.edu.ar

(ii) *Dimensional synthesis*, where the proper dimensions of parts (links lengths, their angles at the starting position and pivots positions) are computed [4–8]. The latter task is also related to that of optimization of the mechanism where the fulfillment of more objectives and restrictions could be taken into account (allowed space, masses, sections, inertias, weight, materials, cost, etc.). The dimensional synthesis phase and all the subsequent stages of detailed design are very costly, so it is essential to have good topology alternatives as output of the type synthesis stage, in order to avoid repeated simulations, and also not to leave alternatives without being explored. Specifically, the aim is to *generate* a list of all non-isomorphic mechanism alternatives potentially suited to develop a required task.

The choice of a suitable mechanism for the desired purpose may be done by visual inspection on an atlas of linkages, but such a visual search is only based on the intuition of a experienced designer; such a procedure may easily lead to neglecting possible solutions. Graph theory can be conveniently applied to do this search automatically.

Since the mid-sixties, graph theory has been used for systematic enumeration and topological analysis of rigid linkages (see reviews of Olson [1] and Mruthyunjaya [9]). In the nineties, flexibility was incorporated by Murphy, Midha and Howell [10, 11] for constructing atlases of flexible linkages. Bar-linkages and other atlases for particular purposes (variable-stroke engine mechanisms [12], planetary gear trains, parallel and robotic mechanisms [3, 13]), were structurally synthesized using a systematic strategy based on the Freudenstein and Maki concept [12] of "separation" of *kinematic structure* (to generate alternatives) from *function* (to evaluate the generated alternatives). In these cases, part of the functional constraints were dealt with after the enumeration process, and the generated alternatives were post-analyzed by hand with the aids of combinatorics to validate the task matching. However, an automated method to generate alternatives satisfying those functional requirements which have structural influence at the outset is less addressed in the literature. We can mention recent computational methods of Chiou and Kota [14], and Moon and Kota [15], which represent and decompose the input/output motion requirements using a matrix representation called Motion Transformation Matrix (MTM), and a properly defined algebra to find all matchings between the MTM representation of the task or sequences of sub-tasks—and the MTMs of stored "design building blocks". While the search takes place, they evaluate functional and operational constraints, thus designing concepts for the desired behavior. They also store parametric values useful for later dimensional synthesis and simulation. Very recently, Chen and Pai [14] presented a design methodology in the same line of research that we have been working on. They parsed the design specification into functional requirements (**a**), structural requirements (**b**), and design constraints (**c**). Using **a**, they construct the *functioning kinematic chains* of a mechanism; from **b**, they search the *admissible kinematic structures* in atlases of kinematic structures; from the analysis of **a** and **b**, they identify the *compatible kinematic chains* which must also satisfy **c**. Finally, after labeling of joints in the compatible kinematic structures they validate the remaining design constraints. Other methods fall in the field of expert systems.

In this setting, we present a subgraph approach where the prescribed parts to move, and the task desired for them, are modeled by an *initial graph* and incorporated into a generator of alternatives (see Fig. 1 as an introductory example). In addition, we adopt an *atlas of kinematic chains (KC)* with simple joints using a graph representation for each KC. Then, we *specialize* each kinematic chain to form various atlases of mechanisms that could be selected as in a finite exploration space. Because both the problem and the mechanisms have graph representation, the generator of alternatives that we propose is an engine for subgraph searching of all non-isomorphic occurrences of the initial graph inside each mechanism of the selected atlas.

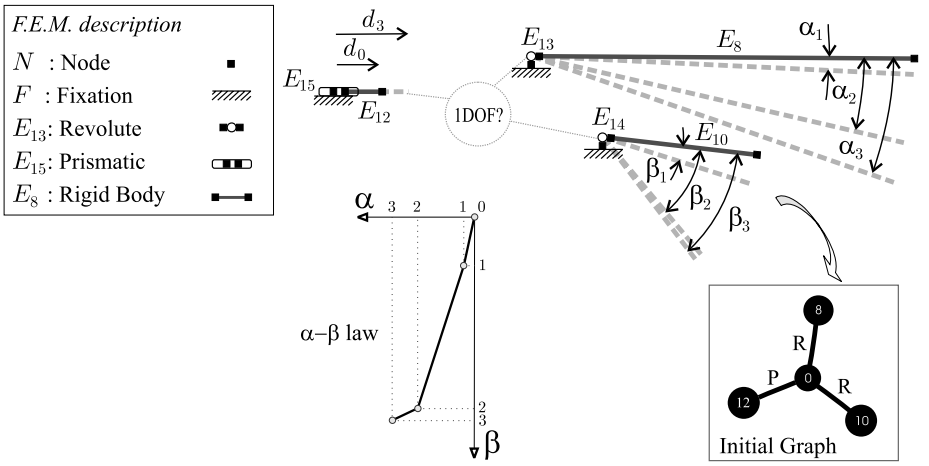

Fig. 1 Graph representation for a combined kinematic task

To avoid isomorphisms, we define a "type adjacency matrix" in conjunction with a new mechanism identifier based on the degree code characterizing unequivocally an alternative mechanism. These tools are used for atlases construction, subgraph occurrences detection, and, if it is desired, *pseudo-isomorphic mechanisms* rejection (a mechanism which presents an "idler loop" with some unloaded links, will be considered as *pseudo-isomorphic mechanism*).

In Fig. 2, we describe the proposed method. The designer interacts in the preprocessing and postprocessing stages, i.e. he enters data into a computer program in stage **I**, then runs the solver **II**, and afterwards evaluates the solutions **III**. In this paper, we are focused mainly on the theory applied in the solver.

The organization of the paper is as follows. A mechanism identifier is presented in Sect. 2. The enumeration of mechanisms is reviewed in Sect. 2.4. The graph representation of kinematic problems is shown in Sect. 4. The proposed number synthesis algorithm is developed in Sect. 5. The detection of pseudo-isomorphic mechanisms is explained in Sect. 6. The methodology is illustrated with examples for several kinematic tasks.

2 A mechanism identifier

The *graph* $G(V, E)$ of a kinematic chain is obtained by representing each link by a vertex v_i and each kinematic pair by an edge e_{ij} connecting the corresponding vertices $\{v_i, v_j\}$. From now on, we will refer to "graph" indifferently, as the graph representing a kinematic chain.

We call $n = |V|$, the size of the set of vertices (links), and $j = |E|$, the size of the set of edges (joints). The set of unlabeled vertices, $V = \{v_0, v_1, \ldots, v_{n-1}\}$ is numbered in zero base, i.e. $V = \{0, 1, \ldots, n-1\}$. To consider links with physical meaning, they are commonly labeled either by the user or by the solver using integer identifiers (IDs). A function $\mathcal{V}(V)$ labels each vertex of the set V with the integer identifiers $\boldsymbol{l}_L = \{l_0, l_1, \ldots, l_{n-1}\}$, so that $\mathcal{V}(v_i) = l_i$. A function $\mathcal{E}(E)$ is also used to match physical joints with edges. A labeled graph is obtained by labeling of their constitutive sets $G(\mathcal{V}(V), \mathcal{E}(E))$.

A graph has matrix representations. One representation is the vertex-to-vertex *adjacency matrix*. The adjacency matrix A of a graph G is a n-by-n matrix in which entry a_{ij} is the

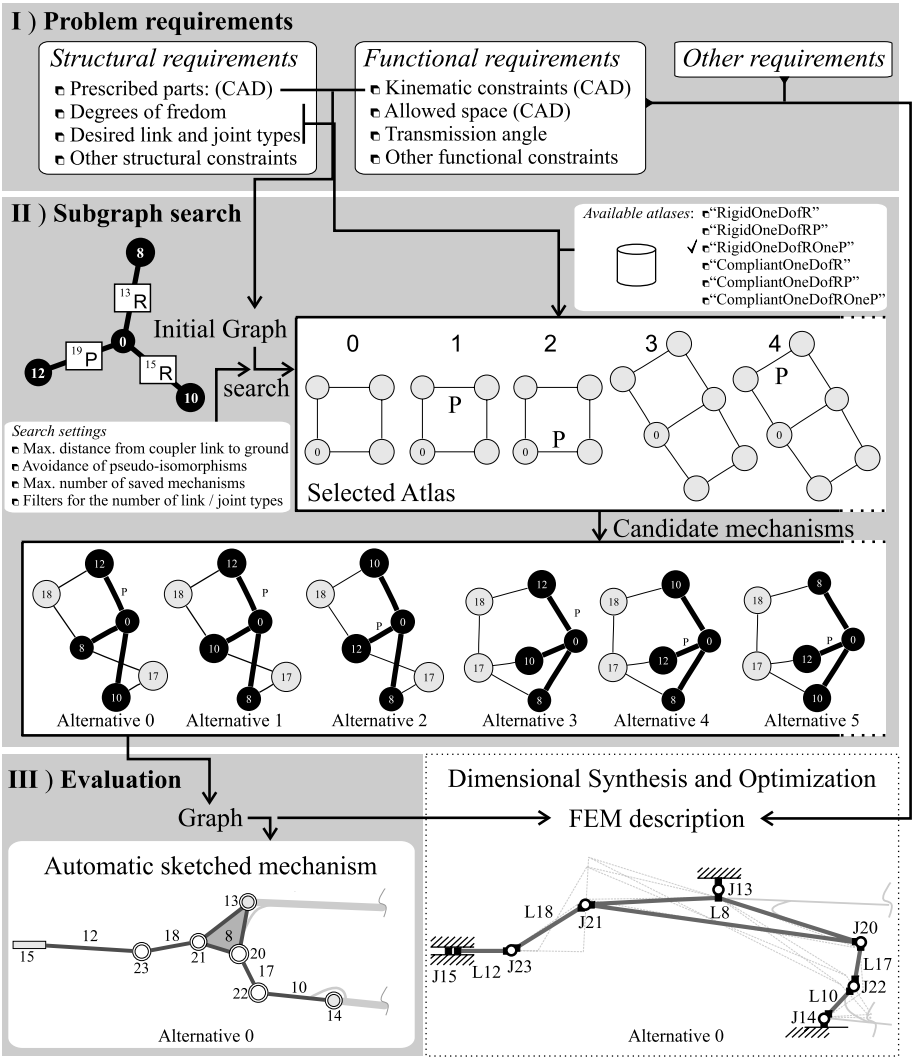

Fig. 2 The proposed type synthesis method

number of edges in G with endpoints $\{v_i, v_j\}$ and $a_{ii} = 0$. Note that permutations of labels change the adjacency matrix A. In other words, it is label order dependent. Two graphs are isomorphic if there is a one-to-one bijection from the vertex set of one graph to the other and edges preserves incidence. Also, two graphs are isomorphic if they share the same adjacency matrix. Many approaches for isomorphism testing were reviewed in Ref. [9]; in this work we will employ the degree code and variations of it.

