
UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas
Instituto de Desarrollo Tecnológico para la Industria Química

# MÉTODOS COMPUTACIONALES PARA DISEÑO Y SÍNTESIS DE MECANISMOS PLANOS

**Martín Alejo Pucheta**

Tesis remitida al Comité Académico del Doctorado
como parte de los requisitos para la obtención
del grado de
DOCTOR EN INGENIERÍA
Mención Mecánica Computacional
de la
UNIVERSIDAD NACIONAL DEL LITORAL

**2008**

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria,
Paraje "El Pozo", S3000, Santa Fe, Argentina.

**UNIVERSIDAD NACIONAL DEL LITORAL**
**Facultad de Ingeniería y Ciencias Hídricas**

Santa Fe, 22 de Abril de 2008.

Como miembros del Jurado Evaluador de la Tesis de Doctorado titulada ***"Métodos computacionales para diseño y síntesis de mecanismos planos"***, desarrollada por el Ing. Martín Alejo PUCHETA, certificamos que hemos evaluado la Tesis y recomendamos que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería – Mención Mecánica Computacional.

La aprobación final de esta disertación estará condicionada a la presentación de dos copias encuadernadas de la versión final de la Tesis ante el Comité Académico del Doctorado en Ingeniería.

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Dr. Oliver BRÜLS | Dr. Adrián CISILINO | Dr. Víctor FACHINOTTI |
| ..... | ..... | |
| Dr. Mario STORTI | Dr. Sergio VERA | |

Santa Fe, 22 de Abril de 2008

Certifico haber leído esta Tesis preparada bajo mi dirección y recomiendo que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería – Mención Mecánica Computacional.

.....
Dr. Alberto CARDONA
Director de Tesis

**Universidad Nacional del Litoral**
Facultad de Ingeniería y
Ciencias Hídricas

Secretaría de Posgrado

Ciudad Universitaria
C.C. 217
Ruta Nacional N° 168 – Km. 472,4
(3000) Santa Fe
Tel: (54) (0342) 4575 229
Fax: (54) (0342) 4575 224
E-mail: posgrado@fich.unl.edu.ar

# Declaración legal del autor

La presente tesis ha sido remitida ante la comisión de Posgrado de la *Universidad Nacional del Litoral (UNL)* como parte de los requisitos para la obtención del grado académico de *Doctor en Ingeniería - Mención Mecánica Computacional*. Una copia de la misma permanecerá depositada en la biblioteca de la *Facultad de Ingeniería y Ciencias Hídricas (FICH)*; su consulta queda supeditada a la normativa legal vigente por el reglamento de la mencionada biblioteca.

Para fines educativos y de investigación, no se requiere autorización especial alguna en lo concerniente a reproducción, copia, distribución o citaciones de la presente tesis.

Cualquier sugerencia o comentario, ya sea de contenido conceptual como de implementación algorítmica, sobre los tópicos desarrollados y discutidos en este trabajo de investigación, será altamente apreciada.

Martín Alejo Pucheta

# Dedicatoria

A mi esposa María Laura,
y a la memoria de mi amigo Marcelo Cañizares.

# Agradecimientos

Esta tesis ha sido financiada por el Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) en el marco del programa de Becas Doctorales Internas, durante el período 2003-2008.

La misma se ha realizado en el *Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC)* del Instituto de Desarrollo Tecnológico para la Industria Química (INTEC), dependiente del CONICET y de la Universidad Nacional del Litoral (UNL). El trabajo de investigación fue dirigido por el Dr. Alberto Cardona, a quien le agradezco su guía y la transferencia de su experiencia durante el transcurso del doctorado. En especial, le agradezco su confianza al permitirme "embarcar" en dos proyectos de nivel internacional, como lo fueron SYNAMEC y SYNCOMECS, de cuyo sinérgico trabajo en equipo esta tesis es una consecuencia. Quiero agradecer la amabilidad de Christian Paleczny, Ettore Baldassin, Aurelio Boscarino, Benoît Colson y Frédéric Cugnon, integrantes de estos proyectos con quienes he interactuado.

Agradezco el esfuerzo de los jurados de tesis al viajar desde distancias tan remotas y, en particular, sus críticas enriquecedoras. En especial, agradezco las correcciones de inglés y estilo del Dr. Víctor Fachinotti.

Luego quiero agradecer a mi esposa María Laura, por su constante apoyo y amor incondicional durante tantos años de noviazgo a la distancia. Sin su fuerza y sacrificio no podría *haberme dado este gusto*. Agradezco a todos los integrantes de mi familia, que me contagiaron de optimismo en cada viaje a Córdoba y que se tomaron la molestia de estar a mi lado siempre, sobretodo para reparar el mal uso que hice de mi salud. En especial, agradezco a mis sobrinos Juana y Benjamín sus inyecciones de ternura tan curativas.

Quiero expresar un agradecimiento especial a mis amigos Angel Zanotti, Federico Cavalieri y Damián Ramajo quienes me han dado ánimo y consejo en mis momentos más difíciles. A todos los integrantes del CIMEC –y sepan reconocer que el listado es largo– les agradezco la generosidad al brindar sus conocimientos, y en especial, el clima de respeto y amistad que se respira en este grupo de Santa Fe. A aquellos generadores y animadores de esas convivencias (cursos, almuerzos, congresos, deportes, asados, mateadas, guitarreadas, cumpleaños, barriles ...) que hicieron tan ameno y llevadero el estudio, los animo a que no decaigan ni en la sana recreación ni en la calidad de la ciencia que desarrollen.

Por último, agradezco a todos aquellos amigos santafesinos que me han alentado profesional y espiritualmente durante esta carrera. Entre ellos, mis profesoras de inglés Raquel Saurit y Andrea Insaurralde, mis amigos del INTEC, del Don Bosco, del Hogar "Miguel Magone" y de los Cooperadores Salesianos de Santa Fe y Córdoba.

Por siempre, un eterno gracias a Dios, a San Juan Bosco y a María Auxiliadora.

**Nota para el lector:** Esta tesis ha sido redactada en idioma inglés para facilitar su difusión. Se ha agregado un resumen extendido en idioma español como apéndice de la tesis.

# COMPUTATIONAL METHODS FOR DESIGN AND SYNTHESIS OF PLANAR MECHANISMS

**Keywords:**

planar linkage mechanisms, Finite Element Method,
type synthesis, Graph Theory, Type Adjacency Matrix,
mechanism isomorphism, Diagonally Extended Degree Code,
mechanisms enumeration, Synthesis Adjacency Matrix,
subgraph search, analytical dimensional synthesis,
Precision Point Method, Genetic Algorithms,
compliant mechanisms synthesis, Rigid-Body Replacement.

# Abstract

The objective of this research work is the study and development of techniques for the design and synthesis of planar-linkage mechanisms. The synthesis of mechanisms consists in finding the suitable mechanism for a given movement. Particularly, this thesis deals with the problem of synthesis of mechanisms starting from the initial specifications or requirements of design, that is to say, starting from zero. Consequently, it is necessary to determine the number and type of components, and the connectivity between them (type synthesis); and then to calculate the dimensions of the components, pivots positions, and the control parameters of the kinematic pairs of the input movement (dimensional synthesis).

This thesis deals with the *kinematic synthesis of position*, whose problem consists in determining the dimensions of a mechanism that satisfies a desired set of *displacements* and *rotations* in certain points of a mechanism and for certain instants of simultaneity. This specification is called kinematic task. The allowed space –for the solution mechanism and the development of the task– is a very common requirement that restricts the solutions to obtain. The problem is highly non-linear. Besides, since it includes the selection of the topology to dimension, it constitutes a discrete problem of combinatorial complexity.

In order to solve this difficult problem, it is proposed to use a representation of the mechanism based on the Finite Elements Method and Graph Theory, managing to preserve and unify both representations to integrate the synthesis into its subsequent stages of detailed analysis and optimization of the mechanism. The original theoretical aspects presented in this thesis are:

- The development of a new identifier of isomorphism of mechanisms and its use in the enumeration of kinematic chains and different atlases of mechanisms.
- The exhaustive enumeration of topologies using sub-graphs search to satisfy structural requirements from the beginning of the design process.
- The automatic decomposition of the closed-loop topologies into single open chains to solve their dimensional synthesis using analytical equations expressed by complex-numbers.
- For the dimensional synthesis, all combinations of single open chains (some of them with multiple solutions) are automatically computed. Among them, that solution which minimizes the summation of link sizes subjected to some design restrictions is retained. In the cases in which there are free parameters, a zero-order optimization technique based on Genetic Algorithms with penalization of restrictions is applied to sweep the design space.
- The modifications for extending the methodology to the design of flexible mechanisms using Rigid-Body Replacement methods are developed and analyzed.

As the final result of the application of this technique, it is obtained a list of alternatives that constitute good initial conditions for subsequent gradient-based optimization already available in commercial software. Throughout the thesis, various test and validation examples are provided, showing the capacity of the inventive tool developed.

# Contents

# Chapter 1

# Introduction

Mechanisms are mechanical devices of intensive use in agricultural, industrial, automotive, aeronautical, and general purpose machinery applications. Nowadays, their use in micro-electromechanical systems (MEMS) is acquiring great importance.

Since ancient times the designer's technical intuition and creative activity have enabled the materialization of a vast quantity of mechanisms that *transmit power while they perform the conversion of movements from some bodies to others*, in a repetitive way, in replacement and/or in multiplication of human or animal work or action coming from another source of power such as river water, wind, a chemical process, etc.

The design of mechanisms is an activity in which the engineer faces the difficult task of applying experience and ingenuity to combine a wide variety of mechanical elements with different functions, seeking the satisfaction of functional requirements and severe space restrictions. The design process is naturally cyclic and iterative. During the first stages of design, the *synthesis* and the *analysis* are used iteratively with the aim of reaching a valid –and if possible "the best"– solution that satisfies the requirements. The automation of these stages by means of the help of computational techniques should lead the designer to *systematize* the definition of the problem, the load of data between stages, the reduction of calculation time, to facilitate results interpretation, and consequently, *to focus the designer's intellectual and creative work only on those decisions of vital importance to the success of the design.*

In this chapter, the historical antecedents of the problem to solve, the hypotheses and scope of the contribution will be introduced. Finally, the content of the thesis will be outlined.

## 1.1. Problem description

Analysis of mechanisms is an activity usually done in industry in order to determine the kinematic and dynamic behavior (speeds, accelerations, efforts, etc.) of points of interest in a mechanism. Based on the results of the analysis, the engineer modifies the mechanism design to find the desired performance. Even though a computational tool for mechanism analysis is used, this process is known as design by trial and error. Nowadays, there are new computational programs that allow to apply numerical optimization techniques efficiently, avoiding unnecessary simulations. Yet, it is possible to find situations in which the design cannot be further improved and, generally, this leads designers to seriously consider trying other topological con-

figurations, analyzing other related inventions, etc., tasks that involve substantial time and cost. Currently, many designers are immersed in this situation.

The fact is that there are design techniques that have been thoroughly studied but did no spread over the work world. These design techniques allow to solve the inverse problem of *synthesis of mechanisms* that consists in *"finding the suitable mechanism for a given movement or task"*. In order to develop a computational tool that allows to solve the synthesis of mechanisms, it is necessary to find an inventive theory and to enunciate its corresponding algorithms, that is, to propose a *systematics of mechanisms design*.

This complex design problem can be enunciated as an optimization problem that is generally subdivided into two sub-problems of synthesis: a problem of topological enumeration that depends on discrete variables, and a dimensional problem with continuous variables. As it will be shown later on, this optimization problem involves theories and studies developed in the last three centuries. The discrete part will be dealt with more thoroughly in this thesis due to the fact that the enumeration and selection of topologies for a given task is an activity that is known, from the computational point of view, as an open problem of mechanisms design. In order to judge the optimality of a topology, the topology must have dimensions; as a result, the problem is extended to find all the dimensioned topologies (mechanisms) for a given task.

It is natural to think that the optimum mechanism is "the simplest" that "best satisfies the requirements"; therefore, the first action immediately after solving the synthesis problem is to evaluate the mechanisms and rank the found solutions in order of complexity and degree of restrictions fulfilment.

### 1.1.1. The design process

Nowadays, the process of *computational design of mechanisms* can be summarized in four stages:

1. **Identification of problem requirements**
   - Desired motion;
   - Allowed space;
   - Minimum power consumption, and others.
2. **Synthesis**

   2-I) Topological or Structural Synthesis:
   - *Type Synthesis*: Combination of the different types of mechanisms for the desired function: cams, gear-trains, linkages, etc.;
   - *Number Synthesis*: Decision about number, type and connectivity of the component parts for the required degrees-of-freedom;

   2-II) Dimensional Synthesis: Calculation of the significant dimensions for each constituent member.
3. **Analysis**

4. **Optimization, detailed design, test and experimentation**.

The first two stages of the process belong to the conceptual design and are of great importance due to the fact that the time and design costs for the analysis, the optimization and the test or experimentation depend strongly on the feasibility of the chosen concept.

### 1.1.2. Type Synthesis

The concept of *type synthesis* is attributed to the German professor Franz Reuleaux (1829-1905) who sets out the mechanism "type" election problem as the first step in the design process. A large quantity of combinations of type mechanisms can potentially satisfy one movement: cam, pulley and belt, gear, linkages, etc. The problem of combining mechanisms for a given specification is not solved in this thesis. Here, *type* will be pre-established in *planar linkages*, whose capacity to transform movements includes the combinations of rotations around an axis perpendicular to the plane and translations along axes contained in the plane, either as input, output, or both. Despite this reduction in the space of solutions, a large quantity of linkages can satisfy the same movement or task; some with less error than others, with a smaller number of constituent parts, among other measures of performance.

### 1.1.3. Number Synthesis

The number synthesis consists in generating feasible alternatives that satisfy the structural requirements of the specifications[1]: degrees-of-freedom required, type and connectivity of the prescribed parts (links and kinematic pairs); type, connectivity and complexity of the desired mechanisms. The structural complexity can be specified in terms of the number of links, number of joints of the kinematic chain, etc. Particularly, for the case in which the requirements are not sufficiently defined, it is the designer who proposes the domain of the desired mechanisms.

### 1.1.4. Dimensional Synthesis

The movements or *desired task* can be of varied complexity (prescription of positions, speeds, accelerations, effort to transmit, etc.). Generally and in the first stage, it is coherent to solve the *kinematic synthesis of position* under the hypothesis of *rigid mechanism* where it is considered that its constituent members do not have mass nor inertia properties [HD64, SE84, ES97, Nor95]. The kinematic synthesis of position consists in finding the mechanisms for the kinematic tasks of body guidance, trajectory generators and functions generation between two or more bodies (multiple coordination). For a simple rigid bar restricted to rotate on one of its points, the position is specified in terms of transcendental equations. Consequently, the kinematic synthesis of position is a non-linear problem in the linkages dimensions (unknowns). The mechanism of simpler degree of freedom is the four-bar mechanism articulated by revolute pairs. In one point of its coupler link (the link not articulated to ground) it is able to develop a planar polynomial curve of up to sixth degree. For this reason the four-bar mechanism is extremely useful, but its inverse design

[1] In recent publications about linkages it is common to find the denomination type synthesis for the complete stage of topological synthesis, including number synthesis.

is also cumbersome: given the curve, to obtain the mechanism. Mechanisms with a higher number of links allow to perform even more complex tasks. In Figure 1.1 the

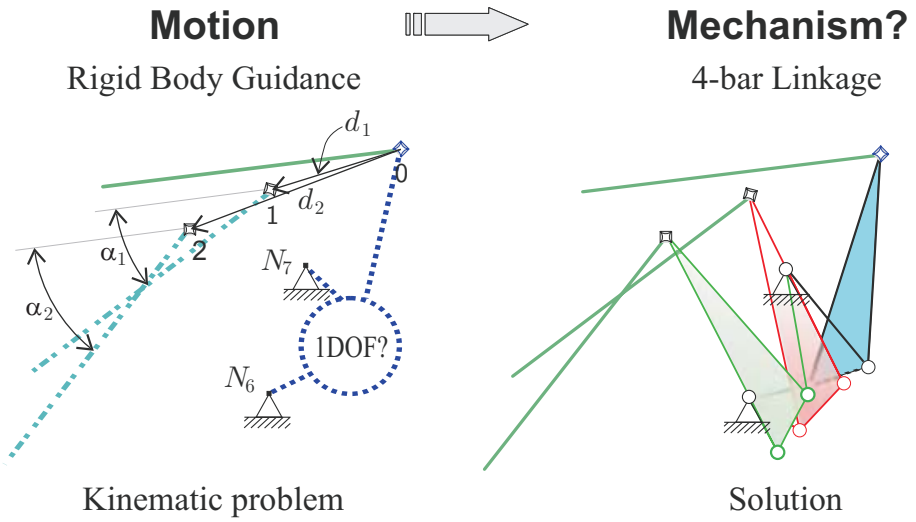



Figure 1.1: Dimensional synthesis of a four-bar linkage.

solution for a kinematic synthesis problem of rigid-body guidance passing through three-precision positions can be seen.

Methods for *dimensional synthesis* are classically divided in three major categories:

- *Graphic or geometrical*,
- *Analytical, exact or by Burmester*, and
- *Approximate*.

The two first methods apply when the desired movement is specified for a finite number of desired positions. In this thesis, they will be referred to as *precision positions*[1] and the associated technique of dimensional synthesis is equivalently called *Precision-Point Method* or *Burmester Theory*[2]. For a given number of positions and a given topology, analytical synthesis allows to know if the solution is unique, multiple, or nonexistent. The analytical expressions are obtained by means of closed-loop equations, which, depending on the number of positions to solve, can lead to systems of linear or non-linear equations. For some non-linear cases, the expressions in complex numbers facilitate to find closed-form equations.

Generally, after using the exact synthesis it is advisable to do a subsequent kinematic analysis in order to assess *the behavior between points of the task*. Then, it may happen that (i) the mechanism passes through the specified points but moving considerably away from the objective in intermediate portions of the task, or (ii) it approaches such portions satisfactorily and does not require further optimization.

[1] Also known as "precision points", "passing points", "accuracy points", "precise positions" and "finitely separated positions".

[2] Also referred as *Algebraic Synthesis*, *Finite Position Synthesis*, *Precision-Position Method* and *Burmester Synthesis*.

The *approximate synthesis* is applied to cases in which the number of specified points is very large, or when the behavior between intermediate points of the task is very important. The technique requires several kinematic analyses, at least one per iteration, where *the generated task is compared to the desired task*. Between both tasks an error index is defined which must be minimized.

A fourth category, that might be added to the above enumeration, is a *mixed – exact and approximate– synthesis*. It was investigated in the past [Sut77, KS75] and has recently acquired a renewed interest [HCE00]. Basically, some exact positions are chosen to have admissible tolerances. They are called *approximate-positions* or *quasi-positions*. The less important exact positions are converted into quasi-positions.

In general, once a mechanism satisfies the prescribed task, its fulfilment of other kinematic and dynamic requirements is pursued.

Among the most commonly found problems when analyzing mechanisms are the blocking or *jamming* and the *bifurcation*. In the first one, the movement is incompatible, for example, because two bars align, which causes the analysis algorithm to interrupt. In the second, for the same mechanism, its parts can go through different circuits depending on the initial condition and it is not possible to choose the desired circuit. Both situations should be taken into account as sources of irregularities of the possible solutions to the synthesis problem, and their correction is called solutions rectification.

## 1.2. Simplificative hypotheses

The following simplificative hypotheses will be made in the context of this thesis:

- Planar linkage synthesis will only be considered.
- The movement is discretized into 3 or 4 precision positions. A separate optimization software will be used to approximate a larger number of positions, using the results of exact synthesis as initial guess.
- Under the rigidity hypothesis, the main characteristics of the mechanism that can be designed are:

  **Task**: to satisfy the *input-motion/s* vs. *output-motion/s* relationship with minimal error. For each point, the goal is to minimize the error between the *desired* and the *generated movement*.

  **Input motion**: when only the motion for output members is specified, part of the problem consists in determining the corresponding states of the remaining members as possible inputs.

  **Space**: the mechanism, at the initial position as well as for other configurations, must be moved inside an *allowed space*.

  **Transmission angle**: it is indispensable that the mechanism does not block during its movement and also be as far as possible from such situation.

  **Weight**: using the mentioned hypothesis, the weight is closely related to the links size and therefore the links minimal size is desired.

## 1.3. Historical antecedents

The design of mechanisms is a task that has been considered, until a few years ago, as a mixture of art and science. In a retrospective look, we can easily be impressed by the current engineering value of Leonardo Da Vinci's inventions (1452-1519)[1] and by the volume of the literature relative to the use of atlases, manuals and catalogues to help the mechanisms designer [SWM+].

Around mid-eighteenth century the main efforts for design systematization arose, especially due to the creation of the first schools for the teaching of the science of mechanisms. These schools were meant to meet the needs created during the so-called Industrial Revolution (1750-1820). The design of mechanisms ceased to be an empirical science to merge into various disciplines: geometry, algebra and kinematics. The ability to design mechanisms considered as innate in a designer up to that moment, began to be taught and learned. One result of these schools is the use of the *steam engine* in the industry first as a source of static power, and then in means of transport (locomotives and ships) as a mobile source, causing the biggest technological and productive boost in the history of the world.

Initially, the synthesis problems were tackled by a few synthesists who rightly began to study a movement from a purely geometric perspective, disregarding its causes. An interesting eighteenth-century historical review was published by Ferguson [Fer62]. Another concise description, even more extended in time, can be found in the first chapter of Hartenberg and Denavit's book [HD64][2]; these authors highlighted the influence of two great figures, the Swiss Leonhard Euler (1707-1783) and the English James Watt (1736-1819), in the progress of the systematization of mechanisms design. They emphasized the difference between their lines of investigation: Euler is identified as the person who first proposed the study of kinematic analysis (the study of motion without considering the causes that produce it), and Watt as a pioneer involved in the synthesis of movement, in particular, the discoverer of the complex motion of a point of the coupler link in the four-bar mechanism. Watt designed a mechanism for which the point in study develops a trajectory approximately straight, and, in 1784 implemented it in the most significant improvement of the steam engine by guiding a long stroke piston through a straight line.

Other scientists, who were recognized in other areas of science, also embarked on the synthesis and analysis of linkages, especially to solve the "benchmark" of that time: *to develop a straight line with minimal error*. Among others, we can mention the Russian mathematician Pafnutij L. Chebyshev (1821-1894), the English mathematician James J. Sylvester (1814-1897), the English William J. M. Rankine (1820-1872) who was one of the most renowned for his postulates in thermodynamics, the French Andrè M. Ampère (1775-1836) more recognized for his contributions to electricity, and the English designer of textile machinery Richard Roberts (1789-1864). In 1873, almost one hundred years after Watt's first invention, the French Charles N. Peaucellier (1832-1913) invented a mechanism to develop an exact straight line; the scientific community was deeply surprised by the simplicity and foundation of the solution [Fer62].

[1] Some of Da Vinci's inventions were not reproducible in his time and regained interest in the present, such as flying machines similar to present-day helicopters, and others biologically inspired machines, similar to birds, comparable to current UAVs (Unmanned Aerial Vehicles)

[2] Both the paper [Fer62] and the book [HD64] are available on-line in the digital library KMODDL [SWM+]

Subsequently, the German Franz Reuleaux (1829-1905) stood out due to a new classification of mechanisms and the definition and classification of kinematic pairs. He was the mentor of the concept of *inversion of the kinematic chain* and thus was a pioneer in the *synthesis and topological analysis* of mechanisms. In the United Kingdom, Robert Willis (1800-1875), in parallel with Reuleaux, made the major contributions to the theory of machines in an integral way: from the point of view of mathematics, engineering and education [Moo03]. In Germany, between the end of the nineteenth- and the beginning of twentieth-century, Ludwig E. N. Burmester (1840-1927) carried out important Geometric contributions to kinematics; among others, he synthesized points on a circumference giving rise to the today so-called *Burmester's Synthesis*. Contemporaneously, in that country, Martin Grübler (1851-1935) provided various theorems to find the degrees-of-freedom of planar and spatial mechanisms, the fundamental basis of the current enumerations of mechanisms. Next, the First and Second World Wars broke out with their logical periods of confidentiality in technological developments [Cec00, Cec03].

An important development in the design of linkages took place between the years 1943 and 1946, during the stay of the Austrian Antonin Svoboda (1907-1980) at the Radiation Laboratory of the Massachusetts Institute of Technology (USA)[1]. Svoboda integrated the linkages in the analogue computers as devices for calculating non-linear functions for the development of control systems for anti-aircraft weapons [She07, Map79].

With the advent of computers towards the middle of the twentieth century, a new science is created, *computer science*, in which the design formulations are expressed through mathematical algorithms. This allowed to largely systematize the *design of mechanisms* and it became a science[2] [Ang97, Cec04].

## 1.4. Computational tools

It is remarkable, as shown by the historical facts, that the computational tools have been developed in the inverse order to that of the stages of the design process (Section 1.1.1). First, the *analysis* by computer was implemented and it was used as a tool to expedite the design by trial and error; then, the *synthesis* programs came up [HD64, SE84, ES97, Nor95, OER87]. Later on, there appeared programs capable of classifying and combining mechanisms automatically from the desired movements [CK99, YO05].

The programs for *analysis* were the first incorporated to CAD/CAE (*Computer-Aided Design/Computer-Aided Engineering*) systems of industrial use. Regarding the numerical techniques for the analysis, it is noteworthy that in the last two decades the modeling and analysis of mechanisms using Finite Elements have advanced considerably achieving great accuracy in simulations [WN03]. It is also worth mentioning that in Argentina, for the last twenty years, the topic of flexible

[1] Considered among the pioneers of computer science, author of *Computing Mechanisms and Linkages*, volume 27 of MIT Radiation Laboratory Series, McGraw-Hill, New York, 1948. Designer, between the years 1950 and 1956 of the first fault-tolerant computer, known as SAPO, built in Czechoslovakia.

[2] Recently, Professor Marco Ceccarelli has been given numerous historical research on the evolution from the *"Theory of Machines and Mechanisms (TMM)"* towards the now called *"Science and Mechanisms of Machines (SMM)"*

mechanisms analysis has been a very fruitful line of investigation for the CIMEC group [Car89, GC01, Len06]. Besides, there are efficient programs for mechanisms *optimization* such us the SAMCEF BOSS/Quattro software that allows to iterate with the non-linear analysis module SAMCEF Mecano [Car89, SAM07].

Among the pioneers of the computational *synthesis* of mechanisms of the mid-twentieth century there are Ferdinand Freudenstein (1926-2006) [FS59] and George N. Sandor (1912-1996) [San59] in the United States and a N. I. Leviskii and K.K. Shakvazian in the Soviet Union [Ang97]. Contemporaneously, Richard Hartenberg (1907-1997) and Jacques Denavit [HD64], Erskine F.R. Crossley, Oene Bottema (1901-1992), Kurt Hain (1908-1995), Bernard Roth, Allen S. Hall Jr. [Hal61] stood out. In the last decades, Arthur G. Erdman [ES97, SE84], Kenneth J. Waldron, Jorge Angeles, Lung-Wen Tsai (1945-2002) [Tsa01], Robert L. Norton [Nor95], Hon-Sen Yan [Yan98], and J. Michel Mc Carthy [McC00], among others. In their researches it can be seen how *kinematics*, *geometry*, *algebra*, *discrete mathematics and combinatorial analysis*, *numerical methods and optimization techniques* have shown their intersection with the *science of mechanisms* imparting it a multi-disciplinary character. The science acquired its true amplitude nowadays if we consider the mechanisms as parts interacting in more complex environments, e. g. with *electronic*, *control*, *robotic* and *mechatronic* systems.

Since more than three decades ago, there have been academic programs that solve satisfactorily some dimensional synthesis problems. Referring to planar mechanisms, among the pioneers, there are KINSYN [Kau71], LINCAGES [EG77], and RECSYN [WS81]. Then, SYNMECH [PK97] appeared and later on some others still in development such as LINCAGES 2000 [Erd, YEB02], SAM [Ran07], TADSOL [CKv], SYNTHETICA [McC], WATT [DK07], and SyMech [Coo], among others. The use of these programs requires a wide experience as a kinematicist, which is simplified by the development of user-friendly interfaces. Some of these programs have exportation capacity to industrial-file formats for a subsequent analysis of the solutions. The CAD/CAE commercial programs most widely known in industry do not have modules of their own for synthesis of mechanisms. Few of the above mentioned synthesis programs have achieved the integration; particular cases are SyMech as "Add-In" of Pro/E [CO02], the study of incorporation feasibility of SYNMECH in ADAMS [PVW97], and the recently Kinzel *et al.* method called Geometric Constraint Programming [KSP06] which uses the "sketching" mode and the constraints handling tools of the commercial CADs. It can be said that the category *Computer Aided Linkage Design* is relatively young[1].

It should be highlighted that the literature related to *type synthesis* topics was scarce and relegated to short communications in specialized magazines until short time ago. Interestingly, a book with very valuable chapters devoted to this field was published in Spanish by Professor Justo Nieto Nieto [Nie77] in 1977. Two decades later, the books by Professors Yan [Yan98] and Tsai [Tsa01] were published, with a more modern approach, both books are provided with algorithms and concepts that have been employed in the present thesis, see Figures 1.2 and 1.3.

[1]Special Session on Computer Aided Linkage Design in proceedings of DECT'02 ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Montreal, Canada, September 29-October 2, 2002.

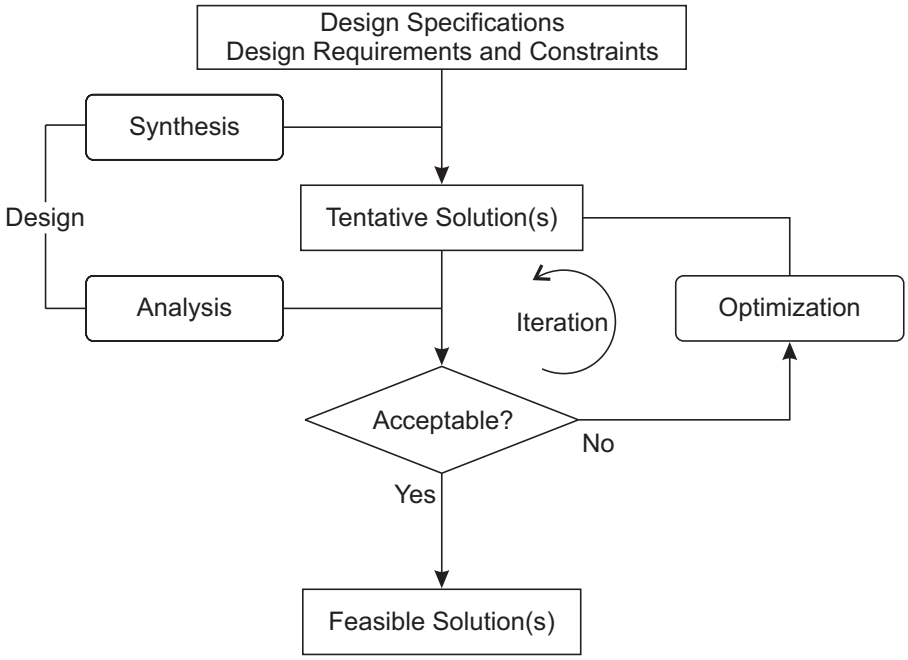



Figure 1.2: "Design, synthesis and analysis for engineering solutions" by Hong-Sen Yan [Yan98].

## 1.5. Current methodologies and state of the art

Lung-Wen Tsai [Tsa01] proposed a systematic methodology for the design of mechanisms and summarized it as follows:

1. *"Identify the functional requirements, based on customer's requirements, of a class of mechanisms of interest.*
2. *Determine the nature of motion (i.e., planar, spherical, or spatial mechanism), degrees of freedom (dof), type, and complexity of the mechanisms.*
3. *Identify the structural characteristics associated with some of the functional requirements.*
4. *Enumerate all possible kinematic structures that satisfy the structural characteristics using graph theory and combinatorial analysis.*
5. *Sketch the corresponding mechanisms and evaluate each of them qualitatively in terms of its capability in satisfying the remaining functional requirements. This results in a set of feasible mechanisms.*
6. *Select a most promising mechanism for dimensional synthesis, design optimization, computer simulation, prototype demonstration, and documentation.*
7. *Enter the production phase."*

*"We note that the methodology consists of two engines: a* ***generator*** *and an* ***evaluator*** *as shown in Figure 1.3. Some of the functional requirements are transformed into the structural characteristics and incorporated in the generator as rules*

*of enumeration. The generator enumerates all possible solutions using graph theory and combinatorial analysis. The remaining functional requirements are incorporated in the evaluator as evaluation criteria for the selection of concepts. This results in a class of feasible mechanisms. Finally, a most promising candidate is chosen for the product design. The process may be iterated several times until a final product is achieved* [Tsa01].

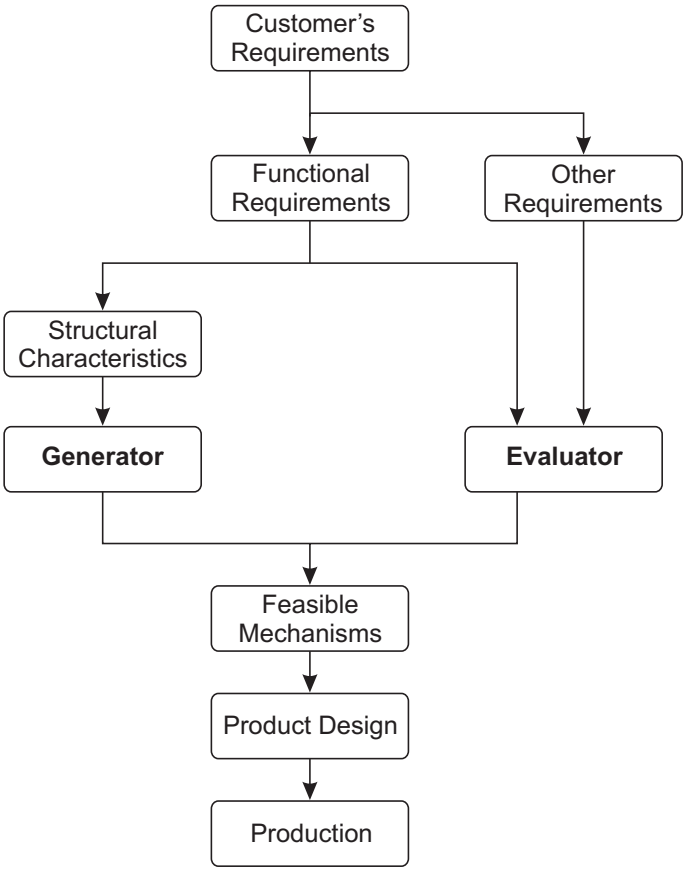



Figure 1.3: "Systematic mechanism design methodology" by Lung-Wen Tsai [Tsa01].

Tsai based his methodology of systematic enumeration on Freudenstein and Maki's concept of "separation" of the kinematic structure (to generate alternatives) from the functional (to evaluate the generated alternatives) [FM79]. He also emphasized that *"The more functional requirements that are translated into structural characteristics and incorporated in the generator, the less work is needed from the evaluator. However, this may make the generator too complex to develop. Generally, if a functional requirement can be written in a mathematical form, it should be included in the generator"* [Tsa98, Tsa01].

Sardain stated that the optimality of a topology cannot be judged until its dimensions are considered and he proved that with an example case [Sar97]. This means that the evaluator should include not only topological matters but also the dimensional ones as well as all those performance measures that can be built from them.

There are several methodologies for topologies enumeration, but few of them are integrated to the dimensional synthesis. In the method proposed by Tsai some

aspects that he solved manually could be implemented computationally. However, though he successfully applied his method to the enumeration of gear-trains, parallel manipulators and link mechanisms [Tsa87, Tsa98, Tsa01]. Recently, Liu and Mc Phee argued that the type synthesis stage is an open problem and made reference to a phrase uttered in 1995 by Arthur Erdman: *"the most critical stage of the entire design process is ... choosing the best topology for a given task"* [LM05]. These researchers applied Genetic Algorithms (GAs) in order to find a topology that satisfies certain topological requirements. In a similar line, also using GAs, Sedlaczek *et al.* [SGE05] proposed to incorporate the dimensional synthesis in the evaluation of each topology.

In an attempt to automatize Tsai's proposal, since 2004, Pucheta and Cardona [PC04, PC05a, PC05b, PC06, PC07] have proposed (i) to solve the type synthesis including the prescribed parts in the enumeration of mechanisms through the use of Graph Theory and (ii) the subsequent use of the GAs in the analytical dimensional synthesis of each topology. A prototype, developed in collaboration with the SAMTECH company [SAM07] enabled these researchers to use the CAD Samcef Field interface for the pre- and post-processing of the synthesis problems and to continue then with the detailed design and optimization stages inside the same environment [CCSP07].

In 2005, Chen and Pai [CP05, CFH+06] presented an exhaustive methodological approach to solve the type synthesis from a classification of the design specifications and solved a problem proposed by Freudenstein and Maki.

## 1.6. Motivation

Synthesis methods are much less developed than analysis methods. For this reason the synthesis is usually the bottleneck in the conception process. In current methodologies, the developments for type synthesis and dimensional synthesis are fragmented. The mechanisms representation through Finite Elements, that implicitly includes the structural representation through Graph Theory, seems to be the adequate tool to unify the task specification and both stages of synthesis to favour its computational implementation.

Based on the above mentioned antecedents, it will be presented in this thesis the experience of incorporating into the alternatives generator:

(i) The prescribed parts and the motions/movements imposed upon them.

(ii) The possibility of pre-evaluating the solvability of the topology for the imposed motions and the number of prescribed precision positions, preventing in this way calculations of dimensional synthesis without solution.

These characteristics are modeled to work in an automatic way, which is highly original and without precedents in the specialized literature of the field. Notice that the second characteristic is applied only if the topology is solved using the Precision Position Method.

In the synthesis of mechanisms there are many complex combinatorial operations impossible to be solved manually. With the current speed of calculation of processors, it is possible to encapsulate in a program and run complex combinatorial algorithms in such a time that does not divert the user's attention from the

design. In the same way, there is a large number of algorithms profoundly studied and used in other branches of science –Computer Science, structural Chemistry, Economy and Business– available to be applied in Mechanics.

## 1.7. Objectives and scope

A computational objective little pursued is the development of applications for the design *from the beginning*, that is, starting from the movement specifications and restrictions imposed upon the existent parts that we wish to move, and manage to enumerate all the possible solution-linkages.

The theoretical objective is to develop a systematic, unified and general design methodology that allows to find all the suitable mechanisms (not restricted to planar linkages) to satisfy a given kinematic task, arranged by increasing order of complexity and without repetition.

Regarding the type synthesis stage, a particular objective is to study in detail the "off-line" enumeration of mechanisms and the "on-line" exhaustive search of the topology of the initial parts inside the already enumerated mechanisms. In this stage, heuristic algorithms will not be applied to reduce the search since they can lead to ignore good alternatives.

Another particular objective is to predict the practical difficulties in the use and extension of the method to the support of more complex tasks and more kinds of mechanisms with different types of members; for example, in the extension to synthesis of mechanisms with elastic members.

The ***selection*** from an ordered enumeration of all the possible solutions is a complex task of decision. In this thesis, it is assumed that the selection will be made manually, although it is also an automatable aspect of the design.

Taking into account the previous review about the existing programs, it was of special interest to incorporate the algorithms resultant from the thesis into a program capable of interacting with a CAD/CAE system whose use minimizes the designer's training period, and besides minimizes the time needed to produce the results.

## 1.8. Proposed Automation

In order to achieve the mentioned objectives, a conjoint representation of the mechanisms and the prescribed parts based on Graph Theory and the Finite Elements Method is proposed. The proposition consists in using one or another representation conveniently, depending on the problem to solve.

The block-diagram in Figure 1.4 shows the stages in the design of a mechanism using computational tools, and specially highlights the synthesis module which is further detailed in Figure 1.5. The solver was implemented in a general code of finite elements written in C++ language called Oofelie (Oriented Object Finite Elements Led by Interactive Executor) [CKG94, KCG98, BER01, Ope].

For the ***Type Synthesis***, the following procedure is proposed [PC07]:

**T1**: Represent the sub-mechanism or prescribed parts and the kinematic task using a CAD for finite elements, discretizing the task in precision positions.

**T2**: Convert the kinematic problem in a graph called *Initial Graph*, which produces a mathematical model for the prescribed parts and structural restrictions.

**T3**: Select a desired atlas of mechanisms. This implies the possession of several atlases of mechanisms generated with different types of links and joints, for different degrees-of-freedom, and using a univocal and efficient codification.

**T4**: Solve the type synthesis using the search of the *Initial Graph* as a sub-graph of each mechanism of the selected atlas and identifying all the non-isomorphic occurrences.

**T5**: Create a schematic diagram for each alternative.

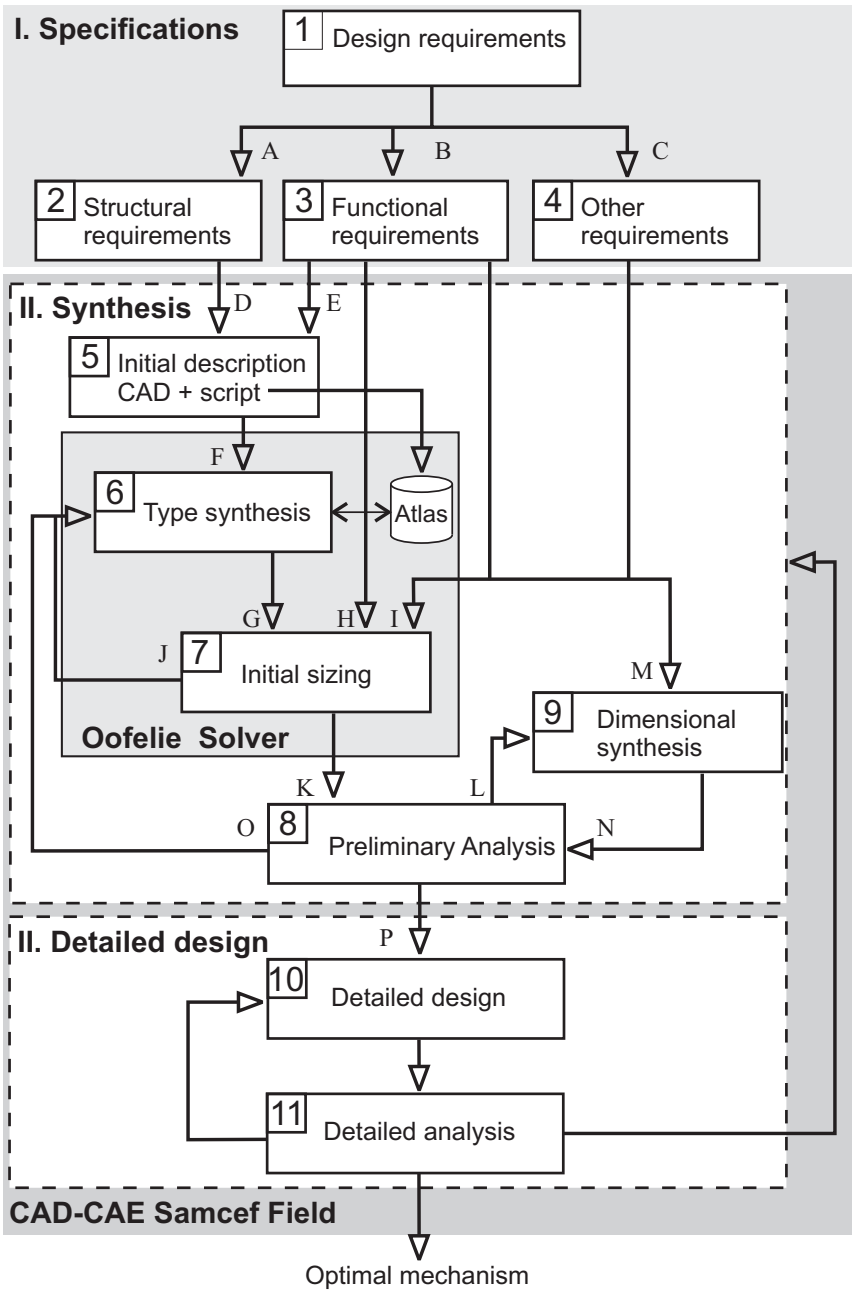



Figure 1.4: Computational process for optimal mechanism design.

For the ***D**imensional Synthesis*, it is proposed the use of exact geometrical methods (already existing) to give dimensions to each abstract topology resulting from the *type synthesis*, developing computationally these steps:

**D1**: Decompose the topology into single open chains [SE84, PC06].

**D2**: Search the optimal ordering of the chains for the available open-chains solvers [PC06].

**D3**: Solve the chains analytically using complex-numbers to represent the links [HD64, SE84].

**D4**: Reassemble the chains to reconstruct the topology.

**D5**: Evaluate the fulfillment of the restrictions.

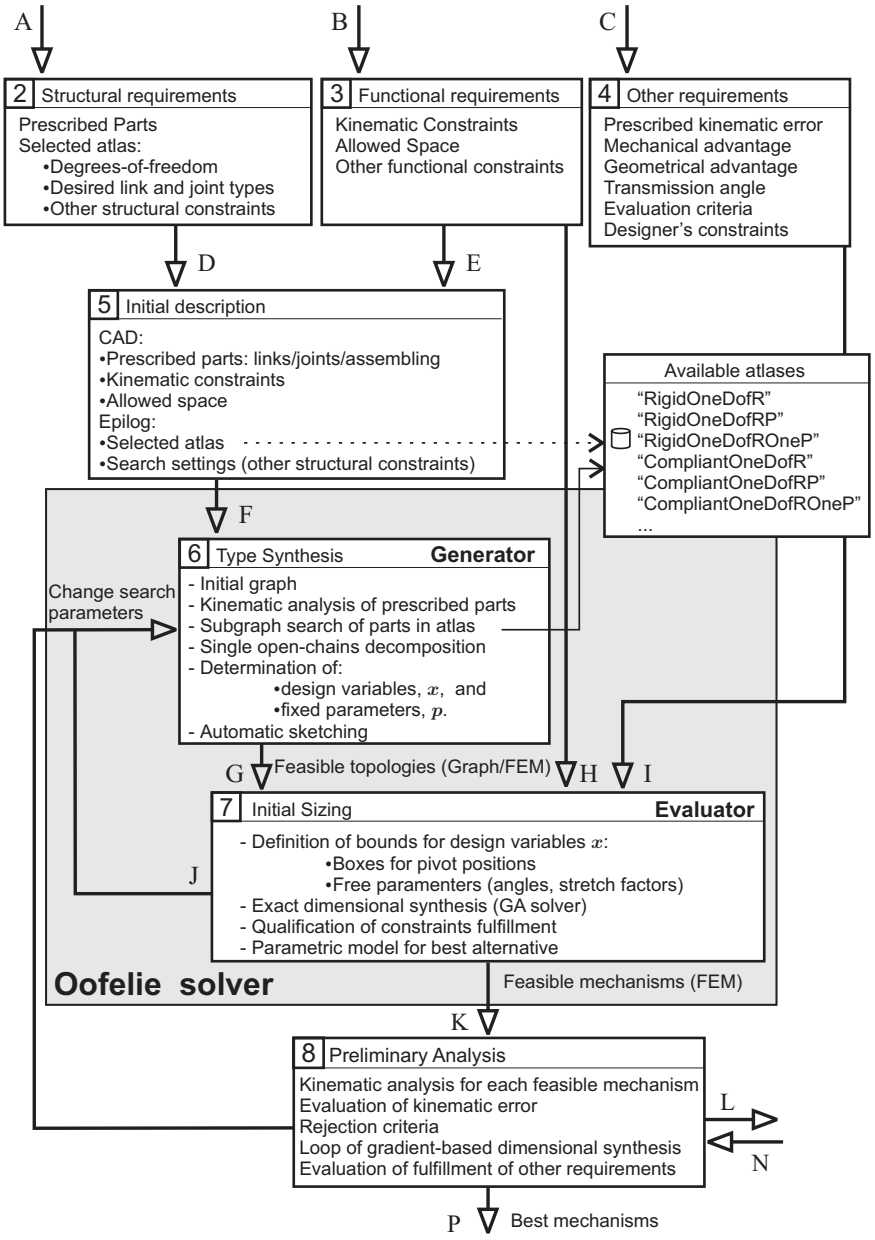



Figure 1.5: Details of the synthesis procedures.