2.1 Degree code

In order to represent graphs and to detect isomorphisms of graphs, Tang and Liu [15] developed the so-called *degree code*. The *code* (C) of the adjacency matrix of a graph is defined

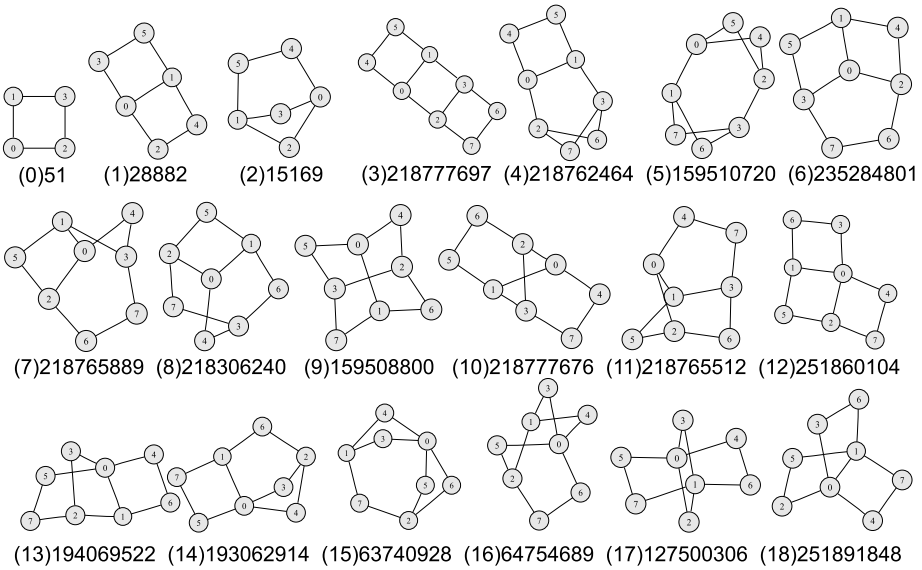

Fig. 3 Labeled kinematic graphs with their respective *level*s and *degree code*s for the atlas of one-DOF kinematic chains with up to 3 independent loops, 8 links, 10 joints

as the decimal value computed by converting the binary string obtained by concatenating the upper triangular elements of the adjacency matrix row by row, excluding diagonal elements. The maximum integer that results from all possible relabellings among those vertices with equal degree is called Degree Code (*DC*).

The degree code is *unique* (two graphs with the same *DC* are isomorphic: $DC(G_1) = DC(G_2) \Leftrightarrow G_1 \cong G_2$) and *decodable* (given the *DC*, we may build the graph). The entire atlas of one-DOF kinematic chain may be very efficiently stored as a sorted list of integers. For each KC, an integer assigning a *level* in the atlas is used to map their respective *degree code* (see Fig. 3).

To arrive to the degree code, we must compute several codes for the defined permutations. For instance, a simple four bar kinematic chain labeled as we see at the top of Fig. 3, has an adjacency matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

and, therefore, can be characterized by the integer 51:

$$DC(A) = \left([110][01][1]\right)_2 = 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^4 + 1 \times 2^5$$
$$= 1 + 2 + 16 + 32 = (51)_{10}.$$

Note that any other permutation of A produces an equal or less code C. Graphs shown in Fig. 3 display their respective degree code labels.

Tang and Liu presented the possibility to take into account *colored edges*. The only change in the degree code algorithm is that the string must be codified in an adequate base,

denoted with b, for the number system. The base is the number of different colors plus one. In our mechanism case, each color represents a joint type, with the type represented by a positive integer number. For example, if the joint is revolute, a color "one" is assigned, while a "two" is assigned for prismatics. Then, the base is the number of different joint types plus one: $b = 2 + 1$.

The diagonal entries in the adjacency matrix A of a KC are null because no *sling edges*[1] are allowed. Also, diagonal entries are permuted to diagonal entries under the same permutation of rows and columns. These two facts allow to easily take into account *colored vertices*, which may represent, e.g. link types. For instance, the "zero" color represents the ground link, while the "one" color indicates a rigid link.

We extend the idea of Tang and Liu to define a *diagonally extended degree code* DC_b^d *in base* b. For example, let A' be a matrix which represents a topology with different types of links and joints. Its code may be computed as follows:

$$A' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 \end{bmatrix} \quad \Longrightarrow \quad C_3^d(A') = \big([0110][101][12][1]\big)_3 = (9034)_{10}.$$

Since all vertices have the same degree, in order to compute the degree code, we need to explore 4! permutations of A'. Among them, $DC_3^d(A')$ is obtained for the permutation $p = \{2, 3, 1, 0\}$, with value

$$DC_3^d(A') = C_3^d\big(p(A')\big) = C_3^d\left(\begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}\right) = 35274.$$

One way of circumventing the computational difficulties that arise when representing a big integer number is to define a more complex but useful identifier, which is formed by the vector of n integers that result by concatenation of the upper triangular elements of the matrix, row by row:

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

Codes comparisons to obtain the degree code are made in lexicographical order: given $C = \{a_0, a_1, \ldots, a_{n-1}\}$ and $C' = \{b_0, b_1, \ldots, b_{n-1}\}$, $C' > (<) C$ if $b_k > (<) a_k$, where k is the first index in $\{0, 1, \ldots, n-1\}$ for which $a_k \neq b_k$. The degree code by rows is obtained for the same permutation of A' with $p = \{2, 3, 1, 0\}$:

$$DC_3^r(A') = C_3^r\big(p(A')\big) = \begin{bmatrix} 48 \\ 10 \\ 4 \\ 0 \end{bmatrix}.$$

In the example, only one comparison is needed ($48 > 12$) to validate that $DC_3^r(A') > C_3^r(A')$.

[1] A sling or self-loop is an edge that connects a vertex to itself.

2.2 Type Adjacency Matrix

Following Murphy, Midha and Howell [10, 11] or Yan [16], we process link types and joint types together in an unique matrix. They respectively defined the *Compliant Matrix* $\boldsymbol{CM}$, and the *Topology Matrix* $\boldsymbol{M}_T$. Murphy et al. proposed the use of the characteristic polynomial $CP(\boldsymbol{CM}, x) = |x\boldsymbol{I} - \boldsymbol{CM}|$ to detect isomorphic mechanisms. However, it has been shown that this mapping is not one-to-one, so we did not use it.

In order to use the *diagonally extended degree code*, we have modified the definition of the entries from that of Murphy et al. to have only positive integer entries, so that we are compatible with the *codability* property of DC_b^r. The *Type Adjacency matrix* $\boldsymbol{T}$ is defined as follows:

$$T_{ii}(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is the ground,} \\ 1 & \text{if } v_i \text{ is a rigid link,} \\ 2 & \text{if } v_i \text{ is a flexible link,} \end{cases}$$

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

Note that this definition could be easily extended to more link and joint types.

We also define two integer mappings,

1. the link types map $\mathcal{T}_L : \mathcal{V}(V) \to \boldsymbol{t}_L$, which maps the labeled vertex set $\mathcal{V}(V)$ to a *link-types vector* $\boldsymbol{t}_L$. For example, let the vertex set be $V = \{0, 1, 2, 3\}$, and their labeled vertex set $\mathcal{V}(V) = \{0, 10, 5, 12\}$ with corresponding link types $\boldsymbol{t}_L = [0, 1, 1, 2]$. Then, the map of link types is

$$\mathcal{T}_L\big(\mathcal{V}(V)\big) = \{0 \to 0, 10 \to 1, 5 \to 1, 12 \to 2\}.$$

   This means that the link with ID 0 is the ground, then links 10 and 5 are rigid, and link 12 is flexible.
2. the joint types map $\mathcal{T}_J : \mathcal{V}(V) \times \mathcal{V}(V) \to \boldsymbol{t}_J$, which maps pairs of labeled connected vertices $\mathcal{V}(V) \times \mathcal{V}(V)$ to a *joint-types vector* $\boldsymbol{t}_J$. For example, a possible joint type map for a given $\boldsymbol{t}_J = [1, 2, 4, 1]$ could be:

$$\mathcal{T}_J\big(\mathcal{V}(V) \times \mathcal{V}(V)\big) = \big\{(0, 10) \to 1, (0, 5) \to 2, (10, 12) \to 4, (5, 12) \to 1\big\},$$

   meaning that the joint between links 0 (ground) and 10 is a revolute joint (type 1), etc.

   The resulting type adjacency matrix of the example considered in Fig. 4 is:

$$\boldsymbol{T} = \begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix}.$$

Note that it can be represented by using the adjacency matrix, formed by zeroes and ones entries, together with the vertex labels $\mathcal{V}(V)$, and the maps $\mathcal{T}_L$ and $\mathcal{T}_J$.

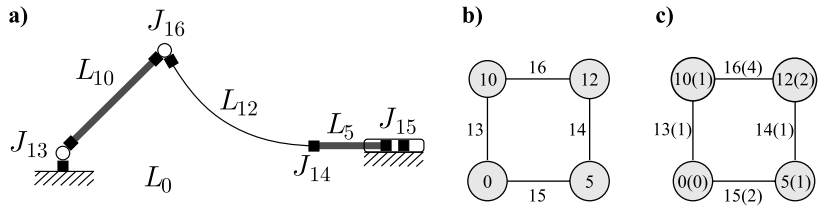

Fig. 4 Mathematical models for a four-bar mechanism: **a** FEM representation; **b** Graph labeled with user's IDs; **c** Graph with link and joint types colors (colored labeled graph)

This form of representation simplifies certain operations. For instance, in order to permute $\boldsymbol{T}$, we permute the adjacency matrix by means of permuting the vertex labels $\mathcal{V}(V)$, but both maps $\mathcal{T}_L$ and $\mathcal{T}_J$ remains unchanged. In this way, the definition of these two maps make the identifier computing more efficient, which is extensively used.

2.3 Synthesis adjacency matrix

We will explore the occurrences of the initial graph, denoted as $G_{\text{ini}}(V_{\text{ini}}, E_{\text{ini}})$, inside a given graph taken from an atlas of mechanisms, denoted as $G_A(V_A, E_A)$. The subgraph occurrence $G_{\text{ini}} \subseteq G_A$ and the matching of the link and joint types of each vertex and edge, respectively, means that G_A is a *structurally feasible* mechanism alternative. However, for a given G_A, there could be many isomorphic occurrences of G_{ini}.

To explore systematically the occurrences $G_{\text{ini}} \subseteq G_A$, we exhaustively compare the vertices of G_{ini} with the subgraph constituted by the first n_{ini} vertices of a $(n_{\text{ini}} - 1)$-permutation of the n_{A} vertices of G_A, and then we make the comparisons related to edges. We denote as $G_A^p(V_A^p, E_A^p)$ to each permuted graph. By definition, the degree code of $\boldsymbol{T}(G_A^p)$ is unique for any permutation p.

We need to make some modification in the $\boldsymbol{T}(G_A^p)$ matrix in a way to identify all non-isomorphic locations of the subgraph G_{ini}. For this purpose, we construct the *synthesis adjacency matrix* ($\boldsymbol{S}$). It has the same definition given to the type adjacency matrix $\boldsymbol{T}$ of G_A^p for synthesized parts and joints, but differs for entries of prescribed parts, i.e. links represented by vertices $v_i \in V_A^p \cong V_{\text{ini}}$. So, we assign different integers to diagonal entries corresponding to each prescribed vertex without regard of its link type. Thus, we consider each prescribed part as *functionally different*:

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

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

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 $\boldsymbol{T}(G_A)$. The number of colors to codify $\boldsymbol{S}$ is the number of prescribed links plus the number of colors of the graphs taken from the atlas, i.e., $n_{\text{ini}} + b$.

2.4 Enumeration of one-DOF kinematic chains

A linkage with n links connected by j simple joints is denoted as a (n, j)-linkage. For a simple-jointed linkage with one degree of freedom $F = 1$, the Grübler movability criterion

Table 1 Number of enumerated non-isomorphic one-DOF kinematic chains

n	j	L	#KC
4	4	1	1
6	7	2	2
8	10	3	16
10	13	4	230
12	16	5	6856
14	19	6	318126
		Total:	325231

asks that $F = 3(n-1) - 2j = 1$. From graph theory, the number of independent loops L could be computed as $L = j - n + 1$.

Methods given by Hwang [17], Hsieh [18], Tsai [3], and Tuttle [19] among others (see Ref. [9]), have allowed to enumerate 325, 231 non-isomorphic one-DOF kinematic chains, see Table 1.

We followed the method found in Ref. [3], to enumerate the first 19 kinematic chains using the degree code as mechanism identifier (see Fig. 3). These kinematic chains were found using graph theory where the feasible graphs have the following characteristics: (i) the minimal vertex degree is 2 ($d(v_i) \geq 2$), i.e. edges are simple, each edge connects only two vertices; (ii) all graphs have no articulation points or bridges; and (iii) partially locked chains/subchains and non-planar graphs were excluded.

3 Generation of atlases of mechanisms

In the proposed method, we can compute the number synthesis in one step since atlases are already computed and stored on disk (see Fig. 2(II)). The user can select an atlas of mechanisms with the desired characteristics for their solutions. Also, the constraints imposed by the prescribed parts—existing joints and links—must be compatible with the selected atlas.

In this section, we will give an alternative methodology to that presented by Murphy et al. [10] for obtaining atlases of mechanisms by assigning link types and joint types on a four-bars kinematic chain. This process is also known as specialization of mechanisms [2]. In this paper, we specialize each kinematic chain of the atlas of Fig. 3 in all non-isomorphic ways. The previously defined tools, the type adjacency matrix and the diagonally extended degree code computed by rows, are extensively used for identifying and codifying the mechanisms. The more different joint/link types are incorporated in the atlas, the more different constraints, and therefore problems, the method will support.

From the list of stored kinematic chains (KCs), we take a degree code and then we retrieve its adjacency matrix A, obtaining a n-vertices and j-edges graph with degree code labeling. The link types are arranged in *link-types vectors* $\boldsymbol{t}_L$ with n entries. Joint types are arranged in *joint-type vectors* with j entries. Then, we specialize the KC in two steps:

Link specialization: specialize all the link-types vectors $\boldsymbol{t}_L$ with a given alphabet (e.g. $\{0 = \text{ground}, 1 = \text{rigid}\}$) and satisfy properly designed link constraints; attach sequentially each $\boldsymbol{t}_L$ on the diagonal entries of the adjacency matrix A constructing the matrix $\boldsymbol{T}'$, compute its DC_b^r and save those which are different. This procedure may result in many non-isomorphic $\boldsymbol{T}'$s. Each of them are used in the second step.

Joint specialization: specialize all joint-types vectors $\boldsymbol{t}_J$ for the given alphabet (e.g. $\{1 = \text{revolute}, 2 = \text{prismatic}\}$); attach each $\boldsymbol{t}_J$ on the corresponding non-null off-diagonal elements of a sequentially selected $\boldsymbol{T}'$ configuring the matrix $\boldsymbol{T}$, check the joint constraints and, if they are satisfied compute its DC_b^r; if it is not yet stored it will be a mechanism of the atlas.

An instance of the first step for a four-bar KC could be:

$$\left(\boldsymbol{t}_L = [0,1,1,2] \wedge A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \right) \longrightarrow \boldsymbol{T}' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \longrightarrow DC_5^r(\boldsymbol{T}'),$$

where the symbol "$\wedge$" means the operation of attaching the link types to the adjacency matrix. Then, for the computed matrix $\boldsymbol{T}'$ the second step is:

$$\left(\boldsymbol{t}_J = [1,2,4,1] \wedge \boldsymbol{T}' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \right) \longrightarrow \boldsymbol{T} = \begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 0 & 4 & 1 & 2 \end{bmatrix}$$

$$\longrightarrow DC_5^r(\boldsymbol{T}) = \begin{bmatrix} 355 \\ 26 \\ 7 \\ 0 \end{bmatrix}.$$

Note that in both steps, after computing $DC_b^r(\boldsymbol{T})$, we check isomorphisms by comparing such codes. For instance, in the second step of the example above, the joint type vector $\boldsymbol{t}_J = [2,1,1,4]$ produces the same $DC_5^r(\boldsymbol{T})$ than vector $\boldsymbol{t}_J = [1,2,4,1]$, so both solutions are isomorphic.

Finally, in case the entry $\boldsymbol{T}_{11}$ be distinct from zero, indeed store $\boldsymbol{T}$, we save a permutation of it, $p(\boldsymbol{T})$ in such a way the first vertex on it has link type zero, i.e. coincident with the ground. This additional restriction allows to reduce the number of permutations needed in the subgraph search.

3.1 Atlas of 1-DOF rigid linkage mechanisms: linkage inversions

The kinematic chains shown in Fig. 3 could be considered mechanisms if the vertex labeled with zero is taken as ground. However, any other link of the KC could have been taken as ground, thus obtaining a mechanism with an eventually new kinematic behavior. Each case for which a new behavior is obtained is known as *linkage inversion*.

The atlas of 1-DOF rigid linkage mechanisms is derived using only link types specialization, since we assumed all joints are of the revolute type (type 1). Therefore, only the first step of specialization is needed here: we need to generate all link-types vectors $\boldsymbol{t}_L$ for an alphabet of two numbers $\{0 = \text{ground}, 1 = \text{rigid}\}$, constrained by a single vertex labeled as zero to be the ground.

For each n-bars KC, each vector of link types $\boldsymbol{t}_L = [t_0, t_1, \ldots, t_{n-1}]$ is attached to the diagonal elements of A and the isomorphism with previously generated $\boldsymbol{T}$'s is checked.

Starting from the atlas of KCs shown in Fig. 3, we can obtain 77 non-isomorphic mechanisms by linkage inversion (see the first row in Table 2). These results coincide with those presented by Tuttle [19], who also obtained 4,351,450 inversions from the 325,231 enumerated one-DOF KCs.

Table 2 Rigid one-DOF linkage atlases (RigidOneDof)

Suffix	BKC Level									
	0	1	2	3	4	5	6	7	8	9
R	1	2	3	2	4	2	5	4	6	2
RP	10	200	232	2,048	2,464	736	4,160	4,096	4,224	1,072
RPrules	7	91	112	564	749	263	1,170	1,192	1,332	381
ROneP	3	13	17	22	34	12	47	44	50	13

BKC Level									Total
10	11	12	13	14	15	16	17	18	
2	8	5	8	7	4	7	3	2	77
1,152	8,192	4,128	8,192	4,864	1,460	4,864	1,312	816	54,222
352	2464	1,180	2,328	1,532	491	1,508	428	247	16,391
16	88	45	88	65	24	65	20	13	679

3.2 Atlas of 1-DOF rigid linkage mechanisms with prismatic joints

In the previous atlas, we assumed all joints are of the revolute type. In order to add prismatic joints, the links specialization is achieved identically as before (it results in 77 alternatives), but for joints specialization we extend the joints alphabet to $\{1 = \text{revolute}, 2 = \text{prismatic}\}$.