In the dimensional synthesis through the decomposition into open chains, there is a calculation order due to the fact that the resolution of one chain depends on the restrictions imposed by a previous one [SE84, ES97]. As a result, the resolution is hierarchical and the multiple solutions make the reassembling also hierarchical. Genetic Algorithms are used in the stages **D3-D4** for the cases in which there are free parameters [PC05a].

As it was mentioned before, in this thesis it is also proposed to incorporate in advance the items **D1-D2** between the items **T4-T5** of the type synthesis, making the generation of topologies more complex but also more useful, see Figure 1.5. In this way, if a mechanism with certain imposed movements cannot be decomposed for the available solvers, the alternative is rejected and the executed type synthesis solver continues looking for the detection of another alternative.

## 1.9. Thesis content

The thesis is divided into two parts, Type Synthesis and Initial Sizing.

In Chapter 2, more antecedents related to Type Synthesis will be introduced and a new strategy of design based on Graph Theory (GT) will be presented. In this chapter, the representation of mechanisms using GT is revised and the characteristics of the possible problems to solve are detailed. Then, the development of a code-based identifier for detecting isomorphisms of mechanisms is emphasized; its use for the non-isomorphic enumeration of diverse atlases of mechanisms is shown. In Chapter 3, the automatic utilization of the search –in an atlas selected by the user– for the parts of an initial mechanism is presented. For this purpose, all non-isomorphic occurrences within each mechanism of the atlas are detected and encoded.

A detailed review of analytical methods for open-chain synthesis is presented in Chapter 4 with the aim of introducing the reader to some conventions about necessary data and solvability conditions used in the subsequent chapters of the thesis.

In Chapter 5, the algorithms employed for the decomposition of a topology in open chains are detailed. The aim pursued with this decomposition is to reduce the number of design variables to its minimum; consequently, this constitutes a problem of discrete optimization. The approach presented is algorithmic and also analytic since it takes into account the necessary (but not sufficient) conditions for the solvability of each open chain.

In Chapter 6, the strategy of initial dimensioning is presented. Its objective is to determine initial dimensions for the subsequent optimization of the mechanism. The algorithms and data structures relative to the degrees-of-freedom of the problem are detailed. The operation with combinations of open-chain solutions with geometric multiplicity is also explained. A running path-following example will be used in Chapter 3 through Chapter 6. In Chapter 7, the resolution of simple validation problems is introduced, together with applications in complex tasks of multiple purposes.

In Chapter 8, the synthesis of flexible binary members is presented in addition to its considerations to form the so-called fully and partially compliant mechanisms.

Finally, Chapter 9 presents the main conclusions of this work, commentaries about efficiency, found difficulties, aspects to improve, and perspectives for the future.

# Part I

# Type Synthesis

# Chapter 2

# Computational tools for Type Synthesis

The mathematical modeling of mechanism synthesis problems using Graph Theory and the Finite Element description of mechanisms is presented in this chapter.

Two computational tools, a graph-matrix representation of mechanisms and a mechanism identifier are developed to cover the two main aspects of topological synthesis:

1. the enumeration, identification, storage and retrieval of mechanisms, and
2. the avoidance of the isomorphic occurrences of the initial graph inside each mechanism of the atlas.

Although the first development is a slight variation of the existing Degree Code [TL93], a discussion over the proper modifications needed to consider different types and links is presented herein for the first time.

The second topic was precisely designed to satisfy an important need of the proposed subgraph search method: the discrimination of *functionally different topologies*. The subgraph search algorithm will be detailed in the next chapter.

## 2.1. The proposed type synthesis methodology

The aim of *Type Synthesis* 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 an experienced designer; such a procedure may easily lead the designer to neglect possible solutions. Graph Theory can be conveniently applied to do this search automatically.

Since the mid-sixties, Graph Theory has been used by Franke, Woo, Freudenstein, Dobrjanskyj, Crossley and Manolescu, among others, for systematic enumeration and topological analysis of rigid linkages (see reviews of Olson [OER85] and Mruthyunjaya [Mru03]). In the nineties, flexibility was incorporated by Murphy, Midha and Howell [MMH96, How01] for constructing atlases of flexible linkages.

Bar-linkages and other atlases for particular purposes (variable-stroke engine mechanisms [FM79], planetary gear trains, parallel and robotic mechanisms [Tsa98,

Tsa01]), were structurally synthesized using a systematic strategy based on the Freudenstein and Maki concept [FM79] of "separation" of *kinematic structure* (to generate alternatives) from *function* (to evaluate the generated alternatives). In these cases, the functional constraints were dealt with after the enumeration process, and the generated alternatives were post-analyzed by hand with the aid of combinatorics to validate the task matching, see Figure 2.1-**i**.

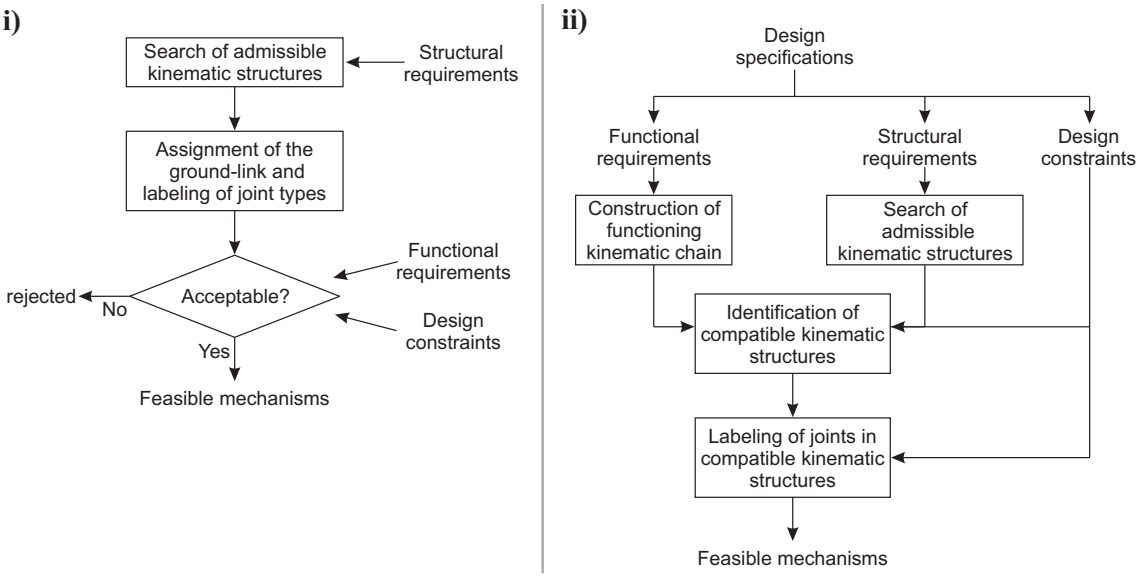



Figure 2.1: Current type synthesis methods [CP05]: (**i**) by Freudenstein and Maki; (**ii**) by Chen and Pai.

However, an *automated method to generate alternatives satisfying those functional requirements which have structural influence at the outset* is less addressed in the literature. Very recently, Chen and Pai [CP05] presented a design methodology in the same line of research presented here. They parsed the design specification into functional requirements (**a**), structural requirements (**b**), and design constraints (**c**). Using **a**, they construct the *functioning kinematic chain* 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 the labeling of joints in the compatible kinematic structures they validate the remaining design constraints, see Figure 2.1-**ii**. These authors did not mention any computational implementation of their method. Other methods fall in the field of expert systems.

In this setting, a subgraph approach is presented. 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 Figure 2.2 as an introductory example). In addition, an *atlas of kinematic chains (KC) with simple joints* is adopted. Then, using a graph representation, each kinematic chain is *specialized* 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 proposed generator of alternatives is an engine for a subgraph search of all non-isomorphic occurrences of the initial graph inside each mechanism of the selected atlas.

Figure 2.3 illustrates the proposed type synthesis method (the example corre-

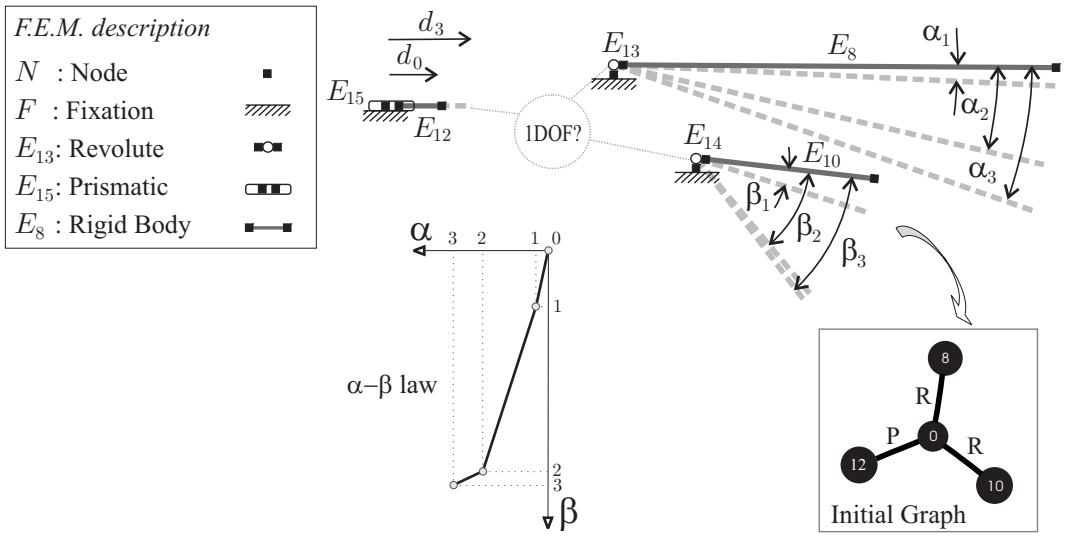



Figure 2.2: Graph representation for a combined kinematic task.

sponds to the kinematic task shown in Figure 2.2). The designer interacts in the preprocessing and postprocessing stages, i.e. he enters data into a computer program in stages **I**, then runs the solver **II**, and finally evaluates the solutions **III**.

To avoid isomorphisms, a topological representation of mechanisms called "Type Adjacency Matrix" is defined 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 "idle loop" with some unloaded links, will be considered as *pseudo-isomorphic mechanism*).

This Chapter is organized in the following way: The graph representation of kinematic chains and mechanisms is presented in Section 2.2. In Section 2.3, methods for isomorphism testing are briefly reviewed. Next, a code-based mechanism identifier is presented in Section 2.4. The enumeration of kinematic chains is briefly reviewed in Section 2.5 and the method for the enumeration of mechanisms starting from kinematic chains is presented in Section 2.6. The graph representation of kinematic problems as well as the subgraph search algorithm will be detailed in the next chapter.

## 2.2. Graph representation of kinematic chains and mechanisms

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 $\{v_i, v_j\}$. Hereafter, links and vertices will be referred to in an homologous way, the same consideration is valid for joints and edges. Then, the size of the set of vertices is denoted by $n = |V|$ and the size of the set of edges as $j = |E|$.The set of unlabeled vertices $V = \{v_0, v_1, ..., v_{n-1}\}$ is numbered in zero base, i.e, $V = \{0, 1, ..., n-1\}$.

Additionally to the graph $G$, two levels of information are needed for defining

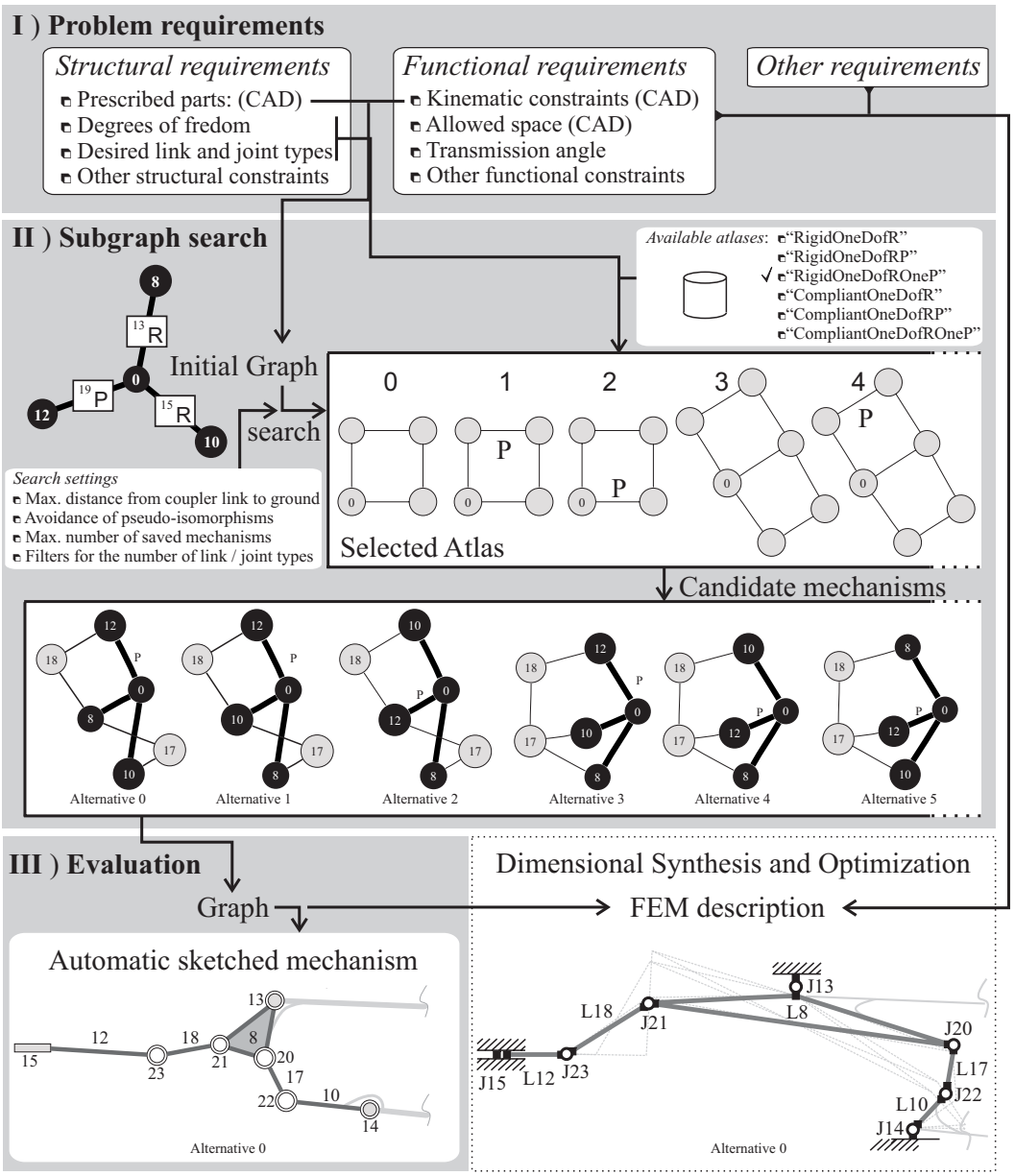



Figure 2.3: The proposed Type Synthesis method

mechanisms:

- ***Labels*** identifying the *functional meaning of each link or joint.*
- ***Types*** identifying the *structural type (or physical behavior) of each link or joint.*

To consider the first item, links with functional meaning (e.g. the ground, a crank, a piston, a flap, etc.) are commonly labeled either by the user or by the solver using integer identifiers (ID's)[1]. A function $\mathcal{V}(V)$ labels each vertex of the set $V$ with the integer identifiers $\boldsymbol{l}_L = \{l_0, l_1, ..., l_{n-1}\}$, so that $\mathcal{V}(v_i) = l_i$. A function $\mathcal{E}(E)$ is also used to match physical joints with edges. The function $\mathcal{E}(E)$ is more

[1]Since the solver runs in batch mode, the user can define the prescribed parts of the problem by using a CAD environment which generates a script or by scripting directly. The solver will add new ID's not used before for those structurally synthesized parts.

conveniently defined by using the pairs of labeled links as $\mathcal{E}(\mathcal{V}(V) \times \mathcal{V}(V))$. A *labeled graph* is obtained by labeling of its constitutive sets $G(\mathcal{V}(V), \mathcal{E}(E))$.

Then, by means of the ID's of the FEM description, the user defines elements with different structural behaviors either for links or joints, which are easily assigned by mapping them to integer numbers: for example, the link types alphabet is $\{0 = \text{ground}, 1 = \text{rigid}, 2 = \text{flexible}\}$, and the joint types alphabet is $\{1 = \text{revolute}, 2 = \text{prismatic}, 3 = \text{flexible\_hinge}, 4 = \text{clamped}\}$; the obtained graph is a *colored labeled graph*.

Once labels and types are assigned on the graph, the *topological structure of a mechanism* is completely defined (see Figure 2.4). The mathematical model of the mechanism is completed with the aid of matricial representation.

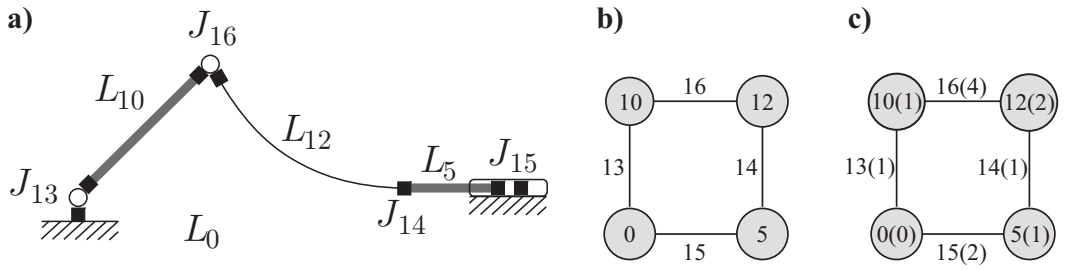



Figure 2.4: Mathematical models for a four-bar mechanism: a) FEM representation; b) Graph labeled with user's ID's; c) Graph with link and joint types colors (colored labeled graph).

A graph has matrix representations which express how vertices are connected by edges. 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 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. A relabeling on the labeled vertex set $\mathcal{V}(V)$ is equivalent to the congruent transformation in the adjacency matrix $A^p = PAP^T$, where the matrix $P$ is a reordering in rows and columns of the identity matrix by $p$, where $p$ is a vector of permuted indexes.

*Two graphs are isomorphic if there is a one-to-one bijection from the vertex set of one graph to the other and edges preserve incidence. Also, two graphs are isomorphic if they share the same adjacency matrix.*

For the purpose of identifying mechanisms, the adjacency matrix must contain information relative to types of links and joints.

Although it is a well-known property of a symmetric matrix, it shall be remarked that under the same permutation of rows and columns (congruent permutation) an entry $a_{ij}$ of $A$ is permuted to the entry $a_{p(i),p(j)} = a^p_{ij}$ in $A^p$. Then, we may observe that:

- an entry on the diagonal changes (or remains) its position to another diagonal entry.
$$a_{ii} \rightarrow a^p_{jj}$$
- a non-null entry on the upper/lower diagonal elements of a matrix, changes (or remains) its position to another one in the same upper/lower block (edges preserve incidence).
$$a_{ij} \neq 0 \rightarrow a^p_{kl} \neq 0.$$

Another important consideration is that no *sling edges*[1] are allowed in kinematic chains, and thus the diagonal elements in $A$ are always null.

So, to form mechanisms, quantities with different physical meaning could be reliably attached as a layer on the adjacency matrix of the kinematic chain. In this sense, different matricial representations were defined in the last decades. Murphy, Midha and Howell [MMH96, How01] defined the *Compliant Matrix* $\boldsymbol{CM}$ considering link-types on the diagonal entries and joint-types on the off-diagonal non-null elements. In a similar fashion, Yan [Yan98] defined the *Topology Matrix* $\boldsymbol{M}_T$ but including the joint-labels on the lower-triangular part and joint-types on the upper-triangular one.

Using these simple concepts and antecedents a choice of matrix representation is made with mind in simplifying operations of permutations and reducing space for storage. A *Type Adjacency matrix* $\boldsymbol{T}$ is defined as follows:

$$T_{ii}(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is the ground,} \\ 1 & \text{if } v_i \text{ is a rigid link,} \\ 2 & \text{if } v_i \text{ is a flexible link,} \end{cases} \quad T_{ij}(e_{ij})_{i<j} = \begin{cases} 0 & \text{no connection,} \\ 1 & \text{if } e_{ij} \text{ is a revolute joint,} \\ 2 & \text{if } e_{ij} \text{ is a prismatic joint,} \\ 3 & \text{if } e_{ij} \text{ is a flexible joint,} \\ 4 & \text{if } e_{ij} \text{ is a clamped joint.} \end{cases} \tag{2.1}$$

Note that this definition could be easily extended to more link and joint types. Also, the entries are defined intentionally by *positive integers numbers* in accordance with an isomorphism identifier explained later in Section 2.4.

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 its 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

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

   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:

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

   meaning that the joint between links 0 (ground) and 10 is a revolute joint (type 1), etc. In order to store a half of the off-diagonal entries, each pair of links in the domain of the map are sorted in an increasing order.

The resulting type adjacency matrix of the example considered in Figure 2.4 is:

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

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

Note that $\boldsymbol{T}$ can be represented by using the Adjacency Matrix, formed by zero and one entries, together with the vertex labels $\mathcal{V}(V)$, and the maps $\mathcal{T}_L$ and $\mathcal{T}_J$. 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 $\mathcal{T}_J$ as well as the joint labels in the function $\mathcal{E}$ remain unchanged. In this way, the definition of these two maps makes more efficient the identifier computing which is extensively used (see next section 2.4 and next chapter).

- In order to enumerate mechanisms (section 2.6), both link-types vectors $\boldsymbol{t}_L$'s and joint-types vectors $\boldsymbol{t}_J$'s can be easily enumerated in vectorial form and then assigned to the adjacency matrix of a given kinematic chain.

## 2.3. Isomorphism testing

In mechanism design, it is very important to recognize if two mechanisms are topologically equivalent. This is done by checking if their colored labeled graphs are isomorphic, or equivalently it implies –for the presented matricial representation– to identify if their two type adjacency matrices are the same for some permutation of rows and columns.

In the last three decades, various lines were followed to test isomorphisms of kinematic chains and mechanisms. In 1975, Uicker and Raicu [UR75] presented a numerical method for computing the coefficients of the *Characteristic Polynomial (CP)* of the adjacency matrix. Then, mainly motivated by kinematic chains enumeration, the interest in computational approaches grew considerably. These methods were briefly based on:

- *Classification of graph/structural properties*: number of links, number of joints, family of links assortments, family of contracted graphs, etc. [BF70, DF67, Tsa01];

- *Matrix Theory and Spectral Methods*: (i) the Characteristic Polynomial ($CP$) of the adjacency matrix ($A$), in both forms, analytical and numerical approaches [UR75, YH81, YH82, Tsa87], (ii) the eigenvectors and eigenvalues spectra of $A$ [ZL99, HZLW82]; these methods were recently reviewed in depth and developed by Sunkari [Sun06];

- *Code algorithms*: They use certificate codes constructed from the upper-right entries of the adjacency matrix and groups of permutations for relabeling the adjacency matrix for producing different certificates, e.g. MAX Code [AA86], MIN Code [AA87a, AA87b] and Degree Code [TL93].

- *Heuristics and artificial intelligence*: they use continuous optimization methods such as neural networks [KLZ99, HZLW82];

Most of the methods were recently reviewed by Mruthyunjaya [Mru03]. On the other hand, it must be remarked that there is another area of research, which is focused on generation methods to *avoid or minimize the isomorphism testing*. This means to generate the biggest possible number of non-isomorphic kinematic chains

or mechanisms at the outset. In this sense, we can mention the Group Theory approaches of Tuttle [Tut96], Yan and Hwang [YH91], and more recently, Yan and Hung [YH06], and Sunkari and Schmidt [Sun06, SS06]. However, they still need to use a mechanism identifier for efficient storage and retrieval of solutions.

The desired properties for an identifier are: *uniqueness*, *efficiency* and *decodability* [TL93, Tsa01]. Murphy *et al.* [MMH96] proposed the use of the characteristic polynomial of the compliant matrix $CP(\boldsymbol{CM}, x) = |x\boldsymbol{I} - \boldsymbol{CM}|$ for isomorphic mechanisms detection. The $CP$ lacks of uniqueness by itself but it was successfully used in combination with other structural properties. Additionally, it is non-decodable. The code-based algorithms have the three desired characteristics for an identifier.

## 2.4. A code-based mechanism identifier

The aim is to assign a code, $C_{\max}(\boldsymbol{T})$, in correspondence with the type adjacency matrix of a mechanism with the following properties:

1. $C_{\max}$ is *unique*: Two colored labeled graphs with the same $C_{\max}$ are isomorphic:

$$C_{\max}(G_1) = C_{\max}(G_2) \Leftrightarrow G_1 \cong G_2;$$

2. $C_{\max}$ is *efficient*: The computational cost grew in order less than factorial.

3. $C_{\max}$ is *decodable*: Given the $C_{\max}$, we may build the graph.

The code-based algorithms use two main fundamental basis:

- *Code* construction: Since the adjacency matrix determines the topological structure of a mechanism up to structural isomorphism, a code, $\mathsf{Code}(A)$, may be constructed from $A$ so as to use less memory. Also, the comparison between two codes must be properly defined.

- *Groups of vertices* $\Pi$: In the worst case, the size of the group of congruent permutations is $|\Pi| = n!$. The number of permutations can be reduced by using graph invariants for grouping the vertices. Then, the permutations are confined inside each group and then combined.

The minimum or maximum value of a code evaluated under all permutations of a given permutation group $\Pi$ is unique. Then, the algorithm consists in running a loop over each permutation in order to find:

$$C_{\max}(A) = \max\{\mathsf{Code}(A^p))\} \quad \forall \quad p \in \Pi$$

Next, a review of other algorithms is given in order to show how to adapt the method for $\boldsymbol{T}$.

### 2.4.1. Degree code

Many code-based methods were firstly designed for identifying isomorphisms of (0-1)-adjacency matrices. In order to represent graphs and to detect isomorphisms of graphs, Tang and Liu [TL93] developed the so-called Degree Code. The Code ($C$)

of the adjacency matrix of a graph is defined 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:

$$C(A) = \{\mathsf{Dec}(\mathsf{String}(A))\}.$$

The maximum integer that results from all $n!$ possible relabelings is called MAX Code, the minimum is the MIN Code. To reduce the number of permutations in $\Pi$ Tang and Liu proposed to add a previous grouping of vertices of equal degree, sort them by decreasing degree, and then combine the permutations inside each group for doing each relabeling. If the obtained group of permutations is denoted as $\Pi_d$, depending upon the graph structure the cardinality of this group is $|\Pi_d| \leq |\Pi|$. The maximum integer acquired by the binary string is called Degree Code ($DC$).

For instance, a simple four-bar kinematic chain labeled as shown on the top of Figure 2.5, has an adjacency matrix:

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

and therefore can be characterized by the integer 51:

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

Note that any other permutation of $A$ produces an equal or smaller code $C$.

The entire atlas of one-DOF kinematic chains may be very efficiently stored as a sorted list of integers (see Figure 2.5). For each KC, an integer assigning a *Level* in the atlas is used to map their respective *Degree Code* and the graphs are displayed with their Degree Code labeling.

To see the improvement of Tang and Liu's algorithm consider the Watt chain. It has six vertices, thus a trivial group of permutations has cardinality $|\Pi| = n! = 720$, but after grouping (in 2 vertices of degree 3, and 4 vertices of degree 2), it is considerably reduced to $|\Pi_d| = 2!4! = 48$ permutations.

#### A change for coding non-binary entries

Tang and Liu presented the possibility of taking into account *colored edges*. The only change in the Degree Code algorithm is that the string must be encoded 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 (see equation (2.1)). For example, if we have revolute and prismatic joints, then the required base is 3 ($b = 2 + 1$).

### 2.4.2. Diagonally Extended Degree Code

We extend the idea of Tang and Liu to define a *Diagonally Extended Degree Code* $DC_b^d$ *in base* $b$. For example, let $\boldsymbol{T}$ be a matrix which represents a topology with

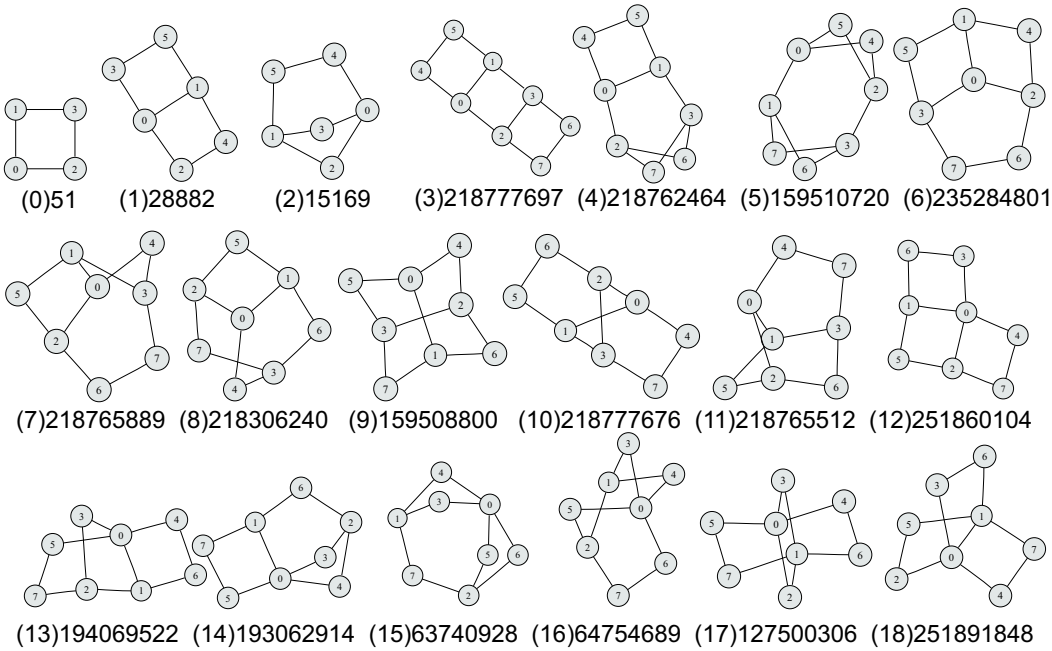



Figure 2.5: Labeled kinematic graphs with their respective *level*s and *Degree Code*s for the atlas of one-DOF kinematic chains with up to 3 independent loops, 8 links, 10 joints.

different types of links and joints, in which the "zero" color represents the ground link, while the "one" color indicates a rigid link. Its code may be computed as follows:

$$\boldsymbol{T} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ & 1 & 0 & 1 \\ & & 1 & 2 \\ \text{sym} & & & 1 \end{bmatrix} \qquad \Longrightarrow \qquad C_3^d(\boldsymbol{T}) = ([0110][101][12][1])_3 = (9034)_{10}.$$

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

$$DC_3^d(\boldsymbol{T}) = C_3^d(p(\boldsymbol{T})) = C_3^d\left(\begin{bmatrix} 1 & 2 & 1 & 0 \\ & 1 & 0 & 1 \\ & & 1 & 1 \\ \text{sym} & & & 0 \end{bmatrix}\right) = 35274.$$

Symbolically, the code can be expressed as

$$C_b^d(A) = \mathsf{Dec}(\mathsf{String}_b(t_{ij}(a_{ij}))), \quad t_{ij}(a_{ij}) = \begin{cases} \mathcal{T}_L(\mathcal{V}(v_i)) & i = j \\ \mathcal{T}_J(\mathcal{V}(v_i), \mathcal{V}(v_j)) & i \neq j \\ 0 & i \neq j \wedge a_{ij} = 0 \end{cases}$$

$$= \mathsf{Dec}([\ t_{11} \quad \dots \quad t_{n-1,n-1} \quad t_{n-1,n} \quad t_{n,n}\ ]_b)$$

$$= t_{n,n}b^0 + t_{n-1,n}b^1 + t_{n-1,n-1}b^2 + \cdots + t_{11}b^{\frac{n(n-1)}{2}+n-1}.$$

The basis $b$ must be saved as an additional part of the code. Data involved in the code are: the $\frac{n^2+n}{2}$ entries of $\boldsymbol{T}(\mathcal{V}, A, \mathcal{T}_L, \mathcal{T}_J)$, and $b$.

### Improvements in coding

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 of the concatenation row by row of the upper-triangular elements of the matrix including the diagonal:

$$\boldsymbol{T} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ & 1 & 0 & 1 \\ & & 1 & 2 \\ \text{sym} & & & 1 \end{bmatrix} \quad \Longrightarrow \quad C_3^r(\boldsymbol{T}) = \begin{bmatrix} (0110)_3 \\ (101)_3 \\ (12)_3 \\ (1)_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 5 \\ 1 \end{bmatrix}$$

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 permutation of $A''$ with $p = \{3, 2, 0, 1\}$:

$$DC_3^r(\boldsymbol{T}) = C_3^r(p(\boldsymbol{T})) = \begin{bmatrix} 48 \\ 10 \\ 4 \\ 0 \end{bmatrix}$$

In the example, only one comparison is needed ($48 > 12$) to validate that $DC_3^r(\boldsymbol{T}) > C_3^r(\boldsymbol{T})$.

The disadvantage of $DC_b^r$ is that the memory requirement for storing this code is increased to $n$, because we need to store $n$ entries of the code, the number of colors $b$, plus an additional integer $r$. For efficiency, $b$ may be the same for a complete family of stored mechanisms.

To analyze the advantage of $DC_b^r$, it must be pointed out that there is a compromise between three elements: (i) the internal data type representation of an integer number in the computer, (ii) the number of different colors $b$ used for enumerating mechanisms; and (iii) the size $n$ of the kinematic chains:

i) Beginning by the integer representation in the computer, we can use either the 32-bits strictly positive integer called "unsigned integer" for which the maximum is $2^{32} = 4,294,967,296$, or the 64-bits integer data type called "unsigned long long int" which doubles the required memory but extends the maximum number of distinct integers to $2^{64} = 18,446,744,073,709,551,616$.

ii) Let us denote $m$ the number of critical entries in the string of the code, which is $m = \frac{n^2+n}{2}$ for $DC_b$, and $m = n$ for the first entry of $DC_b^r$. This is the main advantage of $DC_b^r$.

iii) Suppose we use up to 16-link kinematic chains and we use the $DC_b^r$ represented by 64-bits integers, then we can manage up to 16 different elements (links or joints); see Table 2.1.

Table 2.1 shows that the Degree Code $DC$ can be used to represent binary adjacency matrices (excluding the diagonal), i.e. kinematic chains with up to 8 links with the 32-bits integer, or up to 10 links with the 64-bits one. The $DC_b^d$ allows to represent 10-link KC's using the 64-bits integer, where binary diagonal

| | | $DC$ | | $DC_b^d$ | | $DC_b^r$ | |
|---|---|---|---|---|---|---|---|
| | $m$ | $\frac{(n-1)n}{2}$ | $\frac{(n-1)n}{2}$ | $\frac{n^2+n}{2}$ | $\frac{n^2+n}{2}$ | $n$ | $n$ |
| $n$ | $b$ | $2^{\lfloor\frac{32}{m}\rfloor}$ | $2^{\lfloor\frac{64}{m}\rfloor}$ | $2^{\lfloor\frac{32}{m}\rfloor}$ | $2^{\lfloor\frac{64}{m}\rfloor}$ | $2^{\lfloor\frac{32}{m}\rfloor}$ | $2^{\lfloor\frac{64}{m}\rfloor}$ |
| 4 | | 32 | 1024 | 8 | 64 | 256 | 65,536 |
| 6 | | 4 | 16 | 2 | 8 | 32 | 1,024 |
| 8 | | 2 | 4 | 1 | 2 | 16 | 256 |
| 10 | | 1 | 2 | 1 | 2 | 8 | 64 |
| 12 | | 1 | 1 | 1 | 1 | 4 | 32 |
| 14 | | 1 | 1 | 1 | 1 | 4 | 16 |
| 16 | | 1 | 1 | 1 | 1 | 4 | 16 |
| 18 | | 1 | 1 | 1 | 1 | 2 | 8 |

Table 2.1: Number of vertices in One-DOF KC vs the capacity to store different colors (link/joint types) with 32- and 64-bits integers.

can be used to store inversions. The $DC_b^r$ allows to represent 18-link KC's or their inversions, in 32-bits, or mechanisms with 8 different colors in 64-bits version.

It should be mentioned that another feasible data type that can be used is the 8-*bits character*. Thus, the type adjacency matrix can be represented by a *word* with $\frac{n^2+n}{2}$ characters, i.e. $y_{\text{word}} = \left(\frac{n^2+n}{2}\right) \times 8$ bits without any limitation on the string length. Comparisons between words are trivially made in lexicographical order. Compared against the 64-bits $DC_b^r$ stored in $y_{\text{int}} = 64n$ bits, the word is more convenient for $n < 15$ ($y_{\text{word}} < y_{\text{int}}$), but it is not for mechanisms with more links.

### Improvements in grouping permutations

Consider the set of all graphs obtained by relabeling a graph $G$ having vertex set $V = \{v_0, v_1, \ldots, v_{n-1}\}$. The list

$$\boldsymbol{d}(G) = [\mathsf{deg}(v_0), \mathsf{deg}(v_1), \ldots, \mathsf{deg}(v_{n-1})],$$

called the *degree sequence* of the graph, is not an invariant. However, if the degree sequence is sorted either in ascending or descending order, then it is an invariant [KS99]. The degree sequence in descending order was used by Tang and Liu as *group of vertices* to define the domain of permutations in the Degree Code ($DC$), we denote this group as $\Pi_{\boldsymbol{d}}$.

Let us denote by $\mathsf{deg}_1(v)$ a structural property of a vertex $v$ called *first-neighbors degree of a vertex* defined as the degree of such vertex concatenated in a decimal number with the degrees of their first-neighbors sorted in descending order. We denote $\boldsymbol{d}_1(G)$ another invariant constructed by the list of such structural property $\mathsf{deg}_1(\bullet)$ for every vertex of the graph sorted in descending order. The resulting group of permutations is denoted as $\Pi_{\boldsymbol{d}_1}$.

In some situations, the use of $\boldsymbol{d}_1$ can reduce the number of permutations. For instance, the Stephenson chain shown in the level 2 of Figure 2.5 has the degree sequence:

$$\boldsymbol{d}(G_2) = [\mathsf{deg}(v_0), \mathsf{deg}(v_1), \ldots, \mathsf{deg}(v_5)] = [\underbrace{3,3}_{g_0}, \underbrace{2,2,2,2}_{g_1}],$$

so, it is needed to make $\prod_{i=0}^{|\Pi_{\boldsymbol{d}_1}|} |g_i|! = |g_0|!|g_1|! = 2!4! = 48$ permutations to compute the $DC$; while the sorted sequence with the first-neighbors degrees (the under-braced ones) has three groups:

$$\boldsymbol{d}_1(G_2) = [\mathsf{deg}_1(v_0), \mathsf{deg}_1(v_1), \ldots, \mathsf{deg}_1(v_5)] = [\underbrace{3222}_{g_0}, \underbrace{3222}_{g_0}, \underbrace{233}_{g_1}, \underbrace{233}_{g_1}, \underbrace{232}_{g_2}, \underbrace{232}_{g_2}],$$

thus, the number of permutations is reduced to $2!2!2! = 8$.

Since the kinematic chains have many symmetries, these groups of vertices with equal $\mathsf{deg}_1(\bullet)$, constitute, in general (not always), the sets of similar vertices. The Stephenson chain is a good example of this case. Similar vertices belong to the same cycle in the group of automorphisms $\mathsf{Aut}(G)$ of the graph. The number of groups of vertices cannot be reduced more than the size of the set of the *similar classes of vertices*. This means

$$|\mathsf{Aut}(G)| \leq |\Pi_{\text{Invariant}(G)}| \leq |\Pi_{\boldsymbol{d}_1}| \leq |\Pi_{\boldsymbol{d}}| \leq |\Pi| = n!.$$

The (8,10)-KC with level 13 of Figure 2.5 has 3 groups of vertices with equal $\mathsf{deg}(\bullet)$ while 6 groups of vertices with equal $\mathsf{deg}_1(\bullet)$;

$$[\mathsf{deg}_1(v_0), \mathsf{deg}_1(v_1), \ldots, \mathsf{deg}_1(v_7)] = [\underbrace{43222}_{g_0}, \underbrace{3432}_{g_1}, \underbrace{3322}_{g_2}, \underbrace{243}_{g_3}, \underbrace{242}_{g_4}, \underbrace{242}_{g_4}, \underbrace{232}_{g_5}, \underbrace{232}_{g_5}],$$

but 8 groups, or 8 classes of similar vertices with cardinality 1 each one, so this KC has not any symmetry, it is an *identity graph*. Even though, the number of permutations is reduced from $1!2!5! = 240$ by using $\boldsymbol{d}$ to only $1!1!1!1!2!2! = 4$ if we use $\boldsymbol{d}_1$.

On the other hand, in the Watt KC, the cardinalities of the groups $\boldsymbol{d}(G_1)$, $\boldsymbol{d}_1(G_1)$, and the similar classes of vertices, coincide and the number of permutations cannot be reduced. The main conclusion is that, since the number of permutations is graph structure dependent, then any *invariant* under permutations of the adjacency matrix or graph could be combined to form more groups of vertices with the aim of reducing the number of permutations for improving the efficiency in code computing.

Finally, to maintain an adequate compromise between the computational cost to obtain "groups of vertices" and the obtained "reduction in the number of permutations" the invariant $\boldsymbol{d}_1(G)$ is the preferred choice. On the other hand, the computation of the set of similar classes of vertices requires as many permutations as those needed for arriving to the $DC$ using $\boldsymbol{d}_1(G)$ as group of vertices.

## 2.5. Enumeration of one-DOF kinematic chains

A kinematic chain with $n$ links connected by $j$ simple joints is denoted as a $(n, j)$-KC. A simple-jointed kinematic chain with one degree of freedom $F = 1$ satisfies the Grübler movability criterion: $F = 3(n-1) - 2j = 1$. From Graph Theory, the number of independent loops $L$ could be computed as $L = j - n + 1$.

Methods given by Hwang and Hwang [HH92], Hsieh [Hsi92], Tsai [Tsa01], Tuttle [Tut96], Butcher and Hartman [BH05], and recently, Sunkari and Schmidt [SS06] among others, have allowed to enumerate 20,143,918 non-isomorphic one-DOF kinematic chains, see Table 2.2.

| $n$ | $j$ | $L$ | #KC |
|---|---|---|---|
| 4 | 4 | 1 | 1 |
| 6 | 7 | 2 | 2 |
| 8 | 10 | 3 | 16 |
| 10 | 13 | 4 | 230 |
| 12 | 16 | 5 | 6,856 |
| 14 | 19 | 6 | 318,162 |
| 16 | 22 | 7 | 19,819,281 |
| | | Total: | 20,143,918 |

Table 2.2: Number of enumerated non-isomorphic one-DOF kinematic chains.

The method found in Ref. [Tsa01] was followed to enumerate the first nineteen kinematic chains using the Degree Code as mechanism identifier, see Figure 2.5. 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; (iii) partially locked chains/subchains (degenerate of rigid mechanisms) and non-planar graphs were excluded. Kinematic chains with these characteristics are also referred in the literature as Basic Kinematic Chains (BKC). Hereafter, a one-DOF KC will be referred either as a $n$-link KC or $n$-bar KC.

The last items can be detected by means of different degeneracy theorems. Several computer implementations of these theorems lead to different results, for instance, Tuttle obtained 318, 126 14-link KC's against the 318, 162 KC's obtained by Butcher and Hartman [BH05]. In contradiction with the latter item, Belfiore [BH05] proved that the assumption "the graph of a kinematic chain is a planar graph" is erroneous. Sunkari [Sun06] confirmed this fact and also enumerated up to the 16-link KC's shown in Table 2.2, and proposed the fastest methods for enumeration and degeneracy testing.

## 2.6. Generation of atlases of mechanisms

In the proposed method, the number synthesis is computed in only one step thanks to the stored atlases (see Figure 2.3-**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, an alternative methodology to that presented by Murphy *et al.* [MMH96] for obtaining atlases of mechanisms by *assigning link types and joint types on a* four-bar *kinematic chain* will be given. This process is also known as specialization of mechanisms [YH91]. In this section, each kinematic chain of the atlas of Figure 2.5 is specialized 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. It should be mentioned here that the more different joint/link types are incorporated in the atlas, the more different constraints, and therefore problems, the method

will support. On the other hand, the results of this section will show that the small alphabet of links and joints shown in equation (2.1) will lead us to a big combinatorial explosion.

From the list of stored kinematic chains (KCs) a Degree Code is taken and then its adjacency matrix $A$ is retrieved, 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* $\boldsymbol{t}_J$ with $j$ entries. Then, the KC is specialized in two steps:

**Link specialization:** specialize all the link-types vectors $\boldsymbol{t}_L$ with a given alphabet (e.g. {0=ground,1=rigid}) and satisfying 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=revolute,2=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:

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

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:

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

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

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

Finally, when the entry $\boldsymbol{T}_{11}$ is distinct from zero, instead of storing $\boldsymbol{T}$, a permutation of it $p(\boldsymbol{T})$ is saved in such a way that the first vertex on $\boldsymbol{T}$ has always link type zero, i.e. coincident with the ground. This additional restriction allows to reduce the number of permutations needed in the subgraph search.

### 2.6.1. Atlas of 1-DOF Rigid Linkage Mechanisms: Linkage inversions

The kinematic chains shown in Figure 2.5 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 all joints are assumed of the revolute type (type 1). Therefore, only the first step of specialization is needed here to generate all link-types vectors $\boldsymbol{t}_L$ for an alphabet of two numbers $\{0 = \text{ground}, 1 = \text{rigid}\}$, constraining a single vertex labeled as zero to be the ground.

For each $n$-link 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 KC's shown in Figure 2.5, 77 non-isomorphic mechanisms by linkage inversion can be obtained (see the first row in Table 2.3.) These results coincide with those presented by other authors.

### 2.6.2. Atlas of 1-DOF Rigid Linkage Mechanisms with prismatic joints

In the previous atlas all joints were assumed to be 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 the joints alphabet is extended to $\{1 = \text{revolute}, 2 = \text{prismatic}\}$.

The resultant atlas has $54,222$ mechanisms (see the second row in Table 2.3). 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 [Sar97] used a rule developed by Freudenstein and Maki [FM79] 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 [Nie77] listed three restrictions: "**N1**: No link of a chain can contain only P-pairs whose 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 were designed, rule **F&M** is more restrictive than **N1**, **N2**, and **N3**. Furthermore, rule **N1** can be computed only after the dimensional synthesis stage since it is metric-dependent. Inspired on these rules, the following rules are proposed here:

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

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

| | KC Level | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| suffix | 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 |

| KC Level | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | Total |
| 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 |

Table 2.3: Rigid one-DOF linkage atlases (RigidOneDof)

Applying these restrictions in the second step of specialization of the mentioned 19 KC's, a total of 16,391 non-isomorphic mechanisms were identified.

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 it is shown in the last row of Table 2.3, the assignment results in 679 non-isomorphic mechanisms.

### 2.6.3. 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 $\{1 = \text{revolute}, 2 = \text{prismatic}, 3 = \text{flexible\_hinge}, 4 = \text{clamped}\}$.

Two important rules defined by Murphy [MMH96] are applied to constraint the enumeration.

**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.

| | KC Level | | | |
|---|---|---|---|---|
| suffix | 0 (Four-bar) | 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 |

Table 2.4: Compliant one-DOF linkage atlases (CompliantOneDof)

Table 2.4 shows that the number of solutions grows considerably with the number of links (KC level) and type of links and joints (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 allow 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.

From these atlases, only the CompliantOneDofR has been validated with the atlas presented by Howell in the Appendix G of his book [How01]. Note however that he displayed only 209 solutions for the four-bar compliant enumeration instead of 211. It was 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:

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

**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-bar 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-bar KC, there were $50,267$ non-isomorphic mechanisms from the $52 \times 2,187 = 113,724$ generated ones. In the Stephenson six-bar 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-bar KC's (not shown in Table 2.4).

The method presented by Yan [YH91] based on Group Theory, generates non-isomorphic mechanisms directly from the feasible space and could lead to more efficient ways of generating atlases. Nevertheless, it should be remarked that the cost in the presented approach 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.

# Chapter 3

# Synthesis by means of a subgraph search

In this chapter, the interrelation between mechanisms in the form of non-isomorphic subgraph occurrence is used as a new method for Number Synthesis.