The resultant atlas have 54, 222 mechanisms (see the second row in Table 2). However, it is well known that in RP-mechanisms, prismatic joints behavior introduces singularities depending on the number and orientation of prismatic joints over a link/circuit. Some topologies can be discarded while the enumeration takes place. This avoids working with an "a priori identifiable" unfeasible topology.

In order to avoid singularities, Sardain [7] used a rule developed by Freudenstein and Maki [12] for P-joints assignment, "**F&M**: The maximum number of prismatic joints should be equal to one per link, except at the ground and the effector level". Nieto [20] listed three restrictions: "**N1**: No link of a chain can contain only P pairs which directions are parallel"; "**N2**: Binary links of a chain with only P-pairs cannot be connected directly"; "**N3**: No closed circuit of a chain can have less than two R pairs". These rules are heuristic, and not fully compatible. In fact, because of the application for which the rules where designed, rule **F&M** is more restrictive than **N1**, **N2**, and **N3**. Further, rule **N1** can be computed only after the dimensional synthesis stage since it is metric dependent. Inspired on these rules, we propose the following ones:

R1: No closed circuit of a chain can have less than two non-prismatic pairs. By "non-prismatic pair", we mean other one-DOF connection, either revolute (rigid) or revolute or clamped (both in the flexible sense).

R2: No closed circuit of a chain can have three consecutive P pairs.

Applying these restrictions in the second step of specialization of the mentioned 19 KCs, we get a total of 16,391 non-isomorphic mechanisms.

Often, the degree of freedom of the mechanism is obtained by the actuation of one prismatic joint as driver (for instance, by means of a hydraulic or pneumatic cylinder). So, another atlas was developed for only one prismatic joint. As is shown in the last row of Table 2, the assignment results in 679 non-isomorphic mechanisms.

Table 3 Compliant one-DOF linkage atlases (CompliantOneDof)

Suffix	BKC Level			
	0 (Four-bars)	1 (Watt)	2 (Stephenson)	Total
R	211	50,267	52,507	102,985
RP	731	448,673	459,482	908,886
RPrules	683	385,218	396,328	782,229
ROneP	506	178,845	183,623	362,974

3.3 Atlas of 1-DOF compliant linkage mechanisms

Here, the link alphabet is $\{0 = \text{ground}, 1 = \text{rigid}, 2 = \text{flexible}\}$, and the joint alphabet is $\{1 = \text{revolute}, 2 = \text{prismatic}, 3 = \text{flexible\_hinge}, 4 = \text{clamped}\}$.

We have applied two rules defined by Murphy [10].

M1: At least one rigid segment must be present (the ground).
M2: Two or more rigid links (including the ground) cannot be connected by a clamped connection.

We can see in Table 3 that the number of solutions grows considerably (as a consequence of the so-called combinatorial explosion). The results for the four and six-bar mechanisms without prismatic joints are shown in the first row. Results in the second row, allowed the use of prismatic joints without any restriction. In the third row, the mentioned rules for prismatic pairs were considered. Finally, the last row displays the number of solutions obtained when only one prismatic joint per mechanism was permitted.

We validated part of our atlases with that presented by Howell in Appendix G of his book [11]. Note, however, that he displayed only 209 solutions for the four-bars compliant enumeration instead of 211. We have detected that he missed two solutions with *CP*'s $x^4 - 11x^2 + 9$ and $x^4 - 8x^2 + 2$, respectively (represented here using his notation). The corresponding matrices (using again Howell's notation) are:

$$\begin{bmatrix} -1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \\ 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} -1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

3.3.1 Some comments about efficiency

The presented method for specialization of kinematic chains is easy to program and also, well adapted to the presented identifier. In contrast, it is not the most appropriate from the point of view of computational efficiency. This method requires to explore a large number of unfeasible mechanisms, and the cost grows considerably with the number of links. For instance, in the specialization of the compliant four-bars KC without prismatic joints, there were 6 solutions in the first step and 81 in the second one, so that, $6 \times 81 = 486$ alternatives were explored from which 211 were non-isomorphic. In the Watt six-bars KC, there were $50,267$ non-isomorphic mechanisms from the $52 \times 2,187 = 113,724$ generated ones. In the Stephenson six-bars KC, there were $52,507$ non-isomorphic ones from the $68 \times 2,187 = 148,716$ generated alternatives. Finally, $256 \times 59,049 = 15,116,544$ alternatives were generated for the first 8-bars KCs (not shown in Table 3).

The method presented by Yan [2] based on group theory, generates non-isomorphic mechanisms directly from the feasible space and could lead to more efficient ways of generating atlases. Nevertheless, we should remark that this cost does not affect the speed of computations of the synthesis process, since the atlases generation is performed only once, and the results are stored for exploration during the synthesis of the intended mechanism, which is, in fact, the purpose of our work.

4 Graph representation of kinematic problems

Starting from functional requirements, the designer selects the *structural characteristics* to draw the existing parts like a *skeleton diagram*.

A mechanism is represented internally in a multiple-set of data $\mathcal{M} = \{\boldsymbol{N}, \boldsymbol{F}, \boldsymbol{E}\}$, consisting of nodes, fixations and elements with attributes like positions, type of elements, connectivities, etc. [21]. In a synthesis problem, the *skeleton diagram* results in only some parts that are imposed to exist in the final synthesized mechanism. In addition, on the imposed parts the user specifies requirements for the synthesis task: *motion constraints*, allowed space, minimum and maximum transmission angle, etc.

In planar problems, we can impose three motion 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.

The subgraph search process is general and suits to a wide range of problems; nevertheless, we can distinguish three typical cases:

Path Following (PF). To define a trajectory, we give a set of node displacements

$$\mathcal{D} = \left\{N_{ID}, j, (d_x, d_y)_j; \ldots\right\},$$

where ID is the node identifier, j is the passing point number in the sequence of precise positions, and $(d_x, d_y)_j$ are the passing point displacements expressed in relative coordinates from the initial node position. For instance, in Fig. 5, if two displacements are desired on node N_4, we declare three triplets

$$\mathcal{D} = \left\{N_4, 0, (0, 0); N_4, 1, \boldsymbol{d}_1; N_4, 2, \boldsymbol{d}_2\right\}.$$

Rigid-Body Guidance (RBG). Here, displacements and also orientations are defined for an isolated node. For instance, we declare the displacements

$$\mathcal{D} = \left\{N_5, 0, (0, 0); N_5, 1, \boldsymbol{d}_1; N_5, 2, \boldsymbol{d}_2\right\}$$

on node N_5, and the rotations for rigid-body E_1 (Fig. 6) with the triplets

$$\mathcal{L} = \{E_1, 0, 0; E_1, 1, \alpha_1; E_1, 2, \alpha_2\}.$$

Function Generation (FG). Now, two (or more) sequences of displacements or orientations are specified for two (or more) rigid-bodies. For instance, a law $\boldsymbol{\beta} = f(\boldsymbol{\alpha})$ is given for two bodies hinged to ground in Fig. 7, by defining the sets

$$\mathcal{J} = \{E_1, 0, 0; E_1, 1, \alpha_1; E_1, 2, \alpha_2; E_1, 3, \alpha_3\}$$

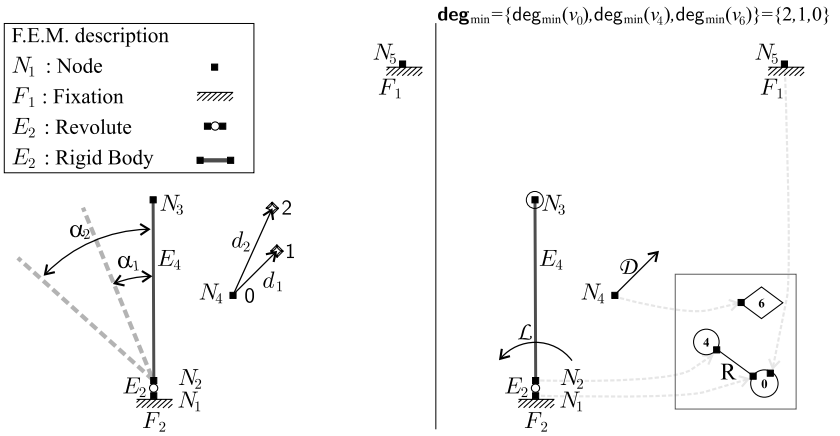

Fig. 5 Path following

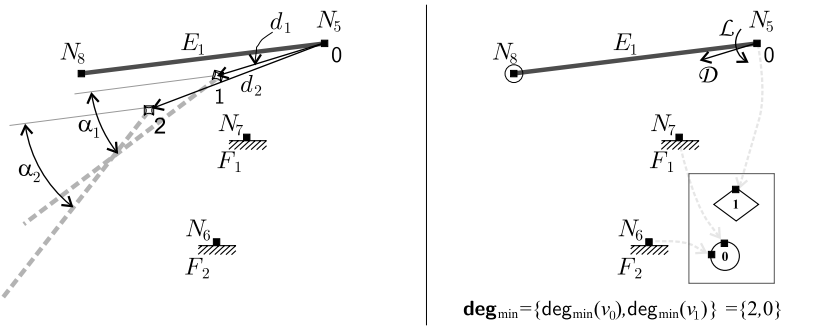

Fig. 6 Rigid-body guidance

and

$$\mathcal{J}=\{E_2,0,0;\ E_2,1,\beta_1;\ E_2,2,\beta_2;\ E_2,3,\beta_3\}.$$

Motorized prismatic joints can also be present in a mechanism, and in this case input displacements may be given for them. For instance, in Fig. 1, a *double function generation* problem is defined where the objective is to move both rigid bodies in a synchronized mode using a prismatic actuator. The initial and last positions are prescribed for the prismatic joint E_3 in the form $\mathcal{J}=\{E_3,0,d_0;\ E_3,3,d_3\}$. The intermediate inputs are unknown and are computed by the dimensional synthesis program. Data is completed by imposing the angular displacements at link E_1: $\mathcal{J}=\{E_1,0,0;\ E_1,1,\beta_1;\ E_1,2,\beta_2;\ E_1,3,\beta_3\}$ and at link E_2: $\mathcal{J}=\{E_2,0,0;\ E_2,1,\alpha_1;\ E_2,2,\alpha_2;\ E_2,3,\alpha_3\}$.