The main contribution here is to take into account prescribed parts to move (such as fixations, moving bodies, joints and their interconnections) and the kinematic constraints imposed on them, to build a colored labeled graph called *initial graph*. This initial graph containing structural characteristics of the problem is used as a pattern to search inside a selected atlas of mechanisms also represented by graphs. The atlases –obtained as was detailed in Chapter 2– configure the space of desired solutions. Thus, the method for number synthesis consists in *a subgraph search subject to satisfy other requirements and design constraints of topological nature*.

The methodology is illustrated with examples for several kinematic tasks.

## 3.1. Graph representation of kinematic problems

Starting from functional requirements, the designer selects the *structural characteristics* to draw the existing parts as 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. [GC01]. In a synthesis problem, the *skeleton diagram* results in only some parts that are imposed to exist in the final synthesized mechanism. In addition, the user specifies on the imposed parts 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 $\mathcal{J}$; the entered parameter must be coherent with the joint type element, i.e. angles $\alpha_j$ for revolute joint elements, and displacements $\rho_j$ for the prismatic joint ones, both expressed in relative coordinates from the initial position. We shall remark that all prescriptions are expressed "in relative coordinates from the initial position" because this particular way to define the problem allows the user to leave some angles or displacements

without being defined for some precise positions.

The subgraph search process is general and suits to a wide range of problems; nevertheless, three typical cases may be distinguished by combinations of the sets $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{J}$:

**Path Following (PF).** To define a trajectory, a set of node displacements is given

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

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 Figure 3.1, if two displacements are desired on node $N_1$, we declare three triplets

$$\mathcal{D} = \{N_1, 0, (0,0);\ N_1, 1, \boldsymbol{d}_1;\ N_1, 2, \boldsymbol{d}_2\}.$$

If timing is prescribed, joint parameters can be declared as

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

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

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

on node $N_5$, and the rotations for rigid-body $E_1$ (Figure 3.2) 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 Figure 3.3, by defining the sets

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

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 Figure 2.2 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 final positions are prescribed for the prismatic joint $E_3$ in the form

$$\mathcal{J} = \{E_3, 0, \boldsymbol{d}_0;\ E_3, 3, \boldsymbol{d}_3\}.$$

The intermediate inputs are unknown and are computed by the dimensional synthesis program. Data is completed by imposing the angular displacements on link $E_8$ by rotating joint $E_{13}$:

$$\mathcal{J} = \{E_{13}, 0, 0;\ E_{13}, 1, \beta_1;\ E_{13}, 2, \beta_2;\ E_{13}, 3, \beta_3\}$$

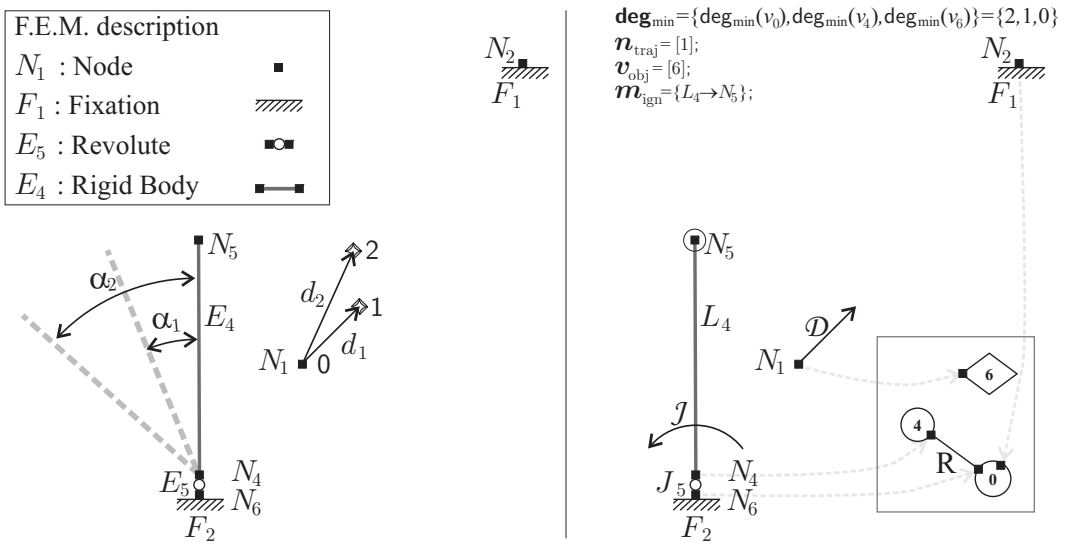



Figure 3.1: Path following.

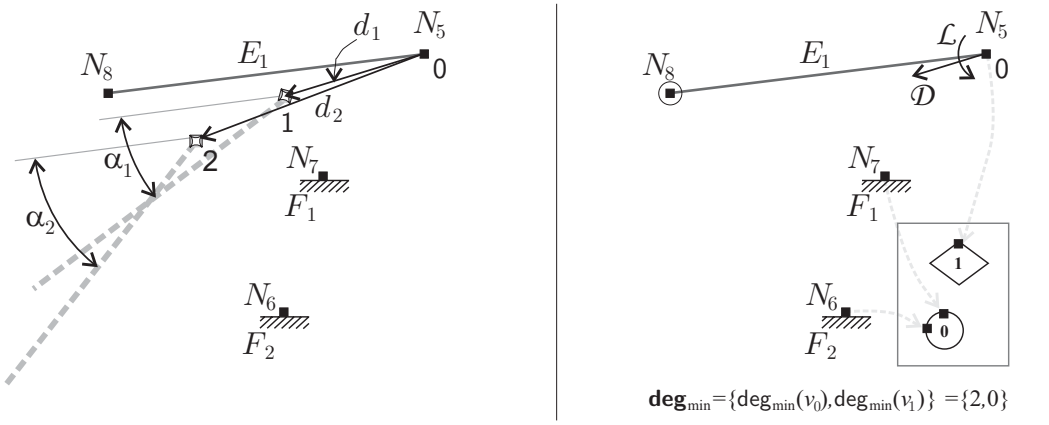



Figure 3.2: Rigid-body guidance.

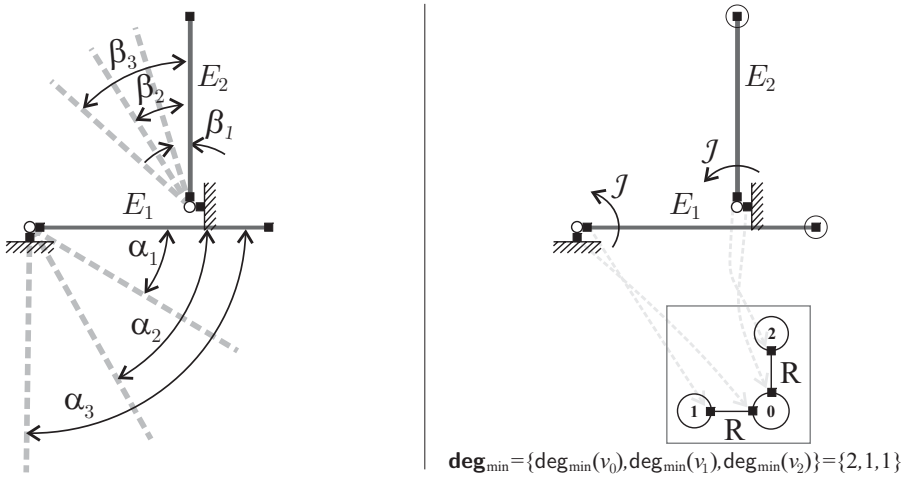



Figure 3.3: Function generation.

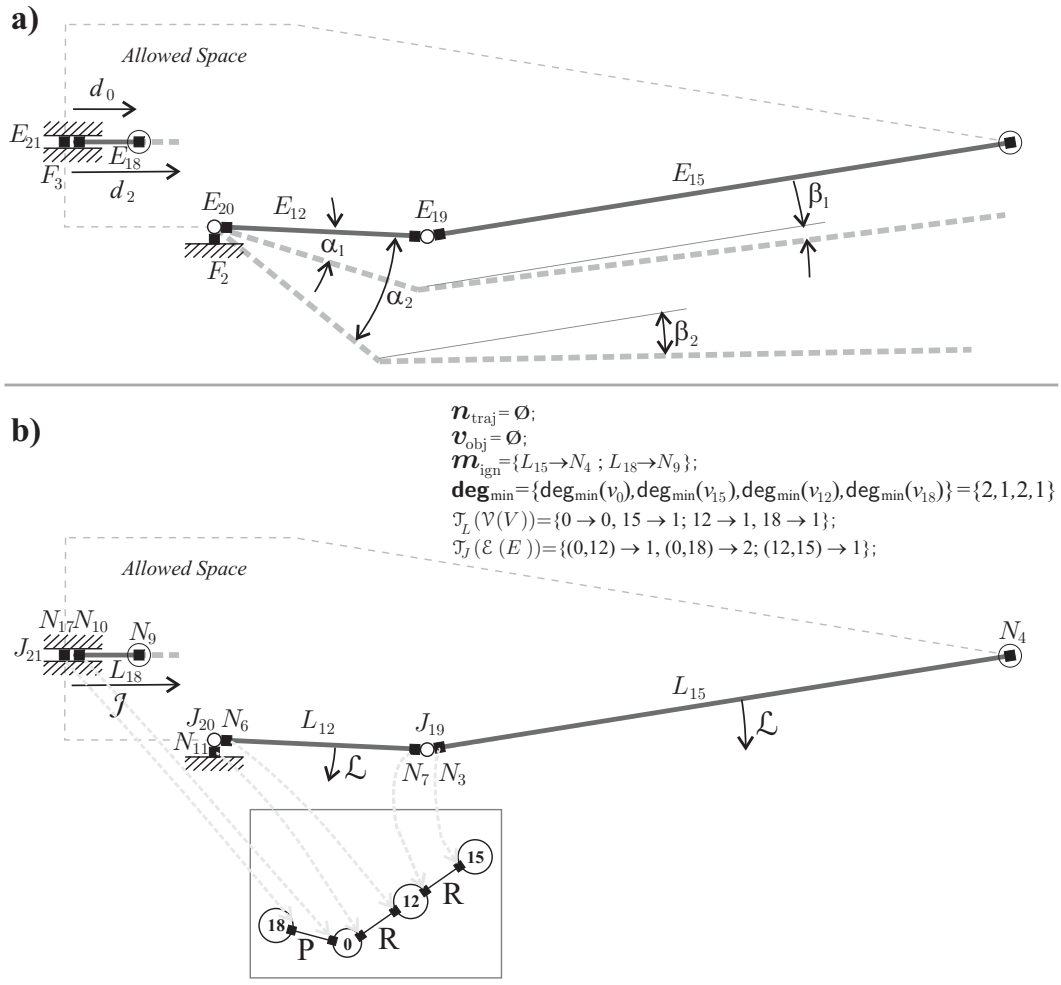



Figure 3.4: Combined double function generation: a) FEM description and defined motion constraints, b) Initial graph construction and auxiliary data collection.

and on link $E_{10}$ by rotating joint $E_{14}$:

$$\mathcal{J} = \{E_{14}, 0, 0;\ E_{14}, 1, \alpha_1;\ E_{14}, 2, \alpha_2;\ E_{14}, 3, \alpha_3\}.$$

A similar prescription for the actuator is the three-position coordination problem illustrated in Figure 3.4.

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.

### 3.1.1. FEM to graph translation –Initial graph

In a FEM description, the nodes can have different attributes: e.g. clamped node (with fixations), isolated node with prescribed movement, node taking part in the geometry of a rigid-body. A rigid-body can have $n_b$ nodes (with $n_b \geq 1$),

$$n_b = n_c + n_p + n_s,$$

where, $n_c$ nodes are used to assemble the body with nodes of other bodies by means of joint *connections*, $n_p$ nodes are not assembled but have *prescribed* movements $\mathcal{D}$, and $n_s$ nodes only give the *shape* of the body and can be used later in the dimensional synthesis stage only as checking points of the task, or to define the restricted area (or space) where the other links must be contained for each configuration.

The *initial graph* represents the initially imposed parts. From the starting parts definition $\mathcal{M}_{\text{ini}}$, we build the associated graph $G_{\text{ini}}$ following these simple rules:

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

V1) **Isolated nodes:** For each isolated node with prescribed movements, an isolated vertex in the graph is assigned; such nodes will be considered as trajectory nodes or "tracer points" and do not constraint the degree of the assigned vertex.

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

V3) **Fixations:** Conventionally, the ground link will be the vertex labeled as zero: $\mathcal{V}(v_0) = 0$. This link could also have a number of nodes connected to bodies $n_c$, and a number of isolated fixed nodes $n_p$. Depending on the number of grounded bodies and fixations, this vertex may be binary, ternary, etc. The prescribed degree of the vertex zero is

$$\mathsf{deg}_{\min}(v_0) = n_c + n_p.$$

**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).

Apart from the initial graph construction, other *auxiliary data* must be collected and stored to be used in the number synthesis and dimensional synthesis stages:

*Trajectory Nodes:* Every trajectory node is stored in a vector denoted $\boldsymbol{n}_{\text{traj}}$ and its corresponding assigned vertex is stored in a separated vector called $\boldsymbol{v}_{\text{obj}}$. For convenience, the two vectors are preferred to a map.

*Ignored Nodes:* These nodes are ignored for the type synthesis purposes. They are stored in an auxiliary element-to-node integer multi-map[1] $\boldsymbol{m}_{\text{ign}}$. These nodes which do not belong to any joint neither have prescribed motion are shown as "circled" in Figures 3.1 to 3.4.

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

[1] A multi-map is a correspondence data structure that allows to assign "multiple values" to the same "key", i.e., many nodes (values) for the same link (key).

An example for the application of these rules will be given for the path following with prescribed timing problem shown in Figure 3.1. The rigid-body element $E_4$ has $n_b = 2$ nodes, where $n_p = 0$, $n_j = 1$ and $n_c = 1$. Following rule V1 the element is translated to a vertex $v_4$ with minimum degree 1. The node $N_5$ shown as "circled" in the element does not participate in the initial graph; however, it is stored in a map indicating $\boldsymbol{m}_{\text{ign}} = \{L_4 \rightarrow N_5\}$. Then, for the isolated node $N_1$ which will develop the required trajectory $\mathcal{D}$, rule V2 is used to create a new vertex $v_6$ with zero degree. Additionally, two integer variables are set and used later in the synthesis process. They are the ID of the node which will develop the required trajectory $\boldsymbol{n}_{\text{traj}} = [1]$, and the objective vertex defined as the vertex of the initial graph which has the trajectory node $\boldsymbol{v}_{\text{obj}} = [6]$. Finally, by means of rule V3, the two fixed nodes $N_2$ and $N_6$, introduce two constraints to the degree of the ground vertex $v_0$. A list called *minimum degree of vertices* has the *minimum degree constraint* for each vertex. For example, it is filled as $\mathbf{deg}_{\min} = \{\mathsf{deg}_{\min}(v_0), \mathsf{deg}_{\min}(v_4), \mathsf{deg}_{\min}(v_6)\} = \{2, 1, 0\}$.

### Structure of the initial graph

Note that the freedom in the problem definition 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 (Figure 3.3), the initial graph is a tree where the ground can be considered as the root. Another problem which has a tree as initial graph is shown in Figure 3.4.

Tasks can be classified according to the number of graph components in the initial graph associated with the problem (e.g. compare Figures 2.2, 3.1, 3.2, and 3.3). For PF (Figure 3.1) and RBG (Figure 3.2) 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*[1]. After applying the rules for construction of the initial graph we can get more than one objective vertex. For instance, a wing flap/tab coordination problem can be divided and then stated as a double "flap guidance" and "tab guidance" problem.

In a few words, the initial graph must correspond to a substructure of a mechanism, with the constraint coming from the FEM convention that states that defined joints must always have their two nodes defined and belonging to an existing body. In some cases where there are more complex situations than bodies connected to ground and free bodies, parts with known dimensions must be checked. 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 Grübler equation [PC08].

### Type Adjacency Matrix of the initial graph

From the processing of the FEM description we get all elements ($A$, $\mathcal{V}$, $\mathcal{E}$, $\mathcal{T}_L$ and $\mathcal{T}_J$) to build the initial graph as well as the type adjacency matrix $\boldsymbol{T}_{\text{ini}}$. However, it must be noted that the representation of isolated vertices is not considered in the classical definition of the adjacency matrix $A$. Isolated vertices are simply added as a null row and null column in such a way that $A$ allows the representation of

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

non-connected graphs. Then, $\boldsymbol{T}$ is implicitly defined taking the structure of $A$ and the link and joint type integer mappings, $\mathfrak{T}_L$ and $\mathfrak{T}_J$.

For example, the adjacency matrix of the initial graph shown in Figure 3.1 is:

$$
\begin{aligned}
V_{\text{ini}} &= \{\ 0 \quad 1 \quad 2\ \}, \quad E_{\text{ini}} = \{(0,4)\} \\
\mathcal{V}(V_{\text{ini}}) &= \{\ 0 \quad 4 \quad 6\}\ , \quad \mathcal{E}(E_{\text{ini}}) = \{(0,4) \to 5\} \\
A_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\
\mathfrak{T}_L(\mathcal{V}(V_{\text{ini}})) &= \{0 \to 0, 4 \to 1, 6 \to 1\}, \\
\mathfrak{T}_J(\mathcal{V}(V_{\text{ini}}) \times \mathcal{V}(V_{\text{ini}})) &= \{(0,4) \to 1\}, \\
\boldsymbol{T}_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
$$

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

## 3.2. Relationships between initial and stored mechanisms

Once the initial graph is defined, denoted as $G_{\text{ini}}(V_{\text{ini}}, E_{\text{ini}})$, it can be looked for as subgraph occurrences inside a given graph taken from an atlas of mechanisms, denoted as $G_A(V_A, E_A)$. But, prior to the description of the algorithm for subgraph detection it is necessary to define a *distance constraint* and a matricial tool used inside such algorithm after each subgraph detection. Both definitions express a relationship between $G_{\text{ini}}$ and $G_A$.

### 3.2.1. Distance from the Objective Vertex

The objective vertex $v_{\text{obj}}$ in PF and RBG problems, i.e. the vertex containing the node that develops the prescribed task, is chosen counting a given distance from the ground $v_0$. This distance is defined as the minimal number of vertices going through a path from the ground to the objective vertex: $\min d(v_0, v_{\text{obj}})$. The lower bound for this distance is set to 2, while the upper bound 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:

$$2 \leq \min\left(d(v_0, v_{\text{obj}})\right) \leq n_{pp} - 1 \tag{3.1}$$

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), additionally more than one objective vertex might be defined on the kinematic task. The subgraph search proposed in the next section (i) explores automatically all possible ways of assigning the objective vertices that satisfy the distance constraint and (ii) avoids isomorphisms due to the matricial tool introduced in the following sub-section.

### 3.2.2. Synthesis Adjacency Matrix

The subgraph occurrence $G_{\text{ini}} \subseteq G_A$ and the matching of the link and joint types of each vertex and edge, respectively, imply that $G_A$ is a *structurally feasible* mechanism alternative. However, for a given $G_A$, there could be many isomorphic occurrences of $G_{\text{ini}}$.

To explore systematically the occurrences $G_{\text{ini}} \subseteq G_A$ the vertices of $G_{\text{ini}}$ are exhaustively compared with the subgraph constituted by the first $n_{\text{ini}}$ vertices of a $(n_{\text{ini}}-1)$-permutation of the $n_{\text{A}}$ vertices of $G_A$ and then the comparisons related to edges are made. Each permuted graph is denoted as $G_A^p(V_A^p, E_A^p)$. By definition, the Degree Code of $\boldsymbol{T}(G_A^p)$ is unique for any permutation $p$.

In order to identify all non-isomorphic locations of the subgraph $G_{\text{ini}}$ it is necessary to make some modifications in the $\boldsymbol{T}(G_A^p)$ matrix. For this purpose the *Synthesis Adjacency matrix* ($\boldsymbol{S}$) is constructed. 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, different integers are assigned to diagonal entries corresponding to each prescribed vertex regardless of its link type. Thus, each prescribed part is considered 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 $k = b+j$; $j = 0, 1, ..., n_{\text{ini}}-1$, 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 encode $\boldsymbol{S}$ is the number of prescribed links plus the number of colors of the graphs taken from the atlas, i.e., $n_{\text{ini}} + b$.

## 3.3. Subgraph search

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:

$$G_{\text{ini}} \subseteq G_A \tag{3.2}$$

with $G_A$ a graph from the atlas. However, the following constraints have to be also satisfied:

- the equality constraint

$$\boldsymbol{T}(G_{\text{ini}}) = \boldsymbol{T}(H_A), \quad G_{\text{ini}} \cong H_A, \quad H_A \subseteq G_A, \tag{3.3}$$

  i.e. the link and joint types in $G_{\text{ini}}$ should match exactly those of the corresponding subgraph, $H_A$, in $G_A$.

- the distance constraint given by equation (3.1),

- the isomorphism constraint, $DC_b^d(\boldsymbol{S}(G_{\text{ini}}, G_{\text{A}}))$ must be different from all previous answers, and
- the pseudo-isomorphism constraint; no solution has a previous one as subgraph (explained later).

Let $\boldsymbol{l}_{\text{ini}} = \{l_0, l_1, ..., l_{n_{\text{ini}}-1}\}$ be the labels of the initial graph $G_{\text{ini}}$, and $\mathcal{V}_{\text{ini}}$ the function which applies the set $\boldsymbol{l}_{\text{ini}}$ to the set of unlabeled vertices $V_{\text{ini}} = \{v_0, v_1, ..., v_{n_{\text{ini}}-1}\} = \{0, 1, ..., n_{\text{ini}} - 1\}$, 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 $G_{\text{ini}}$. 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 the 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**; else, 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 $G_{\text{ini}}$ 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.:

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

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 of the first $n_{\text{ini}}$ 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 $G_{\text{ini}}$ 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 label functions $\mathcal{V}_{\text{ini}}$ and $\mathcal{E}_{\text{ini}}$ for vertices and edges homologous to $G_{\text{ini}}$ in the previous step, and for vertices and edges which are not in $G_{\text{ini}}$ using new labels starting from

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

The new label functions are denoted 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 the

$DC_b^d(\boldsymbol{S})$. If the code is not already stored, save $DC_b^d(\boldsymbol{S})$, 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 encoded 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 $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$, 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 [PC05a].

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 Figure 3.1.

From the available atlases of rigid mechanisms (see Table 2.3), 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 equation (3.1). This search led to 489 non-isomorphic solutions of increasing complexity for this problem. A second running in which pseudo-isomorphic mechanisms were rejected (see section 3.4) resulted in 214 solutions. The first 19 solutions found in the latter case are shown in Figure 3.5.

Figure 3.6 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-bar 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.

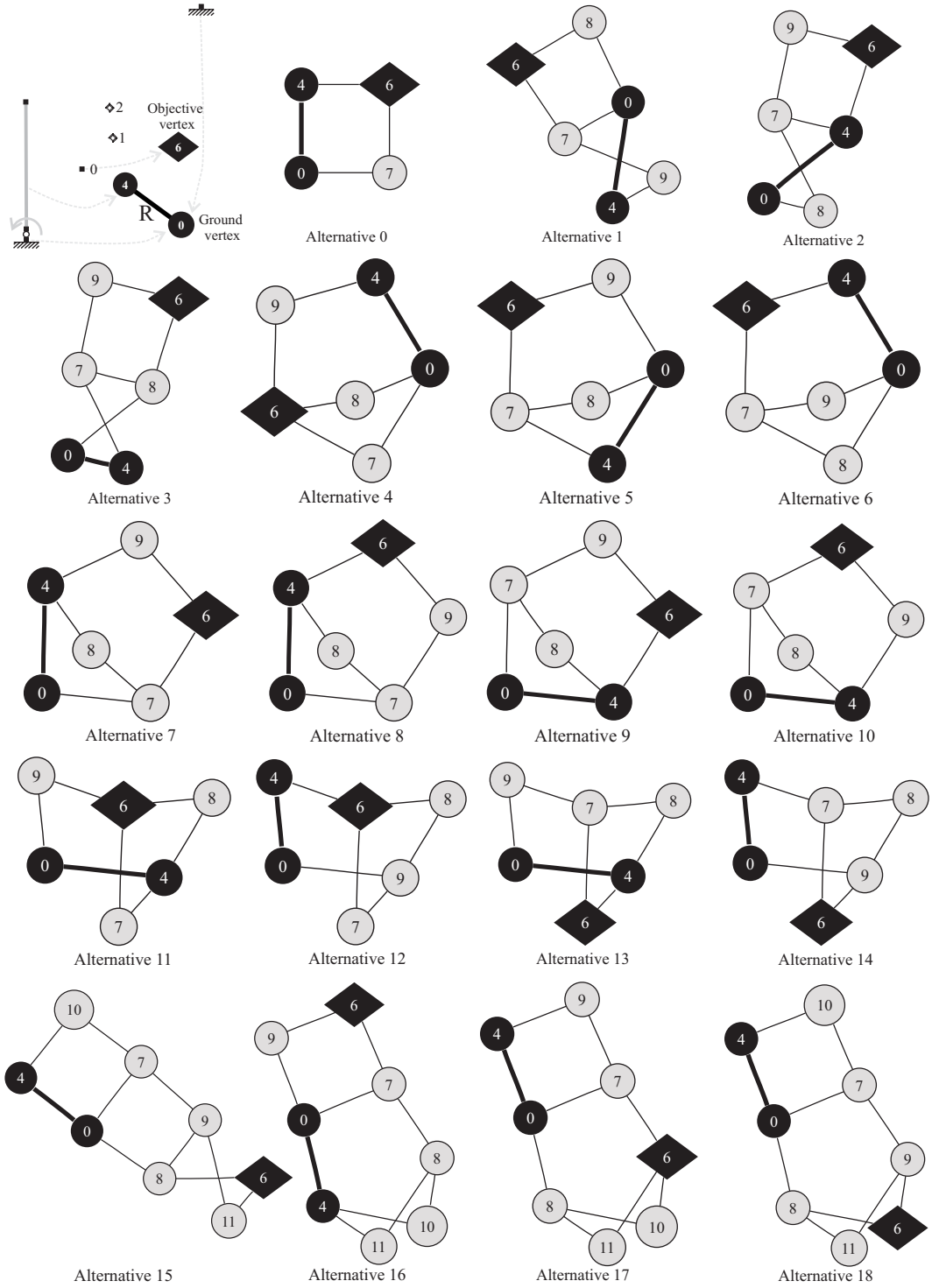



Figure 3.5: First 19 non-isomorphic occurrences of an initial graph

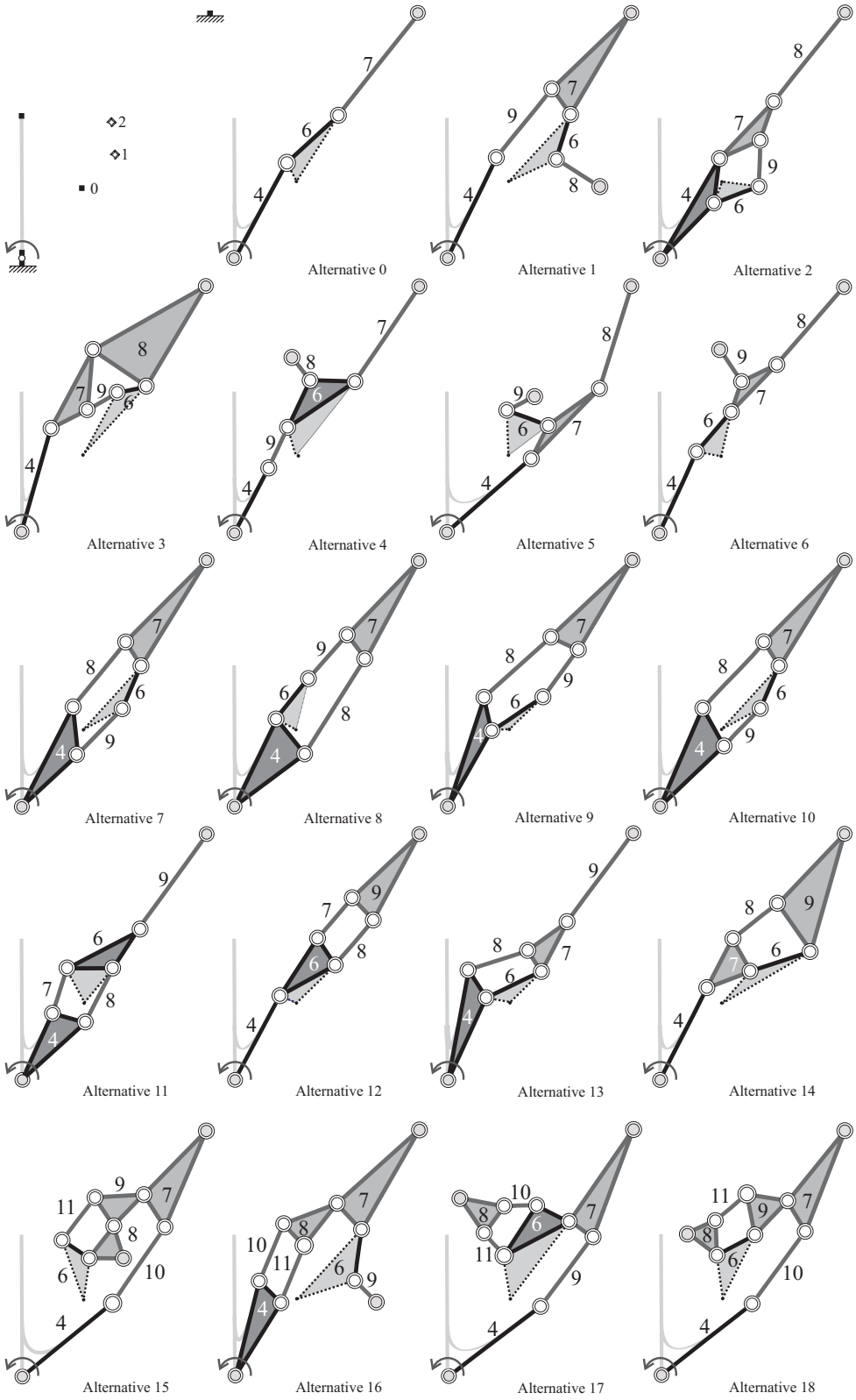



Figure 3.6: Physical sketches for a path following task.

## Example 2: Double Function Generation

The problem shown in Figure 2.2 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 $\alpha$ vs $\beta$ 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 figure. Note that there is no objective vertex, so the distance constraint was not imposed in this case. The selected atlas was RigidOneDofROneP.

The first 10 solutions obtained are shown in Figures 3.7 and 3.8. An automatic sketch is drawn below each solution for clarity.

Note that the first two synthesized mechanisms (*Alternatives 0* and *1*) shown in Figure 2.3 were found inside the same level in the specialized atlas, i.e, from permutations of the same 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:

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

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

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

for the first alternative, and

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

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

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

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

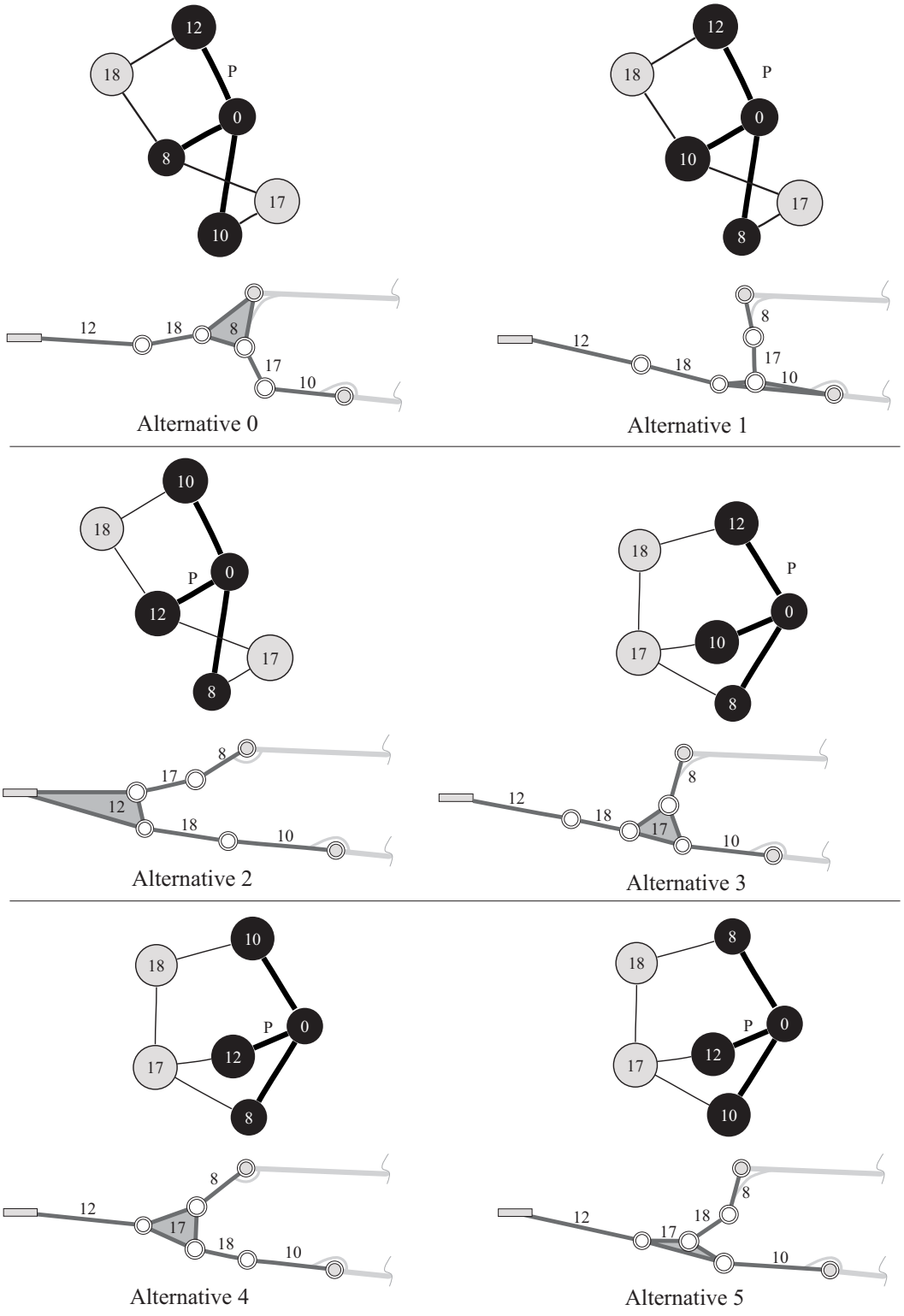



Figure 3.7: First 10 non-isomorphic occurrences of an initial graph inside the atlas (continued).

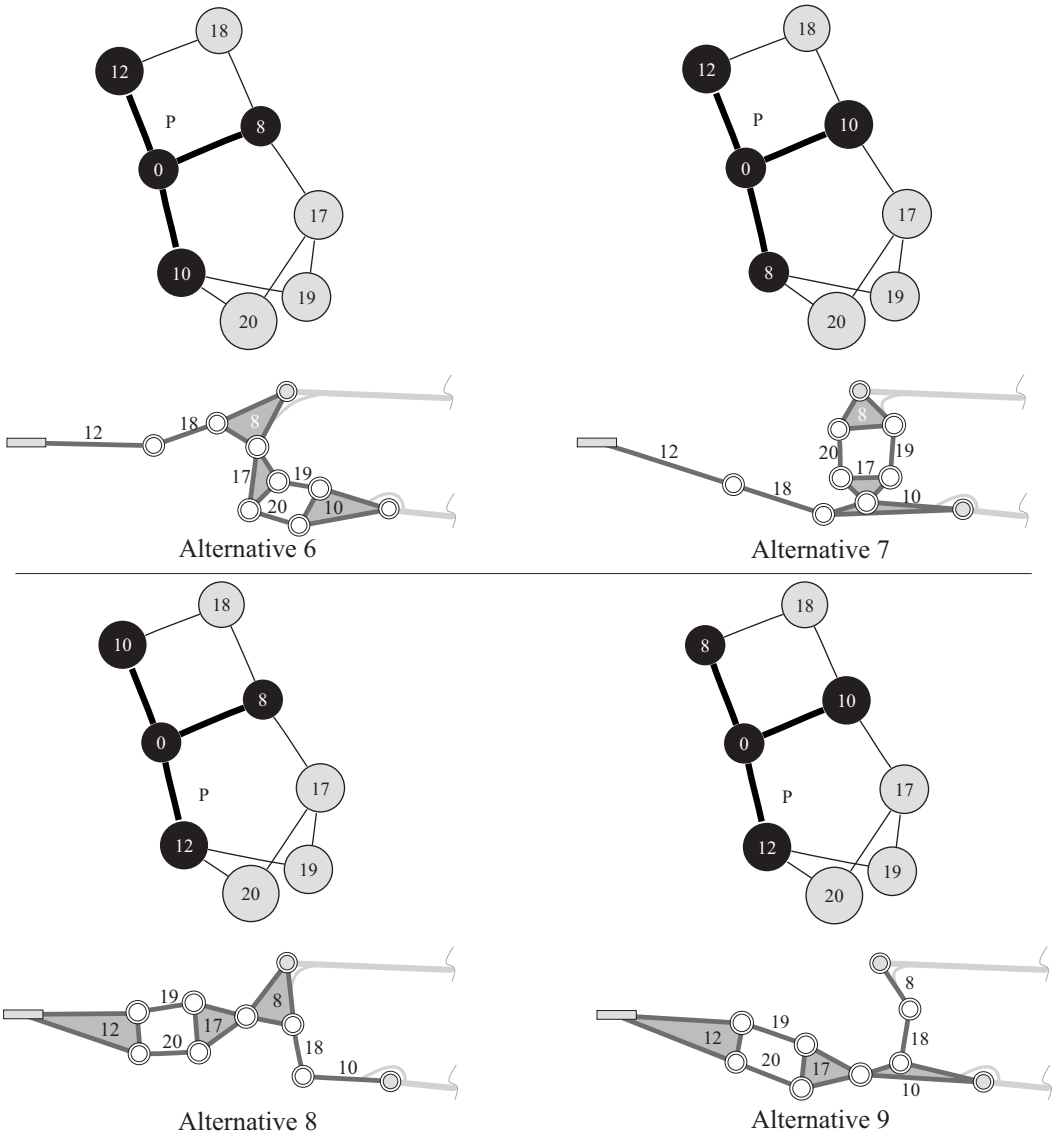



Figure 3.8: First 10 non-isomorphic occurrences of an initial graph inside the atlas.

## 3.4. 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[1].

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 is 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 a 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 Figure 3.9 as *Alternative 1'*, it has *Alternative 0* as a 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 Figure 3.6.

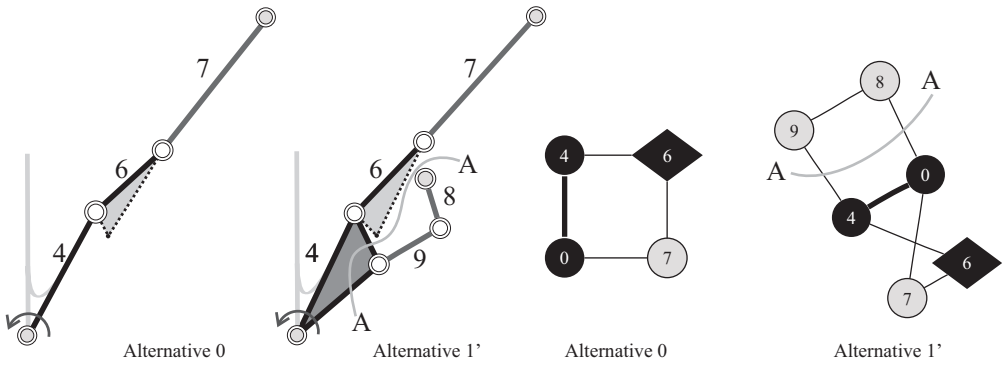



Figure 3.9: First pseudo-isomorphic occurrence in the path following example.

## 3.5. Chapter 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 eliminated the human effort to find the mechanisms that have the prescribed parts and also diminished considerably the computation time in the remaining 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 [JÁCC97, LCL00] or in a complex number formulation based on the decomposition of the topology into open chains [SE84, Car02, PC05a, PC06].

[1] 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.

Other remarkable characteristics are:

- The FEM description of the kinematic problem in conjunction with the rules given for the initial graph construction are the key to adapt the problem into a graph problem. Note that this technique can be easily extended to three-dimensional space.
- The use of a previously specialized atlas assures that all candidate mechanisms satisfy the required degree of freedom without containing rigid sub-chains, and reduces 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 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.

In the literature we can find fragmented developments on type synthesis. A lot of research was done on kinematic chains enumeration and atlas construction [Mru03], but few works explain how to make use of these atlases. The presented method will be a useful point of reference for the automated use of all the enumerations of mechanisms nowadays available.

# Part II

# Initial Dimensioning

# Chapter 4

# Analytical Synthesis

*Analytical Synthesis* is used for sizing a mechanism by means of closed-form equations. The method is applicable when the topology is decomposed into single sub-systems called *Single Open Chains (SOCs)* and the *task is simplified* by defining a number of finitely separated displacements and/or orientations called *precision positions*. The method for this simplified synthesis problem is known as the Precision-Point Method (PPM).

Some introductory topics about dimensional synthesis are needed to justify (a) some design constraints used in the subgraph search, and (b) some necessary conditions implemented as rules in the SOCs decomposition procedures. The decomposition method for obtaining SOCs will be presented in the next chapter, so the study of open chains in this chapter will be given *without regard for the function of each link or joint in the mechanism* (motorized joint, driver-link, floating-link, etc.).

In this chapter, the exact equations for dimensional synthesis of single open chains are reviewed and some non-standard equations are developed. Finally, the data structure used to combine graph theory, complex-numbers, and the Finite Element description is shown.

## 4.1. Introduction

Throughout the last century, the Precision-Point Method has been widely developed for planar as well as for spatial mechanisms. Freudenstein and Sandor [FS59] proposed the use of *complex numbers* for the representation of the significant dimensions of links in planar mechanisms. Complex-number algebra proved to be useful for solving the displacement, velocity and acceleration equations for kinematic synthesis of pure linkages (with lower-pairs: revolute and prismatic) but also for cam- and geared-linkages (higher-pairs) [Hal61, RF63, HD64, SE84, ES97].

For planar open chains, the joint preceding each link admits either the *rotation* or the *sliding* of the link, which is easily modelled in terms of complex numbers by the *Euler operator* and a *stretch factor* respectively. Using these tools, the Loop-Closure equations can be stated for a given number of precision positions. Then, depending upon both the number of links in the chain and the number of precision positions of the problem, the resultant system of equations can be linear or non-linear in the link dimensions (unknowns). Fortunately, some non-linear equations can be solved by manipulating compatibility conditions and their geometrical relationships with the data, using complex numbers [HD64, SE84, ES97, LEJ96].

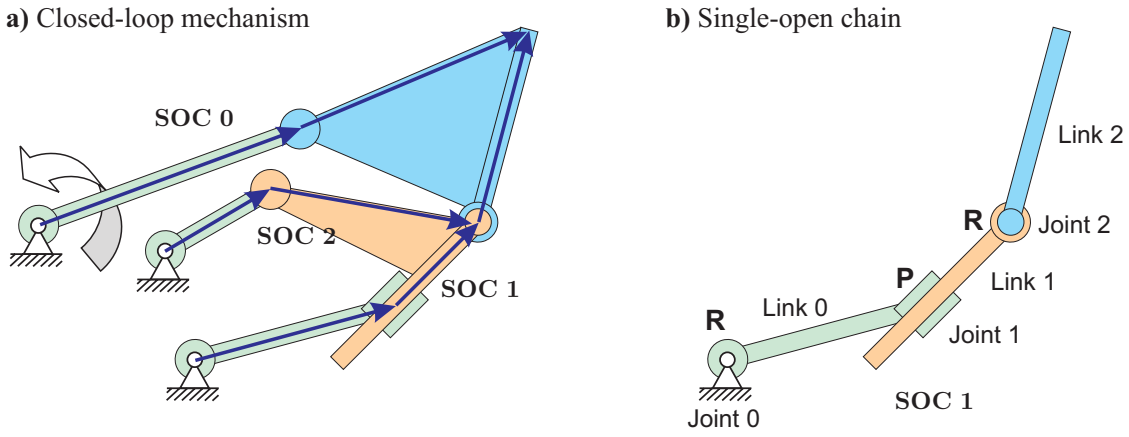



Figure 4.1: Notation for numerating links and joints in SOCs.

## 4.2. Data of the problem

The data necessary to solve dimensionally a single open-chain by means of analytical methods using the complex-number approach are (see notation in Figure 4.1-b and examples in Figure 4.2):

- **Number of links** $n_L$ in the open chain. The SOCs were traditionally called *dyad* ($n_L = 2$), *triad* ($n_L = 3$), *quadriad* ($n_L = 4$), etc. We numbered the links in zero base as $l = 0, \ldots, n_L - 1$.

- **Prescribed motion constraints**. Rotations and translations required at a given time may be applied either on joints or links. An individual motion constraint specified for time $j$ must be relative[1] to time 0.

- **Number of precision positions** $n_{pp}$ defined in the task. The notion of finite *precision position* for a SOC is used to describe the states or configuration of the chain members.

- **Time** $j$. In many problems of position coordination, variable $j$ is considered as a pseudo-time because it only expresses simultaneity within which the motion constraints must coincide. Therefore, the real elapsed time between configurations may differ from 1 sec.

- **End-points displacements**. Two sets of displacements on the *tail* $\boldsymbol{h}^j$ of the SOC and on the *tip* or effector point $\boldsymbol{g}^j$ must be defined.

- **End-points positions**. The position of at least one of the end-point nodes, $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$, must be known.

[1]Note that this way of defining coordinates "relative to the initial position" is characteristic of synthesis problems. This definition has a two-fold purpose: (i) at the definition level, to enable the user for leaving a motion undefined, and (ii) at the solution stage, to facilitate the easily decoupling of the unknowns from the set of Loop-Closure Equations. It is different from other coordinate formalisms, such as *relative* (to the previous position), *absolute*, and *natural* coordinates.

- **Type of joint preceding each link**. In the complex-number approach, a joint is on the tail of each complex number representing each link (compare for example, the different behaviors of the second link in Figure 4.2-a and b).

Given some data, the problem consists in finding a set of $n_L$ links represented by the complex-numbers $\boldsymbol{Z}_l$, which pass through a number of $n_{pp}$ precision positions, subject to prescribed motion constraints. It is also important to determine the minimal motion constraints needed to well pose a problem. Note that in synthesis problems, motion constraints of a SOC are imposed either by the user in the initial task definition or by a previous SOC that shares a link if we follow a sequential procedure for solving *ordered* SOCs (presented in the next chapter), see Figure 4.1-left.

## 4.3. Review of algebraic methods of synthesis

Since the manipulation of links as complex numbers is convenient, the system of coordinates is interchangeably taken as $x$-$O$-$y$ in vectorial form or $x$-$O$-$iy$ in the complex plane. Symbol $i \equiv \sqrt{-1}$ denotes the Euler's imaginary unit.