In this way, the task desired for the synthesized mechanism is entered by specifying sets $\mathcal{D}$, $\mathcal{J}$, and $\mathcal{L}$. The problem of synthesis is then stated as that of finding a mechanism capable of approximately satisfying the prescribed task, minimizing the error between the objective and the generated movements.

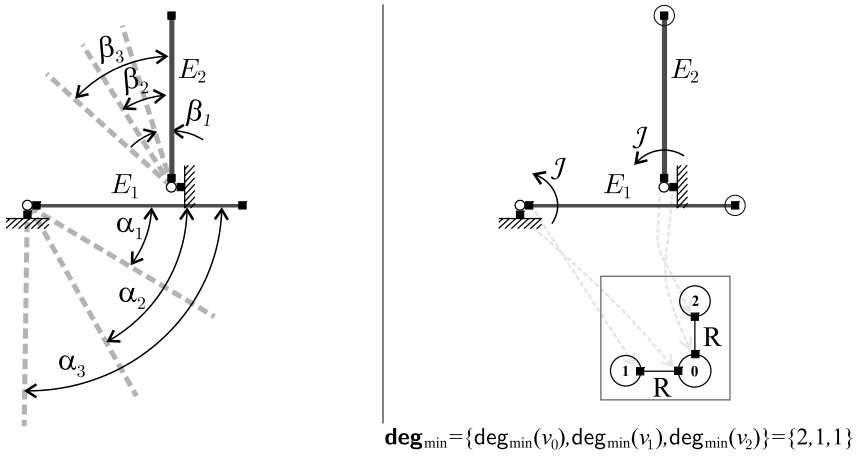

Fig. 7 Function generation

4.1 Initial graph

The *initial graph* represents the initially imposed parts. From the starting parts definition $\mathcal{M}_{\text{ini}}$, we build the associated graph G_{ini} following these simple 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 $\mathcal{M}_{\text{ini}}$) is used to prescribe the degree of vertex zero (ground). For each isolated node with prescribed movement in $\mathcal{M}_{\text{ini}}$, 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).

After construction of the initial graph, a vector called minimum degree of vertices, $\mathsf{deg}_{\min}$ is filled to be used later to accelerate the subgraph search.

Note that nodes which neither pertain to any joint nor have prescribed motion, are filtered out (these nodes are shown as "circled" in Figs. 5–7). 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.

Tasks can be classified according to the number of graph components in the initial graph associated with the problem (e.g. compare Figs. 1, 5, 6, and 7). For PF (Fig. 5) and RBG (Fig. 6) tasks, the initial graph has two separated components. One component includes the ground and the other component has one isolated vertex which is taken as *objective vertex*.[2] If after applying the rules for construction of the initial graph, we get more than two components, the problem is subdivided into successive synthesis tasks, dealing with one

[2]Since the name "coupler or floating link" comes from studies on the traditional four-bar mechanism, in our graph approach we rename it to "*objective vertex*".

objective at each time. For instance, a wing flap/tab coordination problem can be divided into a "flap guidance" problem followed by a "tab guidance" problem [22].

Note that the representation of isolated vertices is not considered in the classical definition of the adjacency matrix A. We simply add them as a null row and column in such a way that A allows the representation of non-connected graphs. Then, $\boldsymbol{T}$ is implicitly defined taking the structure of A and the link and joint type integer mappings, $\mathcal{T}_L$ and $\mathcal{T}_J$.

For example, the adjacency matrix of the initial graph shown in Fig. 5 is:

$$
\begin{aligned}
V_{\text{ini}} &= \{0 \quad 1 \quad 2\}, & E_{\text{ini}} &= \big\{(0,4)\big\} \\
\mathcal{V}(V_{\text{ini}}) &= \{0 \quad 4 \quad 6\}, & \mathcal{E}(E_{\text{ini}}) &= \{2\} \\
A_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\
\mathcal{T}_L\big(\mathcal{V}(V_{\text{ini}})\big) &= \{0 \to 0, 4 \to 1, 6 \to 1\}, \\
\mathcal{T}_J\big(\mathcal{V}(V_{\text{ini}}) \times \mathcal{V}(V_{\text{ini}})\big) &= \big\{(0,4) \to 1\big\}, \\
\boldsymbol{T}_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
$$

This new definition of the adjacency matrix is the key aspect to deal with a general graph and subgraph isomorphism problem.

4.1.1 Distance from the objective vertex

The objective vertex v_{obj} in PF and RBG problems, i.e. the vertex containing the node which develops the prescribed task, is chosen counting a given distance from the ground v_0. This distance $d(v_i, v_j)$ is defined as the minimal number of vertices going through a path beginning from the objective vertex to the ground: $\min d(v_0, v_{\text{obj}})$. The minimum distance from the objective vertex to the ground is set to 2, while the maximum is either fixed by the user or taken to be equal to the number of passing points n_{pp} specified in the task minus one.

$$
2 \leq \min\big(d(v_0, v_{\text{obj}})\big) \leq n_{\text{pp}} - 1. \tag{1}
$$

For instance, if three passing points are specified for a PF task, the vertex is chosen at a distance of two from the ground. Of course, depending on the topology, there can be more than one option for the objective vertex that verifies this requirement (although some of them can be isomorphic). The subgraph search proposed in the next section explores automatically all possible ways of assigning the objective vertex that satisfies the distance constraint and avoids isomorphisms.

5 Number synthesis by means of a subgraph searching

The initial graph represents the initial situation. In order to get a mechanism that matches this initial situation, the initial graph should be a subgraph of any valid mechanism of the

selected atlas of mechanisms. The problem consists in looking for the simplest mechanism in the atlas for which the initial graph is a subgraph:

$$G_{\text{ini}} \subseteq G_A \tag{2}$$

with $G_{A,}$ a graph from the atlas. However, the following constraints have to be also satisfied:

- the equality constraint

$$\boldsymbol{T}(G_{\text{ini}}) = \boldsymbol{T}(H_A), \quad G_{\text{ini}} \cong H_A,\ H_A \subseteq G_A, \tag{3}$$

  i.e. the link and joint types in G_{ini} should match exactly those of the corresponding subgraph, H_A, in G_A.
- the distance constraint given by (1),
- the isomorphism constraint, $DC_b^d(\boldsymbol{S}(G_{\text{ini}}, G_A))$ be different from all previous answers, and
- the pseudo-isomorphism constraint, no solution has a previous one as subgraph (explained later).

Let $\boldsymbol{l}_{\text{ini}} = \{l_0, l_1, \ldots, l_{n_{\text{ini}}-1}\}$ be the labels of the initial graph G_{ini}, and $\mathcal{V}_{\text{ini}}$ the function which applies the set $\boldsymbol{l}_{\text{ini}}$ to the set of unlabeled vertices $V_{\text{ini}} = \{v_0, v_1, \ldots, v_{n_{\text{ini}}-1}\} = \{0, 1, \ldots, n_{\text{ini}} - 1\}$, so that $\mathcal{V}_{\text{ini}}(v_i) = l_i$. Also, let us consider the function $\mathcal{E}_{\text{ini}}(e_{ij}) = m_{ij}$ for labeling the edges of G_{ini}. These labels were assigned by the user when the problem was defined.

The search begins at the lowest level of complexity in the selected atlas. In this way, we try to *minimize* the number of links in the solution and get the simplest possible mechanism. The algorithm for selection of a suitable mechanism is the following:

S0. Initialize search level $A = -1$, and the number of alternatives $a = -1$.

S1. Increase level index A and take a graph G_A from the atlas, with n_A vertices. If $n_{\text{ini}} \leq n_A$, continue to **S2**; otherwise, repeat **S1**.

S2. Do a $(n_{\text{ini}} - 1)$-permutation of the vertex set of G_A, V_A, excluding vertex v_0 which is always present in both G_{ini} and G_A. We call p' this set of $V_A \setminus v_0$. A special permutation vector p is formed by concatenating v_0, the permutation p', and the remaining vertices of V_A which are not in p' sorted in ascending order, i.e.:

$$p = \Big\{\underbrace{v_0, v_1^{p'}, \ldots, v_{n_{\text{ini}}-1}^{p'}}_{V(H_A)},\ \underbrace{\ldots, v_{n_A-1}}_{v_i < v_{i+1}, v \ni p'}\Big\}.$$

  Do a permutation of G_A using p. Take a subgraph H_A of the permuted graph G_A^p in the following way: the vertices of H_A are composed by the first n_{ini} vertices of the permuted list of vertices of G_A^p, and connect H_A as in G_A^p. Label H_A using $\mathcal{V}_{\text{ini}}$ and $\mathcal{E}_{\text{ini}}$. There is a number of $\binom{n_A-1}{n_{\text{ini}}-1}(n_{\text{ini}} - 1)!$ permutations p' to be explored following the lexicographic order of the unlabeled vertices. If all the $(n_{\text{ini}} - 1)$-permutations have already been tested in the explored level, return to **S1**.

S3. If edges of G_{ini} are included in H_A and also the link and joint types match, then $G_{\text{ini}} \subseteq G_A^p$; else, return to **S2** and choose a new subgraph H_A in lexicographic order of p' permutations.

S4. Relabel G_A^p using the labels functions $\mathcal{V}_{\text{ini}}$ and $\mathcal{E}_{\text{ini}}$ for vertices and edges homologous to G_{ini} in the previous step, and for vertices and edges which are not in G_{ini} using new

labels starting from

$$\max(l_i, m_{ij}) + 1, \quad i, j = 0, 1, \ldots, n_{\text{ini}},$$

we denote the new label functions as $\mathcal{V}_{a+1}$ and $\mathcal{E}_{a+1}$.
If there is an objective vertex and the distance constraint is violated return to **S2** and choose a new subgraph H_A in lexicographic order. Else, construct the matrices $\boldsymbol{T} = \{\boldsymbol{l}_L \wedge \boldsymbol{l}_J \wedge G_A^p\}$ and $\boldsymbol{S}$ from the labeled G_A^p and compute its $DC_b^d(\boldsymbol{S})$. If the code is not already stored, set $a \leftarrow a + 1$ and save a new alternative, $\mathcal{M}_a \leftarrow [C_b^d(\boldsymbol{T}), \mathcal{V}_a, \mathcal{E}_a]$, then return to **S2**. Else, return to **S2** and choose a new subgraph H_A in lexicographic order.

In this way, we performed the *number synthesis* and the *specialization* process, that is, we determined the mechanism (degrees of freedom, number of links, number of joints and their interconnections, and the type of links and joints) that could answer the requirements. All this information is codified in the n_A-vector $C_b^d(\boldsymbol{T})$. In addition, the vertex and edge labels $\mathcal{V}_a$ and $\mathcal{E}_a$ are saved to retrieve the physical meaning of the links and joints.

At the end, the process results in a list of mechanisms with admissible topologies $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$, …, where the mechanism $\mathcal{M}_0$ with labeled and colored graph G_0 is the simplest admissible solution. These mechanisms inherit synthesis data definitions ($\mathcal{D}$, $\mathcal{J}$, $\mathcal{L}$) from $\mathcal{M}_{\text{ini}}$, which are useful to compute the missing data (i.e. vertices representing new links have unknown node positions) at later dimensional synthesis stages [23].

Applications of the subgraph search algorithm are illustrated next with practical examples.

Example 1: Path following with prescribed timing

A subgraph search in an atlas of rigid mechanisms was executed for the problem shown in Fig. 5.

From the available atlases of rigid mechanisms (see Table 2), we select the atlas RigidOneDofR with 77 candidates to explore.

Because the problem has an objective vertex with 3 passing points, the allowed distance from the objective vertex to ground was automatically set to take the value 2 by the constraint defined in (1). This search lead to 489 non-isomorphic solutions of increasing complexity for this problem. A second running in which pseudo-isomorphic mechanisms were rejected (see Sect. 6), resulted in 214 solutions. The first 19 solutions found in the latter case are shown in Fig. 8.

Figure 9 displays their corresponding sketches (they were generated automatically with manual relocation of some nodes to avoid crossing edges). We can see that solutions result in increasing order of complexity as they come out from the selected atlas. Alternative 0 is a four-bars mechanism, 1 is a Watt-I mechanism; 2 and 3 are Watt-II mechanisms; 4, 5 and 6 are Stephenson-I mechanisms; 7 to 10 are Stephenson-II mechanisms; and 11 to 14 are Stephenson-III mechanisms.

Example 2: Double function generation

The problem shown in Fig. 1 schematizes the prescribed coordination between the horizontal movement of an hydraulic cylinder and the rotations of two flaps of a turbine engine following a prescribed α vs. β law between them. The primary flap is displayed as body 8, body 10 is the secondary flap and the hydraulic cylinder is displayed as body 12 in the

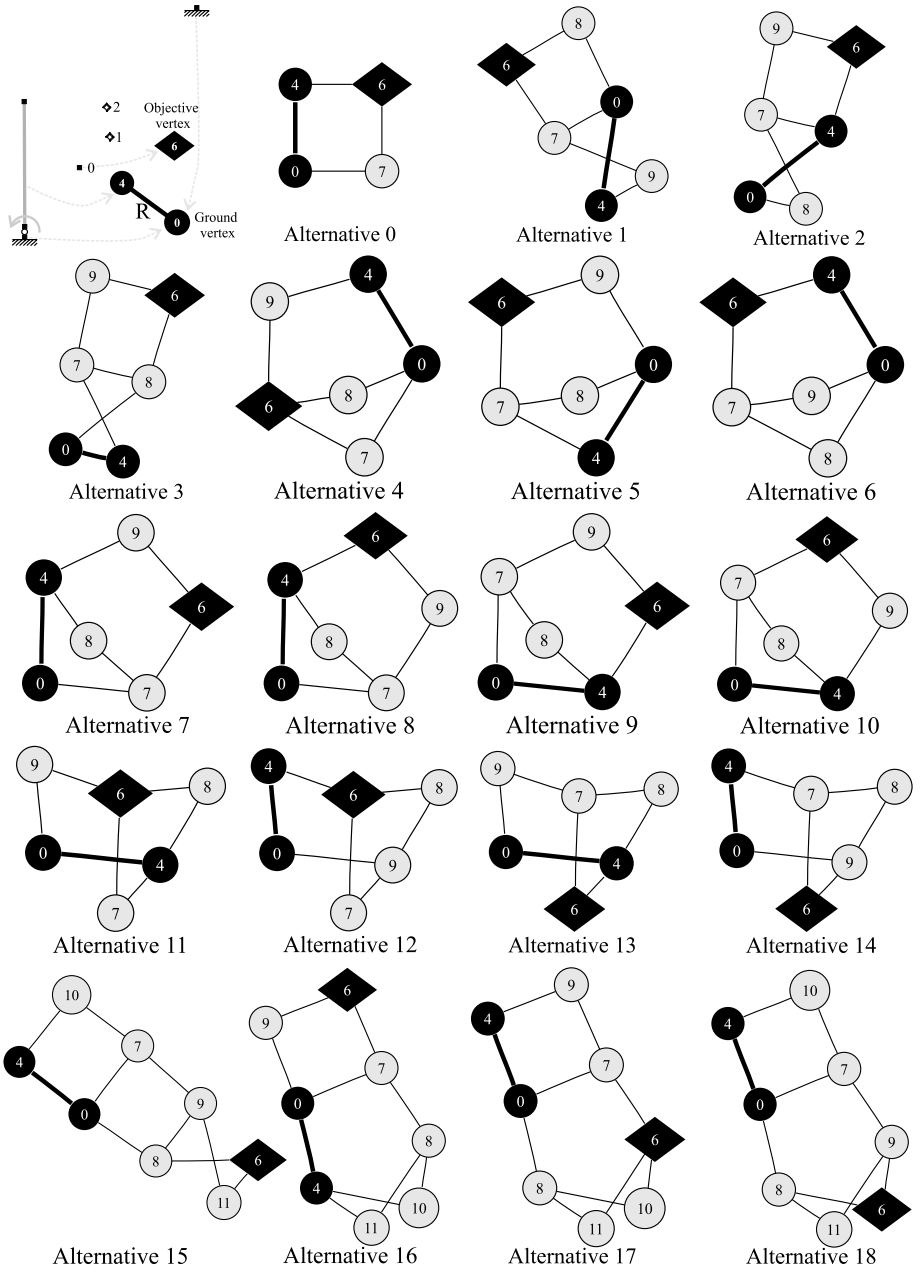

Fig. 8 First 19 non-isomorphic occurrences of an initial graph

figure. Note that in this case there is not any objective vertex, so the distance constraint was not imposed. The selected atlas was RigidOneDofROneP.

The first 10 solutions obtained are shown in Figs. 10 and 11. An automatic sketch is drawn below each solution for clarity.