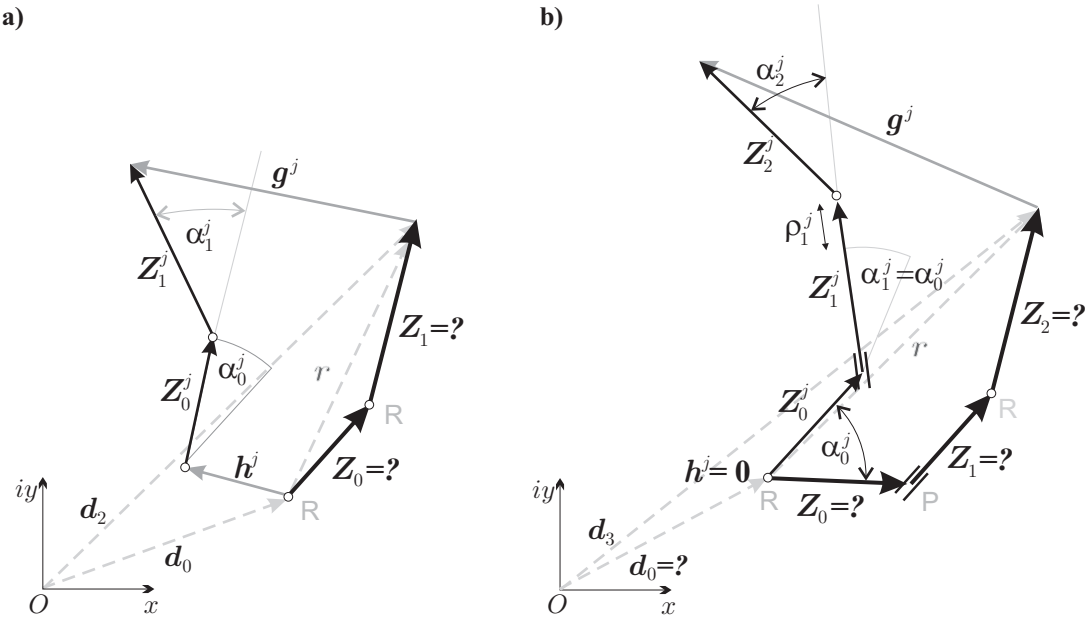



Figure 4.2: Examples of data modeled by complex numbers: (a) for a RR-Dyad and (b) for a RPR-Triad. The initial and $j$-th precision positions are shown.

When the chains move to the $j$-th precision position, the obtained configuration is characterized by the nature of each joint, that is, revolute joints permit link rotations $\alpha_l^j$ (see for example, the first link of Figure 4.2-a where $\boldsymbol{Z}_0^j = \boldsymbol{Z}_0 e^{i\alpha_0^j}$), and prismatic joints permit stretch factors $\rho_l^j$ through joint direction of the subsequent link but preserve a fixed angle with the previous link/complex number, e.g. the second link of Figure 4.2-b where $\boldsymbol{Z}_1^j = \rho_1^j \boldsymbol{Z}_1 e^{i\alpha_0^j}$.

### 4.3.1. Standard synthesis

Using this notation we can write the *Loop-Closure Equations* to solve. For example, the RR-dyad has the expression

$$\underbrace{\boldsymbol{Z}_0 + \boldsymbol{Z}_1}_{\text{initial}} + \underbrace{\boldsymbol{g}^j - \boldsymbol{Z}_0 e^{i\alpha_0^j} - \boldsymbol{Z}_1 e^{i\alpha_1^j} - \boldsymbol{h}^j}_{j-\text{th position}} = \boldsymbol{O}, \quad j = 1, \ldots, n_{pp} - 1. \tag{4.1}$$

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

$$\boldsymbol{Z}_0(e^{i\alpha_0^j} - 1) + \boldsymbol{Z}_1(e^{i\alpha_1^j} - 1) = \boldsymbol{\delta}^j, \quad j = 1, \ldots, n_{pp} - 1. \tag{4.2}$$

Expression (4.2) is known as the *Standard-Form Equation* for a Dyad.

### 4.3.2. Linear solution

When $n_{pp} = n_L + 1$, the associated nonhomogeneous complex-number system is linear and can be easily solved. For instance, if three positions (initial and $j = 1, 2$) are prescribed for a dyad, and (i) one of the sets $\boldsymbol{h}^j$ or $\boldsymbol{g}^j$ is null (it is said that the pivot location will be computed), and (ii) one of the end-point positions $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$ is unknown, the resultant system is written as,

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \end{bmatrix} \tag{4.3}$$

or in a more compact way as

$$\mathbb{C}\mathbb{Z} = \mathbb{D}. \tag{4.4}$$

Therefore, if compatibility condition $\det(\mathbb{C}) \neq 0$ is satisfied, the links can be computed by solving the equation (4.4), either by inverting $\mathbb{C}$ or by applying the Cramer's rule. The pivot location is then computed using links and the known end-point position. For example, if $\boldsymbol{h}^j = 0$ and $\boldsymbol{d}_0$ is unknown, then the pivot position can be found by

$$\boldsymbol{d}_0 = \boldsymbol{d}_{n_L} - \underbrace{\sum_{l=0}^{n_L - 1} \boldsymbol{Z}_l}_{\boldsymbol{r}}. \tag{4.5}$$

If one of the two end-point positions $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$ is unknown, but the sets of end-point displacements $\boldsymbol{h}^j$ and $\boldsymbol{g}^j$ are both non-null data, all links may have imposed rotations, but this does not assure the existence of a solution. The solution existence only depends on the determinant of the system, $\det(\mathbb{C})$. An example with similar data is shown in Figure 4.2-b for a RPR-triad with a free-pivot, where $\boldsymbol{h}^j = 0$ and $\boldsymbol{d}_0$ is unknown. After computing the links that satisfy $\boldsymbol{g}^j$ and the motion constraints, the pivot position can be found by equation (4.5).

### 4.3.3. Non-linear solution

The first case where $n_{pp} > n_L + 1$ is a four-positions dyad without imposed pivot, i.e., three complex equations in two complex unknowns for which the matricial form is

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ (e^{i\alpha_0^3} - 1) & (e^{i\alpha_1^3} - 1) \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \boldsymbol{\delta}^3 \end{bmatrix}, \quad \mathbb{C}\mathbb{Z} = \mathbb{D}. \tag{4.6}$$

Since there are two unknowns, the non-square matrix of coefficients $\mathbb{C}$ must be of rank 2 to obtain a solution. It is required that the determinant of every $2 \times 2$-sub-matrix of $\mathbb{C}$ be non-null

$$\det \mathbb{C}_a = \begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \end{vmatrix} \neq 0 \wedge \det \mathbb{C}_b = \begin{vmatrix} (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ (e^{i\alpha_0^3} - 1) & (e^{i\alpha_1^3} - 1) \end{vmatrix} \neq 0, \quad (4.7)$$

and every third-order determinant of the system augmented by the independent term $\mathbb{D}$, be null; for this example, since the augmented matrix results square, there is a unique *compatibility condition*:

$$\det([\mathbb{C}|\mathbb{D}]) = \begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ (e^{i\alpha_0^3} - 1) & (e^{i\alpha_1^3} - 1) & \boldsymbol{\delta}^3 \end{vmatrix} = 0. \quad (4.8)$$

Note that conditions (4.7) and (4.8) would be hardly ever satisfied by constraints imposed by the user at the outset, thus in our synthesis methodology we are interested in problems where only either $\alpha_0^j$ or $\alpha_1^j$ are known data and will force our decomposition method to retain open chains with such kind of imposed motions. Since we have a column with unknown data, a trivial solution for condition (4.8) can be found by proposing $\alpha_0^j = \alpha_1^j$ but it contradicts conditions (4.7), while the non-trivial is deduced by expanding the determinant of the augmented matrix by cofactors of the column with unknown data. For example, if the unknown data are in the first column, $\alpha_1^j$ and $\boldsymbol{\delta}^j$ are given, we shall proceed as:

$$\begin{aligned} (e^{i\alpha_0^1} - 1) \underbrace{\begin{vmatrix} (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ (e^{i\alpha_1^3} - 1) & \boldsymbol{\delta}^3 \end{vmatrix}}_{\Delta_1} + (e^{i\alpha_0^2} - 1) \underbrace{\left( - \begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_1^3} - 1) & \boldsymbol{\delta}^3 \end{vmatrix} \right)}_{\Delta_2} \\ + (e^{i\alpha_0^3} - 1) \underbrace{\begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \end{vmatrix}}_{\Delta_3} = 0 \end{aligned} \quad (4.9)$$

where minors $\Delta_i$ have known data. Then

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 + \underbrace{e^{i\alpha_0^3}\Delta_3 \underbrace{-\Delta_1 - \Delta_2 - \Delta_3}_{\Delta_0}}_{\Delta_0'} = 0 \quad (4.10)$$

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 = -\Delta_0' \quad (4.11)$$

The compatibility equation (4.10) can be seen as a one-DOF four-bar linkage called *Compatibility Linkage*, whose links are the minors $\Delta_i$ $(i = 0, 1, 2, 3)$, see Figure 4.3-a. Since it has one-DOF, by proposing one of the infinite values of $\alpha_0^3$ as free choice, a link $\Delta_0' = e^{i\alpha_0^3}\Delta_3 + \Delta_0$ is obtained; see Figure 4.3-b. The reduced compatibility equation (4.11) can be seen as two known complex numbers $\Delta_1$ and $\Delta_2$ rotated by $\alpha_0^1$ and $\alpha_0^2$ respectively, whose summation results in the known complex number $-\Delta_0'$; they form a three-bar loop called *Solution Structure* that can be assembled in two different ways called geometric inversions and thus helps to obtain two sets of $\alpha_0^1$ and $\alpha_0^2$, shown in Figure 4.3-c as $\{\alpha_{0,a}^1;\alpha_{0,a}^2\}$ and $\{\alpha_{0,b}^1;\alpha_{0,b}^2\}$.

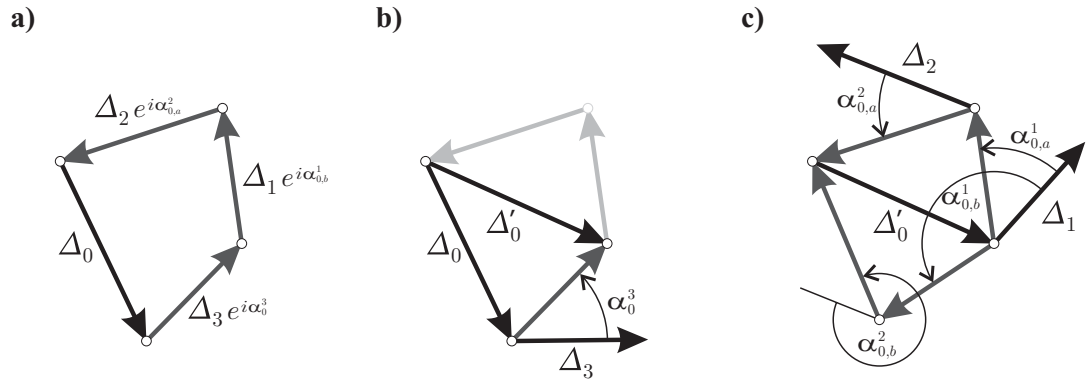



Figure 4.3: a) *Compatibility linkage* drawn with supposedly known rotations; b) Free-choice proposal ($\alpha_0^3$); c) *Solution structure* in its two geometric inversions.

Finally, for a free choice $\alpha_0^3$ and every set of $\alpha_0^1$ and $\alpha_0^2$ the coefficients of $\mathbb{C}$ are determined, and a set of links can be computed from any two equations of the system of equations (4.6) (for example by inverting $\mathbb{C}_a$).

Since one free choice must be made, and multiplicity is 2, the problem has $2(\infty)$ possible solutions. This procedure of building the *Compatibility Linkage* and then solving the *Solution Structure* was firstly introduced by Sandor and Freudenstein [San59], Denavit and Hartenberg [HD64], Sandor and Erdman [ES97], and then extended by Lin *et al.* in a systematic way [LEJ96]. They gave methods to linearize the non-linear systems when the number of prescribed positions is higher than the number of equations needed to obtain a linear solution. These cases are dyads in 4 to 5 positions, triads in 5 to 7 positions, quadriads in 6 to 9 positions. Note that there is a maximum number of equations that can surpass the number of unknowns to get a closed-form solution. They are: dyads in 5 positions (4 equations), triads in 7 positions (6 equations), and quadriads in 9 positions (8 equations) [LEJ96].

### 4.3.4. Synthesis with imposed offset

If the two end-point positions are known, an additional equation must be considered for the initial situation constraining the system (4.2) with the equation:

$$\boldsymbol{Z}_0 + \boldsymbol{Z}_1 = \boldsymbol{r}, \tag{4.12}$$

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

A dyad example is shown in Figure 4.2-a where the sets $\boldsymbol{h}^j$ and $\boldsymbol{g}^j$ are both non-null. Then, we have a system modified as

$$\begin{cases} \boldsymbol{Z}_0(e^{i\alpha_0^j} - 1) + \boldsymbol{Z}_1(e^{i\alpha_1^j} - 1) = \boldsymbol{\delta}^j, \\ \boldsymbol{Z}_0 + \boldsymbol{Z}_1 = \boldsymbol{r}. \end{cases} \quad j = 1, \ldots, n_{pp} - 1 \tag{4.13}$$

If, for example, we need to solve three positions, this system would involve three equations in two unknowns, its matricial form can be written as

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \boldsymbol{r} \end{bmatrix}, \tag{4.14}$$

or briefly as

$$\mathbb{C}^{\text{off}}\mathbb{Z} = \mathbb{D}^{\text{off}}. \tag{4.15}$$

Solution/s will be possible only if the non-square coefficient matrix of the system (4.15) is of $\text{rank}(\mathbb{C}^{\text{off}}) = 2$, i. e.

$$\det \mathbb{C}_a^{\text{off}} = \begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \end{vmatrix} \neq 0 \wedge \det \mathbb{C}_b^{\text{off}} = \begin{vmatrix} (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ 1 & 1 \end{vmatrix} \neq 0, \tag{4.16}$$

and the third-order characteristic determinant of its augmented matrix is zero,

$$\det([\mathbb{C}^{\text{off}}|\mathbb{D}^{\text{off}}]) = 0, \tag{4.17}$$

$$\begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ 1 & 1 & \boldsymbol{r} \end{vmatrix} = 0. \tag{4.18}$$

Note that the trivial solution for this condition is found by proposing $\alpha_1^1 = \alpha_0^1$ and $\alpha_1^2 = \alpha_0^2$ in $\mathbb{C}^{\text{off}}$ but it contradicts equations (4.16).

The third column includes the known data, $\boldsymbol{\delta}^j$ and $\boldsymbol{r}$, and one of the first columns must be completely unknown. Then, we can expand the determinant by the unknown column to find the geometrical relationships between the parameters. Suppose, for example, that $\alpha_1^1$ and $\alpha_1^2$ are the given motion constraints on the second link, thus known data are in the middle and last columns. If we develop the determinant by cofactors of the first column, we get the *Compatibility equation* for system (4.15):

$$(e^{i\alpha_0^1} - 1)\underbrace{\begin{vmatrix} (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ 1 & \boldsymbol{r} \end{vmatrix}}_{\Delta_1} + (e^{i\alpha_0^2} - 1)\underbrace{\left(-\begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ 1 & \boldsymbol{r} \end{vmatrix}\right)}_{\Delta_2} + \underbrace{\begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \end{vmatrix}}_{\Delta_3} = 0$$

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 \underbrace{-\Delta_1 - \Delta_2 + \Delta_3}_{\Delta_0} = 0 \tag{4.19}$$

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 = -\Delta_0. \tag{4.20}$$

Note that the triangle solution structure of equation (4.11) is reusable here to obtain two sets of $\alpha_0^1$ and $\alpha_0^2$. By replacing every set into coefficients of $\mathbb{C}_a^{\text{off}}$, two sets of links can be computed.

Therefore, when the two end-point positions, $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$, and the sets of end-point displacements, $\boldsymbol{h}^j$ and $\boldsymbol{g}^j$, of the SOC are known, the problem (4.14) is known as *synthesis with imposed offset* or *synthesis with ground-pivot specification* if one of the sets $\boldsymbol{h}^j$ or $\boldsymbol{g}^j$ is null. A three-position dyad with ground-pivot specification case was presented by Erdman [ES97][1]. We shall remark that both sets, $\boldsymbol{h}^j$ and $\boldsymbol{g}^j$, can

[1] Another equivalent notation for (4.1) uses the closing vectors for each position $\boldsymbol{R}^j$ as function of the offset vector; $\boldsymbol{R}^j = \underbrace{\boldsymbol{Z}_0 + \boldsymbol{Z}_1}_{\boldsymbol{r}} + \underbrace{\boldsymbol{g}^j - \boldsymbol{h}^j}_{\boldsymbol{\delta}^j}$. Then, the compatibility equation would look like:

$$\begin{vmatrix} e^{i\alpha_0^1} & e^{i\alpha_1^1} & \boldsymbol{R}^1 \\ e^{i\alpha_0^2} & e^{i\alpha_1^2} & \boldsymbol{R}^2 \\ 1 & 1 & \boldsymbol{r} \end{vmatrix} = e^{i\alpha_0^1}\underbrace{\begin{vmatrix} e^{i\alpha_1^2} & \boldsymbol{R}^2 \\ 1 & \boldsymbol{r} \end{vmatrix}}_{\Delta_1} + e^{i\alpha_0^2}\underbrace{\left(-\begin{vmatrix} e^{i\alpha_1^1} & \boldsymbol{R}^1 \\ 1 & \boldsymbol{r} \end{vmatrix}\right)}_{\Delta_2} + \underbrace{\begin{vmatrix} e^{i\alpha_1^1} & \boldsymbol{R}^1 \\ e^{i\alpha_1^2} & \boldsymbol{R}^2 \end{vmatrix}}_{\Delta_0} = e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 + \Delta_0 = 0,$$

which coincides with equation (4.20).

be simultaneously null for a triad, quadriad, and so on. But it configures a structure for a dyad.

Lin *et al.* [LEJ96] as well as many other authors did not consider cases where there is the additional equation imposed by *offset distance*. However, in this thesis, the decomposition method is only based on the fact that both end-point positions of the open chains are known. So, all programmed modules that solve the equations of dimensional synthesis were written by including offset. Note that for three-position problems and imposed offset, a dyad has the easy solution structure shown above, while a triad has an easy linear solution.

A triad passing through three positions with imposed offset has a nonhomogeneous system

$$\begin{bmatrix} (e^{i\alpha_0^1}-1) & (e^{i\alpha_1^1}-1) & (e^{i\alpha_2^1}-1) \\ (e^{i\alpha_0^2}-1) & (e^{i\alpha_1^2}-1) & (e^{i\alpha_2^2}-1) \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \\ \boldsymbol{Z}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \boldsymbol{r} \end{bmatrix}, \tag{4.21}$$

which has solution iff $\det(\mathbb{C}^{\text{off}}) \neq 0$.

Note that for direct linear solutions, apparently, there are not restrictions on the imposed angles, however, those trivial cases that make the determinant null must be avoided: (i) a null column or row, (ii) at least two linearly-dependent columns or rows, for example $\alpha_l^1 = 0$ or $\alpha_0^j = C\alpha_1^j$ with $C = \text{cte}$.

## 4.4. Generalizing loop-closure equations

A general statement of the displacement equations would be classified into two categories where the main difference is that the offset imposes one additional equation:

**Standard synthesis.** Loop-closure equations are:

$$\boldsymbol{Z}_0(\lambda_0^j - 1) + \boldsymbol{Z}_1(\lambda_1^j - 1) + \cdots + \boldsymbol{Z}_{n_L-1}(\lambda_{n_L-1}^j - 1) = \boldsymbol{\delta}^j, \quad j = 1, \ldots, n_{pp} - 1. \tag{4.22}$$

where $\lambda_l^j = e^{\alpha_l^j}$ if joint preceding link $l$ is of revolute type or $\lambda_l^j = \rho_l^j$ for prismatic type.

- One of the end-point positions, $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$, is unknown. The associated end-point displacements are null.
- If $n_{pp} = n_L + 1$ the system has a direct linear solution.
- For $n_{pp} > n_L + 1$ a compatibility linkage procedure may be used. Among other properties, Lin [LEJ96] found that the number of degrees-of-freedom (number of free choices) of a compatibility linkage is $F = 2n_L - n_{pp} + 1$.
- After solving the system of equations, the pivot is computed from the non-null end-point position.

**Synthesis with imposed offset.** One more equation is taken into account:

$$\begin{cases} \boldsymbol{Z}_0(\lambda_0^j - 1) + \boldsymbol{Z}_1(\lambda_1^j - 1) + \cdots + \boldsymbol{Z}_{n_L-1}(\lambda_{n_L-1}^j - 1) = \boldsymbol{\delta}^j \\ j = 1, \ldots, n_{pp} - 1 \\ \boldsymbol{Z}_0 + \boldsymbol{Z}_1 + \cdots + \boldsymbol{Z}_{n_L-1} = \boldsymbol{r}, \end{cases} \tag{4.23}$$

- Both end-point positions, $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$, are known.
- If $n_{pp} = n_L$ the system has a direct linear solution.
- For $n_{pp} > n_L$ a compatibility linkage procedure may be used in a way similar to those cases without offset.

For both categories:

- The sets of end-point displacements, $\boldsymbol{h}^j$ and $\boldsymbol{g}^j$, are considered known data.
- Motion constraints that can be imposed are subjected to:
  - The number of equations. It coincides with the number of precision points $n_{pp}$ in the standard form (4.22), and is $n_{pp} + 1$ for imposed offset (4.23).
  - The number of unknowns (number of links).

  If the number of equations exceeds the number of unknowns, the system results over-determined (rank-deficient and non-linear). For these cases, the necessary condition to be fulfilled by motion constraints is that *there must be at least one link with completely unknown motion constraints to develop the characteristic determinant of* $(n_L + 1)$*-th order*. A system of coupled characteristic determinants (or compatibility equations) results if the number of equations surpasses the number of links plus 2.

### Differences between the two categories

In equations with imposed offset, the advantage of taking both $\boldsymbol{d}_0$ and $\boldsymbol{d}_{n_L}$ always as known data is to have a control over the place where the synthesized pivots are positioned. This gave us a robust design strategy in such a way that if a new pivot is synthesized by the type synthesis solver, then (i) pivot position is considered known for decomposition purposes; (ii) displacements are imposed to be null; (iii) the dimensional synthesis solver asks the user for an area for pivot location, i.e. the user proposes bounds for the pivot position in the $x$-$y$ plane usually defined as a *boxed area* inside the *allowed space*; and (vi), when the dimensional synthesis solver is run, the position of such pivot, $\boldsymbol{d}_0$ or $\boldsymbol{d}_{n_L}$, will be generated as data for the SOC module solver.

On the other hand, if the pivot was computed by using standard equations, the resultant pivot may fall far from the allowed space for a wide range of the free parameters. Standard equations only admit defining free parameters in the form of motion constraints (angles and displacements).

The definition of bounds for pivot location is more intuitive than the definition of bounds for motion constraints. Furthermore, locating a boxed area inside the allowed space suits for automation.

## 4.5. Programmed modules

Since a set of Loop-Closure Equations are solved for each single-open chain, Sandor [SE84, Chap. 2, Sec. 19] called "loop" an individual open chain. However, the representation of each single open-chain that will be assigned in this thesis does

not match with any element of Graph Theory, so it is called a *SOC module* and the term "loop" is reserved to refer exclusively to closed paths in the graph[1].

A *SOC module* is defined by following a *path of nodes* involving both links (vertices) and joints (edges). In Figure 4.4 we can see the available modules to analyze and solve the SOCs. Using the FEM description, a module begins in a node and ends in other node and between two links there is always a pair of nodes constrained by a joint. For the internal nodes, the symbol □ means that the position and the set of displacements for each node are unknowns. For the end-point nodes, the symbol ■ means that the position and the set of displacements for each node are known.

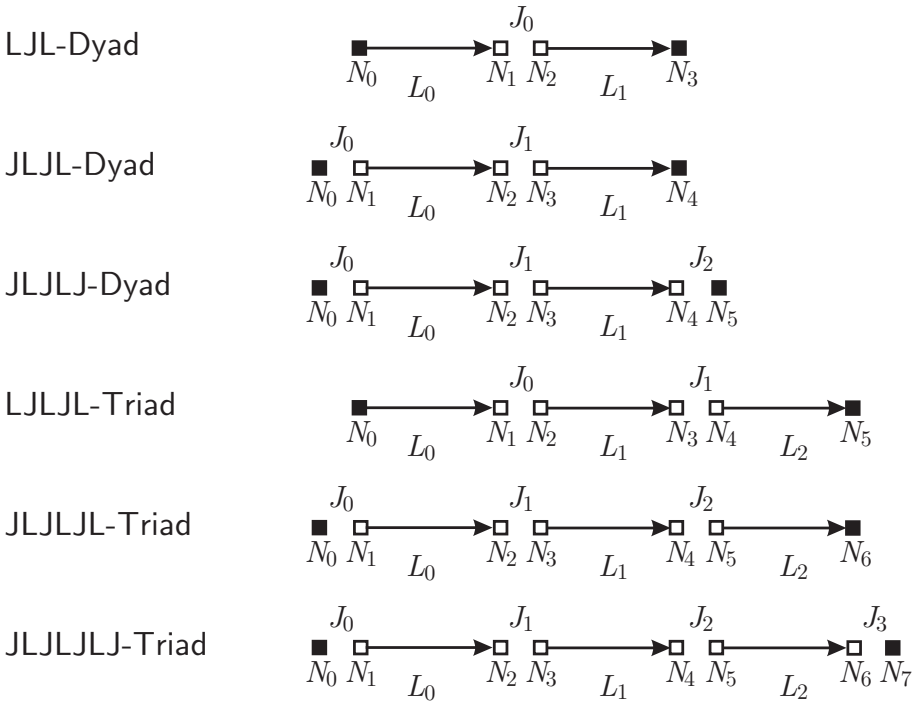



Figure 4.4: Modules for solving SOCs.

As we will see in the next chapter, the SOC decomposition algorithm will identify particular subcases of a SOC module. For instance, the subcases for a Dyad could be LJL or JLJL, where J means joint and L denotes link (JLJLJ will never occur for the presented rigid-body examples, but it has a meaning in flexible mechanisms). The second subcase is also used to solve the symmetric occurrence LJLJ.

According to joint types of the SOC module, the proper Loop-Closure Equations solver is linked. Solvers are shown in Figure 4.5. Joint types can be either R (revolute)[2], P (prismatic), or PA (prismatic with imposed angle). For instance, the LJL- and JLJL-Dyad modules with revolute joints, will be linked to the RR-Dyad solver.

[1]From Graph Theory, a *path* is a sequence of vertices in the graph.

[2]In Chapter 8, the treatment for solving joints of *clamped* and *living hinge* type used in compliant mechanisms is explained. These joints are solved using modules for revolute type but a proper restriction for limiting the range of movement is added.

Given the number of prescribed precision positions $n_{pp}$ and given motion constraints, each SOC module can execute two functions:

`Assemble`: Gives the quantity of free parameters needed and the multiplicity of solutions.

`Solve`: Solves the Loop-Closure Equations system for a given set of parameters and an externally chosen multiplicity. It may return either that there is a successful answer or that no solution exists.

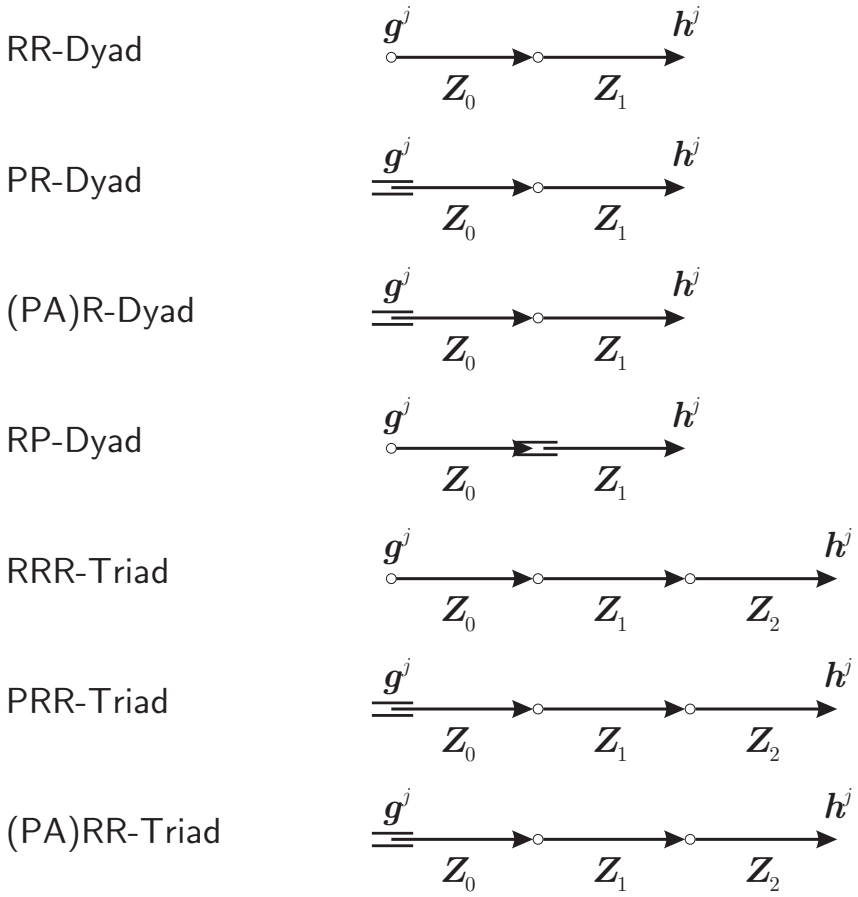



Figure 4.5: Loop-Closure Equations solvers.

The number of free choices and multiplicity of solutions are the main characteristics that will be automatically managed by an optimizer. For a given SOC, the exploration of its free choices within their ranges as well as its multiplicities will result in a finite or infinite number of solutions that will be assembled with the solutions of other SOCs to form mechanisms. A computer program may find the optimal free parameters and combination of solutions for some design criterion (detailed in Chapter 6 later on).

## 4.6. Chapter conclusions

In this chapter, the use of Loop-Closure Equations for sizing open chains was reviewed. Among the various existing strategies for solving open chains, methods of *synthesis in standard form* from *synthesis with imposed offset* were distinguished. To reduce the number of forms in which a problem can be solved, the latter is chosen as the solution procedure and was programmed in several modules. Additionally, the necessary data and solvability conditions were defined. In the next chapter, an automated decomposition method will be designed for the solution capabilities of these SOC modules. A few conventions established in this chapter will serve as background for developing an automatic strategy for dimensional synthesis of multi-loop linkages.

# Chapter 5

# Decomposition of topology into open chains

In the *Type Synthesis* stage, as an answer of the type synthesis software, the *list of all non-isomorphic topologies* for the given kinematic problem was obtained. The goal in the initial sizing part is to evaluate and sort the topologies in a ranking of optimality, and therefore, they must necessarily have dimensions.

For the purpose of sizing, it was computationally implemented a classical strategy for representing the closed-chain mechanism by means of its decomposition into several Single Open-Chains (SOCs) which contain the significant dimensions of the mechanism. This strategy consists in the following steps:

(i) decompose the topology into an ordered list of SOCs;

(ii) solve analytically each SOC in the order given by the previous step using complex numbers for representing the links [San59, SE84, LEJ96], and

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

The decomposition of the topology into SOCs is not unique; moreover, the first two steps (i-ii) are strongly interrelated because the solvability of one SOC may be dependent on the solvability of a previous SOC. So all possible decompositions must be carefully analyzed. In this chapter the aim is to develop a method for the step (i) which best considers the given task and the subsequent solvability of the resulting SOCs using the modules programmed by analytical synthesis techniques presented in Chapter 4. The required SOCs decomposition algorithm is the key step between the stages of *Type Synthesis* and *Dimensional Synthesis by analytical methods*. The problem is solved using Graph Theory and a FEM-like description for the topology and the SOCs.

## 5.1. Introduction

The *decomposition of complex multi-loop linkages into single subsystems* was deeply studied for automated kinematic and dynamic analysis [CFG96, KKH97, YYZ98a, STY00]. However, its use in automated synthesis applications is less addressed in the literature [YYZ98b].

On the other hand, the Sandor and Erdman's strategy was successfully implemented in academic and commercial computer programs (mentioned before in Chap-

ter 1.4), and used to solve most of the particular-purpose linkages employed in industry and life. It is supposed that these programs have a data base of decompositions for the particular cases that they offer (four-, six-, and some eight-bar linkages). In this chapter, a general automated method for decomposing any mechanism on-line (i.e. dynamically computed while the solver is running) is presented.

The main characteristics of the problem in hand can be summarized as follows:

- The resultant set of SOCs is not unique.
- For a given set of SOCs:

  - the order is not unique, so there could be many valid orders;

  - data between SOCs could be either dependent or independent;

  - SOCs may present different multiplicity, so the solutions must be properly combined.

The proposed SOCs decomposition algorithm uses the graph structure and information relative to the prescribed parts and the motion constraints data imposed on them. The goal is to obtain an ordered set of SOCs that satisfies the solvability conditions, and involves the biggest number of prescribed motion constraints in the given order. Only one ordered set of SOCs will be retained to pass to the next stage.

In the previous chapter, we could see that many modules were programmed to solve analytically all solutions (there could be multiplicity) of the different SOCs. The proposed decomposition method does not need to execute the modules for solving SOCs, but it requires: (a) to analyze their solvability (by contrasting against data) and (b) to ask for multiplicity (already stored in modules) for a number of precision positions and given motion constraints.

Although the method is general, it is only limited by the programmed modules: dyads and triads with revolute and prismatic pairs passing through three or four precision positions.

The chapter organization is as follows: In Section 5.2 the data of the problem collected at the type synthesis stage are reviewed. Section 5.3 explains the classification of the significant dimensions of the linkage by considering the prescribed data and the type synthesized topology. Section 5.4 presents the proposed method for graph decomposition. Finally, Section 5.5 describes the applications to solve kinematic problems for single- and multi-loop linkages.

## 5.2. Starting from a type synthesis output

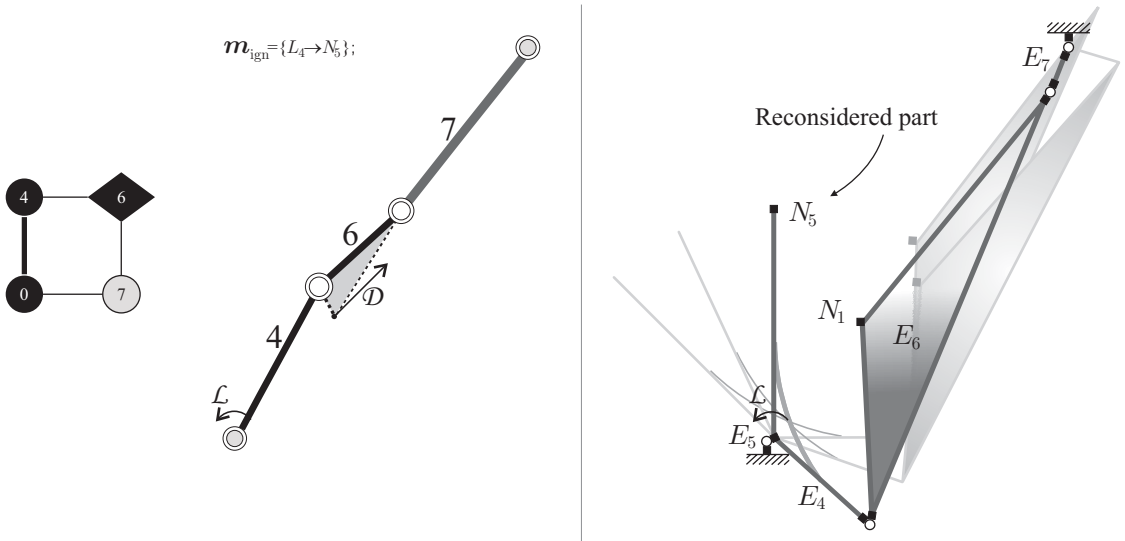
After the execution of a type synthesis software the initial graph allows to find potential mechanism alternatives. They can be visually judged by the user in order to determine if another type synthesis execution with other search constraints will be needed. Otherwise, if a satisfactory list is obtained, the decomposition for every structurally feasible mechanism is made. The required data for executing the decomposition algorithm are:

- Graph $G_a(E, V)$ and Type Adjacency Matrix $\boldsymbol{T}_a$ of the mechanism alternative, which are retrieved from $C_b^d(\boldsymbol{T}_a)$, $\mathcal{V}_a$, and $\mathcal{E}_a$;
- Finite Element description of the mechanism $\mathcal{M}_a$: $\boldsymbol{N}$, $\boldsymbol{F}$, and $\boldsymbol{E}(\boldsymbol{L}, \boldsymbol{J})$;

- Sets of the imposed Motion Constraints: $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{J}$;

- Trajectory Node Vector $\boldsymbol{n}_{\text{traj}}$ and its associated Objective Vertex Vector $\boldsymbol{v}_{\text{obj}}$.

For example, Figure 3.1 shows the problem of guiding one point of an unknown mechanism by three positions with prescribed timing: the set of displacements $\mathcal{D}$ defined on node $N_1$ must be in coordination with the set of rotations $\mathcal{J}$ defined on the joint $E_5$ of the crank $E_4$. Then, following the instructions given in Section 3.1.1, the initial graph is built and the vectors $\boldsymbol{n}_{\text{traj}} = [1]$, and $\boldsymbol{v}_{\text{obj}} = [6]$ are saved. The subgraph search parameters were set as: (a) atlas of rigid one-degree of freedom mechanisms RigidOneDofR, (b) maximum distance from the objective vertex to ground equal to 2, and (c), avoidance of pseudo-mechanisms. Each matching is a feasible mechanism topology $\mathcal{M}_a$ which inherits the motion constraints $\mathcal{D}, \mathcal{J}, \mathcal{L}$ of the task. The results were shown in Figures 3.5 and 3.6. Figure 5.1-a shows the graph and the sketch for the simplest first alternative, $\mathcal{M}_0$, found by this type synthesis execution.




**a)** Type Synthesis: graph and sketch.

**b)** Dimensional Synthesis: FEM description.

Figure 5.1: The simplest solution for the two stages of synthesis for a path following problem.

In the graph, the ground has the label zero, and the objective vertex has a diamond shape. The grey vertex (7) and its incident edges are the new type synthesized link and joints, respectively. In Figure 5.1-b the dimensional synthesis result is displayed only to remark that the ignored part is restored to its original element to complete and evaluate the mechanism. In these figures, it can be seen that body $E_6$ is binary in terms of number of joints as considered in the graph; but it is physically ternary in terms of number of nodes. In the same mechanism, the body $E_4$ is binary in the graph, but it has three nodes after reconsidering the ignored part.

## 5.3. Significant dimensions

Next, the steps for decomposing a mechanism will be developed. The *Trajectory Node* vector $\boldsymbol{n}_{\text{traj}}$ and its associated *Objective Vertex* vector $\boldsymbol{v}_{\text{obj}}$ are of great importance since they *modify (add) the significant dimensions* to be computed.

Other important aspect that will change the behavior of the following decomposition method is the identification of the *previously-known significant dimensions*. By executing a kinematic analysis of the initial parts, the set of displacements $\mathcal{D}$ of prescribed nodes connected by joints, and other sets of motions of links $\mathcal{L}$ and joints $\mathcal{J}$ must be computed and updated. This execution, called *Initial Kinematics*, is run once for the initial parts before the type-synthesis running and does not have any influence on the type-synthesis stage.

Filtering of *Ignored Nodes* can be simply justified by the fact that they modify the shape of links but do not have influence on the kinematic behavior of the mechanism, i.e. they are not nodes defining significant dimensions.

Therefore, for the dimensional synthesis purpose, *dimensions of the mechanism* can be classified into two categories:

I) Significant: Between nodes connected by joints and trajectory nodes.
   - Unknown: to be computed.
   - Known: prescribed by the user.

II) Obsolete: They can be ignored during the whole initial dimensioning stage.

Note that this classification is possible only if we use the FEM description: in terms of positions and displacements of nodes. In the following section, a method for automatically assigning SOCs to the significant dimensions of any linkage is presented.

## 5.4. The proposed decomposition method

The decomposition method consists in the following steps:

S1) **Topology decomposition:** The kinematic chain (closed-loops chain mechanism) is decomposed into a set of separated closed-loops[1] of minimal length called minimum cycle basis. It should be pointed out that for most graphs, the minimum cycle basis is not unique. Hereafter, it will be called *minimal loop basis*.

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

S3) **SOCs evaluation:** For each decomposition, using the resultant order, the data transfert between SOCs is simulated while an index, for evaluating the SOCs solvability and the number of solved constraints, is generated.

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

[1] Known as *Cycles* in Graph Theory [Har69] or *Circuits* [Tsa01], and also called *Kinematic Loops* by [KKH97].

### 5.4.1. Topology decomposition

In the *Type Synthesis* stage, a mechanism structure represented by a *connected graph* $G(V, E)$ is obtained, where the vertex set $V$ has cardinality $v$, and the edge set $E$ has cardinality $e$. It is well-known from Graph Theory that a planar graph has $\nu = e - v + 1$ *independent closed loops*, and particularly, it is possible to find one or all of the basis of minimal length loops or *minimal independent loops*. The main characteristics of these basis are: a) no loop is contained into another loop, b) any other loop of the graph can be spanned by sums of the loops of the basis.

Using these loops all the *significant dimensions* of the links can be efficiently explored in a systematic way.

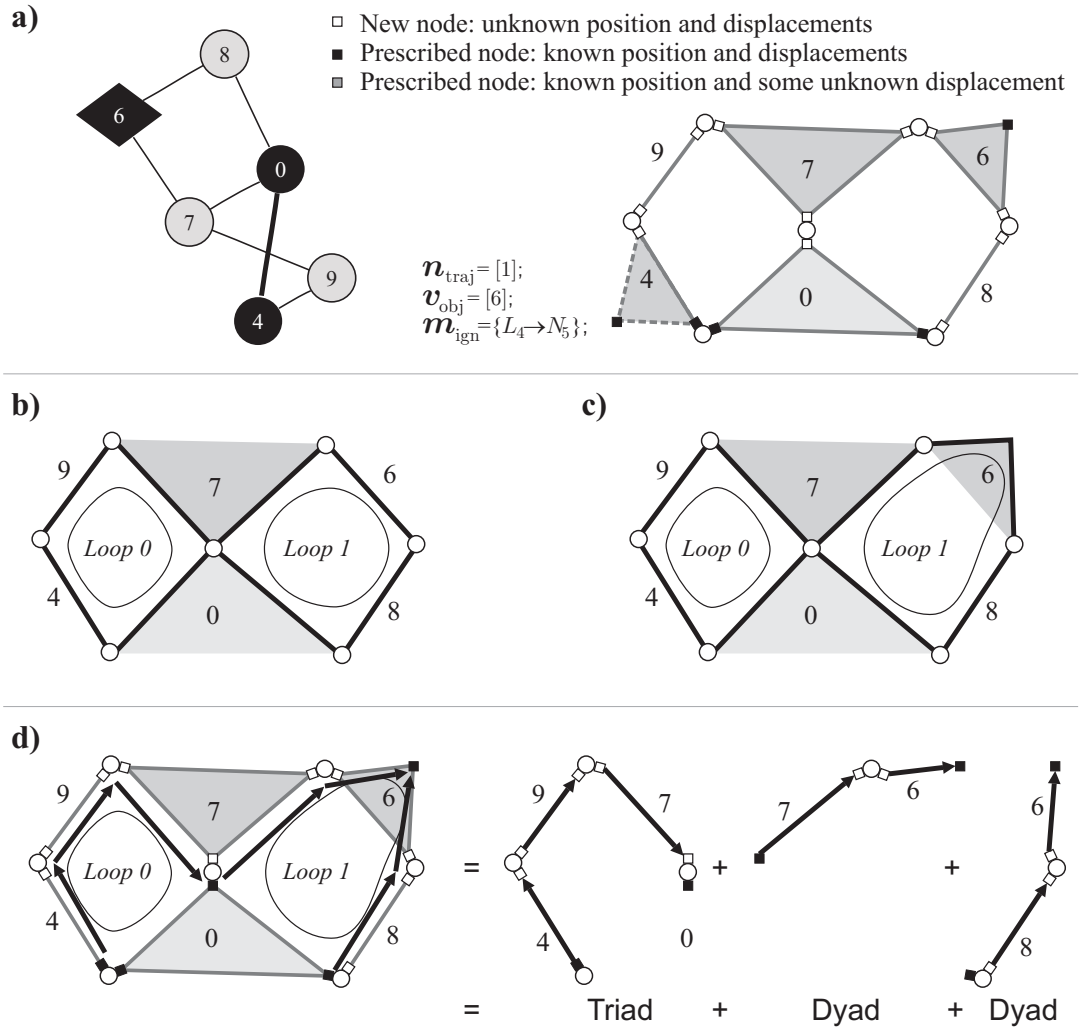



Figure 5.2: The set of independent loops of minimal length allows to find the significant dimensions of links.

The Watt-II kinematic chain has $\nu = 7 - 6 + 1 = 2$ independent closed loops (Figure 5.2-a). One line per non-binary link is left without being explored by the independent loops, e.g. links 7 and 0 in Figure 5.2-b, but their end-points are visited by the loops and consequently taken into account. Thus, the loops, traditionally given in terms of edges in Graph Theory, visit all edges of the graph and therefore all –kinematically important– nodes connected by joints of the FEM description.

There is only one exception: if the loop visits an *Objective Vertex*, the loop must be extended to pass through its associated *Trajectory Node* (see Figure 5.2-c). This extension must be repeated for every objective vertex (trajectory node) making use of the complementary data structure $\boldsymbol{v}_{\text{obj}}$ and $\boldsymbol{n}_{\text{traj}}$. That is, when a given vertex $\boldsymbol{v}_{\text{obj}}[i]$ is visited by the loop, its corresponding node $\boldsymbol{n}_{\text{traj}}[i]$ is inserted on the loop. Figures 5.2-c and 5.2-d show an schema of the loop extension effect on the SOCs decomposition (also detailed later with true coordinates in Figures 5.6-b and 5.7).

### Closed-loops determination

A *spanning tree* $T$, is a tree containing all the vertices of a connected graph $G$. Therefore, $T$ is a subgraph of $G$. In general, the spanning tree of a connected graph is not unique. For a given spanning tree $T$, the edge set $E$ of $G$ can be decomposed into two disjoint subsets, namely the arcs and chords. The arcs of $G$ consist of all the elements of $E$ that form the spanning tree $T$, whereas the chords consist of all the elements of $E$ that are not in $T$. The union of the arcs and chords constitutes the edge set $E$. The addition of a chord to a spanning tree forms one and only one circuit [Tsa01].

The collection of all the circuits with respect to a spanning tree forms a set of independent loops or fundamental circuits. The fundamental circuits constitute a basis for the circuit space. Any arbitrary circuit of the graph can be expressed as a linear combination of the fundamental circuits using the arithmetic of *modulo* 2, i.e., $1 + 1 = 0$ [Har69, Tsa01].

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

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

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

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

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

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

**S6** Up to this point, the algorithm does not necessarily lead to loops with "minimal lengths", i.e., with minimal number of edges per loop. Therefore, we process the obtained loops making all possible sums of *modulo* 2 between them while saving those with minimal lengths to form the minimal loop basis.

The stages produced when applying the algorithm are illustrated in Figure 5.3. Loops are represented by vectors indexed by the $e$ edges, in which every entry is 1 if the edge belongs to the loop or 0 otherwise. This representation facilitates loops addition operations. For instance, a resultant basis for the Watt chain shown in

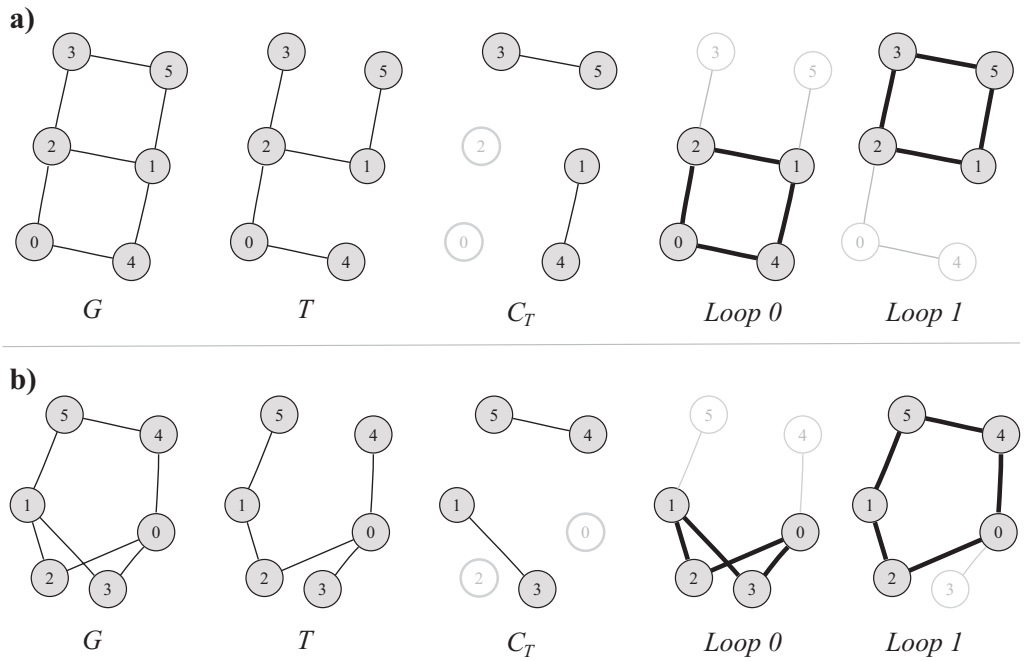



Figure 5.3: Stages of the algorithm to find a set of independent loops in the graphs of (a) a Watt-I topology and (b) for a Stephenson topology.