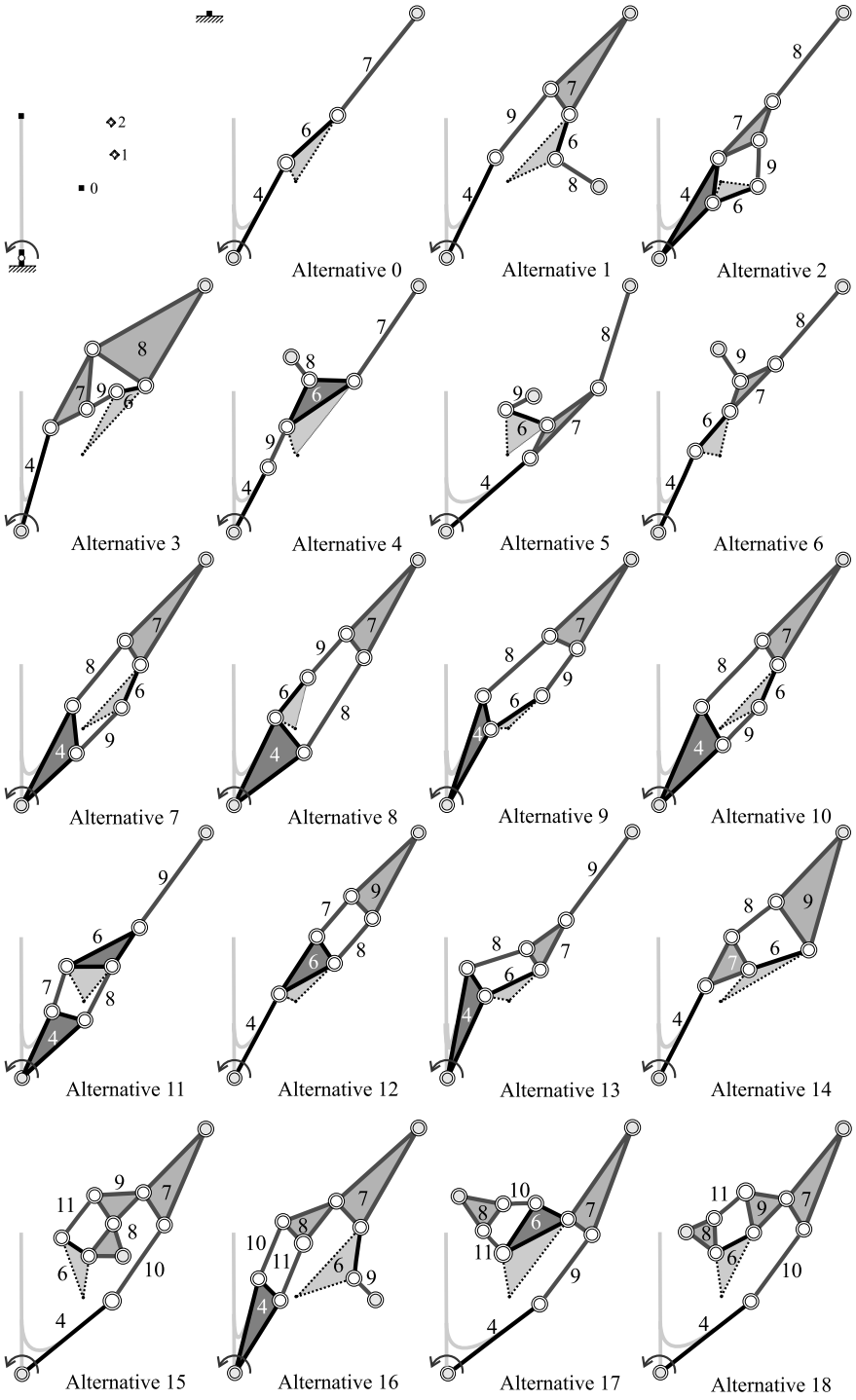

Fig. 9 Physical sketches for a path following task

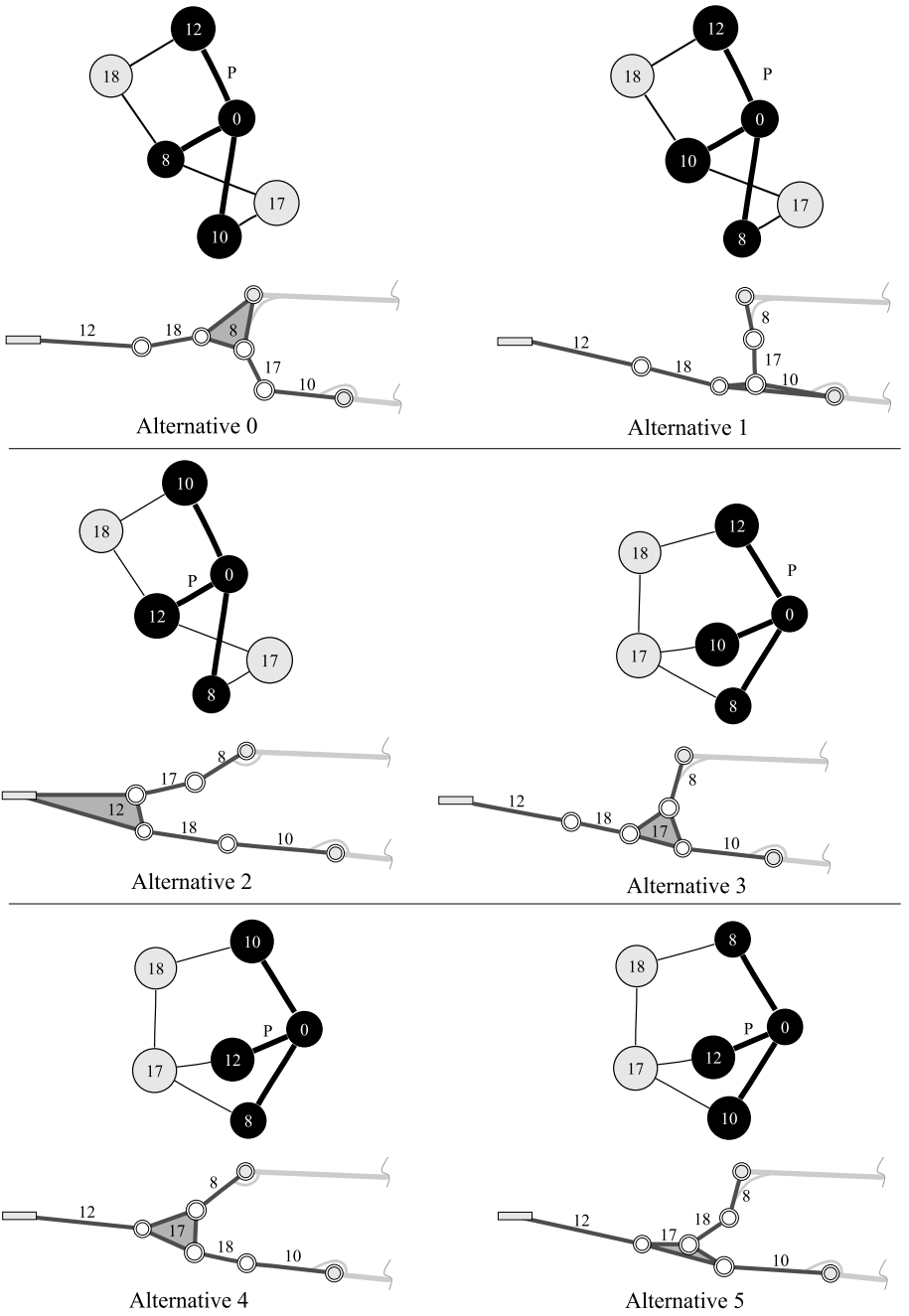

Fig. 10 First 10 non-isomorphic occurrences of an initial graph inside the atlas (continued)

Note that the first two synthesized mechanisms (*Alternatives 0* and *1*) shown in Fig. 2 were found inside the same level in the specialized atlas, i.e. from permutations of the same

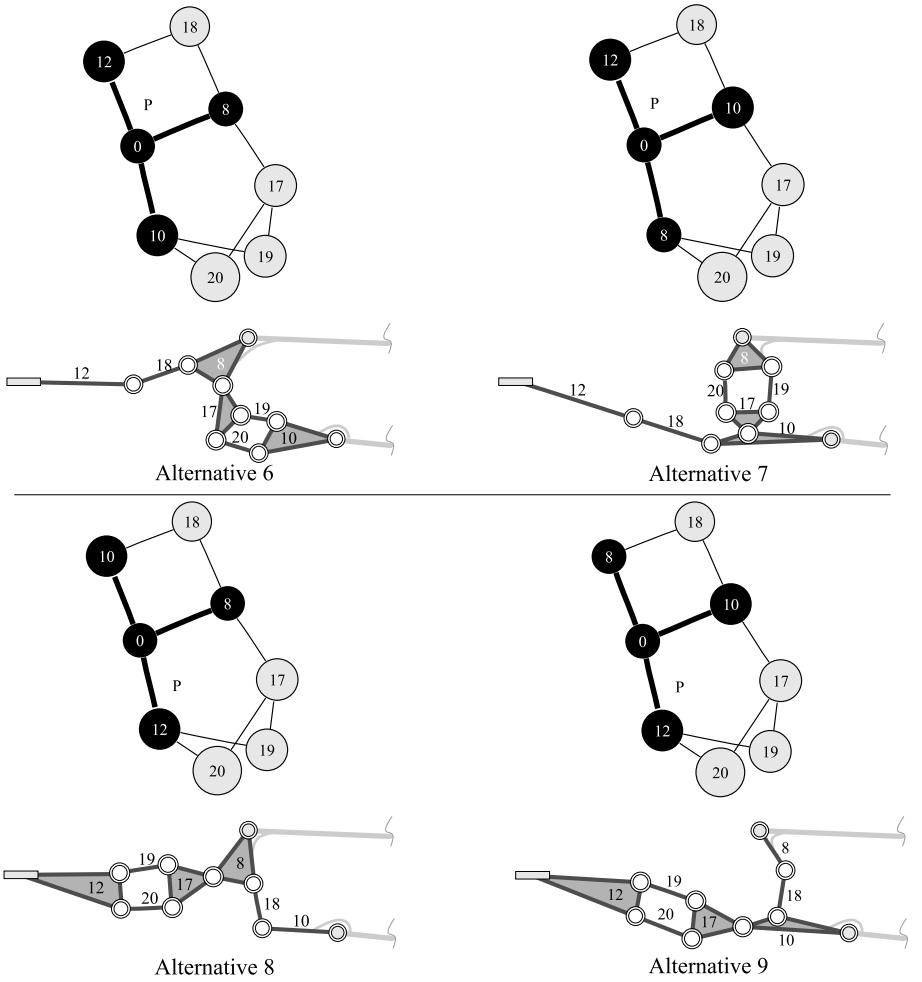

Fig. 11 First 10 non-isomorphic occurrences of an initial graph inside the atlas

Watt-I mechanism (with a prismatic joint between the ternary ground and one of their adjacent binary links). These subgraph occurrences are *structurally isomorphic but functionally different*. These two valid alternatives have the following $\boldsymbol{T}$, $\boldsymbol{S}$ matrices and degree codes

$$\mathcal{V}\left(V_A^{p^0}\right) = \{0 \quad 8 \quad 10 \quad 12 \quad 17 \quad 18\},$$

$$\boldsymbol{T}\left(G_A^{p^0}\right) = \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{S}^0 = \begin{bmatrix} 3 & 1 & 1 & 2 & 0 & 0 \\ 1 & 4 & 0 & 0 & 1 & 1 \\ 1 & 0 & 5 & 0 & 1 & 0 \\ 2 & 0 & 0 & 6 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow DC_7^r\left(\boldsymbol{S}^0\right) = \begin{bmatrix} 70021 \\ 7218 \\ 350 \\ 50 \\ 42 \\ 5 \end{bmatrix}$$

for the first alternative, and

$$\mathcal{V}\left(V_A^{p^1}\right) = \{0 \quad 10 \quad 8 \quad 12 \quad 17 \quad 18\},$$

$$\boldsymbol{T}\left(G_A^{p^1}\right) = \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{S}^1 = \begin{bmatrix} 3 & 1 & 1 & 2 & 0 & 0 \\ 1 & 5 & 0 & 0 & 1 & 1 \\ 1 & 0 & 4 & 0 & 1 & 0 \\ 2 & 0 & 0 & 6 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow DC_7^r\left(\boldsymbol{S}^1\right) = \begin{bmatrix} 86828 \\ 7218 \\ 350 \\ 50 \\ 42 \\ 4 \end{bmatrix}$$

for the second one. The degree code of the $\boldsymbol{S}$ matrices effectively determine if the resultant occurrences leads to functionally different mechanisms.

6 Rejection of pseudo-isomorphic mechanisms

A mechanism has an idle loop if it contains a sub-chain with inactive links which carry no loads. These links increase unnecessarily the complexity.[3]

The algorithm for idle loop detection is simply another subgraph search: In step **S4**, before saving a candidate solution $\mathcal{M}_a$, we do a new subgraph search checking to see if any stored mechanism solution $\mathcal{M}_s$ is a subgraph of $\mathcal{M}_a$, i.e. if there exists an occurrence of $\mathcal{M}_s \subset \mathcal{M}_a,\ s = 0, \ldots, a-1$.

Note that, in the presented examples, no solution has a previous one as subgraph. This is a consequence of the implemented second subgraph search.

For instance, in the path following example, although the second valid subgraph occurrence is that given in Fig. 12 as *Alternative 1'*, it has *Alternative 0* as subgraph so it is rejected. Note that the new grounded link 8 and the new link 9 do not carry any load for the imposed input. By ignoring the parts selected by line A-A, the mechanism is identical to that of *Alternative 0*. The following valid alternative (*Alternative 1*) is that shown in Fig. 9.

[3] In special cases, an idle loop could be desired to produce some particular kinematic or dynamic behavior, e.g. to produce a locked end-position (with zero transmission angle), reinforce the planar stability, or for balancing purposes.

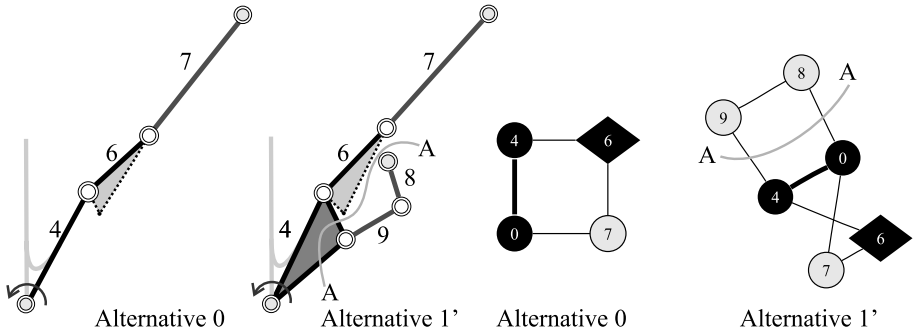

Fig. 12 First pseudo-isomorphic occurrence in the path following example

7 Conclusions

The main contribution of this work to linkage mechanisms design is to offer a systematic procedure to obtain topological alternatives for a given kinematic problem. New algorithms based on graph theory and combinatorial analysis were developed to search and codify the solutions in a non-isomorphic way.

In this work, we incorporated structural characteristics in the alternatives generator by using the initial graph concept. This eliminates the human effort for task matching and also diminished considerably the computation time in the rest stages of synthesis. In the subsequent dimensional synthesis stage, the information of known nodes and type-synthesized graph structure can be used either in a general absolute/natural coordinate formulation [24, 25] or in a complex number formulation based on the decomposition of the topology into open chains [4, 22, 23, 26].

Other remarkable characteristics are:

- The F.E.M. description of the kinematic problem in conjunction with the rules given for the initial graph construction is the key to adapt the problem into a graph problem one. Note that this technique can be easily extended to three-dimensional space.
- The use of an specialized atlas assures that all candidate mechanisms satisfy the required degree of freedom without containing rigid sub-chains, and reduce the time consumed for specialization.
- The number of solutions is finite and the combinatorial explosion is manageable. The method allows the user to look for all solutions for a given planar problem in a selected atlas with a defined number of candidate mechanisms.
- The designed diagonally extended degree code allows coding and decoding of solutions in an efficient and straightforward way.
- The CPU time consumption is quite small, and the examples shown were computed in just a few seconds of CPU time on a modern PC.
- The method is being used as part of a complete system for type and dimensional synthesis of rigid and flexible mechanisms implemented in the context of a finite elements program [27–30].

The algorithms and the identifier are useful for dealing with more types of links and joints in the mechanism, and even more complex tasks. However, adequate rules to reject kinematically invalid solutions must be properly designed, as we have seen for prismatic joints. Although it was not presented in the examples, the method is capable to deal with

compliant members. In future work, we will use the method for exploring the large number of mechanisms available in the atlases of compliant mechanisms.

Acknowledgements 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 from the *European Community* through grant *SYNCOMECS* (SYNthesis of COmpliant MEChanical Systems) project UE FP6-2003-AERO-1-516183.

References

1. Olson, D.G., Erdman, A.G., Riley, D.R.: A systematic procedure for type synthesis of mechanisms with literature review. Mech. Mach. Theory **20**, 285–295 (1985)
2. Yan, H.S., Hwang, Y.W.: The specialization of mechanisms. Mech. Mach. Theory **26**, 541–551 (1991)
3. Tsai, L.W.: Mechanism Design: Enumeration of Kinematic Structures According to Function. CRC Press, Boca Raton (2001)
4. Sandor, G.N., Erdman, A.G.: Advanced Mechanism Design: Analysis and Synthesis, vol. 2. Prentice-Hall, New Jersey (1984)
5. Lin, C.S., Erdman, A.G., Jia, B.P.: Use of compatibility linkages and solution structures in the dimensional synthesis of mechanism components. Mech. Mach. Theory **31**, 619–635 (1996)
6. Erdman, A.G., Sandor, G.N.: Mechanism Design: Analysis and Synthesis, vol. 1, 3rd edn. Prentice-Hall, New Jersey (1997)
7. Sardain, P.: Linkage synthesis: Topology selection fixed by dimensional constraints, study of an example. Mech. Mach. Theory **32**, 91–102 (1997)
8. McCarthy, J.M.: Geometric Design of Linkages. Springer, Berlin (2000)
9. Mruthyunjaya, T.S.: Kinematic structure of mechanisms revisited. Mech. Mech. Theory **38**(4), 279–320 (2003)
10. Murphy, M.D., Midha, A., Howell, L.L.: The topological synthesis of compliant mechanisms. Mech. Mach. Theory **31**, 185–199 (1996)
11. Howell, L.L.: Compliant Mechanisms. Wiley, New York (2001)
12. Freudenstein, F., Maki, E.R.: Creation of mechanisms according to kinematic structure and function. J. Environ. Plan. B **6**, 375–391 (1979)
13. Tsai, L.W.: Systematic enumeration of parallel manipulators. Technical report, Institute for Systems Research, College Park, MD, USA (1998)
14. Chen, D.Z., Pai, W.M.: A methodology for conceptual design of mechanisms by parsing design specifications. ASME J. Mech. Des. **127**(6), 1039–1044 (2005)
15. Tang, C.S., Liu, T.: The degree code—a new mechanism identifier. ASME J. Mech. Des. **115**, 627–630 (1993)
16. Yan, H.S.: Creative Design of Mechanical Devices. Springer, Singapore (1998)
17. Hwang, W.M., Hwang, Y.W.: Computer-aided structural synthesis of planar kinematic chains with simple joints. Mech. Mach. Theory **27**, 189–199 (1992)
18. Hsieh, H.I.: Sistematic methodologies for the automatic enumeration of topological structures of mechanisms. Master's thesis, University of Maryland, USA (1992)
19. Tuttle, E.R.: Generation of planar kinematic chains. Mech. Mach. Theory **31**(6), 729–748 (1996)
20. Nieto Nieto, J.: Síntesis de Mecanismos. Editorial AC, Madrid (1977)
21. Geradin, M., Cardona, A.: Flexible Multi-Body Dynamics. A Finite Element Approach. Wiley, New York (2001)
22. Cardona, A.: Computational methods for synthesis of mechanisms. Technical report, CIMEC-INTEC (2002)
23. Pucheta, M.A., Cardona, A.: 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 (2005)
24. Jiménez, J.M., Álvarez, G., Cardenal, J., Cuadrado, J.: A simple and general method for kinematic synthesis of spatial mechanisms. Mech. Mach. Theory **32**(4), 323–341 (1997)
25. Da Lio, M., Cossalter, V., Lot, R.: On the use of natural coordinates in optimal synthesis of mechanisms. Mech. Mach. Theory **35**(10), 1367–1389 (2000)
26. Pucheta, M.A., Cardona, A.: A decomposition method for modular dimensional synthesis of planar multi-loop linkage mechanisms. In: Mecánica Computacional, XV Congreso sobre Métodos Numéricos y sus Aplicaciones, ENIEF 2006, vol. XXVII, pp. 351–373, Santa Fe, Argentina, November 2006

27. Cugnon, F., Cardona, A., Selvi, A., Paleczny, C.: Synthesis and optimization of flexible mechanisms. In: Bottasso, C.L., Masarati, P., Trainelli, L. (eds.) Multibody Dynamics 2007, ECCOMAS Thematic Conference on Multibody Dynamics, Milan, Italy (2007)
28. Pucheta, M.A., Cardona, A.: Kinematics synthesis of compliant mechanisms using rigid-body replacement. In: Bottasso, C.L., Masarati, P., Trainelli, L. (eds.) Multibody Dynamics 2007, ECCOMAS Thematic Conference on Multibody Dynamics, Milan, Italy (2007)
29. SAMTECH S.A. SAMCEF, http://www.samcef.com
30. Open Engineering S.A. OOFELIE: oriented object finite elements led by interactive executor. http://www.open-engineering.com. University of Liège, Belgium and INTEC, Argentina