Figure 5.3-a may be arranged as the so-called ($\nu \times e$) *Fundamental Circuit Matrix*:

$$C = \begin{matrix} \\ l_0 \\ l_1 \end{matrix} \begin{matrix} \begin{matrix} e_{02} & e_{04} & e_{12} & e_{14} & e_{15} & e_{23} & e_{35} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}.$$

Note that the vector addition modulo 2 between the loops of the basis results in the peripheral loop:

$$l_0 \oplus l_1 = l_\mathrm{p} = \begin{matrix} \begin{matrix} e_{02} & e_{04} & e_{12} & e_{14} & e_{15} & e_{23} & e_{35} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{matrix}.$$

Finally, for convenience, every loop is represented by a vector containing only the consecutive edges embraced by the loop[1], sorted in such a way that indexes to vertices result concatenated. For the previous example, they are $l_0 = [e_{02}, e_{21}, e_{14}, e_{40}]$ and $l_1 = [e_{21}, e_{15}, e_{53}, e_{32}]$ which can be condensed as $B = \{l_0, l_1\}$.

In some cases, like in the Stephenson chain shown in Figure 5.3-b, there are two loops of length five, and consequently more than one minimal loop basis:

$$B = \{l_0, l_1\} = \{[e_{02}, e_{21}, e_{13}, e_{30}], [e_{02}, e_{21}, e_{15}, e_{54}, e_{50}]\},$$

and

$$B' = \{l_0, l'_1\} = \{[e_{02}, e_{21}, e_{13}, e_{30}], [e_{03}, e_{31}, e_{15}, e_{54}, e_{50}]\}.$$

[1]Other possible representation is the list of vertices $l_0 = [v_0, v_2, v_1, v_4]$.

### 5.4.2. SOCs decomposition

Once the loops are computed and stored, they will be decomposed into SOCs based on some necessary conditions for dimensional synthesis. The method uses the FEM description for each loop (at the nodes level) and the initial motion constraints coming from the topology.

Depending on the programmed modules two alternatives for dividing the closed loops into SOCs may be distinguished:

**Standard equations**: A loop is divided to form a SOC by going through the nodes chained by the loop, *starting from a node with known displacements and ending in another node with known displacements*. These nodes will be the tail and the tip of each SOC. It is also required that the position of at least one of these end-point nodes must be known.

**Equations with prescribed offset**: A loop is divided to form a SOC *starting from a node with known position and displacements and ending in another node with known position and displacements*.

Only the second class of equations were programmed in SOC modules due to the advantages remarked in Chapter 4. The use of this class of equations requires a particular treatment for the *fixed nodes of new prescribed pivots* that, obviously, have an unknown position, but for convenience (read Sub-section 4.4) these nodes are considered as known for the decomposition stage.

#### Loop-basis, loop-order and loop-orientation

This decomposition is "loop order dependent", so, for every basis all the $\nu!$ *Loop-Orderings* are analyzed in lexicographical order.

Additionally, since the decomposition is also "loop orientation dependent" when there are more than one SOC per loop, the opposite orientation must also be explored. For example, in the presence of an *objective vertex*, the number of loop-orderings is multiplied, i.e., we need to explore all possible orderings for each orientation of the loops containing this kind of vertex. By using this discrimination we can consider the graph as "undirected".

The algorithm for SOCs decomposition will be illustrated using the path following example, which is shown in Figure 5.4. Part of the mechanism is prescribed by user's data or computed by the Initial Kinematics execution: the nodes filled in black have known positions and known sets of displacements (see Figure 5.4-c). The remaining part needs to be synthesized: white or grey filled nodes have completely or partially unknown motion constraints, respectively.

#### Circular table

An individual closed-loop is decomposed into SOCs with the aid of a *circular table* that has information only relative to that loop. This table contains the FEM description (nodes, links and joints) of the loop. Additionally, a Boolean variable, denoted as stdispl, is added for each node of the table and used to *simulate* the states (known (1) or unknown (0)) of the node displacements. Initially, each fixed node has a null set of displacements, so fixed nodes are known (see nodes $N_2$ and $N_6$ in Figure 5.4-c); the *trajectory nodes* (like node $N_1$) are a second source of nodes with

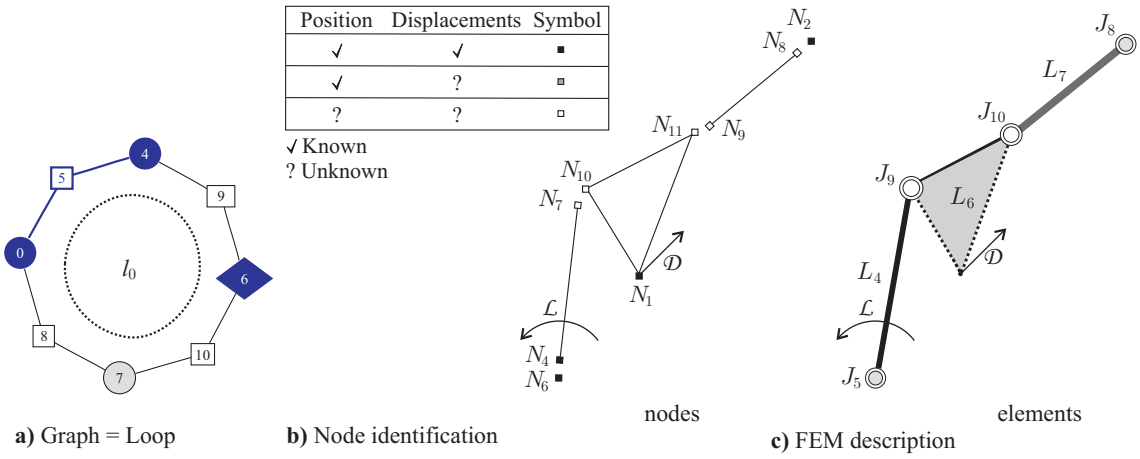



Figure 5.4: A first feasible graph for a path following problem.

prescribed displacements. Other previously-known node displacements are those computed in the *Initial Kinematics* running (e.g., displacements of node $N_4$).

Additionally, a vector indexed by the IDs of links is denoted as stlinks and is used to simulate the *states of links*. By "state" of a link, we mean the cardinality of the sets of motion constraints $\mathcal{L}(L_i)$ of link $L_i$. This vector is used later for the evaluation of the decomposition. Before the decomposition, the vector stlinks is filled with the number of motion constraints prescribed or computed by the initial kinematics. For example, the initial states of links vector for the problem shown in Figure 5.4-c is:

$$\begin{array}{c} \\ \mathsf{stlinks} = \end{array} \begin{array}{c} \begin{array}{ccc} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{array} \\ \left[\begin{array}{ccc} 3 & 0 & 0 \end{array}\right] \end{array} \quad (|\mathcal{L}_0| = 3 \quad \text{by default}).$$

Once a SOC is identified, it is solved: the initial positions and the sets of displacements for all its underlying nodes are computed. So, they will impose new constraints for decomposing/solving the subsequent SOCs lying either in the same or in the following loop. Therefore, after one SOC is identified the Boolean variable stdispl is updated as true for all the involved nodes, and links are then updated with the number of solved constraints on variable stlinks. This procedure will be illustrated using the path following example shown above.

For the unique loop, $l_0 = [e_{04}, e_{46}, e_{67}, e_{70}]$, the auxiliary *circular table* (see Table 5.1) has four rows and as many columns as nodes in the loop ("circular" denotes that the last column is connected to the first one).

In order to fill the table, a cursor starts pointing to a node of the first vertex of the loop, and writes (see Figures 5.4-a and 5.4-c) :

1. the node ID in the first row;
2. the ID of the link to which the node belongs in the second row;
3. the ID of the joint to which the node belongs in the third row;
4. the boolean state variable, indicating if this node has a prescribed displacement.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 2 | 6 | 4 | 7 | 10 | 1 | 11 | 9 | 8 |
| linkID | ⟼ | 0 | 0 | 4 | 4 | 6 | 6 | 6 | 7 | 7 |
| jointID | ⟼ | 8 | 5 | 5 | 9 | 9 | 0 | 10 | 10 | 8 |
| stdispl | ⟼ | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| SOC 0 | ⟼ | | | ▲. | ... | ... | .▼ | | | |
| SOC 1 | ⟼ | .▼ | | | | | ▲. | ... | ... | ... |

Table 5.1: Example of a circular table for the identification of single-open chains.

When the loop contains an objective vertex ($L_6$), the table has an additional column for the trajectory node ($N_1$) inserted between the two nodes ($N_{10}$ and $N_{11}$) that connect the associated objective vertex to other vertices of the loops ($L_4$ and $L_7$).

Two SOCs are identified by analyzing the fourth row content. In the example, we have one SOC going clockwise from node 4 to node 1, and a second SOC continuing clockwise from node 1 to node 2. The orientations of the complex numbers in the **SOC 1** are inverted to make them compatible with the available solver module JLJL-Dyad. The program automatically inverts the orientations of the complex-number chain whenever it finds that the SOC is started by a link and ended by a joint.

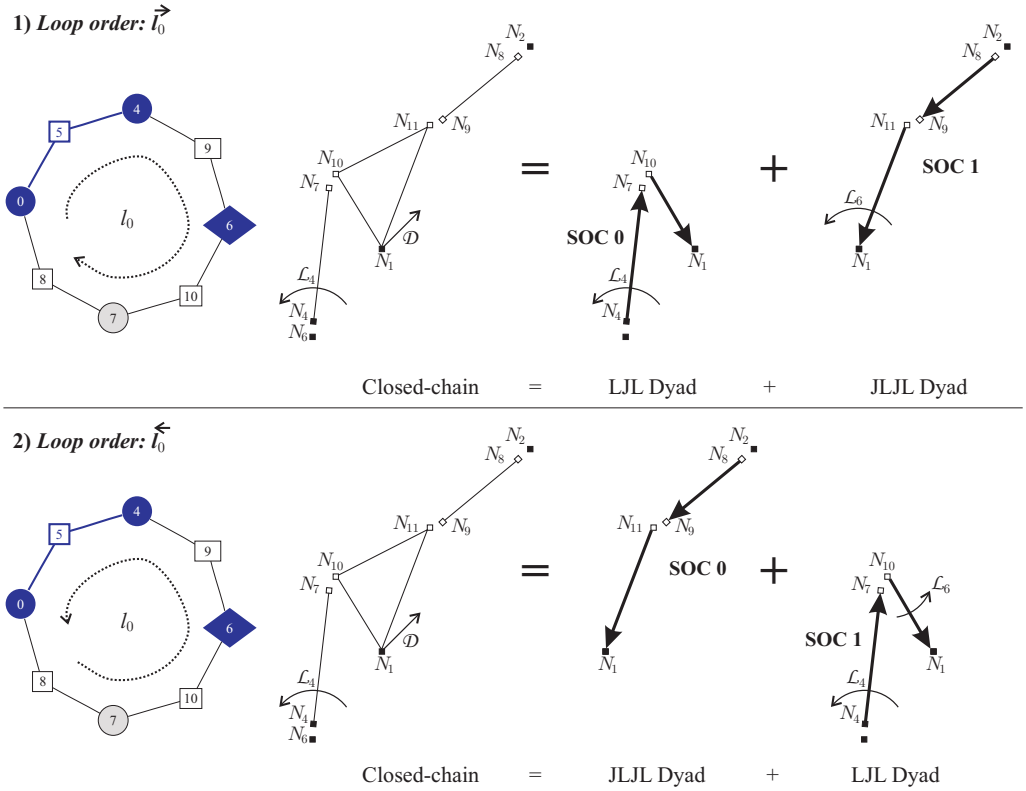



Figure 5.5: Complex numbers built for both decompositions.

For the running of the loop with the inverted orientation, the decomposition shown in Table 5.2 is obtained. This second decomposition resulted in two SOCs with the same type as the first one, that is, a JLJL-Dyad and a LJL-Dyad module, but in different order.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| nodeID | ⟼ | 6 | 2 | 8 | 9 | 11 | 1 | 10 | 7 | 4 | ⟲ |
| linkID | ⟼ | 0 | 0 | 7 | 7 | 6 | 6 | 6 | 4 | 4 | ⟲ |
| jointID | ⟼ | 5 | 8 | 8 | 10 | 10 | 0 | 9 | 9 | 5 | ⟲ |
| stdispl | ⟼ | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | ⟲ |
| SOC 0 | ⟼ | | ▲. | ... | ... | ... | .▼ | | | | ⟲ |
| SOC 1 | ⟼ | | | | | | ▲. | ... | ... | .▼ | ⟲ |

Table 5.2: Circular table for the counter-clockwise orientation.

The resultant complex-number models for both decompositions can be seen in the illustration of Figure 5.5. In the next section, the decompositions will be evaluated to determine which of them is more convenient.

This is a simple example with a unique loop. In Section 5.5, results for multi-loop mechanisms will be shown.

### 5.4.3. SOCs evaluation

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

The reasons for simulating instead of solving the open-chains are:

- The number of possible decompositions depends on the set of loop-basis, loop-orderings, and loop-orientation detected in the topology.
- From the set of possible decompositions, many of them may be "a priori" identifiable as unsolvable.
- The execution of the precision-point method for every decomposition requires a computational cost primarily dependent on the number of variables (free choices, boxes for new pivots, missing motion constraints), which can be high if the decomposition is not adequate.

#### Evaluation rules

In the previous example, no direct advantage from the change of orientation in the loop decomposition was appreciated. For both decompositions, two pairs of SOCs of the same modules were obtained. However, the SOCs ordering will be very important in the dimensional synthesis stage. The solution procedure for each decomposition is as follows:

First decomposition (Figure 5.5-1): The **SOC 0** has the first link ($L_4$) with its motion completely defined, that is the prescribed timing transmitted by the input joint ($J_5$). Then, parameters of the second link of the SOC, $\alpha_1^1$ and $\alpha_1^2$, can be solved by means of the solution procedure provided in the previous Chapter (Section 4.3.4). So, the sets of rotations of link ($L_6$) are known. To solve the **SOC 1**, the set of rotations of the second link ($L_6$) are imposed by the results

of **SOC 0**; here the parameters of the first link of the SOC ($L_7$), $\alpha_0^1$ and $\alpha_0^2$, can be computed similarly. Cardinalities of the solved motion constraints per link change as follows:

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 0 & 0 \end{bmatrix} \quad \text{at the initial situation,}$$

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 3 & 0 \end{bmatrix} \quad \text{after solving } \textbf{SOC 0},$$

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 3 & 3 \end{bmatrix} \quad \text{after solving } \textbf{SOC 1}.$$

Second decomposition (Figure 5.5-2): For the inverted orientation decomposition, two free choices on one of the links of **SOC 0** ($L_7$ or $L_6$) must be defined; parameters on the other link can be computed. Then, by transmitting the results of **SOC 0**, we have the **SOC 1** with completely defined movements on both links ($L_4$ and $L_6$). As the offset is imposed, only if $\det([\mathbb{C}^{\text{off}}|\mathbb{D}^{\text{off}}]) = 0$ the **SOC 1** will have a solution, which is a very odd case. Cardinalities of the solved motion constraints per link change as follows:

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 0 & 0 \end{bmatrix} \quad \text{at the initial situation,}$$

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 3 & 3 \end{bmatrix} \quad \text{after solving } \textbf{SOC 0},$$

$$\mathsf{stlinks} = \begin{matrix} |\mathcal{L}_4| & |\mathcal{L}_6| & |\mathcal{L}_7| \end{matrix} \\ \begin{bmatrix} 3 & 3 & 3 \end{bmatrix} \quad \text{after solving } \textbf{SOC 1}.$$

The second decomposition violates the important necessary design condition $\boldsymbol{E}_I$: *when the system becomes non-linear or has rank deficiency, at least one link must have completely unknown rotations.* In the example, the imposed offset produces a rank deficiency in the system of equations that solves the three-positions dyad (read Section 4.3.4).

Additionally, to choose a SOC decomposition $\boldsymbol{E}_{II}$: *it will be preferred the decomposition that solves a maximum number of imposed constraints –transferred from one SOC to the following– in the computing order.* This rule, combined with the first one, leads to impose the smallest number of missing motion constraints, and therefore, the number of free parameters is reduced to a minimum.

For example, in the first decomposition, the **SOC 0** solves three motion constraints prescribed by the user (timing) and the **SOC 0** of the second decomposition solves none. The whole problem can be solved without the need to define any free parameter. So, the first decomposition is preferred for this reason.

**Vectorial form of the evaluation criterion**

For each SOC, we count the *number of links with completely undefined constraints* $n_U(\mathbf{SOC})$. As we have reviewed in Chapter 4, it must be

$$n_U(\mathbf{SOC}) \begin{cases} \geq 1 & \text{if } n_{pp} > n_L \text{ and, } d_0 \text{ and } d_{n_L} \text{ are known (imposed offset),} \\ \geq 0 & \text{if } n_{pp} = n_L \text{ and, } d_0 \text{ and } d_{n_L} \text{ are known (imposed offset),} \\ \geq 1 & \text{if } n_{pp} > n_L + 1 \text{ and, } d_0 \text{ or } d_{n_L} \text{ is unknown,} \\ \geq 0 & \text{if } n_{pp} = n_L + 1 \text{ and, } d_0 \text{ or } d_{n_L} \text{ is unknown.} \end{cases} \tag{5.1}$$

Only the first two cases must be considered for modules that solve synthesis with imposed offset. Such choice greatly simplifies the evaluation because the number of decomposition cases and combinations of modules is reduced.

Rules are mathematically modelled by means of:

- a Boolean vector indexed by SOCs, $\boldsymbol{R}_I$, for which an entry has a value of true if the underlying SOC satisfies the condition (5.1), and false otherwise.
- an integer vector indexed by SOCs,

  $$\boldsymbol{R}_{II} = \{n_C(\mathbf{SOC\ 0}), n_C(\mathbf{SOC\ 1}), \dots, n_C(\mathbf{SOC}\ \boldsymbol{\nu}), \dots\},$$

  where function $n_C(\mathbf{SOC\ k})$ simply counts the number of motion constraints solved by $k$-th SOC. This vector can be easily filled by reading the previously defined vector stlinks.

Finally, a criterion that combines both rules is arranged as a vector of integers denoted as $\boldsymbol{R}^i = \{\boldsymbol{R}_I, \boldsymbol{R}_{II}\}$, where the supra-index $i$ denotes the number of decomposition. Boolean conditions are translated into integers: 1 for true and 0 for false.

For example, the evaluation for the decompositions shown in Figure 5.5 gives:

1. $\boldsymbol{R}_I = \{\text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{3, 3\}$; $\quad \rightarrow \boldsymbol{R}^1 = \{1, 1, 3, 3\}$.
2. $\boldsymbol{R}_I = \{\text{true}, \text{false}\}$, $\boldsymbol{R}_{II} = \{0, 6\}$; $\quad \rightarrow \boldsymbol{R}^2 = \{1, 0, 0, 6\}$.

### 5.4.4. Retained ordered SOCs

Two evaluations $\boldsymbol{R}^1$ and $\boldsymbol{R}^2$ are easily compared in lexicographical order, but first of all, it must be required that the part of the evaluation belonging to solvability rules be all true, otherwise the solution procedure would be partially solved and interrupted. For the second part, the lexicographical order comparison will give the right importance to the relevant decompositions. If no decomposition fulfils the first part of the evaluation, the topology is discarded; otherwise, the best valued decomposition of open-chains is stored for dimensional synthesis. The latter will be the *computing order*.

More than one ordering could be well-posed in terms of these two evaluation criteria, $\boldsymbol{R}_I$ and $\boldsymbol{R}_{II}$, so all of them would be retained to pass through the dimensional synthesis stage to give a valid judgement. Note that it cannot be predicted if a set of SOCs has a solution until the free parameters (variables) are proposed for

their whole ranges. However, the computational cost of solving dimensionally every feasible decomposition may be substantial, either in terms of computation time or in terms of complexity in the dimensional synthesis stage. This complexity arises from the fact that: (a) the procedure has a branch for every decomposition, and (b) each decomposition may branch due to the multiplicity of the individual SOCs.

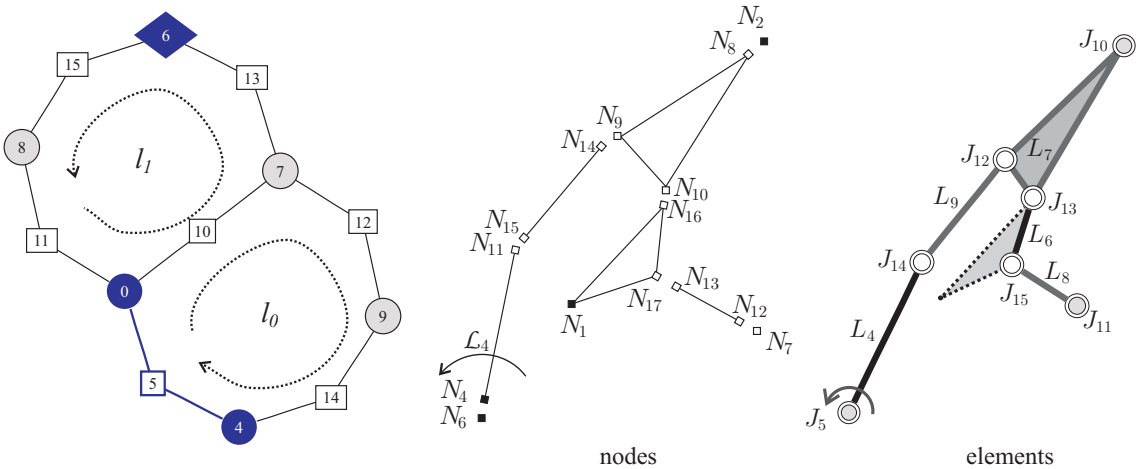
Following the example given above, since $\boldsymbol{R}^1 > \boldsymbol{R}^2$, the first one is considered at the first position in the ranking, and thus it is selected for dimensional synthesis. Its result looks like the shown in Figure 5.1-b.

## 5.5. More decomposition examples

Several kinematic problems were tested and used as feedback to develop the method. Two multi-loop examples: (a) with presence of an objective vertex and (b) without it, were selected for presenting the steps of the decomposition algorithm.

### 5.5.1. A multi-loop curve path generator

The study of the path following example shown in Figures 3.5 and 3.6 will be considered again. The **Alternative 1** of such type synthesis output is a two-loop chain (see Figure 5.6), where a new pivot was synthesized. Their minimal independent loops were shown in Figure 5.2-c with link labels. In Figure 5.6-a they are shown with link and joint labels.




**a)** Graph and minimal loop basis

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

Figure 5.6: Initial situation for the decomposition of the second alternative of the path following problem.

First, we identify those loops where the objective vertex is located. Using the additional data $\boldsymbol{v}_{\text{obj}}$ and the loop basis (shown in Figure 5.2-a), the loop $l_1$ is identified as containing the objective vertex $v_6$. Then, the two orientations for the loop $l_1$ are denoted as $\overrightarrow{l_1}$ and $\overleftarrow{l_1}$. The possible loop orderings are $l_0 - \overrightarrow{l_1}$ and $\overrightarrow{l_1} - l_0$, the inverted cases are $l_0 - \overleftarrow{l_1}$ and $\overleftarrow{l_1} - l_0$. Thus, the number of possible decompositions is increased to $2\nu!$.

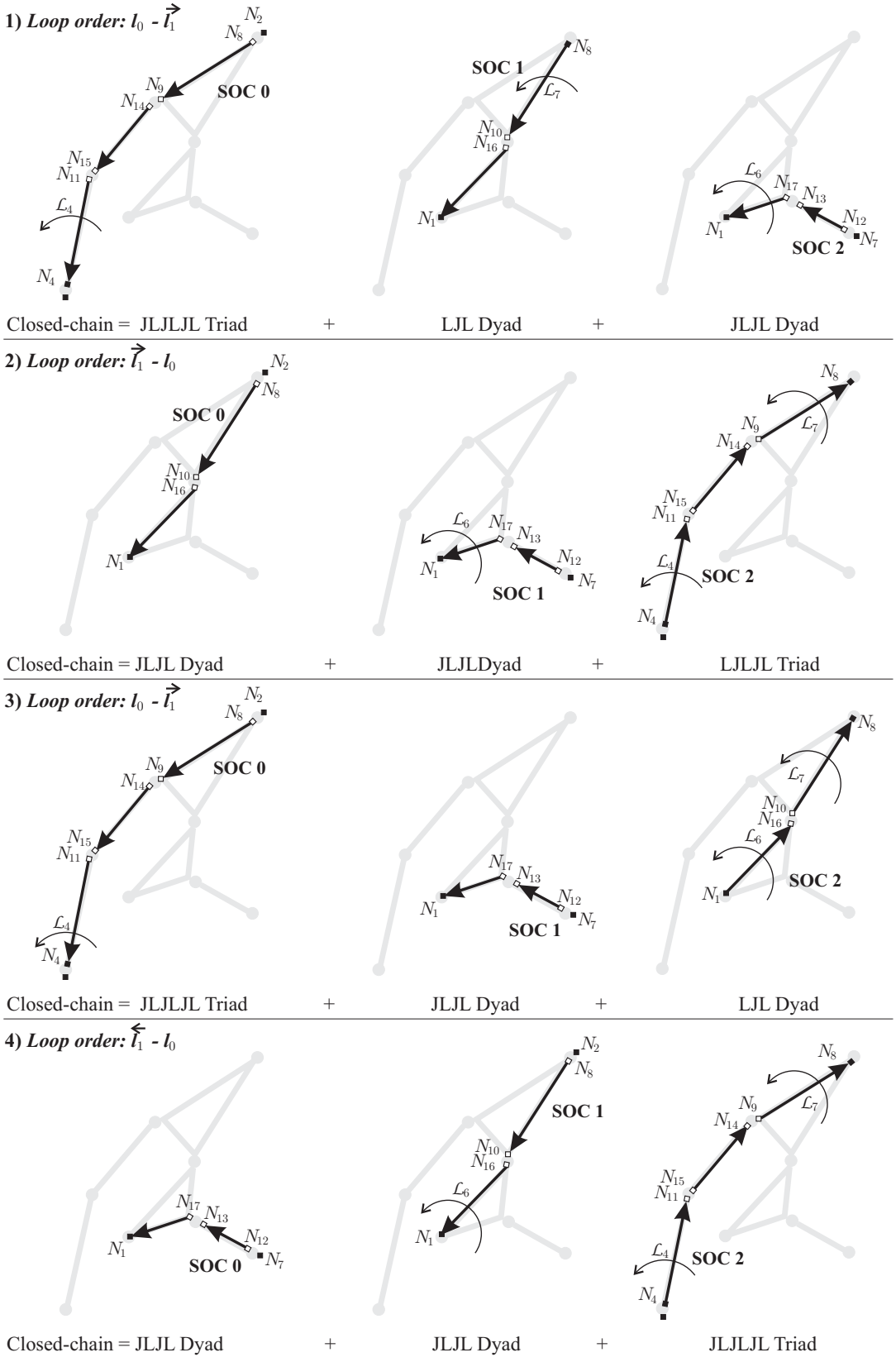



Figure 5.7: All decompositions for the second alternative of the path following problem. Only nodes belonging to SOCs are labeled.

In this case, for every loop-ordering, the topology is decomposed into $\nu + 1$ SOCs. The resultant decompositions are shown in Figure 5.7 where the sets of motion constraints $\mathcal{L}_i$ transferred between SOCs were drawn.

The obtained evaluations are:

1. $\boldsymbol{R}_I = \{\text{true}, \text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{3, 3, 3\}$; $\quad \rightarrow \boldsymbol{R}^1 = \{1, 1, 1, 3, 3, 3\}$.
2. $\boldsymbol{R}_I = \{\text{true}, \text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{0, 3, 6\}$; $\quad \rightarrow \boldsymbol{R}^2 = \{1, 1, 1, 0, 3, 6\}$.
3. $\boldsymbol{R}_I = \{\text{true}, \text{true}, \text{false}\}$, $\boldsymbol{R}_{II} = \{3, 0, 6\}$; $\quad \rightarrow \boldsymbol{R}^3 = \{1, 1, 0, 3, 0, 6\}$.
4. $\boldsymbol{R}_I = \{\text{false}, \text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{0, 3, 6\}$; $\quad \rightarrow \boldsymbol{R}^4 = \{0, 1, 1, 0, 3, 6\}$.

Since $\boldsymbol{R}^1 > \boldsymbol{R}^2 > \boldsymbol{R}^3 > \boldsymbol{R}^4$, the final ranking is 1, 2, 3, 4, but only the first two decompositions are fully feasible.

Using the same procedure, the best decompositions shown in Figure 5.8 were chosen for the following topological solutions for the same problem.

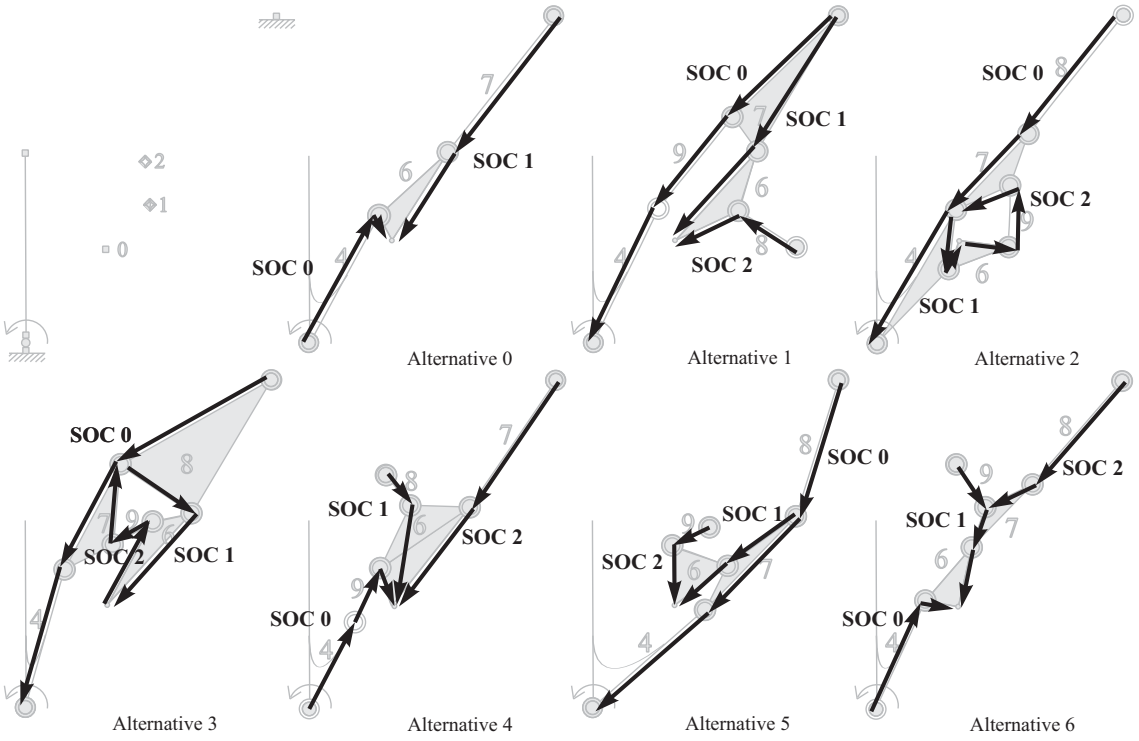



Figure 5.8: Best decompositions for **Alternatives 0** to **6** of the path following problem.

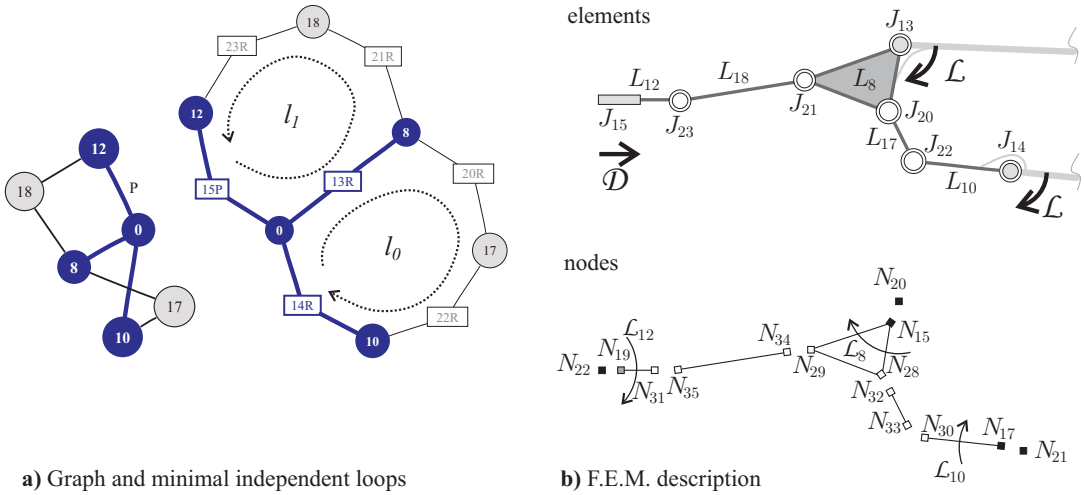
## 5.5.2. Nozzle of a turbine engine

The problem shown in Figure 2.2 schematizes the prescribed coordination between two flaps of a turbine engine and the horizontal movement of a hydraulic cylinder. The physical meaning of the initial graph vertices is 8 (primary flap), 10 (secondary flap), and 12 (hydraulic cylinder). Four positions are given for each

flap while only the starting and ending positions are prescribed for the cylinder (2 positions).

The first solutions available from the type synthesis stage were previously shown in Figures 3.7 and 3.8.

The synthesized nodes and elements for the **Alternative 0** are illustrated in Figure 5.9. From the kinematic analysis of the initial parts, the sets of all displace-




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

Figure 5.9: Initial situation for the decomposition of the simplest alternative for double function generation.

ments for nodes $N_{15}$ and $N_{17}$ were obtained, so they are shown in black color in Figure 5.9-b. On the other hand, since the prismatic joint $J_{15}$ has only initial and final prescribed motions, the displacements for intermediate positions of node $N_{19}$ cannot be determined until the synthesis equations and parameters for the prismatic joint are computed.

The decomposition process results in two SOCs for each loop-ordering. Below, the circular tables obtained for the first loop-ordering $l_0 - l_1$ are described.

The circular Table 5.3 for SOC identification in loop $l_0$ is filled as follows.

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

Table 5.3: Circular table for loop $l_0$ in the nozzle problem.

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

The next loop obtained, $l_1$, is filled as it is shown in Table 5.4. The corresponding decomposition is illustrated in Figure 5.10-1. The orientations of the complex numbers in the **SOC 1** are inverted to make them compatible with the available solver module JLJLJL-Triad.

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

Table 5.4: Circular table for loop $l_1$.

The decomposition for the loop-ordering $l_1 - l_0$ is treated in the same form, and its result is shown in Figure 5.10-2.

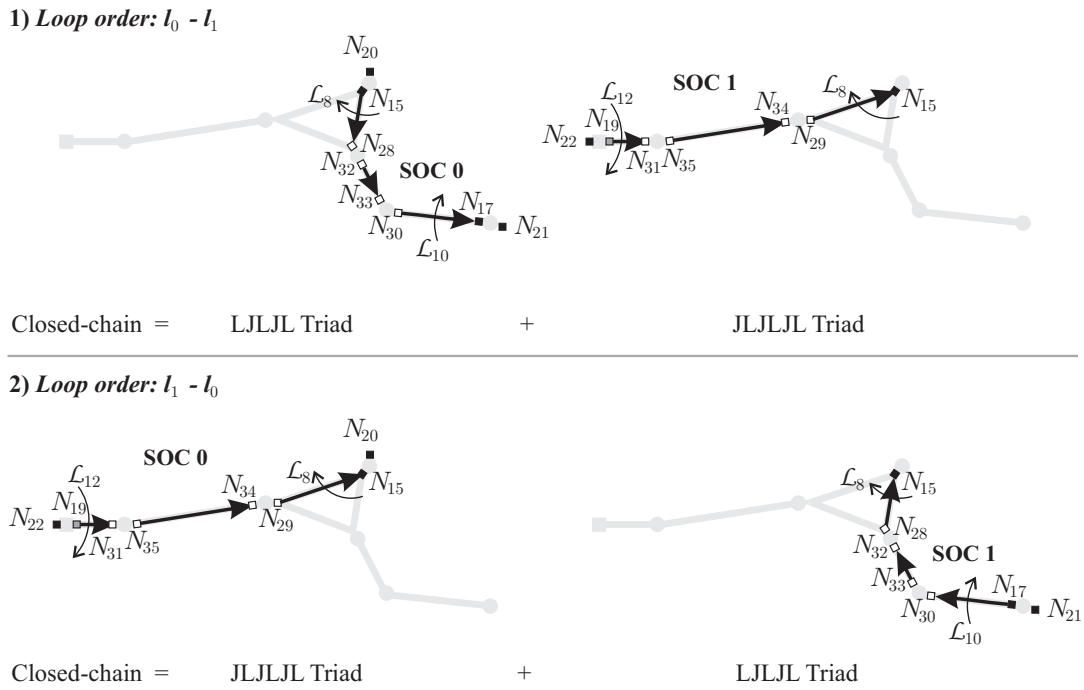



Figure 5.10: All decompositions for the double function generation problem.

The obtained evaluations for both decompositions were:

1. $\boldsymbol{R}_I = \{\text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{8, 6\}$; $\rightarrow \boldsymbol{R}^1 = \{1, 1, 8, 6\}$.
2. $\boldsymbol{R}_I = \{\text{true}, \text{true}\}$, $\boldsymbol{R}_{II} = \{6, 8\}$; $\rightarrow \boldsymbol{R}^2 = \{1, 1, 6, 8\}$.

From the first part of the evaluation, both decompositions result solvable, then, following the lexicographical comparison, the algorithm finds that $\boldsymbol{R}^1 > \boldsymbol{R}^2$; therefore, the first one is chosen for dimensional synthesis (see Figure 5.11).

From all decompositions, it may be observed that SOCs are coupled by link 8. Then, since no "new" motion constraint solved in one SOC is transferred to the other, the synthesis of SOCs may be considered as *order-independent*. For this case, the decomposition algorithm made superfluous evaluations; however, a general theorem for detection of "order-dependency" is a difficult task and an almost unexplored field of research [YYZ98a, YYZ98b]. On the other hand, the presented algorithm is computationally cheap and fast.

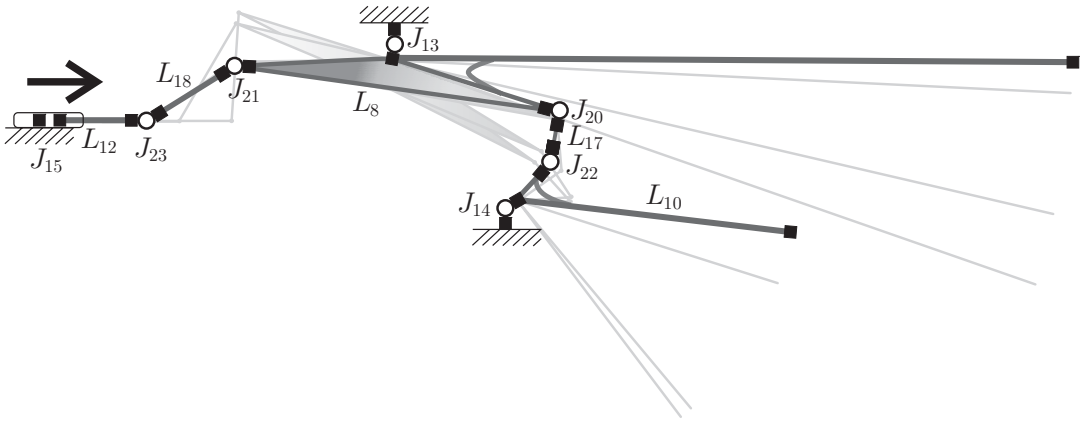



Figure 5.11: Dimensional synthesis of the nozzle problem using the first decomposition.

## 5.6. Chapter conclusions

An automated method to generate, evaluate and rank all SOCs decompositions from the sets of minimal independent loops of the graph representation of a mechanism was proposed. The decomposition can be used for the synthesis of any kinematic problem based on the Precision Point Method. The rules for evaluation are heuristic, and the examples showed that they do not reject any feasible solution and offer the best sequence of SOCs ranked in the first place. However, this best sequence is not always the optimal, since there are decompositions in which the SOCs belong to different loops in an alternated way. Thus, the decomposition algorithm can still be improved.

The presented algorithm for computing the set of *minimal independent loops* works well with graphs without attributes on their vertices and edges –link and joint types, respectively. However, when attributes are considered, loops with equal lengths could be isomorphic and therefore, two minimal loop bases could be isomorphic. So, in the presented method unnecessary evaluations would be avoided by detecting isomorphic bases. In future research, an isomorphism identifier for colored loops must be defined to find *all non-isomorphic minimal loop bases*.

The rules for decomposition based on FEM description can be extended to tridimensional problems.

One may easily deduce that it is not possible to build a data base of decompositions for all kinematic problems. However, the presented method can be useful to construct a data base for the most popular kinematic tasks such as PF, FG and RBG, for mechanisms of up to eight links.

Recently Sunkari [SS06] has pointed out that some kinematic chains, with more than 10 links, have associated *non-planar graphs*. For the family of mechanisms derived from those kinematic chains the proposed method would lack generality and its treatment must be investigated.

In Chapter 8, the minimal loop basis computed in this decomposition stage will be used for the synthesis of planar compliant mechanisms.

# Chapter 6

# Initial Sizing strategy

This chapter deals with the automated method applied for sizing a mechanism by means of analytical synthesis. The proposed strategy starts from a given topology with motion constraints in some of its parts, decomposed in a sequence of single-open chains (SOCs) as was described in the previous chapter.

In order of complexity, this chapter addresses the algorithms and data structures for solving two typical cases of synthesis: (i) without free parameters and (ii) with them.

The Loop-Closure Equations programmed as modules –or black boxes– for solving SOCs (Chapter 4) are also suited for the use of a Genetic Algorithm (zero order search) for sweeping the design space in case of presence of free parameters.

The aim is to find, among all combinations of multiple solutions of ordered SOCs, the best set of variables that minimizes a goal function. As a first criterion, the *summation of link sizes* of the mechanism is proposed to be minimized subject to some design restrictions that are crucial for any linkage design problem: *minimal link lengths*, *non-inversion of transmission angles*, and *allowed space*.

## 6.1. Introduction

Most of the computational methods for exact synthesis are based on the Burmester curves –also called M-K curves– that describe the locus of all possible locations of the pivot (circle-point curve) and moving joint (center-point curve) for a given task specified by a number of precision positions [ES97, SE84, Sar97, ME05, BCCA07]. Software for planar linkages like KINSYN, RECSYN and LINCAGES allowed the user to graphically interact with these curves, and then to see in real-time their associate mechanism with its generated coupler-point curve. The expert synthesist can interpret, from the Burmester curves and "solution maps" [YEB02], zones of the variables for which the mechanism violates the constraints or has no solution.

Of course, these programs also admit modifying the mechanism geometry directly and watching the generated task at the same time. This is a more intuitive and direct interaction that is nowadays available in several commercial software, such as SAM, WATT and SyMech. No coupler curve is displayed if the linkage has no solution; in this sense, the designer is not able to know the mathematical reason for the lack of solutions. Therefore, different design philosophies are focused on reducing the time for problem comprehension and designer training. On the other hand, academic programs are massively using Graphic User Interfaces (GUI) to facilitate the study

of the exact synthesis problem [YEB02, SCM02, McC, BCCA07].

It is noteworthy that in all of these programs, the user must choose the topology or build it by addition of sub-chains to a basic mechanism (WATT [DK02], SyMech [CO02]).

Software like SAM and WATT automatically computes a set of initial guesses for later executing their optimization solvers.

Due to the non-linearities of the design space and the mandatory *good initial guesses* claimed by gradient-methods [FS59, RF63, SVGF04, Col07], several researchers have combined, in different ways, strategies for global optimization. Most of these methods use a preliminar zero-order search for "exploring" the design space, followed by gradient-methods for doing a local search ("exploitation") of the optimum mechanism. The former zero-order search algorithms were based on:

1. Experimental design techniques, such as *Monte-Carlo*, *Orthogonal Arrays* [KC93], *Gross-Fine search* [MG03];

2. Simplex methods, for example, the *Nelder-Mead* algorithm [NM65][1] used by Da Lio *et al.* [LCL00];

3. Heuristic and stochastic algorithms, such as *Random Search* [Han93], *Genetic and Evolutionary Algorithms* [KK95, CSP02, ZC02, FBAAA05, Col07], *Simulated Annealing*, *Tabu Search* and *Ants Colony* [SD07].

4. Combined methods, for example, the Monte-Carlo and Genetic Algorithms combination used in SAM [Ran07].

In this setting, an initial sizing strategy based on the use of Genetic Algorithms is proposed to sweep the design space defined by free parameters that are automatically identified and bounded by the user. This optimization strategy does not require any kinematic analysis, thus the execution of the analytical evaluation is fast and often leads to good solutions. However, geometrically valid but kinematically invalid exact solutions may occur. Several constraints were designed for obtaining better mechanisms. An *allowed space* constraint uses simple algorithms based on concepts of computational geometry to penalize solutions that fall out of this area. The main disadvantage of GAs is the elevated number of function evaluations. However, the sizing of a complex topology can be achieved in few seconds due to the current speed of the modern PCs.

In order to avoid "linkage defects" such as *branching* and *wrong order*, the same strategy can be improved by incorporating as a criterion the measure of the kinematic error at the required precision positions. This evaluation requires one kinematic analysis per set of proposed parameters.

A resultant solution obtained through the method presented in this chapter can then be refined by available commercial software[2].

The organization of the chapter is as follows. The data of the problem is presented in Section 6.2. In Section 6.3, the determination of the variables and their bounds are detailed with aid of the used data structure. The models for the objective

[1] Implemented in the mincon function of MATLAB®.

[2] Specifically, the Svanberg *Globally Convergent Method of Moving Asymptotes* (GCMMA) was used in combination with a *Sensitivity Analysis* inside the SAMCEF BOSS/Quattro® environment [RR02, SAM07, Car02, CCSP07].

function and restrictions are explained and illustrated with examples in Section 6.4. A path following problem previously used in the type synthesis part of this thesis is used to illustrate the solution scheme in Section 6.5 and to show the results for its two particular cases in Section 6.6.

## 6.2. Data of the problem

In Chapter 4 we reviewed the use of complex numbers to model links, from which we can write *the displacement equations* for the different ordered SOCs that contain all the significative dimensions of a mechanism, and for a number of $n_{pp}$ prescribed positions, in the form:

$$\mathbb{C}^k\mathbb{Z}^k = \mathbb{D}^k \qquad k = 0, \ldots, s-1 \quad \text{(no summation over } k \text{ implied)} \tag{6.1}$$

where $s$ is the number of single open chains in which the mechanism has been decomposed (Chapter 5); the supra-index $k$ denotes the SOC number or *computing order*; $\mathbb{C}^k$ is a complex matrix whose data depend on the task and joint types, $\mathbb{D}^k$ is a complex vector that depends on the end-point displacements (and implicitly on the imposed pivot positions), and $\mathbb{Z}^k$ is a complex vector that represents the links of the single open-chain to compute, i.e., the unknowns of the problem. Note also that, using the decomposition of the previous chapter, all vectors $\mathbb{Z}^k$ result different. However, motion constraints and displacements may be coupled in matrices $\mathbb{C}^k$ or vectors $\mathbb{D}^k$, i.e., data change hierarchically in *computing order*. After a solution of the system of equations (6.1) is computed, the full mechanism configuration is known since all joint parameters are computed by post-processing of link rotations inside the proper SOC module.

Let us denote $\boldsymbol{x}$ the set of identified free parameters, and $\boldsymbol{p}$ the set of fixed parameters. An open-chain $k$ in the computing order has a $k$-th system of equations in (6.1) that may have a unique solution or a finite multiplicity of solutions, for instance, 2 or 4. If the SOC has free parameters, it is said that the SOC might have an infinite or multiple infinite number of solutions. Each of the multiple solutions defines a new subproblem for the consecutive chains –to which intersect or share parts– with different sets of fixed parameters $\boldsymbol{p}_m^{k+1}$; the index $m$ denotes the $m$-th solution for the SOC $k$.

Lack of solution may also occur and therefore such topology must be discarded. The main source of interruption is the non-existence of an analytical solution for one of the SOCs solvers in computing order. On the other hand, if an analytical solution exists, it can be qualified poor or bad due to (i) a marked violation of constraints, and/or (ii) the solution is unacceptable because of branch and circuits defects [BC02a]. The lack of solution for so simplified –3 or 4 precision positions– synthesis problem is a good index of its difficulty. Since this is a highly non-linear and non-convex problem, we employ a zero order search method to propose a set of free parameters, and then compute the links by means of the SOCs sequence.

Therefore, either with or without free parameters, in case of multiple solutions we have a solutions tree with several branches that have to be covered successively. Thus, the strategy involves a branched process in a mix of combinatorial and zero-order optimization techniques, locating each feasible mechanism near global optimum for the considered criterion. So, the assembly is not unique because the solution

is not generally unique, and those solutions with best performance index and best satisfaction of restrictions are retained for the preliminary analysis stage. In the presented approach, only one solution per topological alternative is retained.

## 6.3. Determination of variables–Free parameters

Compared to other synthesis methods, the Precision-Point Method is useful for reducing the number of variables of the problem. This reduction is notable particularly when the number of prescribed precision positions is low: 3 or 4. The *determination of the variables of the problem* is a complex subject of the PPM. It is complicated mainly because the user has the possibility of leaving undefined parameters of varied nature: node displacements, joint rotations, link rotations, etc. Fortunately, modules for solving SOCs facilitate its automated identification.

The decomposition algorithm provides a set of SOCs that satisfies the solvability conditions. Then, for a given SOC $k$ with $n_L^k$ links, $n_{pp}$ precision positions, and given motion constraints, the system of Loop-Closure Equations may be linear or non-linear, and the offset is always included.

The following rules were programmed in SOCs modules for variables identification:

1. In the linear case, all entries of $\mathbb{C}^k$ may be imposed, and the missing ones are considered as *free* parameters.

2. In the non-linear case, one column of $\mathbb{C}^k$ with completely undefined motion constraints is chosen to develop the compatibility linkage and to solve then its associated solution structure; the DOFs proposed in the compatibility linkage are *free parameters*. The "other columns" of $\mathbb{C}^k$ must be data. Missing data in such columns will be also considered as *free parameters*.

3. A third source of free parameters are the positions of pivots. A new pivot position is located inside a box, and then the offset $\boldsymbol{r}$ can be defined.

4. For a trajectory node, a given component ($x$ or $y$) of the displacement at a given time $j$ can be left free to take any –judiciously bounded– value. The missing data in nodes displacements will be taken as *free parameters* in such a way that displacement vectors $\boldsymbol{g}^j$ and $\boldsymbol{h}^j$ are completed, and consequently $\mathbb{D}^k$ vectors are known inside the SOC modules.

Let us denote $\boldsymbol{x}$ the set of identified free parameters. For example, it could have the form:

$$\boldsymbol{x} = [\underbrace{\alpha_0^1 \quad \alpha_0^2}_{\textbf{SOC 0}} \quad \underbrace{\rho_3^1 \quad d_{x\,N_2}^{\;1}}_{\textbf{SOC 1}} \quad \underbrace{x_{N_5}^0 \quad y_{N_5}^0}_{\textbf{SOC 2}}]$$

The position of a variable $x_i$ in $\boldsymbol{x}$ is algorithmically determined by executing the rules mentioned above for the sequence of SOCs. Vector $\boldsymbol{x}$ can also be seen as partitioned into SOCs:

$$\boldsymbol{x} = [\underbrace{\boldsymbol{x}_0}_{\textbf{SOC 0}} \mid \underbrace{\boldsymbol{x}_1}_{\textbf{SOC 1}} \mid \underbrace{\boldsymbol{x}_2}_{\textbf{SOC 2}}].$$

### 6.3.1. Bounds for variables

In order to restrict the search space, lower and upper bounds are needed for the free parameters, denoted as $\boldsymbol{x}_{\min}$ and $\boldsymbol{x}_{\max}$.

Three categories of parameters can be identified:

- *Angular displacements*, whose allowable variation is $\theta \in [0, 2\pi]$, default values are set in $\theta \in [-j, j]$, where $j$ is the number of precision position or pseudo-time;
- *Linear displacements*, whose variation is fixed by default to $\rho \in [0, j]$ but it necessarily must be defined by the user, and
- *Pivot positions* $(x; y) \in ([x_{\min}, x_{\max}]; [y_{\min}, y_{\max}]) \subset A$, that are confined in boxes inscribed in the allowed space $A$ (defined by a polygonal area).

We should point out that the "sliding" in a prismatic joint is modelled by a *stretch factor* applied on the attached complex number $\boldsymbol{Z}$ in the way that it is stretched by $\rho\boldsymbol{Z}$. For instance, a value $\rho = 1.7$ means that the magnitude of $\boldsymbol{Z}$ is increased in 70%. But, in principle, the magnitude of $\boldsymbol{Z}$ is unknown, so the definition of bounds for $\rho$ variations is a difficult and non-intuitive task for the user. To circumvent this problem of SOCs with prismatic joints, the Loop-Closure Equations must be written *for linear displacements* instead of the stretch factors. This greatly facilitates bounds definition to the user.

### 6.3.2. Synthesis degrees-of-freedom table

An auxiliary table called synthesis degrees-of-freedom, SyDOFs, is used to store and link all data involved in the mechanism synthesis problem. Although the SyDOFs data structure is presented as a "table", it is a "class" with a complex interface for many reading and writing operations.

The SyDOFs table enables to connect the decomposed single-open chains to a unique data structure (in such a way that no repeated data is created), and implicitly enables to connect the solver of SOCs with this table. This table is built after the type synthesis execution. Thus, the SyDOFs data structure is used in three main stages:

1. To simulate the dimensional synthesis process by assembling the open-chains in the computing order provided by SOC decomposition.
2. To determine the free parameters $\boldsymbol{x}$ of the problem for which their bounds $\boldsymbol{x}_{\min}$ and $\boldsymbol{x}_{\max}$ must be defined.
3. For computing the dimensional synthesis: in the *computing order* every solved SOC modifies its associated set of SyDOF's in this table. Since a SOC could share nodes, links or joints with the following SOCs, the solving process must dynamically change in a proper way.

An individual synthesis degrees-of-freedom SyDOF is a row of the table. It is uniquely characterized by a set of three identifiers {Type, ID, Physical meaning} plus an additional variable I/O for defining its function. Types of mechanism components are grouped in Node and Elem, which can have different physical meanings:

Node: Value for
- x, or
- y component.

Elem: Parameter for
- Link, or

| SyDof | | | | | Precision Position | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i | Type | ID | Phys | I/O | 0 | | | | 1 | | | | 2 | | | |
| | | | | | ord | kind | sub | state | ord | kind | sub | state | ord | kind | sub | state |
| 0 | Node | 4 | x | | 0 | fix | | 0.00 | 0 | fix | | 0.00 | 0 | fix | | 0.00 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 1 | Node | 4 | y | | 0 | fix | | 0.00 | 0 | fix | | 0.00 | 0 | fix | | 0.00 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 2 | Node | 7 | x | | 0 | cmp | post | ? | 0 | cmp | post | ? | 0 | cmp | post | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 3 | Node | 7 | y | | 0 | cmp | post | ? | 0 | cmp | post | ? | 0 | cmp | post | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 4 | Node | 10 | x | | 0 | cmp | post | ? | 0 | cmp | post | ? | 0 | cmp | post | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 5 | Node | 10 | y | | 0 | cmp | post | ? | 0 | cmp | post | ? | 0 | cmp | post | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 6 | Node | 1 | x | O | 0 | cmp | pre | 0.40 | 0 | cmp | pre | 0.60 | 0 | cmp | pre | 0.58 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 7 | Node | 1 | y | O | 0 | cmp | pre | 0.50 | 0 | cmp | pre | 0.70 | 0 | cmp | pre | 0.90 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 8 | Elem | 9 | J | | 0 | cmp | post | ? | 0 | cmp | post | ? | 0 | cmp | post | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 9 | Elem | 4 | L | | 0 | fix | | 0.00 | 0 | fix | | 0.44 | 0 | fix | | 0.80 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 10 | Elem | 6 | L | | 0 | fix | | 0.00 | 0 | cmp | solve | ? | 0 | cmp | solve | ? |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 11 | Node | 2 | x | | 0 | fix | | 1.20 | 0 | fix | | 1.20 | 0 | fix | | 1.20 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 12 | Node | 2 | y | | 0 | fix | | 1.60 | 0 | fix | | 1.60 | 0 | fix | | 1.60 |
| | | | | | 1 | cmp | pre | | 1 | cmp | pre | | 1 | cmp | pre | |
| 13 | Node | 8 | x | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 14 | Node | 8 | y | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 15 | Node | 9 | x | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 16 | Node | 9 | y | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 17 | Node | 11 | x | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 18 | Node | 11 | y | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 19 | Elem | 8 | J | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 20 | Elem | 10 | L | | 0 | und | | | 0 | und | | | 0 | und | | |
| | | | | | 1 | cmp | post | ? | 1 | cmp | post | ? | 1 | cmp | post | ? |
| 21 | Elem | 7 | L | | 0 | und | | 0.00 | 0 | und | | | 0 | und | | |
| | | | | | 1 | fix | | | 1 | cmp | solve | ? | 1 | cmp | solve | ? |
| 22 | Node | 6 | x | | 0 | fix | | 0.00 | 0 | fix | | 0.00 | 0 | fix | | 0.00 |
| | | | | | 1 | fix | | | 1 | fix | | | 1 | fix | | |
| 23 | Node | 6 | y | | 0 | fix | | 0.00 | 0 | fix | | 0.00 | 0 | fix | | 0.00 |
| | | | | | 1 | fix | | | 1 | fix | | | 1 | fix | | |
| 24 | Elem | 5 | J | I | 0 | fix | | 0.00 | 0 | fix | | 0.44 | 0 | fix | | 0.8 |
| | | | | | 1 | fix | | | 1 | fix | | | 1 | fix | | |

Table 6.1: Example of SyDOFs table for a four-bar case (problem shown in Figure 5.5)

- Joint.

An additional variable is referred to as the Input/Output function of the mechanism component.

I/O: there are three possible functions
- Input,
- Output, or
- passive.

A state is a field of `double` precision data type that stores the value of a node position or element parameter. A SyDOF has as many states as passing points were defined in the task. The execution of the solver for a given SOC must properly modify the state of itsr corresponding SyDOF's. The aim is to compute all states of the table. Each state may have three different kinds of data, namely:

- fixation: is a state imposed by the user at the initial definition or imposed by a previous open-chain.
- genetic: is assigned if the state is a free parameter.
- computed: is assigned if the state is either computable from the other states of the same SOC or by using *PPM*. Between the different computing orders, these states could have different instants or sub-orders of computation, namely:
  - ♢ pre: the state is computable by pre-processing other states.
  - ♢ solve: the state is solved by the Loop-Closure Equation module.
  - ♢ post: the state is computable from post-processing of other states of the same SOC.

For explanation purposes, the four-bar mechanism, whose decomposition was shown in Figure 5.5-1, is taken as an example. In order to build the Table 6.1, SOCs are successively loaded. A SyDOF row is created for any component of their nodes, joints and links (in that order). Since fields kind and sub-order are order-dependent, each SyDOF has as many sub-rows as SOCs the mechanism has. Although there is only one state per SyDOF, for a clearer explanation the symbol ? is used in the proper sub-row of the table for indicating for which SOC (order) its state must be computed.

The **SOC 0** is constituted by nodes $N_4$, $N_7$, $N_{10}$ and $N_1$; they are indexed in the Table 6.1 by rows $i$ = 0-7; joint $J_9$ is then added, and next, links $L_4$ and $L_6$. The **SOC 1** is embraced by rows $i$ = 11-21 and it shares node $N_1$ and link $L_4$ with **SOC 0**. The last rows in the table, $i$ = 22-24, are completed with those parts that were completely determined after the initial kinematic analysis of prescribed parts. Note that nodes $N_4$, $N_1$, $N_2$ and $N_6$ have known positions (with fixation kind) for all precision positions, while nodes $N_7$ and $N_{10}$ (with post kind) will be computed after solving the Loop-Closure equations of **SOC 0**. Regarding links, link $L_4$ has imposed movements (with fixation kind ) while the rotations of link $L_6$ have solve kind because they must be solved analytically. Then, the parameters of joint $J_9$ are computed by post-processing of rotations of links $L_4$ and $L_6$.

The next open chain **SOC 1** will be correspondingly solved in order 1. Here, the rotations of $L_6$ have computed-pre kind, that is, they are obtained from single use of the state computed in a previous SOC. Using the Loop-Closure equations, the rotations of $L_7$ will be solved, and from their results, the positions of nodes $N_8$,

$N_9$, $N_{11}$, and rotations of joint $J_8$ will be post-processed. This logical procedure is programmed in the form of the virtual member function `Assemble` of the SOC modules. Another output of this function is detailed in the next sub-section.

After assembling, kinds and sub-orders are determined. By means of this process, the free parameters result identified as genetic variables. If they exist, bounds definition is required before the execution of the initial sizing solver. Thus, the intervention of the user is mandatory to set bounds and limit the design space, otherwise default values for bounds are used.

Table 6.1 is used to solve the SOCs. Then, after solving all SOCs, the state columns are filled. The results for the given example are shown in Table 6.2.

| SyDof | | | | | Precision Position | | |
|---|---|---|---|---|---|---|---|
| i | Type | ID | Phys | I/O | 0 | 1 | 2 |
| | | | | | state | state | state |
| 0 | Node | 4 | x | | 0.00 | 0.00 | 0.00 |
| 1 | Node | 4 | y | | 0.00 | 0.00 | 0.00 |
| 2 | Node | 7 | x | | 0.3867 | 0.5222 | 0.5597 |
| 3 | Node | 7 | y | | -0.4047 | -0.2014 | -0.004547 |
| 4 | Node | 10 | x | | 0.3867 | 0.5222 | 0.5597 |
| 5 | Node | 10 | y | | -0.4047 | -0.2014 | -0.004547 |
| 6 | Node | 1 | x | O | 0.40 | 0.60 | 0.58 |
| 7 | Node | 1 | y | O | 0.50 | 0.70 | 0.90 |
| 8 | Elem | 9 | J | | 0.00 | -5.772 | -5.475 |
| 9 | Elem | 4 | L | | 0.00 | 0.44 | 0.80 |
| 10 | Elem | 6 | L | | 0.00 | 6.212 | 6.275 |
| 11 | Node | 2 | x | | 1.20 | 1.20 | 1.20 |
| 12 | Node | 2 | y | | 1.60 | 1.60 | 1.60 |
| 13 | Node | 8 | x | | 1.20 | 1.20 | 1.20 |
| 14 | Node | 8 | y | | 1.60 | 1.60 | 1.60 |
| 15 | Node | 9 | x | | 1.15 | 1.411 | 1.337 |
| 16 | Node | 9 | y | | 1.382 | 1.527 | 1.777 |
| 17 | Node | 11 | x | | 1.15 | 1.411 | 1.337 |
| 18 | Node | 11 | y | | 1.382 | 1.527 | 1.777 |
| 19 | Elem | 8 | J | | 0.00 | 1.462 | 2.709 |
| 20 | Elem | 10 | L | | 0.00 | -4.749 | -3.567 |
| 21 | Elem | 7 | L | | 0.00 | 1.462 | 2.709 |
| 22 | Node | 6 | x | | 0.00 | 0.00 | 0.00 |
| 23 | Node | 6 | y | | 0.00 | 0.00 | 0.00 |
| 24 | Elem | 5 | J | I | 0.00 | 0.44 | 0.8 |

Table 6.2: Solution table for a four-bar problem.

Note that, since both nodes of revolute joints share positions and displacements, Table 6.2 could be presented even in a more condensed way, but that is not the case for sliding joints.

### 6.3.3. Vector for combinations of solutions

After the type synthesis process is finished, each alternative can be decomposed into SOCs. The different open chains may have different multiplicity of solutions.

Then, two data are loaded by the execution in succession of the member function `Assemble` of each SOC module: (1) the SyDOFs table and (2) a vector indexed by SOCs that contains the number of solutions for each SOC, denoted as the multiplicity vector $\boldsymbol{m}_{\max}$. This execution is automatically done just after a decomposition is selected as feasible.

So, a loop over SOCs ($k = 0, \ldots, s-1$) in the form of

$$\mathbf{SOC}[k].\texttt{Assemble}(\mathsf{SyDOFs}, \boldsymbol{m}_{\max}[k])$$

is executed in computing order, while the proper part in the SyDOFs is modified by each SOC module and the corresponding component in the multiplicity vector $\boldsymbol{m}_{\max}$ is loaded.

For instance, the previous four-bar example decomposed in two SOCs results in:

$$\boldsymbol{m}_{\max} = \begin{bmatrix} \textbf{SOC 0} & \textbf{SOC 1} \\ 2 & 2 \end{bmatrix}.$$

Using $\boldsymbol{m}_{\max}$, a matrix with indexes to all possible combinations of solutions is built:

$$\boldsymbol{M}(\boldsymbol{m}_{\max}) = \begin{bmatrix} \textbf{SOC 0} & \textbf{SOC 1} \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{6.2}$$

so that, a row in $\boldsymbol{M}$ is simply denoted as the *combination vector* $\boldsymbol{m}$ and it is used to choose a set of solutions in the system of equations (6.1). Suppose that $\boldsymbol{M}$ consists of $c$ combinations, then a number of $c$ problems must be solved. No distinction is made if the problem presents or does not present free parameters.

For instance, for solving the previous four-bar example shown in Figure 5.5 whose associated SyDOFs table is shown in Table 6.1 and multiplicity vectors are taken from $\boldsymbol{M}$ (equation (6.2)), four executions of the equations (6.1) are needed. Results for the *best solution* are presented in Table 6.2 and corresponds to Figure 5.1-b. The next section is dedicated to define what "best" means, and therefore to define the criterion for retaining solutions among all combinations.

## 6.4. Objective function

In the sizing algorithm, the objective function to minimize is the size of the mechanism, which is defined as the summation of the link sizes as follows:

$$F^*(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_L - 1} s(L_k) \tag{6.3}$$

where $n_L$ is the number of links in the mechanism. Then, for a given link $k$, function $s(\ )$ returns a measure of its *size* by considering the summation of all distances between their $n^k = n_c^k + n_p^k$ nodes, where $n_c^k$ nodes are connected by joints, and eventually, $n_p^k$ nodes have prescribed movements (trajectory nodes); see Figure 6.1. All distances between pairs of nodes are computed without repetition, i.e,

$$s(L_k) = \frac{1}{\binom{n_c^k}{2}} \sum_{i=0}^{\binom{n_c^k}{2}} d(N^k_{(C[i,0])}, N^k_{(C[i,1])}). \tag{6.4}$$

Function $d(\ ,\ )$ takes two nodes as arguments and returns its Euclidean distance, while the matrix $C_{ij}$ contains the $\binom{n_c^k}{2}$ pairs of combinations of nodes arranged by rows, in such a way that, nodes can be properly indexed.

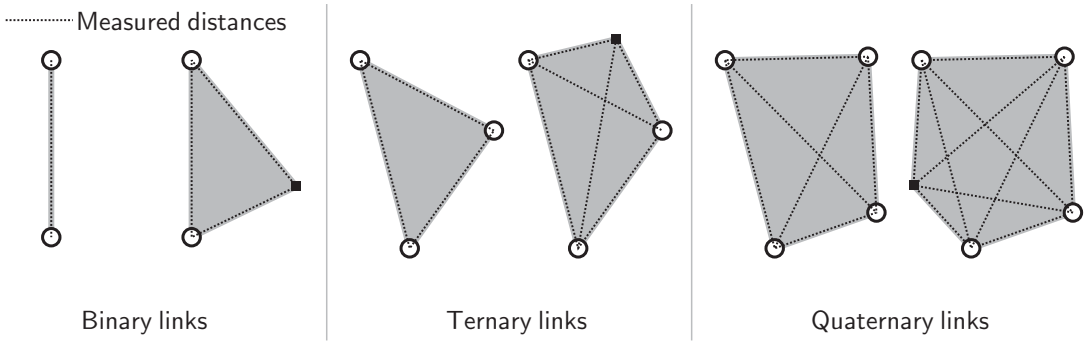



Figure 6.1: Distances to be considered in *Objective function* and *Minimal link lengths* restriction.

### 6.4.1. Evaluation of restrictions

Even with or without free parameters, restrictions are evaluated after the SOCs were successfully computed and assembled. Three restrictions are available:

*Minimal link lengths*: this restriction is added (mainly due to constructive fundamentals) to avoid too short links, while too long links are implicitly avoided by the minimization of the objective function.

*Allowed space*: it represents a restricted area where all joints and links must lie at initial position and during the mechanism movement.

*Non-inversion of transmission angles*: this restriction avoids kinematically invalid movements between passing points (sometimes, the solution is geometrically correct but not kinematically correct). It minimizes *jamming risk*, avoiding any pair of links going through a dead-center position [Hal61, BC02b].

#### Minimal link lengths:

While the objective function presented in equation (6.3) is computed, an internal decision can be incorporated after computing every distance between nodes to check the violation of the minimal link length allowed.

$$q_L(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_L-1} s_q(L_k, L_{\min}) \tag{6.5}$$

where $L_{\min}$ is the *minimal link length* parameter, and function $s_q(\ )$ is the accumulative sum of distances $d_q$ that are required to be less than $L_{\min}$:

$$s_q(L_k) = \sum_{i=0}^{\binom{n_c^k}{2}} d_q(N^k_{(C[i,0])}, N^k_{(C[i,1])}), \tag{6.6}$$

where

$$d_q(\ ,\ ) = \begin{cases} L_{\min} - d(N^k_{(C[i,0])}, N^k_{(C[i,1])}) & \text{if} \quad d(N^k_{(C[i,0])}, N^k_{(C[i,1])}) < L_{\min}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

**Allowed space:**

$$q_A(\boldsymbol{x}, \boldsymbol{p}) = \max_{\substack{i=0,\dots n_J-1 \\ j=0,\dots n_{pp}-1}} < d(N_i(j), A) >, \quad \forall N_i(j) \subset \mathbb{R}^2 - A \tag{6.8}$$

where function $d(\ ,\ )$ returns the outer-distance from node $N_i$ at precision position $j$ to the allowed space $A(P_0, P_1, \dots, P_{n_A-1}, P_0)$ (the latter being defined as an oriented and closed polygon joining $n_A$ points, see Figure 6.2), $n_J$ is the number of joints in the mechanism, $N_i$ is the position of one node of $i$-th joint.

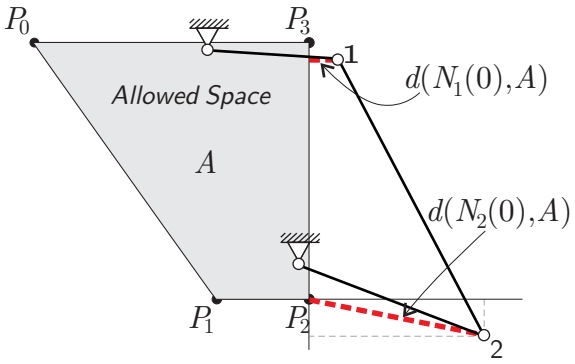



Figure 6.2: Space violation example, where distances of nodes defining joints 1 and 2 to allowed space are shown.

The algorithm for computing distance $d(\ ,\ )$ is a two step procedure: (1) detect if node $N_i$ is outside the allowed space[1], (2) if such point is outside, compute the penetration of that point outside the region. The highest penetration is updated in succession.

This constraint can be checked for initial position only ($j = 0$), or for all configurations, and thus an outer loop over precision positions is run ($j = 0, \dots, n_{pp} - 1$) as it is shown in equation (6.8).

**Non-inversion of transmission angles:**

This restriction is mathematically expressed as follows:

$$q_T(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_J-1} \sum_{j=0}^{n_{pp}} \Delta\psi_k^j \tag{6.9}$$

where $n_J$ is the number of joints in the mechanism, function $\Delta\psi_k^j$ returns the violation of the movement in joint $k$ at precision position $j$ when their connected links align, i.e. they start, end or pass through a dead-center position. The alignment of links is identified as follows:

Let $L_0$ and $L_1$ be two vectors that represent the initial position of two links connected by a revolute joint $J_k$. Then, the included angle between them is defined as $\psi_k^0$, computed as $\text{ mod } (\psi_k^0, 2\pi)$. Suppose $L_0$ as fixed, then $L_1$ is allowed to have a relative movement respect to $L_0$ inside a half space defined by $L_0$. That is, the included angle $\psi_k^j$ between $L_0$ and $L_1$ at a given position $j$ is bounded to be inside one of two open intervals $(l_{\min}, l_{\max})$ defined by $\psi_k^0$ in the following way

[1] by means, for example, of a "winding number" computation [O'R98, Chap. 7, Sec. 4].

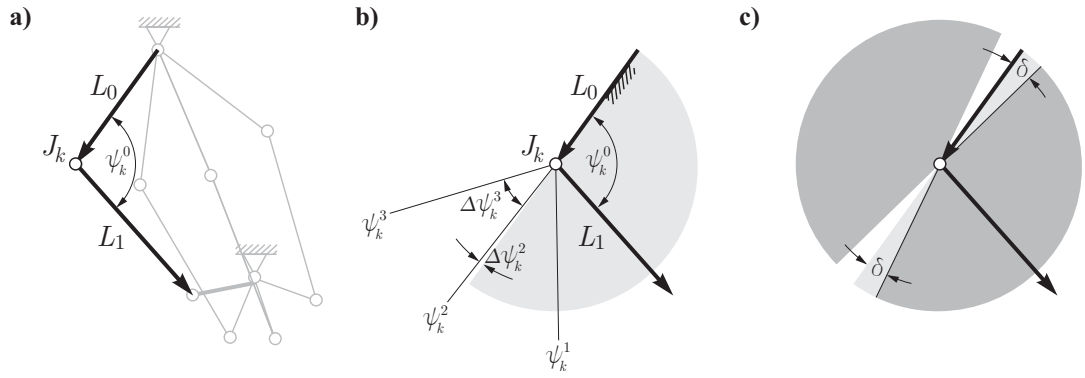



Figure 6.3: Graphic interpretation of *non-inversion of transmission angles* restriction: (a) given solution, (b) measured angles and violations, (c) narrowed intervals.

$$\psi_k^j \subset \begin{cases} (0,\pi) & \text{if} \quad \text{round}\left(\frac{\psi_k^0}{\pi}\right) = 0 \\ (\pi,2\pi) & \text{if} \quad \text{round}\left(\frac{\psi_k^0}{\pi}\right) = 1, \end{cases} \tag{6.10}$$

then, if a given movement $\psi_k^j$ fall on the wrong interval, the difference

$$\Delta\psi_k^j = \min\{ \quad \text{mod}\,(\psi_k^j - l_{\max}, 2\pi), \quad \text{mod}\,(l_{\min} - \psi_k^j, 2\pi)\ \} \tag{6.11}$$

is accumulated as a constraint violation. Example of such violations are depicted in Figure 6.3-b.

Note that the angle $\psi_k^j$ defined here is not the transmission angle in its classical definition [Hal61, BC02b], but it is an indirect measure of it that coincides in its "sign". Therefore, the restriction assures that while links are moved, the mechanism does not pass through a dead-center position in some joint, i.e. a null transmission angle. A null value can be reached either at starting or ending the movement, but also at an intermediate position (so-called *toggle* position) of the range of movement in which transmission angle would suffer a change of sign. The latter is completely avoided if no contribution of any $\Delta\psi_k$ exists.

Mechanisms solutions with poor transmission angle at starting and ending positions can be conclusively avoided by modifying the allowed limits in equation (6.10)[1]. To do this, the intervals can be narrowed into an appropriate positive value $\delta$; for instance, in a minimal safety amount of 10 degrees. Instead of

$$0 < \psi_k^j < \pi \quad \text{or} \quad \pi < \psi_k^j < 2\pi,$$

it would be allowed

$$0 + \delta < \psi_k^j < \pi - \delta \quad \text{or} \quad \pi + \delta < \psi_k^j < 2\pi - \delta.$$

Following with the example given above, the feasible domain for links movements is shown in Figure 6.3-c.

[1]For specific applications, the designer could desire to obtain linkages which reach an ending limit position, which *must be actuated on a different link to reverse its movement* [Nor95]. The *folding linkages*, where space requirement is mandatory, are typical applications where poor transmission angles are present.

The restriction is computed for all joints. However, it is well-known that a non-actuated joint of a driver-link may work without jamming. For example, the crank-rocker mechanism where crank fully rotates. Therefore, this restriction can prune feasible but less reliable solutions. An example may illustrate this difference more clearly.

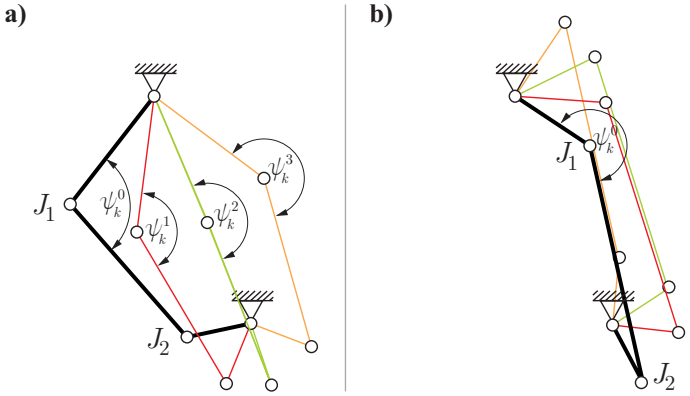



Figure 6.4: Effect of *non-inversion of transmission angle* restriction for two solutions of the same problem where (a) restriction is violated, and (b) restriction is satisfied.

Figure 6.4 shows two solutions for a problem of function generation, in which the upper driver link has the imposed rotations $\boldsymbol{\beta} = \{0, \pi/6, \pi/3, \pi/2\}$ and the inferior driving link rotations are $\boldsymbol{\alpha} = \{0, 1, 1.7, 2.6\}$. Both solutions, (a) and (b), are *geometrically* valid. However, note that in Figure 6.4-a the angles of joint $J_1$ are positive in the first and second positions, $\psi_1^0$ and $\psi_1^1$, whereas they are negative in the other positions. The same phenomenon occurs in joint $J_2$. The mechanism passes through a dead-center position, and it may block. Solution of Figure 6.4-b does not have this problem since it does not violate the restriction of *non-inversion of transmission angle*.

### 6.4.2. Search by using a Genetic Algorithm

Genetic Algorithms (GAs) [Gol89, Mic97] are suitable for problems where neither domain nor goal function and restrictions are known or they are so complicated that gradient computation becomes difficult or impossible. GAs are exploratory processes of optimization based on principles of genetic variation and natural selection. GAs are easy to implement for any number of variables and restrictions. Since GAs do not use derivatives, they need a considerable big quantity of evaluations of the objective function, compared to their gradient-based optimization counterpart.

In presence of free parameters, we use a simple Genetic Algorithm to find those free parameters that minimize the objective function subject to restrictions. Let us suppose that free parameters $\boldsymbol{x}$ of the problem are detected. Their variation bounds are defined depending on the type of variables. A set of values for $\boldsymbol{x}$ is called *individual*. Then, a population

$$\boldsymbol{X}^0 = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_i \quad \dots \quad \boldsymbol{x}_P]^T$$

of $P$ individuals verifying bounds is generated randomly. With index $i$, we will refer to a particular individual. Hereafter, the supra-index of the population will

denote an iteration $t$ of the algorithm whose biological interpretation is the number of generation commonly denoted with $g$.

While the genetic algorithm makes the population evolve naturally, those individuals with the highest fitness –most adapted in the biotope– have more probability to survive and to be *selected* for *reproducing* new offspring. These descendants have more probability to have better or equal fitness and less probability to decrease it. Naturally, after a finite number of generations, the population is standardized to a medium value from which the individual with best fitness is retained.

For this problem, the *fitness function* is designed exactly as the objective function expressed by equation (6.3) subject to the restrictions given in equations (6.5), (6.8), and (6.9) that are taken into account in the form of penalization by defining:

$$F(\boldsymbol{x}, \boldsymbol{p}) = F^*(\boldsymbol{x}, \boldsymbol{p}) + Q(\boldsymbol{x}, \boldsymbol{p}), \tag{6.12}$$

where the term $Q(\boldsymbol{x}, \boldsymbol{p})$ is the contribution from restrictions to the fitness function. Then, *best fitness* means *minimal size* with the best fulfilment of constraints.

The penalization may be composed of three terms weighted by factors $\lambda_i$ that are adjusted empirically:

$$Q(\boldsymbol{x}, \boldsymbol{p}) = \lambda_L q_L + \lambda_T q_T + \lambda_A q_A. \tag{6.13}$$

There are, however, other approaches for constraints management –alternative to the *Weighted Sum* shown in equation (6.13). Joines and Houck [JH94] proposed to consider the so-called *non-stationary penalty function* where constraints are dynamically dependent on the length of the search, i.e. of the generation number $t$:

$$Q(\boldsymbol{x}, \boldsymbol{p}) = (C \times t)^{\alpha} \sum_{k=0}^{n_q - 1} q_k^{\beta}, \tag{6.14}$$

where $C$, $\alpha$ and $\beta$ are constants, in such a way function $(C \times t)^{\alpha}$ is monotonically non-decreasing in value with $t$; $n_q$ is the number of constraints. In order to use this strategy, we define

$$q_0 = \lambda_L q_L, \quad q_1 = \lambda_T q_T \quad \text{and} \quad q_2 = \lambda_A q_A.$$

The evaluated objective function as well as the constraints for a given individual can be arranged in vectorial form as:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{m}) = [F^* \quad \lambda_L q_L \quad \lambda_T q_T \quad \lambda_A q_A]^T.$$

Note that the evaluation is made for a chosen set of solutions $\boldsymbol{m}$.

Default values for the balancing parameters $\lambda_i$ are set in terms of a *characteristic length* $D$, which is computed as the diagonal of the bounding box of the geometrical data (all node positions of the set $\boldsymbol{N}$), and a penalty value defined by the user $\lambda_{\max}$. By default, these parameters are heuristically fixed as follows:

$$L_{\min} = \frac{D}{15}, \quad \lambda_L = \frac{\lambda_{\max 0}}{2D}, \quad \lambda_T = \frac{\lambda_{\max 1}}{2} \quad \text{and} \quad \lambda_A = \frac{\lambda_{\max 2}}{D},$$

$$\text{with} \quad \lambda_{\max 0} = \lambda_{\max 1} = \lambda_{\max 2} = \lambda_{\max} = 1000,$$

but they can be changed by the user.

Additionally, if an analytical solution does not exist for the proposed free parameters, SOC solvers are programmed in such a way they answer with

$$\boldsymbol{f}_{\text{non\_sol}}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{m}) = [\lambda_{\max} \quad \lambda_{\max 0} \quad \lambda_{\max 1} \quad \lambda_{\max 2}]^T$$

since neither the objective function nor the restrictions can be computed.

The evaluation of the objective function and restrictions of the whole population can take the matrix form

$$\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{m}) = [\boldsymbol{f}_1 \quad \boldsymbol{f}_2 \quad \dots \quad \boldsymbol{f}_i \quad \dots \quad \boldsymbol{f}_P]^T.$$

Using this matricial form, the evaluation of the fitness of a single individual $\boldsymbol{x}_i$ is made as:

$$F(\boldsymbol{x}_i, \boldsymbol{m}) = \begin{cases} F^*(\boldsymbol{x}_i, \boldsymbol{p}) & \text{if } \boldsymbol{x}_i \text{ has a feasible solution} \\ F^*(\boldsymbol{x}_i, \boldsymbol{p}) + Q(\boldsymbol{x}_i, \boldsymbol{p}) & \text{if } \boldsymbol{x}_i \text{ has an unfeasible solution} \\ Q_{\max} = \lambda_{\max} + \sum_{k=0}^{n_q - 1} \lambda_{\max k} & \text{if } \boldsymbol{x}_i \text{ has no solution.} \end{cases} \tag{6.15}$$

Once the bounds of the variables are given, the starting population for the GA is generated:

$$\boldsymbol{X}^0 \leftarrow \texttt{initializeGA}(n_{\text{vars}}, n_{\text{bits}}, P, \boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}, n_{\text{constr}}, \boldsymbol{f}_{goal}, p_{\text{cross}}, p_{\text{mut}}).$$

Each variable $x$ of the $n_{\text{vars}}$ variables of an individual $\boldsymbol{x}$ is allowed to take one of the $2^{n_{\text{bits}}} - 1$ values, in which its domain $[x_{\min}, x_{\max}]$ is divided. Then, goals for the objective function and constraints are set to zero, thus the vector of size $n_{\text{constr}} = 1 + n_q$ is set as $\boldsymbol{f}_{goal} = \underline{0}$ for minimization.

For this initial population, the objective function is computed by means of analytical synthesis and restrictions are evaluated $\boldsymbol{F}(\boldsymbol{X}^0, \boldsymbol{m})$, then the first generation of the evolution program can start.

An evolution of a generation can be briefly denoted as

$$\boldsymbol{X}^t \leftarrow \texttt{evolveGA}(t, \boldsymbol{X}^{t-1}, \boldsymbol{F}(\boldsymbol{X}^{t-1}, \boldsymbol{m}))$$

from which a new population $\boldsymbol{X}^t$ is generated. This function achieves a single evolution of the population following these steps:

**S1**: compute the fitness of each individual using $\boldsymbol{F}(\boldsymbol{X}^{t-1}, \boldsymbol{m})$ and the equation (6.15);

**S2**: apply a *scaling* based on the fitness of the individuals for obtaining a ranked population;

**S3**: *select* the individuals to reproduce $\boldsymbol{X}^t$,

**S4**: apply operators of *crossover* and *mutation* to alter the new population $\boldsymbol{X}^t$,

**S5**: use a transition criterion, for instance, the *elitism* criterion is used for preserving the best individual in the new population $\boldsymbol{X}^t \leftarrow \boldsymbol{x}_{\text{best}}^{t-1}$,

In steps **S1** to **S3** a new population is created based on the fitness of the old population while steps **S4** and **S5** alter the new population.

The genetic algorithm may be firstly tuned by defining the population size (default value $P = 10$) and the maximum number of generations (default value $t_{\max} = 30$). Heuristically, 10 individuals per variable are added to the default $P$, and 10 generations per variable are added to the default $t_{\max}$. Secondly, two further parameters should be given to the algorithm: *crossover probability* and *mutation probability*; the values employed in the presented examples were 0.5 and 0.01 respectively. Other parameters may be kept fixed to solve a wide range of problems. The default available algorithms [Gol89, Mic97] chosen for the simple GA are:

Representation: "Binary" $n_{\text{bits}} = 12$,

Penalization type: "Joines-Houck" $C$=1, $\alpha$=1 and $\beta$=2 [JH94].

Scaling: "Standard",

Selector: "Tournament",

Crossover: "OnePoint",

Mutation: "Flip",

Transition: "Elitism".

## 6.5. General solution scheme

As it was shown in Section 6.3.3, a loop over SOCs allows to automatically find the SyDOF's table with the data of the problem and the multiplicity vector $\boldsymbol{m}_{\max}$.

Then, user intervention is required for setting the bounds of the variables, whereas since minimization is desired for the objective function and constraints, their goals are automatically set to zero. Other parameters to be set by the user are: the I/O settings and the GA's parameters. By default, trajectory nodes are chosen as output and joints with imposed movements in prescribed parts as inputs. More than one joint can have imposed movements, therefore the user must only *select the proper input joints*, which must coincide with the degrees-of-freedom desired for the mechanism solutions. Other remaining joints with imposed movements will be taken as output.

The initial sizing procedure is displayed in Algorithm 1. An external loop on solution combination is made, see line 2 and line 33. Then, a distinction between a problem with or without free parameters switches the algorithm to different procedures:

- The first procedure (lines 4-11) consists in a loop for solving SOCs. If an analytical solution exists, a function `SelectOptimum` evaluates the objective function and constraints, and then compares its fitness $\boldsymbol{f}$ against the best solution saved $\boldsymbol{f}_{\text{best}}$. Note that the vector of chosen solutions $\boldsymbol{m}_{\text{best}}$ is also saved.

- A second procedure (lines 13-31) assumes the existence of free parameters, for which a population of individuals is randomly generated using the function `initializeGA`. Then, the genetic algorithm is run iteratively, using the function `evolveGA`, until a maximum number of generations is reached. The function `SelectOptimum` evaluates every individual and saves that with the best fitness $\{\boldsymbol{x}_{\text{best}}, \boldsymbol{f}_{\text{best}}, \boldsymbol{m}_{\text{best}}\}$.

**Algorithm 1** Initial sizing scheme

```
1: bool foundSol=false
2: while nextMCombination(m,m_max,s) do
3:   if x = [∅] then {—Case without free parameters—}
4:     SyDOFsTMP := SyDOFs
5:     bool found = true
6:     for k = 0 to s − 1 do {Loop on SOCs}
7:       found = found and SOC[k].Solve(SyDOFsTMP,m)
8:     end for
9:     if found=true then {Analytical solution exists}
10:      foundSol=SelectOptimum(SyDOFsTMP,f,m,f_best,m_best)
11:    end if
12:  else {—Case with free parameters—}
13:    for t = 0 to t_max do {Loop on generations}
14:      if t=0 then
15:        X^t ← initializeGA(n_vars, n_bits, P, x_min, x_max, n_constr, f_goal, p_cross, p_mut)
16:      else
17:        X^t ← evolveGA(t, X^{t−1}, F(X^{t−1}, m))
18:      end if
19:      for i = 0 to P do {Loop for fitness evaluation of individuals}
20:        SyDOFsTMPi:=SyDOFs
21:        x_i = X^t(i, :)
22:        bool found = true
23:        for k = 0 to s − 1 do
24:          found = found and SOC[k].Solve(SyDOFsTMPi,m,x_i)
25:        end for
26:        if found=true then
27:          foundSol=SelectOptimum(SyDOFsTMPi,x_i,f_i,m, x_best,f_best,m_best)
28:        end if
29:        F^t(i, :) = f_i {This is the evaluation matrix F(X^t, m)}
30:      end for individuals
31:    end for generations
32:  end if
33: end while
34: if foundSol then
35:   Save, run kinematic analysis, export, and show results.
36: else
37:   No solution found.
38: end if
```

After the combinations are exhausted, in line 36 a final evaluation of the best solution is made, and then performance indexes, i.e. the values of the objective function and each restriction, are shown.

## 6.6. Path following example

In order to show the methodology, a very simple path following example with prescribed timing is used. The description of the problem was shown in Figure 3.1-left.

The synthesis task is defined as a set of displacements on node $N_1$

$$\mathcal{D} = \{N_1, 0, (0, 0);\ N_1, 1, (0.2, 0.2);\ N_1, 2, (0.18, 0.4)\},$$

plus a timing given as a set of rotations (in $[rad]$) imposed on the joint element $E_5$

$$\mathcal{J} = \{E_5, 0, 0;\ E_5, 1, 0.44;\ E_5, 2, 0.8\},$$

whereas the coordinates of nodes (in $[m]$) are $N_6 = (0.0, 0.0)$, $N_1 = (0.4, 0.5)$ and $N_2 = (1.2, 1.6)$.

The execution of the type synthesis solver for this example was used through the Chapters 2 and 3. It was also used in Chapter 5 to explain the decomposition method. Here, it is used to show the initial dimensioning without and with free parameters arising in **Alternatives 0** and **1** respectively. Since this problem has not territorial restriction only two restrictions are considered –minimal link lengths and non inversion of transmission angles.

### 6.6.1. Solution for Alternative 0

The type synthesized **Alternative 0** is a four-bar mechanism which is solved analytically without free parameters. Its SyDOFs table and multiplicity vector were used to exemplify the previous sections, see Tables 6.1 and 6.2, and equation (6.2).

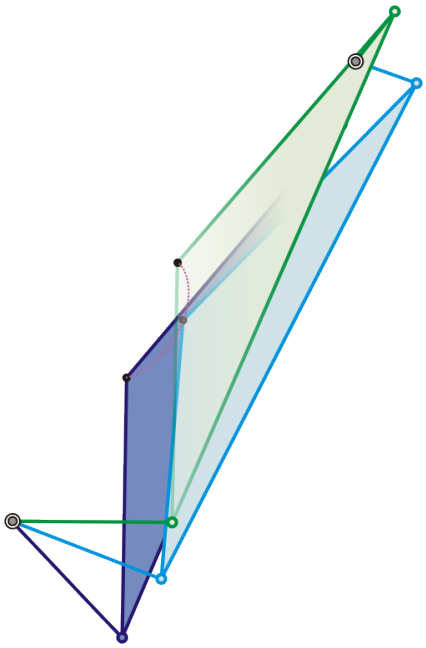


Figure 6.5: Path following problem: initial sizing solution for **Alternative 0**. The generated curve is shown.

The best result is depicted in Figure 6.5. For clarity of the drawing, the crank is not shown.

### 6.6.2. Solution for Alternative 1

The following alternative has a Watt-II topology with a ternary ground and consequently it has a new pivot. Its decomposition results in the three SOCs shown in Figures 5.7-1 and 5.8. Six free parameters can be identified from the corresponding SyDOFs table, which is rather long to be shown here. Four free parameters must be set for two links of the triad SOC 0. Two additional free parameters are needed to define a box for the new pivot position. This pivot is computed by the last SOC module. Table 6.3 presents the description of the free parameters and default bounds for them.

| | SyDOF | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Free param. | Type | ID | Phys | Prec.Pos. | SOC | min | max | best |
| 0 | Elem | 7 | L | 1 | 0 | -1 | 1 | -0.54872 |
| 1 | Elem | 7 | L | 2 | 0 | -2 | 2 | -0.90696 |
| 2 | Elem | 9 | L | 1 | 0 | -1 | 1 | 0.052503 |
| 3 | Elem | 9 | L | 2 | 0 | -2 | 2 | -0.36484 |
| 4 | Node | 7 | x | 0 | 2 | 0 | 1.2 | 0.20923 |
| 5 | Node | 7 | y | 0 | 2 | 0 | 1.6 | 0.72283 |

Table 6.3: Free parameters for **Alternative 1** of a path following problem.

For this problem, the default bounds shown in Table 6.3 were used for computation (rotations are in $[rad]$). For the running, the optimization parameters were:

**Constraints**: minimal link length parameter $L_{\min} = 0.20$, transmission angle tolerance $\delta = 10°$;

**GA solver**: population size $P = 70$, maximum number of generations $t_{\max} = 90$, crossover probability $p_{\text{cross}} = 0.5$, mutation probability $p_{\text{mut}} = 0.01$, number of bits $n_{\text{bits}} = 12$, penalty $\lambda_{\max} = 1000$.

The results show a complete fulfilment of constraints (both, *minimal link lengths* and *non-inversion of transmission angles*, are null; there is not *allowed space* defined), and a value of 0.058002 reached by the objective function. The best solution is shown in Figures 6.6 and 6.7.

The evolution of the GA for each combination is shown in Figures 6.8 to 6.11. Then, Figure 6.12 compares the evolutions of the best individuals taking into account restrictions. The minimum is retained from the last combination used. Note that the initial mean fitness value is approximately $3\lambda_{\max} = 3000$, where the number 3 comes from the summation of one objective plus two constraints. It is the expected value when the open-chain synthesis has no solution, and hence, it does not exist any possibility of evaluation. This situation in which a population is completely unfeasible, is a common pattern in this kind of problems.

## 6.7. Discussion

The presented sizing method has some advantages compared to other methods found in the literature and those found in software capabilities:

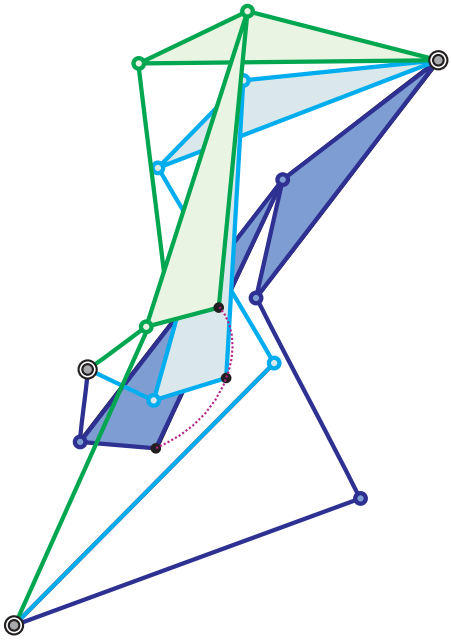


Figure 6.6: Path following problem: initial sizing solution for **Alternative 1**. The generated curve is shown.

- ✓ The method is compatible with an automated SOC decomposition algorithm followed by an automatic detection of the variables of the problem.
- ✓ User intervention is minimized to the actions of (i) selecting the input parameters, (ii) defining the bounds of the variables and (iii) setting the parameters of the GA. The first action is straightforward, while for the second a user without experience in defining bounds may run the solver with the default values and a moderate number of evaluations ($P \times t_{\max} \approx 5000$). For problems without free parameters, actions (ii) and (iii) are not required.
- ✓ A very useful *allowed space constraint* enables the designer to find compact mechanisms. Space requirements are mandatory in almost all real-life applications. This automatic feature is not addressed in specialized literature.
- ✓ Human work for exploring combinations in the search of the best solution is eliminated since combinations of solutions were incorporated in the sizing algorithm.
- ✓ The methodology is integrated to a CAD/CAE system of analysis of mechanisms and structures by finite elements.
- ✓ No initial guess is required.
- ✓ Geometrical advantage can be specified by means of the precision positions.
- ✓ Object-Oriented Programming enables to extend the SOC modules for taking into account more joint and link types as well as to add their own design constraints in the selection of the best qualified mechanism.

Among the disadvantages compared to other methods:

- ↓ If the problem has no solution, no easy recipe may guide the user towards the modification of the search parameters and bounds for missing parameters. On

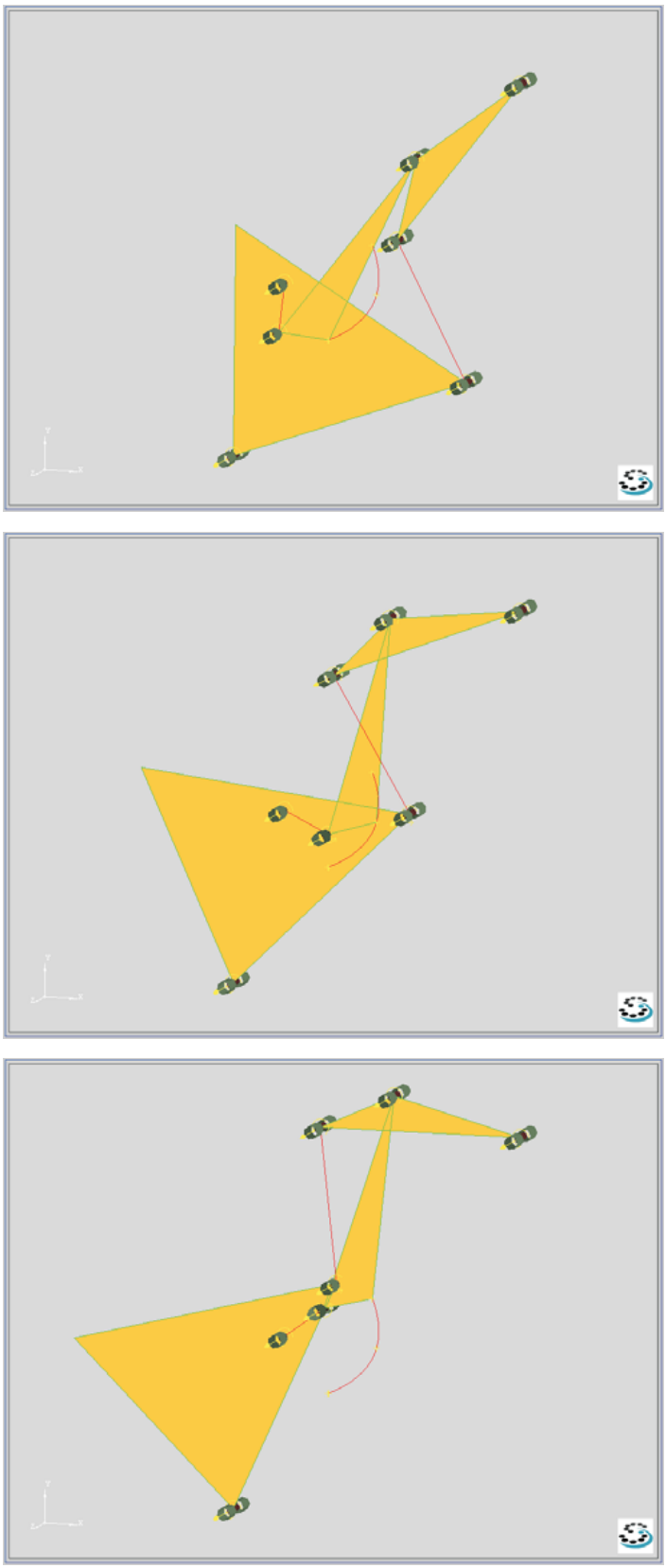


Figure 6.7: Path following problem: aspect of **Alternative 1** exported to SAMCEF software. The desired curve is displayed.

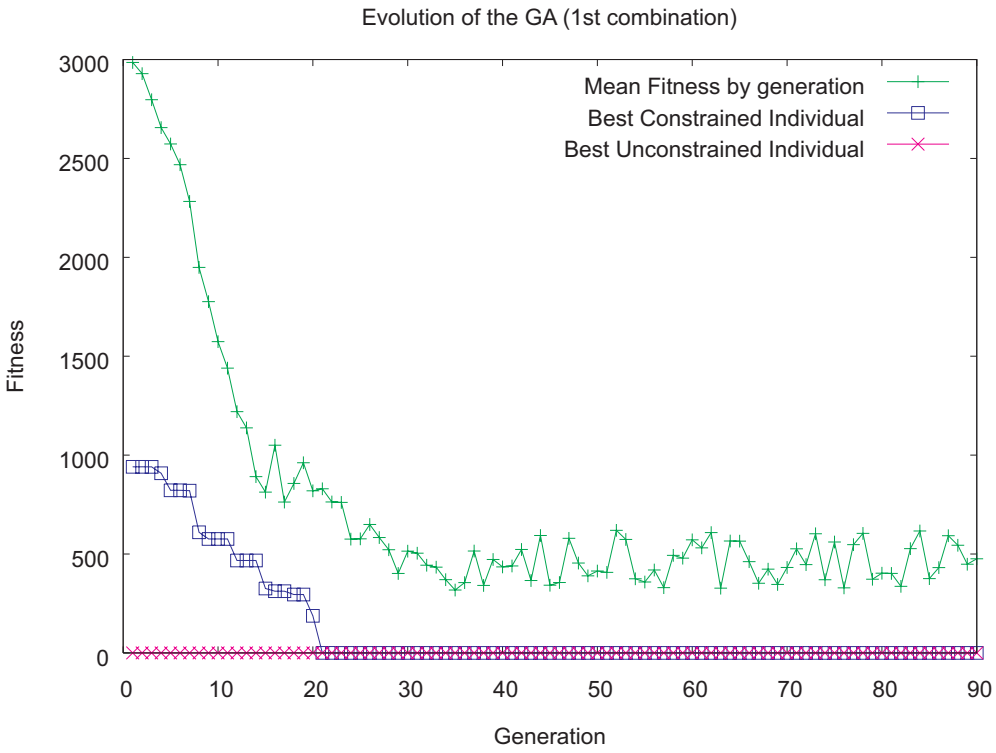



Figure 6.8: Evolution of GA for the initial sizing of **Alternative 1**, first combination.

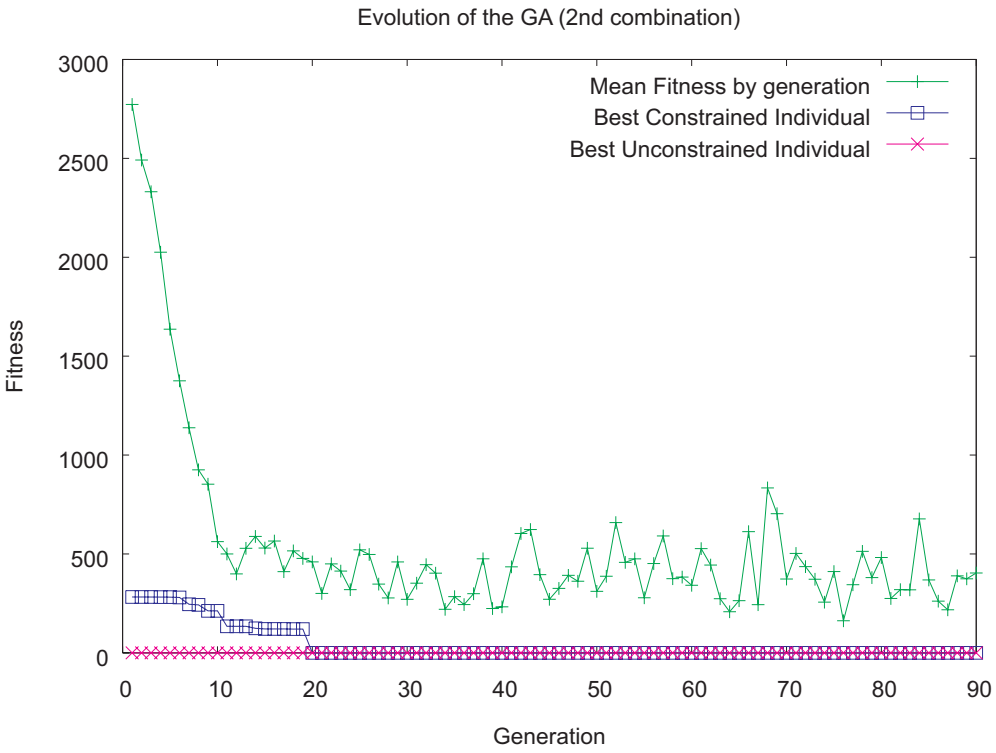



Figure 6.9: Evolution of GA for the initial sizing of **Alternative 1**, second combination.

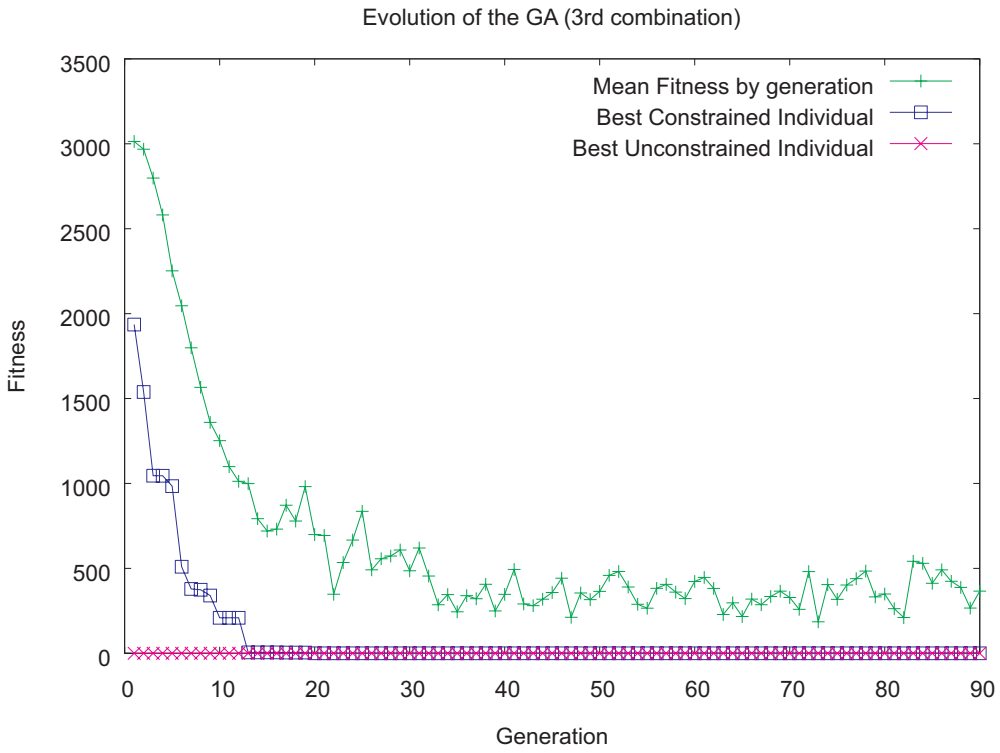



Figure 6.10: Evolution of GA for the initial sizing of **Alternative 1**, third combination.

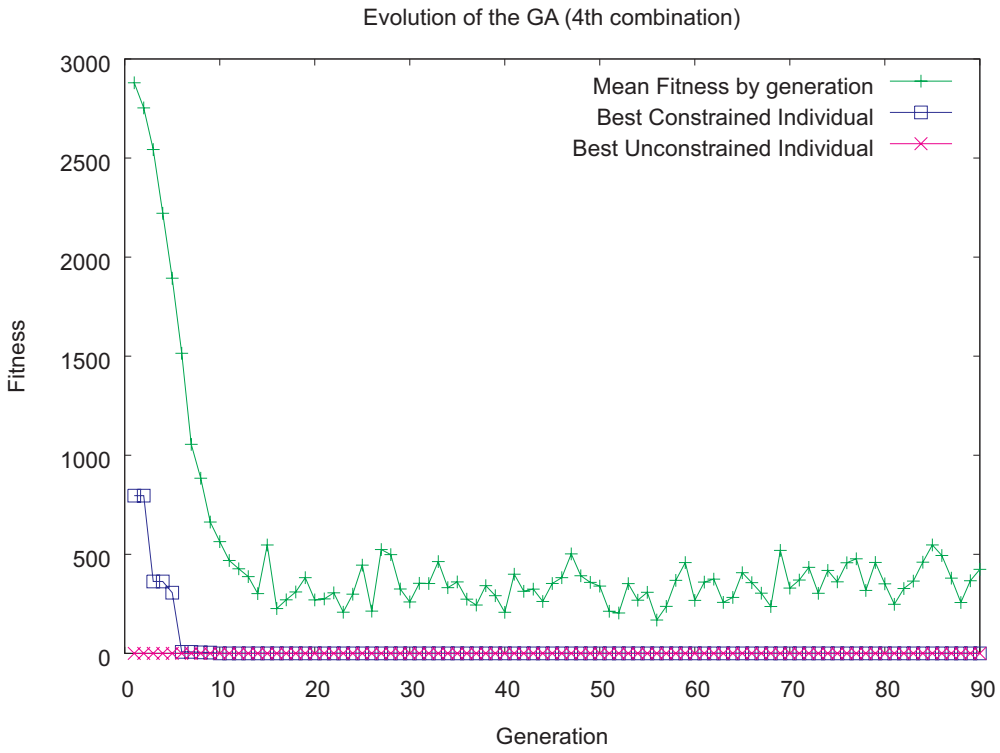



Figure 6.11: Evolution of GA for the initial sizing of **Alternative 1**, fourth combination.

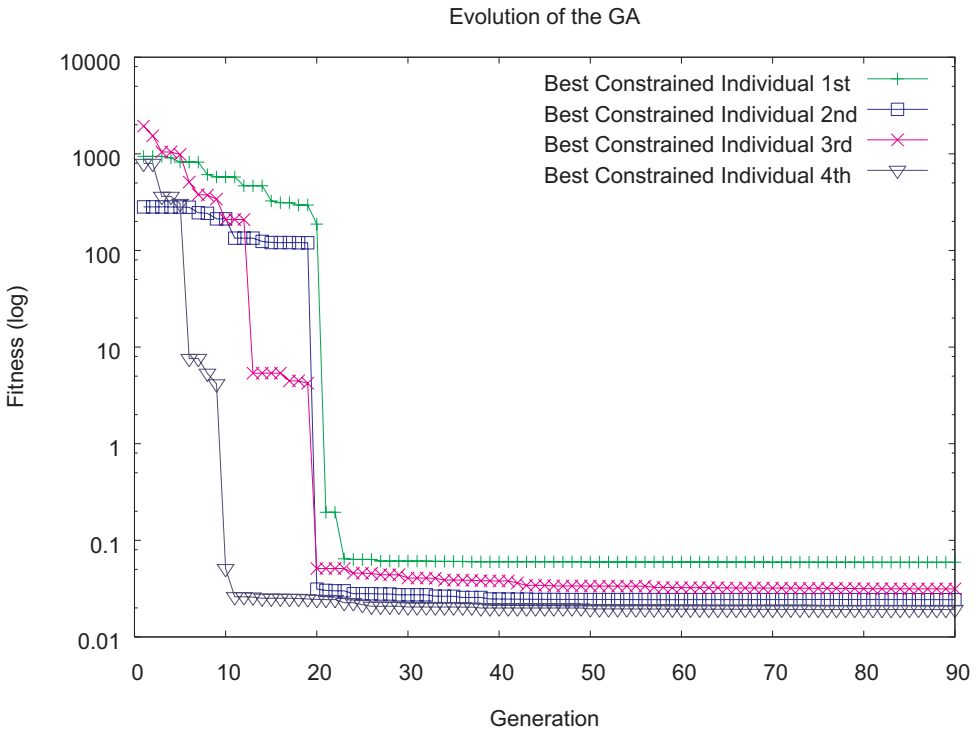



Figure 6.12: Comparison between the evolutions for the best constrained individuals for each combination of solutions (four sub-problems).

the contrary, if solution exists, the user must read fitness of the best result and observe if there are active constraints[1] to take decisions. In this aspect, the *solution map* available in LINCAGES 2000 is more advantageous, but limited to predefined four- and six-bar topologies.

↓ Since the solver runs in "batch-mode" no control is offered to the user while the running takes place. However, the running takes few seconds in a modern PC.

↓ The number of sub-problems increases with the number of combination of solutions.

↓ Exact synthesis, constrained as it was presented here, does not assure for all cases the non existence of branch and circuit defect. Order defect may be rectified by incorporating one kinematic analysis per individual and an equality constraint of desired and generated precision positions.

↓ There is no control over (i) intermediate positions of the task, (ii) optimum transmission angle, and (iii) mechanical advantage. Although the latter does not belong to the field of kinematics it is added to complete the comparison.

[1] A constraint is said to be "active" if it acquires a final value of the variables equal to its bound, e.g. the constraint $q(\boldsymbol{x}) \leqq 0$ has a final value $q(\boldsymbol{x}) = 0$

## 6.8. Chapter conclusions

In this chapter a computational method for the initial sizing of linkage topologies was presented. The strategy exploits the use of analytical equations for exact synthesis that have fast execution. In this sense, the usage of the ordered SOC modules, the auxiliary data structures and the algorithms used in the solution scheme were described.

The developed program allows us to easily configure and explore the feasible solution space, looking for that having the best satisfaction of an optimality criterion. At the same time, it verifies that the proposed solutions do not violate restrictions (space, singularity of the movement, minimum dimensions, etc).

Although solutions may be multiple, only the best of them is saved. The solution obtained, if it exists, may be later evaluated for other requirements, e.g. the continuum task, and also may serve as an initial guess for later optimization using Gradient Techniques.

# Chapter 7

# Results

In order to validate the methodology, different studies have been carried out with several tests, some of them coming from aeronautical applications.

A common pattern will be used to present the examples. Firstly, a brief description of the problem and its real-life application will be introduced. Secondly, the settings chosen for the type synthesis execution and the interpretation of results will be discussed. Thirdly, the settings chosen for the dimensional synthesis execution will be dealt with . Finally, some comments about the best solution found are made.

The presentation is divided into two groups of applications –single and multiple– depending upon the purpose of the kinematic task.

## 7.1. Applications for single tasks

Any complex kinematic problem can be decomposed into a succession of traditional kinematic tasks –function generation, path following and rigid-body guidance.

However, due to the generality of the proposed method, no classification action to indicate the kind of task is required from the user. In this section, the solution procedure for kinematic descriptions that correspond to single tasks of FG and RBG is explained. The PF task was deeply explained in previous chapters.

### 7.1.1. Function generation

A function generation task similar to that shown in Figure 3.3 is applied here to synthesize a *landing-gear* mechanism.

The geometry of the prescribed parts to move is illustrated in Figure 7.1. Body $E_9$ represents a *leg* and the upper body $E_{12}$ is a rotative actuator for retracting/lowering the leg. Both bodies are grounded by revolute joints $E_{13}$ and $E_{14}$ to the aircraft chassis. Their nodes have coordinates in $[m]$ $(0.0, 0.0)$ and (0.324,0.05775), respectively. The end-point node of the leg has coordinates $N_3 = (0.757371, 0.0)$, and the end-point node of the actuator has coordinates $N_6 = (0.323981, 0.107749)$.

Motion constraints are defined on joint-elements like $\mathcal{J} = \{E_{14}, 0, 0;\ E_{13}, 0, 0;\ E_{13}, 1, -\pi/6;\ E_{13}, 2, -\pi/3;\ E_{13}, 3, -\pi/2\}$. Since angular positions for joint $E_{13}$ are not defined, the motorization is unknown. An *allowed space* is added by defining a polygon given by nodes $N_7 = (-0.01, 0.057749)$, $N_8 = (-0.01, -0.28615)$, $N_9 = (0.323981, -0.03615)$, and $N_{10} = (0.323981, 0.057749)$.

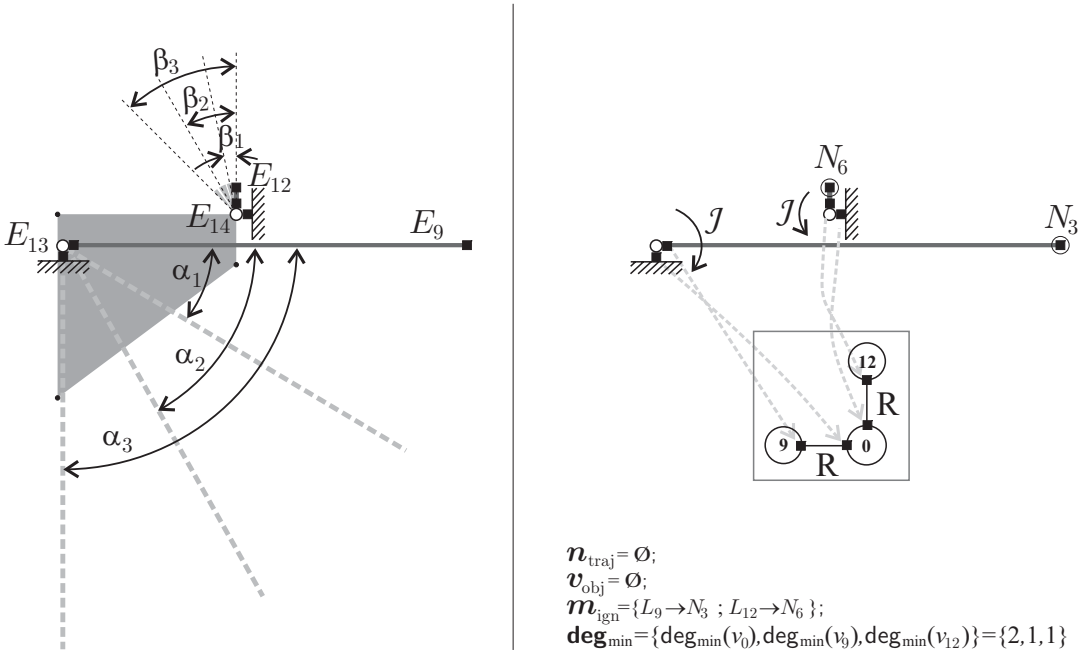



Figure 7.1: Description of a landing gear retraction task.

The data for problem definition are interactively entered by CAD, see Figure 7.2.

Under the solver Oofelie–Mechanism Synthesis, the program Samcef Field allows the user to define the prescribed parts, allowed space and motion constraints. Additional information is entered by the *Solver Epilog script*.

### Type Synthesis

In Figure 7.1-right, the initial graph associated with the problem is shown. It is composed of two vertices (body $E_{13}$ and body $E_{16}$) connected to vertex 0 by two revolute joints which, for reasons of clarity, are not drawn. In the same figure, additional information is shown below the initial graph. Note that the trajectory node and objective vertex vectors are empty.

The Type Synthesis solver is launched with the settings shown in the Solver Epilog script of Figure 7.2: (i) the solution space is defined by selecting the atlas RigidOneDofR, (ii) default filters for link and joint types are used, (iii) ten alternatives are required.

Successfully, the sub-graph search algorithm found the ten alternatives shown in Figure 7.3. Since the default for search settings includes *Avoidance of pseudo-isomorphisms*, no topology is included as sub-graph of a previous one[1].

After SOCs decomposition, a number of four feasible alternatives compatible with dyads and triads passing through four positions is obtained. They are the **Alternatives 0** to **3**. More feasible alternatives would result if quadriads were incorporated, which is still a topic under research.

Note also that from the set of topologies feasible to be solved by the PPM method, **Alternatives 1** to **3** incorporate an additional pivot because the ground

[1]However, a common requirement for designing landing gear mechanisms is the existence of an idle chain for locking it at the lowered position.

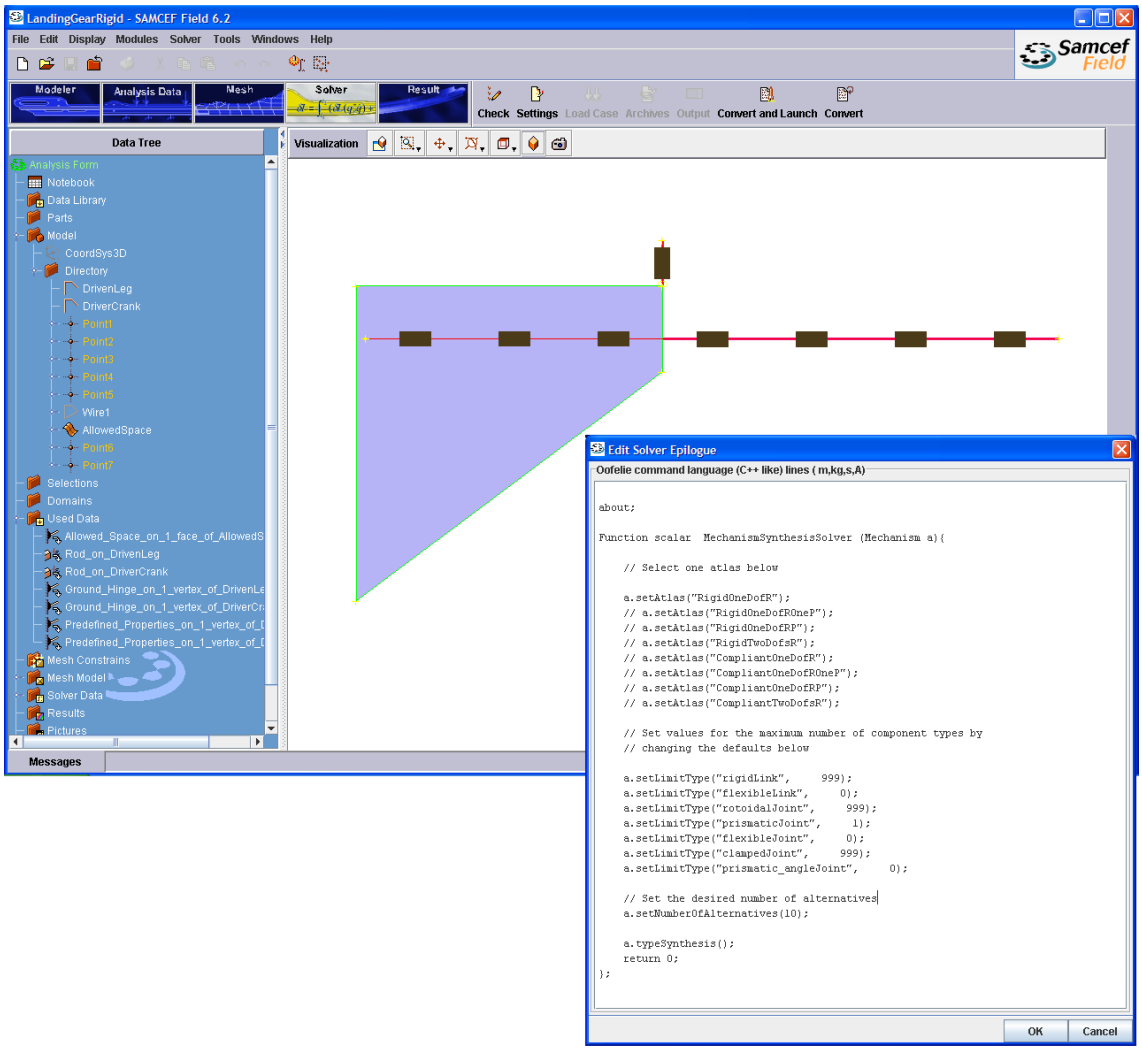



Figure 7.2: Aspect of the CAD environment for entering the kinematic problem.

is ternary.

The first initial graph occurrence was found in a one-loop topology like the one shown in Fig. 7.4-a. A unique single open chain is identified; a JLJLJL-Triad module is highlighted in black in Figure 7.4-c. From the Loop-Closure Equation solvers, it is determined that there are two missing parameters in one link and a free parameter in the middle link.

After the decomposition, the program asks the user for bounds of these free parameters, see Figure 7.5-left.

So, the actuator can be biased to rotate counter-clockwise by means of the defined bounds, see Table 7.1.

### Dimensional Synthesis

The default penalty values are used for launching the initial sizing solver. The presence of an allowed space suggests an increase of the necessary evaluations. So, the main GA parameters were set as $P = 70$ and $t_{\max} = 150$.

Four precision positions are shown in Figure 7.4-d where the positions of the nodes of joints 16 and 17 were computed. The final values for the variables are

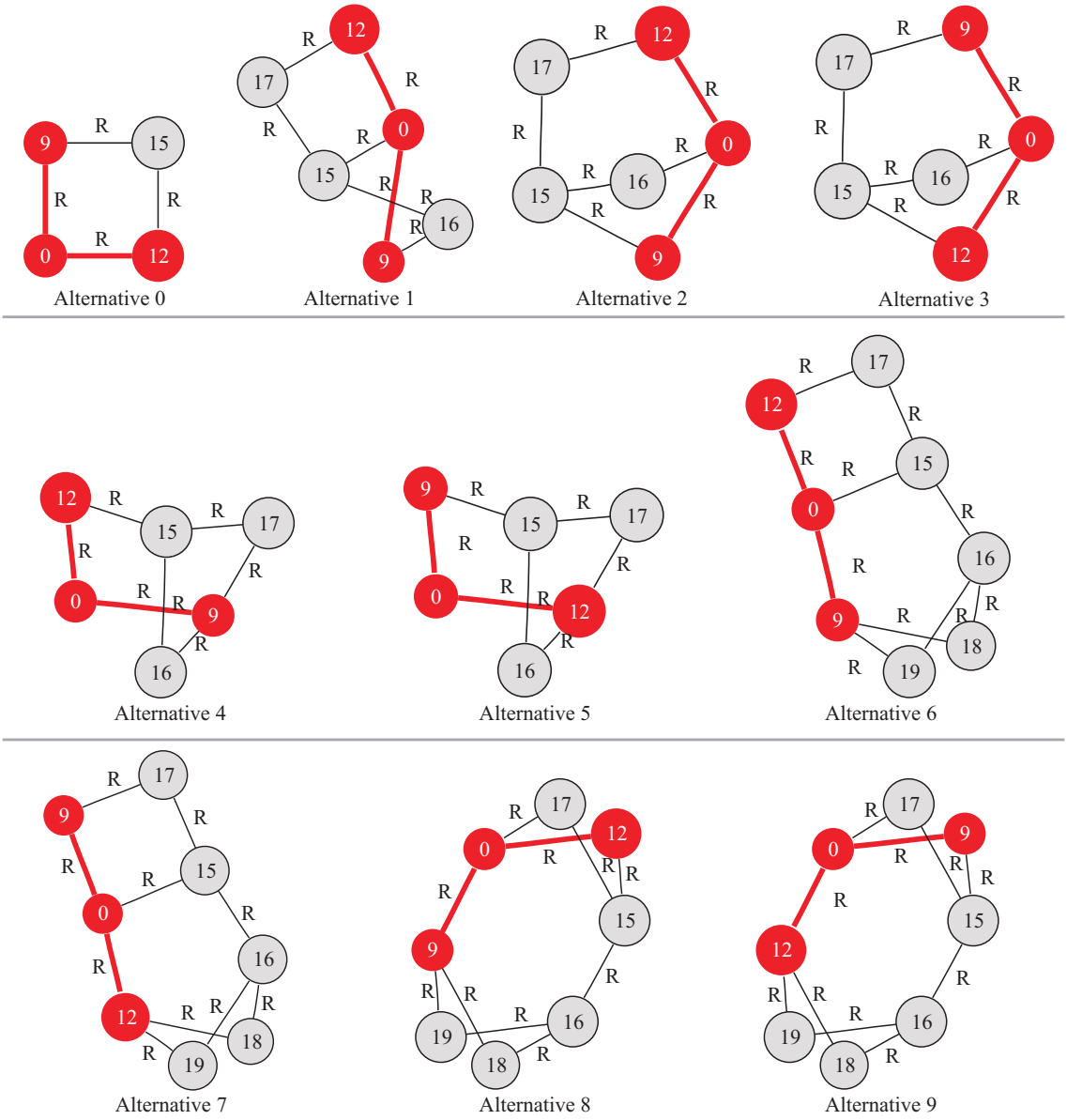



Figure 7.3: Topologies found by the constrained subgraph search algorithm.

| | SyDOF | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Free param. | Type | ID | Phys | Prec.Pos. | SOC | min | max | best |
| 0 | Elem | 14 | J | 1 | 0 | 0.15 | 0.22 | 0.212 |
| 1 | Elem | 14 | J | 2 | 0 | 0.3 | 0.35 | 0.33525 |
| 2 | Elem | 14 | J | 3 | 0 | 0.65 | 0.7 | 0.65088 |
| 3 | Elem | 15 | L | 3 | 0 | -2 | -1.8 | -1.9998 |

Table 7.1: Resultant free parameters for **Alternative 0** of a function generation problem.

shown in the last column of Table 7.1. As it is seen in this Figure, assembling results in a folding-stay retraction mechanism that has a great flexibility for installation in a constricted space.

After the exportation, the mechanism looks like the one shown in Figure 7.6. A

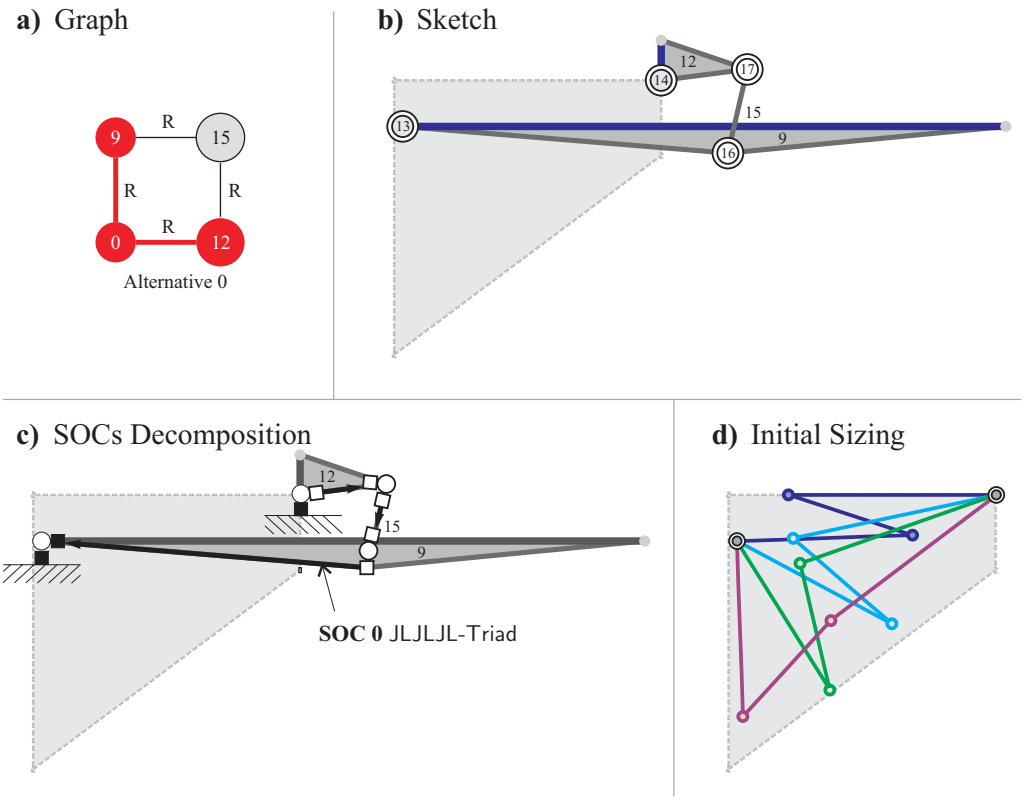



Figure 7.4: Solution stages for the first alternative.

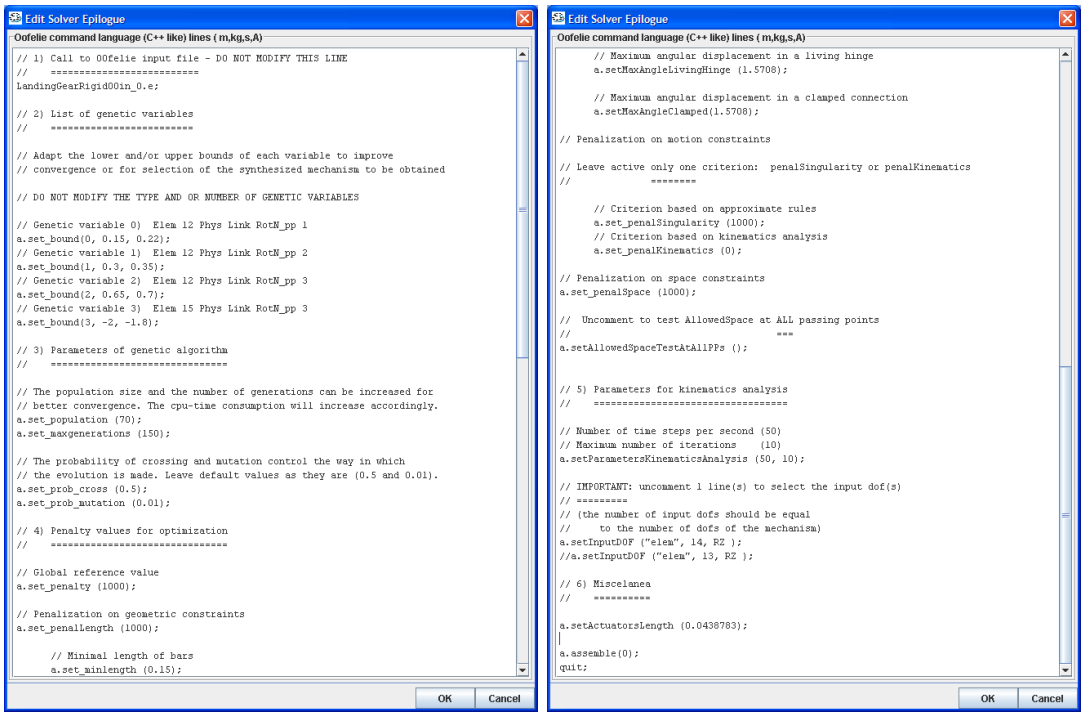



Figure 7.5: Epilog for initial sizing settings.

kinematic analysis of the final mechanism allows the user to check the behavior at the intermediate positions.

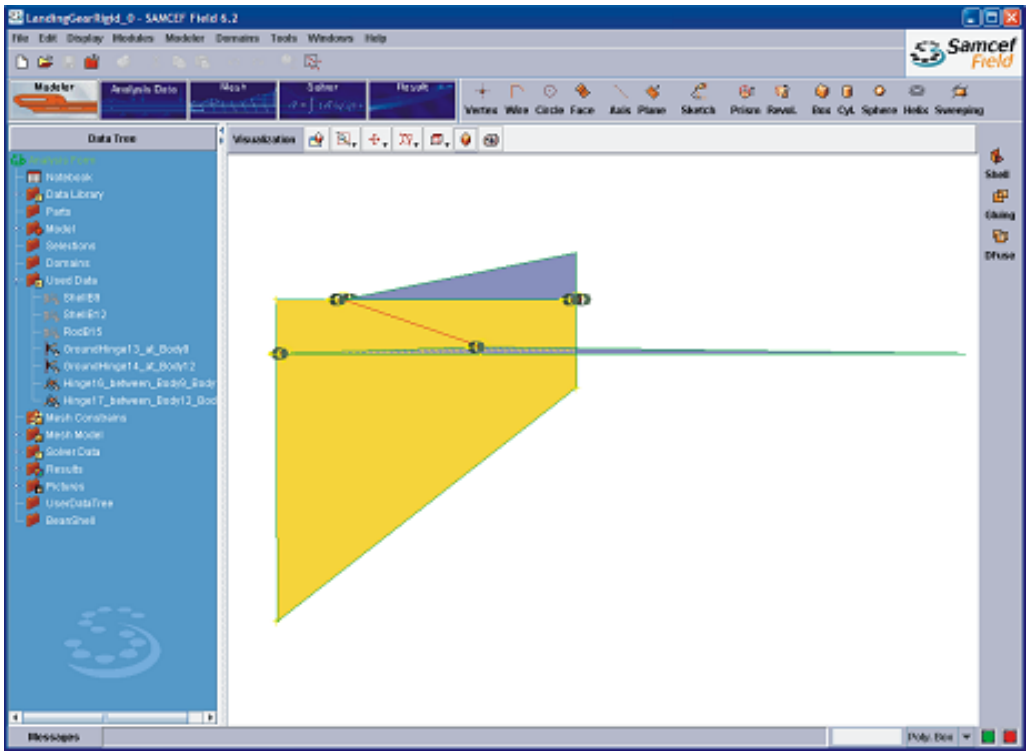


Figure 7.6: First solution for landing gear retraction

By incorporating the detailed designs of the leg, the CAD description shown in Figures 7.7 and 7.8 is obtained.

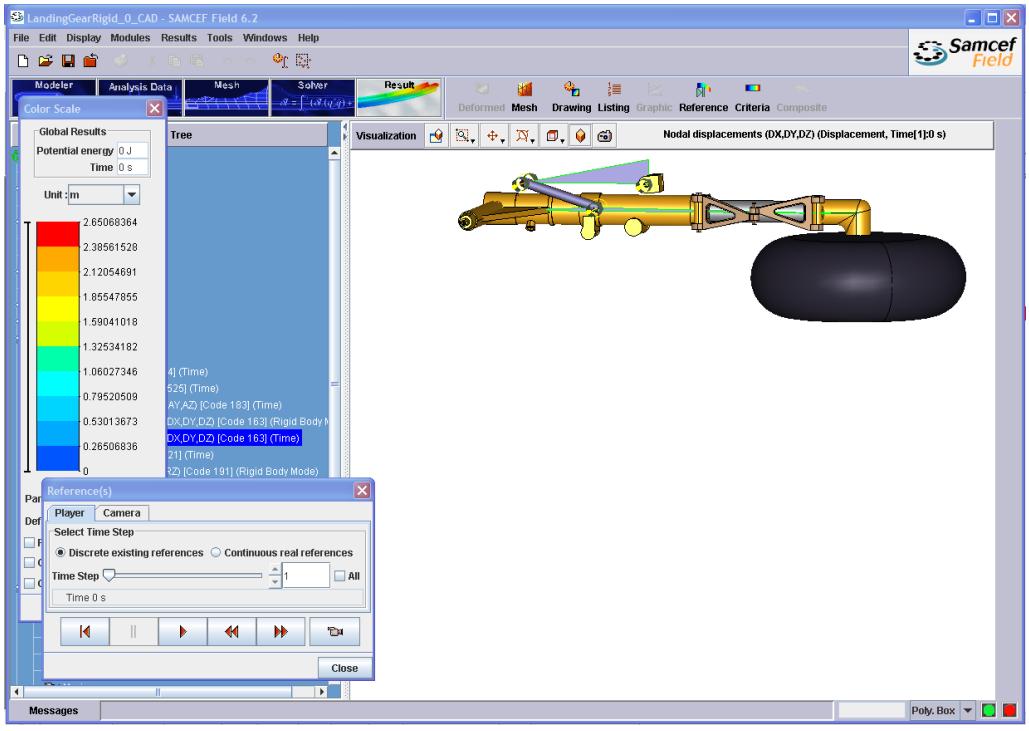



Figure 7.7: Solution with detailed geometry of parts incorporated.

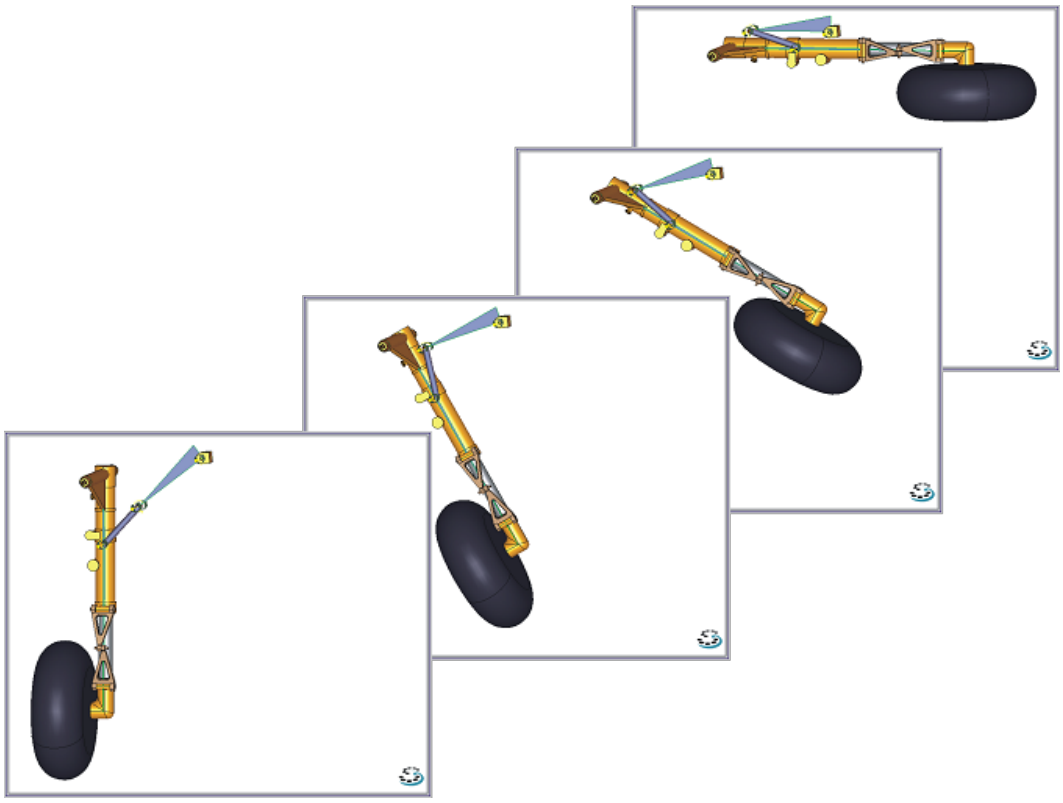


Figure 7.8: Snapshots of animation in four precision positions ($j = 0, 1, 2, 3$).

### 7.1.2. Rigid-body guidance

An industrial test corresponding to a task like that showed in Figure 3.2 was solved. The required movement is the guidance of a *slat* for an airplane wing. Functionally, its motion changes the attack wing angle to drive the airplane.

The task is defined on node $N_5 = (11.84, 2.40)$ that represents the slat, displacements are

$$\mathcal{D} = \{N_5, 0, (0, 0); N_5, 1, (-0.14, -0.04); N_5, 2, (-0.22, -0.08)\},$$

and rotations for rigid-body $E_1$ are

$$\mathcal{L} = \{E_1, 0, 0; E_1, 1, 30°; E_1, 2, 45°\}.$$

Pivot node positions are $N_6 = (11.70, 2.14)$ and $N_7 = (11.74, 2.28)$. The aspect of the CAD environment of the entered data is shown in Figure 7.9.

#### Type Synthesis

The type synthesis solver is executed using the default settings. In Figure 3.2-left we can see the initial graph construction. It is composed of two disconnected vertices: 1 and 0. Vertex 1 is composed of node $N_5$ of body $E1$, and vertex 0 of nodes $N_6$ and $N_7$. Although it is not showed in that figure, additional information can be identified: the map $\boldsymbol{m}_{\text{ign}} = \{L_1 \rightarrow N_8\}$, the ID of the node which will develop the required trajectory $\boldsymbol{n}_{\text{traj}} = [5]$, and the objective vertex $\boldsymbol{v}_{\text{obj}} = [1]$.

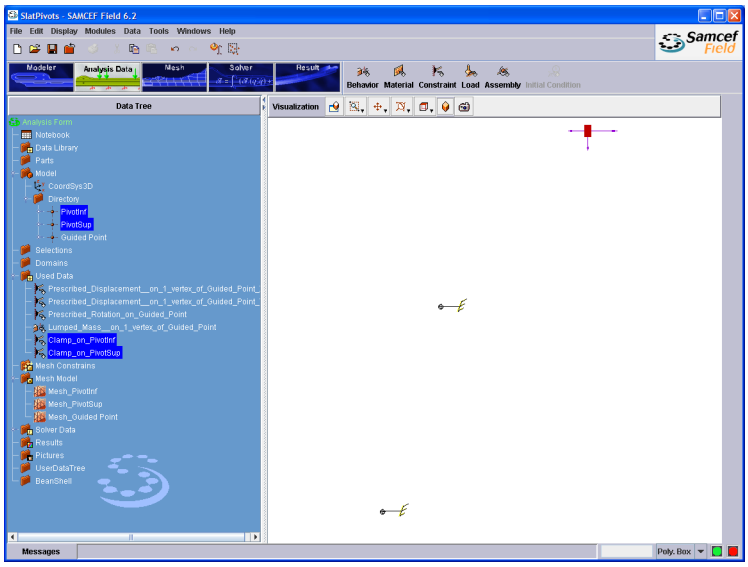



Figure 7.9: Input of data for a rigid-body guidance kinematic task.

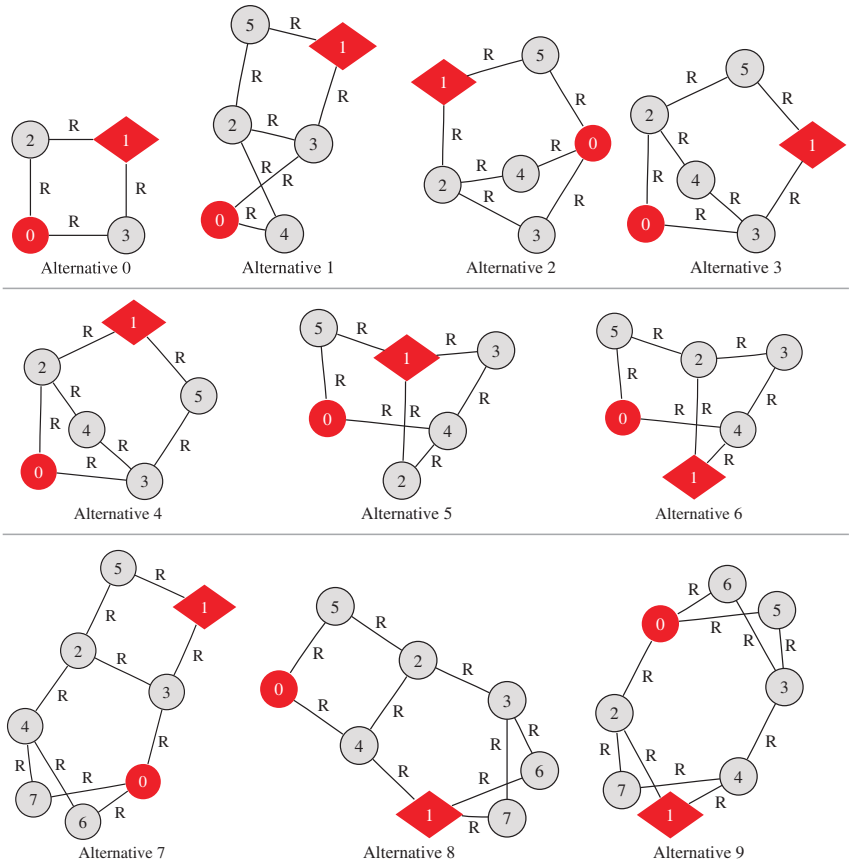



Figure 7.10: Sub-graph occurrences for a rigid-body guidance problem.

Then, the list of the first ten alternatives found is shown in Figure 7.10.

Among these alternatives, **Alternatives 1**, **7** and **8** produce an unsuccessful decomposition because of the identification of quadriads.

The simplest graph found is a four-bar topology, see **Alternative 0** in Fig-

ure 7.11. This graph is decomposed into two JLJL-Dyad modules, see Figures 7.11-b and c.

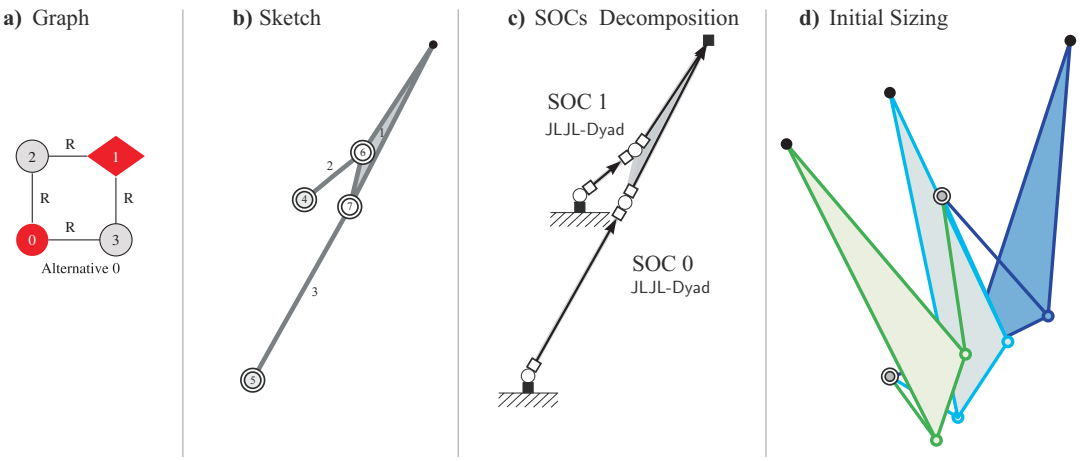



Figure 7.11: Stages for solving the first alternative of the rigid-body guidance problem.

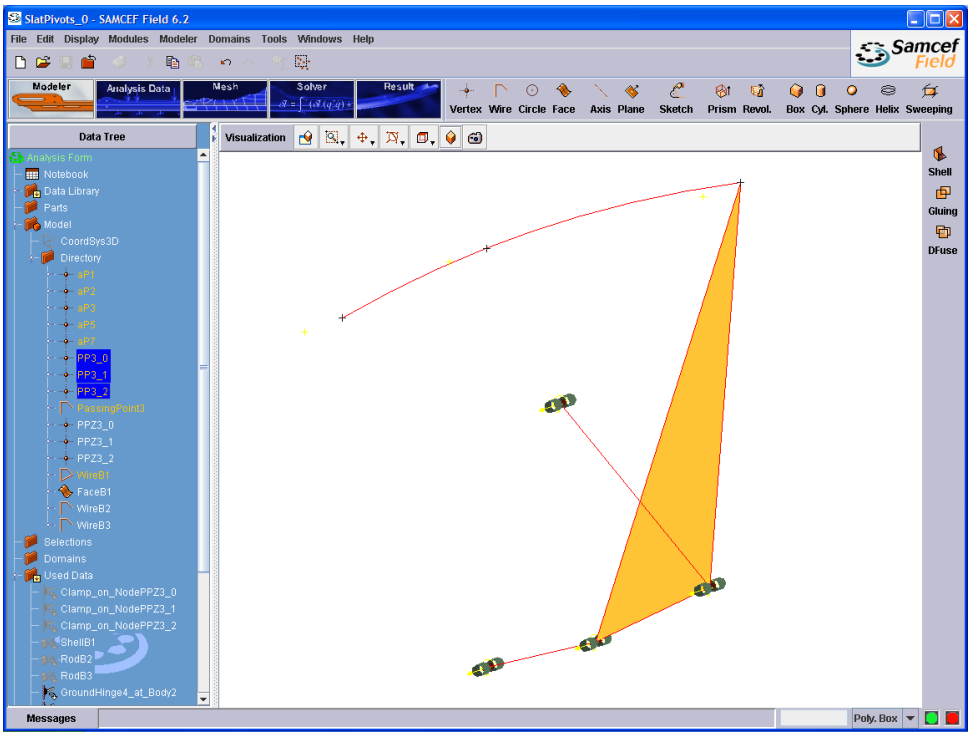



Figure 7.12: Sized **Alternative 0**.

### Dimensional Synthesis

Following with the same alternative, the **SOC 0** is firstly computed. Its links do not impose new constraints over **SOC 1** because the second links in both chains must

rotate the same angles as already prescribed. No free parameter, and consequently, no GA setting is requested from the user. The user must only indicate which joint is the input. This joint can be chosen by reading, from the associated sketch, the IDs of the new joints (Figure 7.11-b); for example, $J_5$ is selected as input.

There are four combinations of solutions from which the best solution is retained, see Figure 7.11-d and the assembled solution exported to CAD environment in Figures 7.12 and 7.13-a. It results in a four-bar motion generator for which the computed motorization is

$$\mathcal{J} = \{J_5, 0, 0.0; J_5, 1, -0.80479; J_5, 2, -1.20423\}.$$

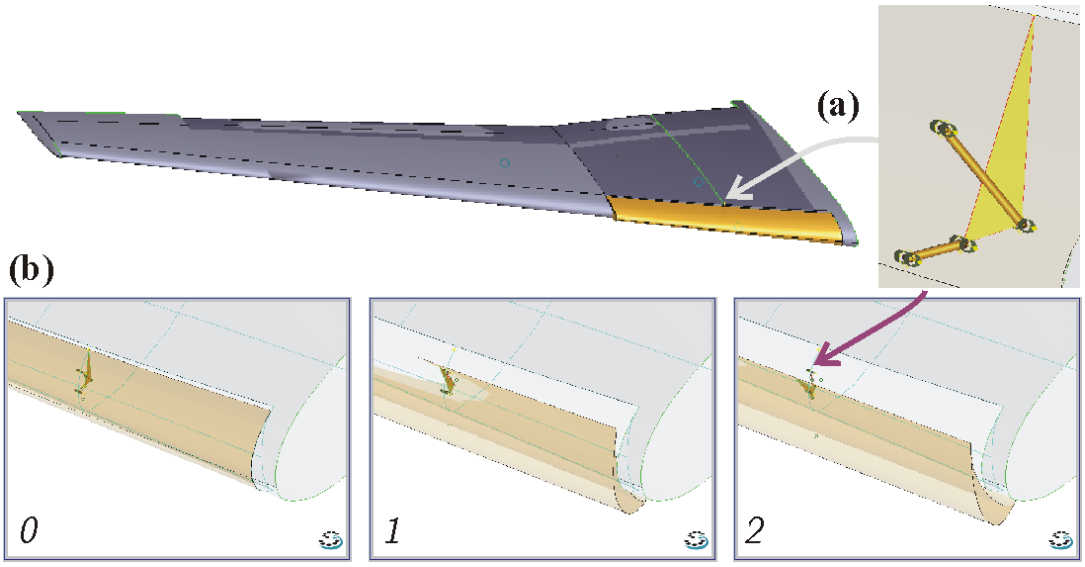



Figure 7.13: Work plane location in the wing (a), and snapshots of three precision positions with rigid slat attached to mechanism (b).

### 7.1.3. Rigid-body guidance with prescribed geometric advantage

The proposed test problem consists in finding a mechanism able to produce the deflection of two curved beams. The boundary conditions are shown in Figure 7.14. The topology can be considered as a compliant mechanism itself. If this mechanism is actuated by the prismatic joint, its full kinematic description at any point of the beams can be obtained.

#### Deflection of compliant members by means of rigid mechanisms

A different way to actuate the beams shown in Figure 7.14 is the topic of research. It is desired to connect an internal one-DOF mechanism to move the beams producing a proper movement of some of its points, considering the prismatic joint as passive. Looking for the simplest design, the tip node ($P_{\text{tip}}$ in Figure 7.15) is taken as a guided point passing through three positions. Then the kinematic description –displacements and rotations– of the tip node is used to state a rigid-body guidance

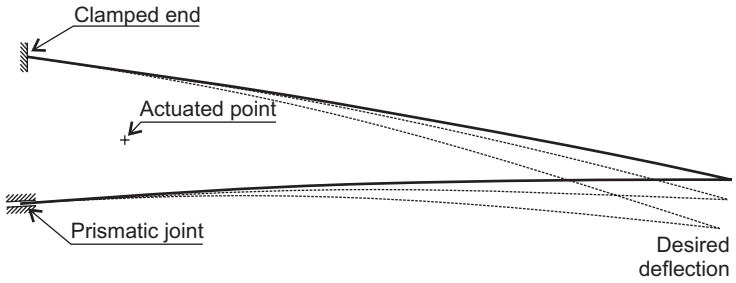



Figure 7.14: Boundary conditions for the deflection of two beams.

kinematic task. The geometric advantage is also prescribed: the input displacement (piezo-electrical motor) is prescribed to be 1/100 of the tip vertical deflection.

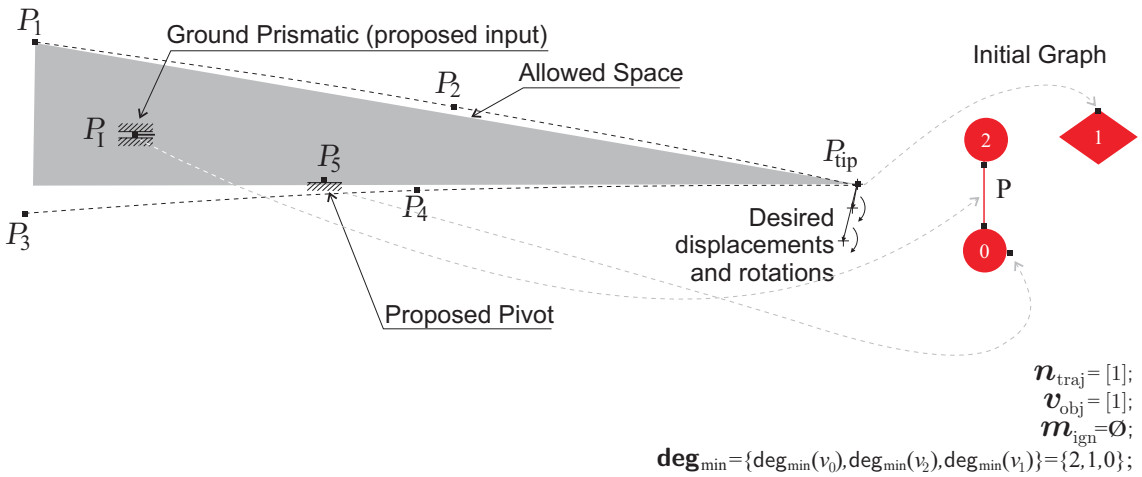



Figure 7.15: Problem description.

Note that the territorial constraint imposed by the beams makes the problem a difficult one. The final design must be completely included inside the interior *allowed space* defined by the beams. A convex polygonal area is used to compute the fulfillment of this constraint.

The data for this problem are (coordinates in $[m]$):

- Tip coordinates: $P_{\text{tip}} = (1.0, 0.0)$
- Tip displacements:
  - $\delta_1^{\text{tip}} = (-0.002089, -0.010085)$, and $\delta_2^{\text{tip}} = (-0.005632, -0.024297)$.
- Tip rotations in $[rad]$:
  - $\alpha_1^{\text{tip}} = -0.0435582$, and $\alpha_2^{\text{tip}} = -0.104336$.
- Beams are defined by interpolating cubic-splines passing through the following points:
  - Upper beam: $P_1 = (0.65085, 0.06106)$, $P_2 = (0.85148, 0.03018)$, $P_{\text{tip}}$.
  - Lower beam: $P_3 = (0.64915, -0.01086)$, $P_4 = (0.79849, -0.00229)$, $P_{\text{tip}}$.

- Prismatic actuator (input): $P_I = (0.7, 0.02)$.
  - ◇ Maximum displacement: $\delta_2^I = (0.0, 0.024297/100)$.
- Boundary conditions:
  - ◇ Clamped constraint on proposed pivot: $P_5 = (0.78, 0.0)$.
  - ◇ Clamped constraint on node $P_1$ of the upper beam.
  - ◇ Grounded prismatic joint constraint on node $P_3$, axis direction $\hat{n} = (0.99835, 0.05729, 0)$.

#### Type synthesis

When the type synthesis solver is executed, the *initial graph* is automatically constructed, see Figure 7.15. Nodes constrain the degree of the graph vertices for the subgraph search. The trajectory node is structurally ignored, but after type synthesis, it is assigned to a new body as it is shown in the sketches.

On the left of Figure 7.16, we can see the graphs obtained by the subgraph search.

#### Initial sizing

A multi-loop solution corresponding to **Alternative 4** is shown in Fig. 7.17 where the beams were attached to the tip.

## 7.2. Applications for multiple tasks

### 7.2.1. A mixed path following and rigid-body guidance

This problem also corresponds to the explanations given in the previous section. In this example, we desire to guide not only the tip but also another point; for example, we can choose the intermediate point, $P_H$, of the lower beam (Figure 7.18). The point is guided by a set of positions, without prescription of orientations. This means that we must add a hinged connection between the mechanism and the lower beam. Conveniently, this hinge is added after solving the trajectory of one of its nodes. Therefore, we reformulate the problem as a multiple-task problem: a path following task is required for node $P_{\mathrm{H}}$ and, simultaneously, a rigid-body guidance for the tip $P_{\mathrm{tip}}$.

Both movements must be coordinated with the rotation of a motorized crank. The mechanism solution must fit inside an allowed space.

Data for this problem are:

- Tip coordinates: $P_{\mathrm{tip}} = (1.0, 0.0)$
- Tip displacements:
  - ◇ $\delta_1^{\mathrm{tip}} = (-0.002089, -0.010085)$, and $\delta_2^{\mathrm{tip}} = (-0.005632, -0.024297)$.
- Tip rotations:
  - ◇ $\alpha_1^{\mathrm{tip}} = -2.4957°$, and $\alpha_2^{\mathrm{tip}} = -5.978°$.
- Guided hinge coordinates $P_{\mathrm{H}} = (0.79849, -0.00229)$

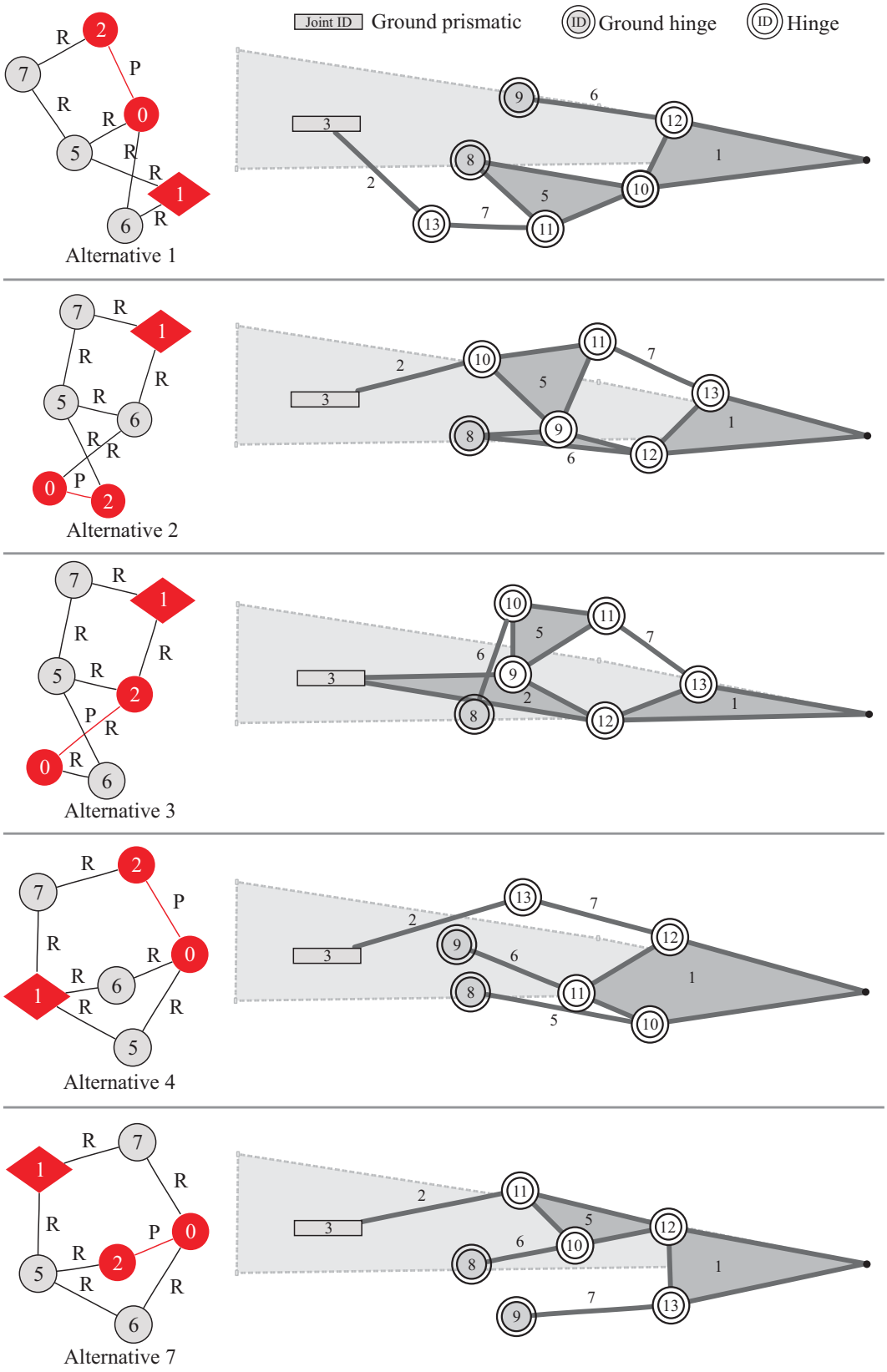



Figure 7.16: Outputs of the type synthesis solver and their corresponding physical sketches.

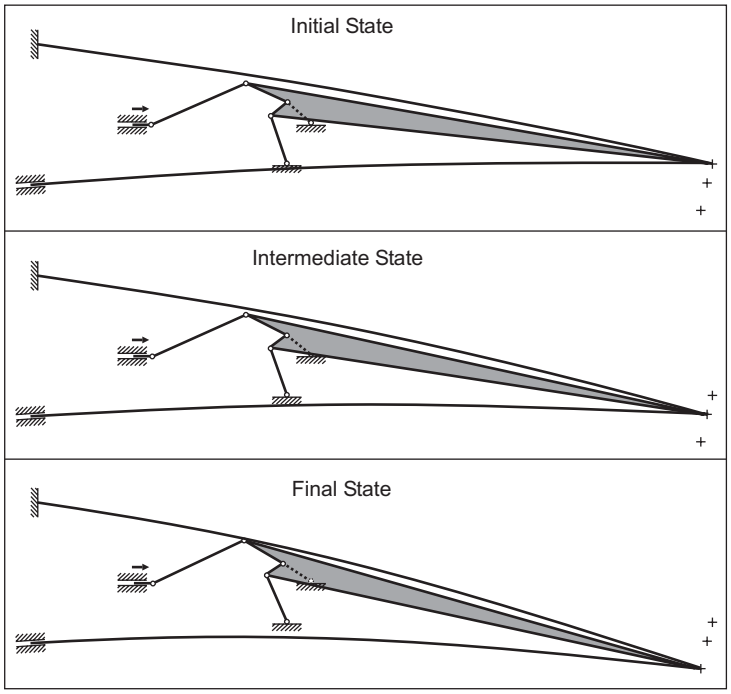



Figure 7.17: Rigid mechanism solution for guiding the tip of two flexible members through three positions.

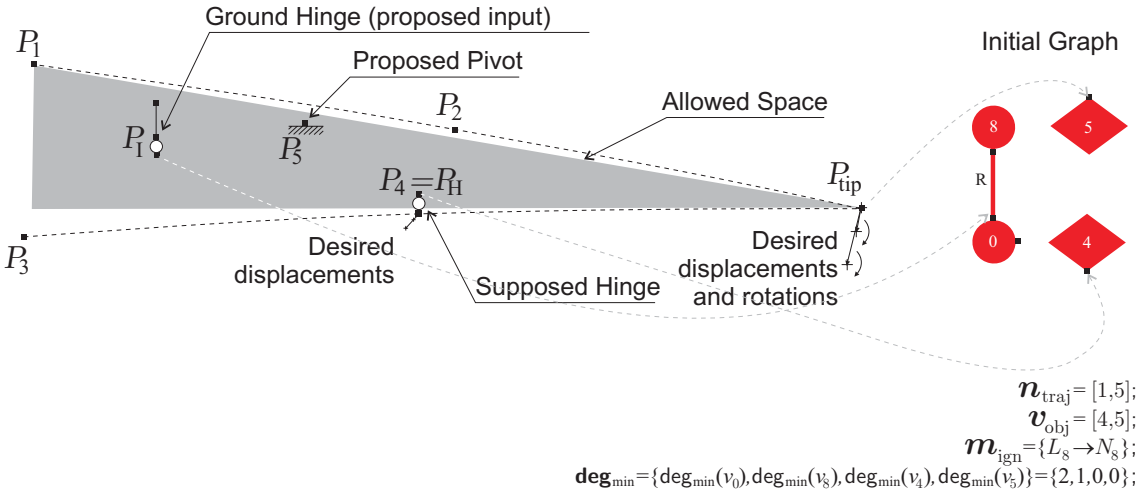



Figure 7.18: Deflection of two beams reformulated as a multiple-task synthesis problem.

- Guided hinge displacements:
  - $\diamond$ $\delta_1^{\mathrm{H}} = (-0.002325, -0.0025888)$, and $\delta_2^{\mathrm{H}} = (-0.005714, -0.0062605)$.
- Beams are defined by cubic-splines interpolation passing through the following points:
  - $\diamond$ Upper beam: $P_1 = (0.65085, 0.06106)$, $P_2 = (0.85148, 0.03018)$, $P_{\mathrm{tip}}$.
  - $\diamond$ Lower beam:$P_3 = (0.64915, -0.01086)$, $P_4 = (0.79849, -0.00229)$, $P_{\mathrm{tip}}$.

- Ground Hinge actuator (input): $P_I = (0.7, 0.02)$.
  - ◇ Maximum displacement: $\alpha_2^I = -3°$.
- Boundary conditions:
  - ◇ Clamped constraint on proposed pivot: $P_5 = (0.76, 0.03)$.
  - ◇ Clamped constraint on node $P_1$ of the upper beam.
  - ◇ Grounded prismatic joint constraint on node $P_3$, axis direction $\hat{n} = (0.99835, 0.05729, 0)$.

### Type synthesis

The graphs for the first ten solutions are displayed in Figure 7.19.

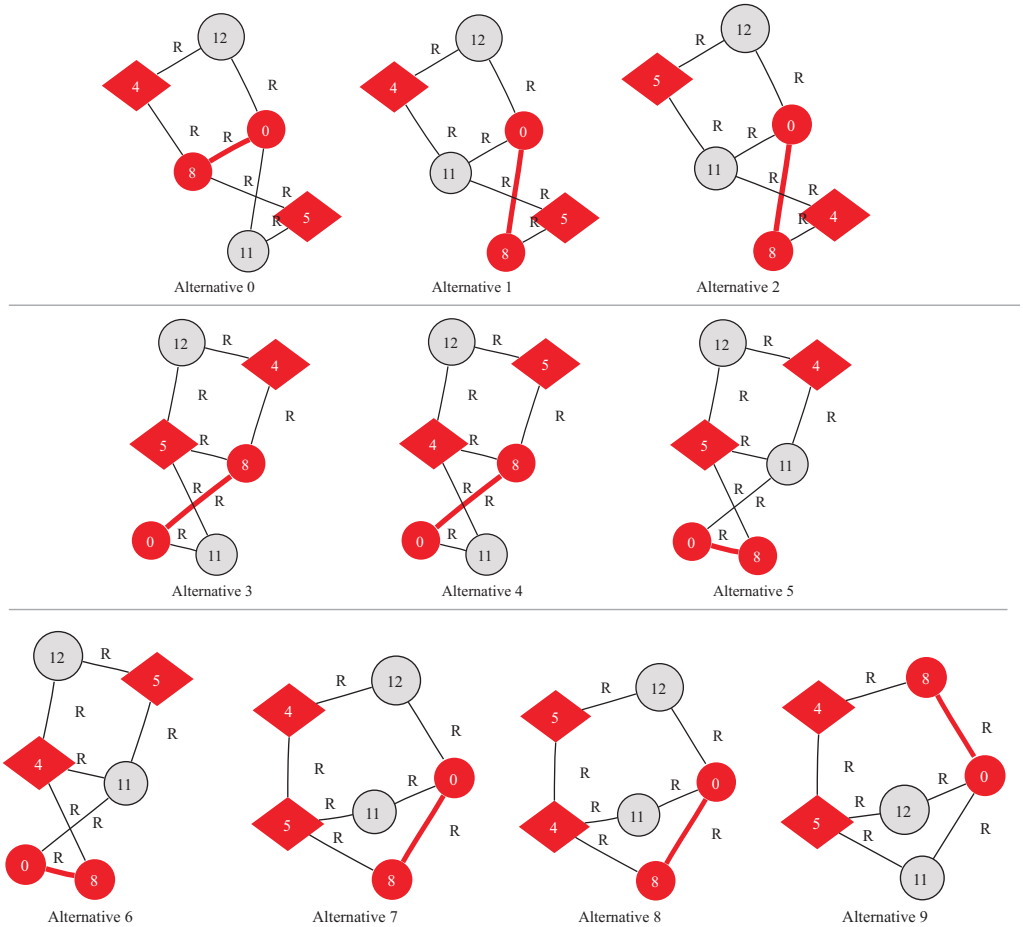



Figure 7.19: Data for the subgraph search and the first ten solutions of the type synthesis running.

The algorithm for SOCs decomposition gives us the **Alternatives 0**, **3**, **4**, **7**, **8** and **9** as compatible to be solved analytically. Their sketches are shown in Figure 7.20.

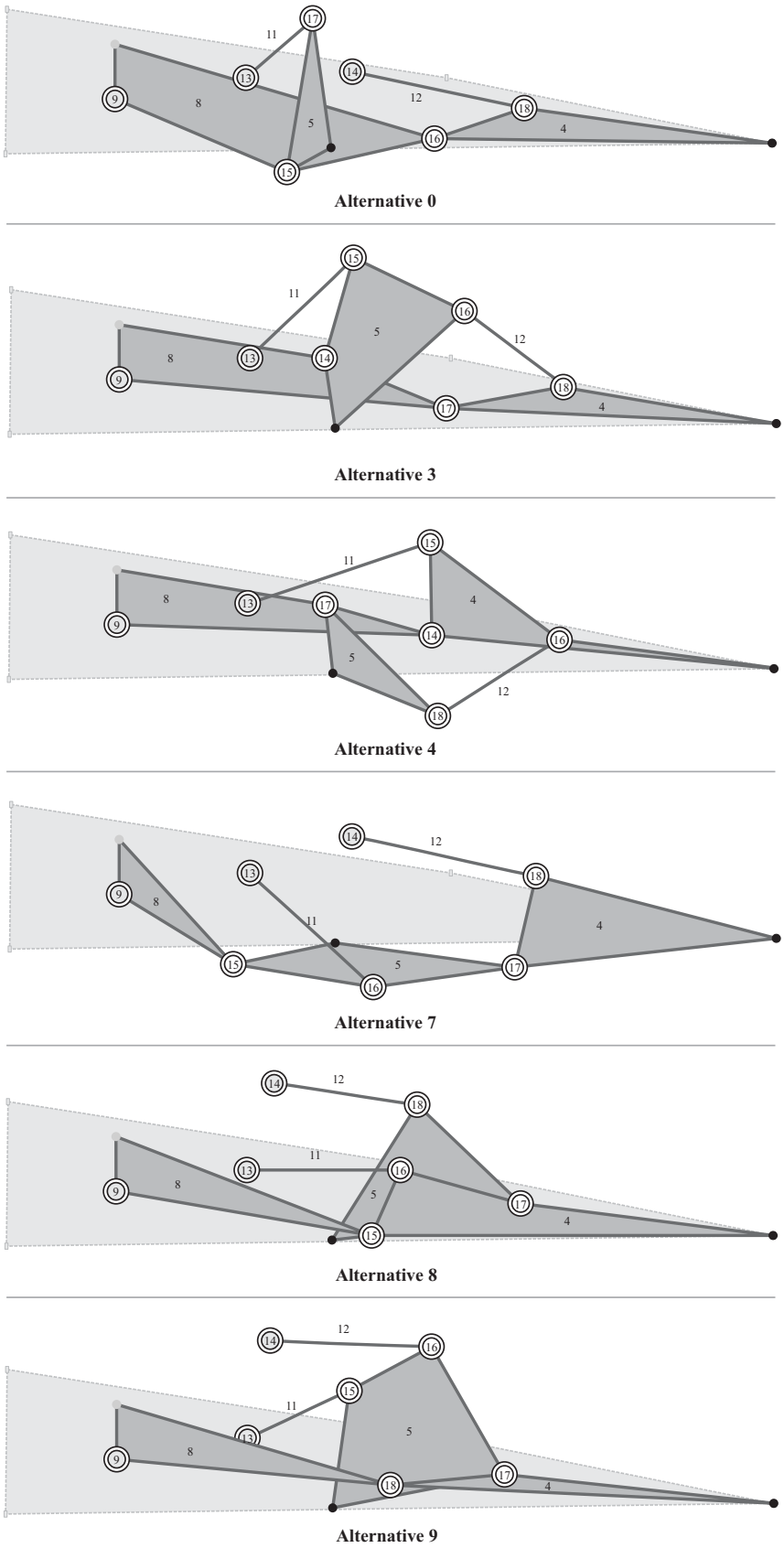



Figure 7.20: Sketches of the feasible decompositions.

### Initial sizing

A multi-loop solution corresponding to **Alternative 7** is shown in Figure 7.21 where the beams were attached to the tip. Among the ten analyzed alternatives, this was the solution that best satisfied the constraints.

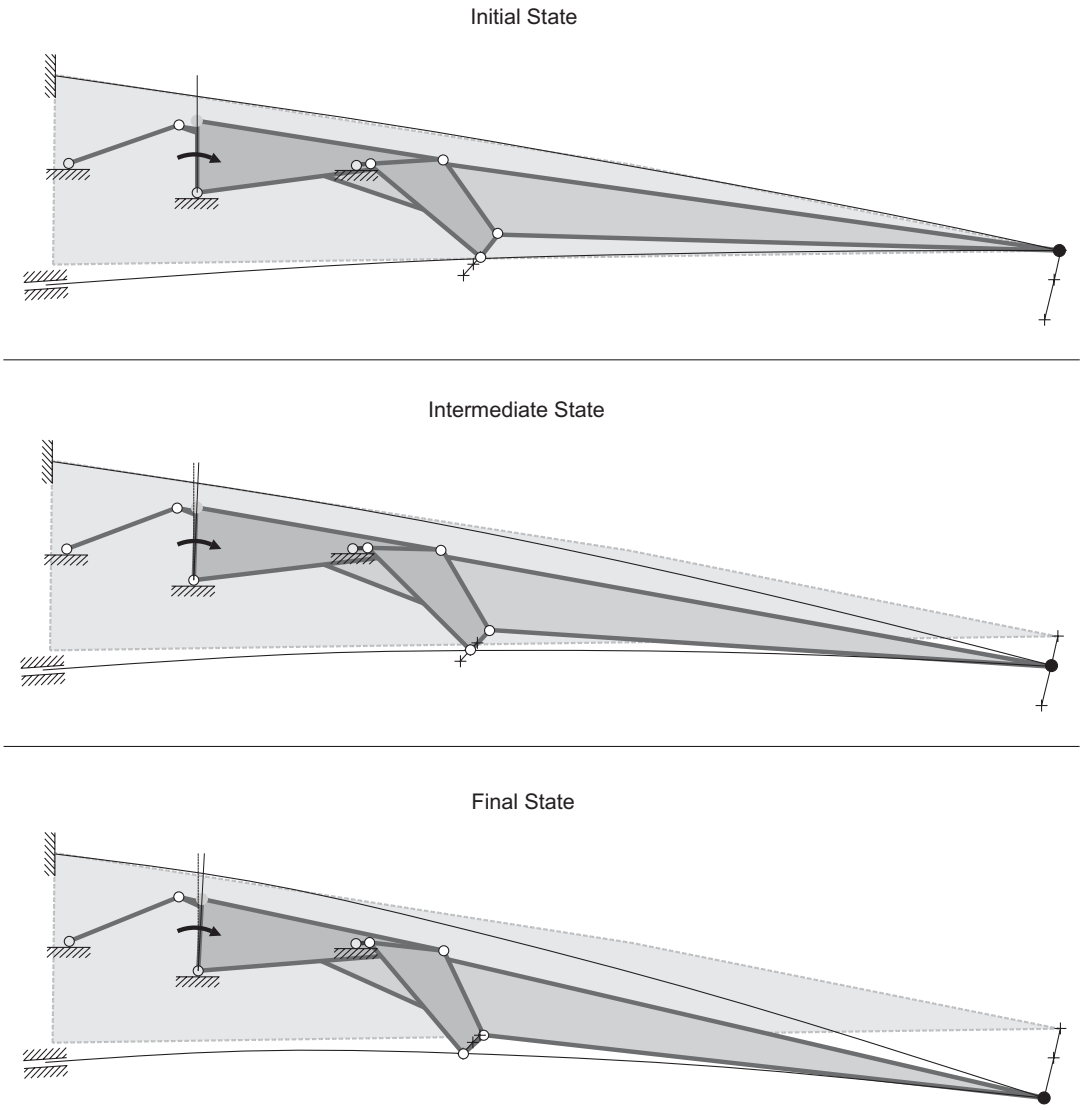



Figure 7.21: **Alternative 7** multi-loop linkage passing exactly through three positions prescribed for a combined task.

As we observe in the topologies found, they form a complex mechanism when the flexible beams are attached. This is an undesired result since the minimization of parts for reducing manufacturing costs and assembly time is always pursued. A different result would be obtained if the beams were considered inside the initial graph. This topic, still under research, requires the proper (i) atlas of flexible mechanisms, (ii) decomposition rules, and (iii) synthesis methods for initially curved beams.

One positive experience when applying the rigid solver to obtain this mix of mechanisms and structures, known as *partially compliant mechanisms*, is the possibility of getting an exact guidance of key points independently of the purpose of the movements of such points. In this example, the key points moved the beams that

represent the trailing edge of an airplane wing. The same procedure was followed for the leading edge. The procedure may find applications in other fields of mechanics.

### 7.2.2. Complex function generation

The kinematic task proposed to solve is showed in Figure 3.4. An unknown 1-DOF-mechanism must guide the rotations of two bodies, $E_{12}$ and $E_{15}$, passing through three prescribed angular positions. The application of such mechanism is to produce the change of section of a convergent-divergent nozzle of a turbine engine. The mechanism must be actuated by means of a grounded prismatic actuator for which the initial and final positions are prescribed ($E_{21}$). The *allowed space* area where the new bodies and new pivots must lie is also prescribed.

For reasons of confidentiality, the values of the functions $\alpha$ vs. $\beta$ cannot be shown explicitly. The procedure to solve it by means of the proposed method will be described.

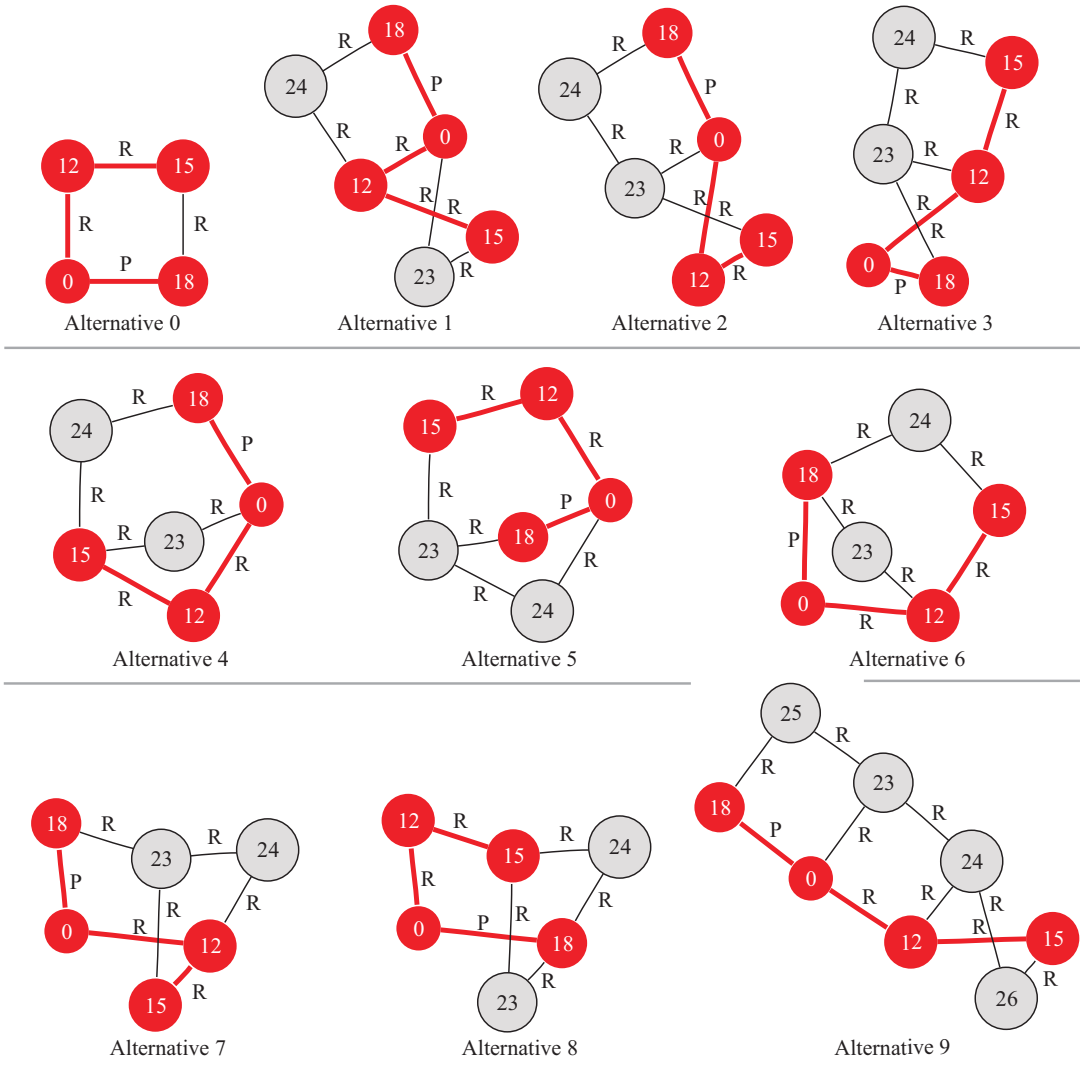



Figure 7.22: Non-isomorphic graph occurrences of the initial graph inside mechanisms of the atlas, obtained in the number synthesis stage. For clarity, edges are only labeled with their joint types (R: revolute, P: prismatic).

### Type synthesis

The initial graph for the example problem was shown in Figure 3.4-b. Below this figure additional information is shown. Note that for synthesis purposes, nodes $N_4$ and $N_9$ are ignored because they are not connected by joints. Neither the trajectory node nor the objective vertex is prescribed.

In this test, the *Kinematic Analysis of the Initial Parts* has a crucial effect on the subsequent SOC decomposition algorithm. Using joints $J_{20}$ and $J_{19}$ as input, the kinematic analysis helps to calculate displacements of nodes $N_3$, $N_7$ and $N_6$. Note that these nodes belong to the group of "*nodes connected by joints*". This will not produce any effect in the sub-graph search.

Since there is one prismatic joint, the selected atlas is RigidOneDofOneP. Then, the type synthesis solver is launched. In Figure 7.22, the first 10 occurrences of the solutions for a sub-graph search are shown. Note that if we did not consider the difference in prescribed parts, **Alternatives 1** and **2** would be isomorphic.

### A note on decomposition

The first topology feasible to be solved analytically by the available modules is **Alternative 1**.

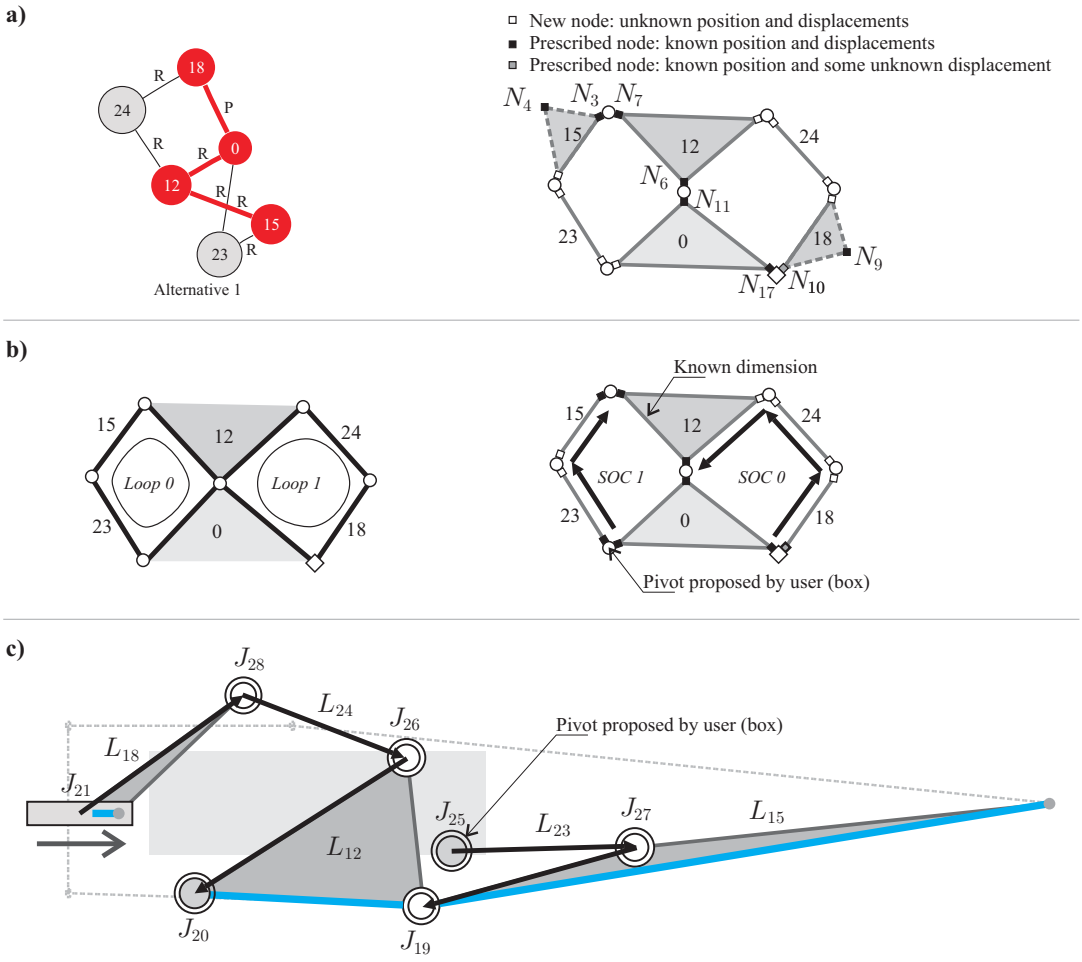



Figure 7.23: Decomposition process for **Alternative 1**

The node classification after the kinematic analysis of the initial parts is shown in Figure 7.23-a. Thus, nodes $N_3$, $N_7$ and $N_6$ are shown with a black-filled square, meaning that their positions and displacements are known. By means of the map $\boldsymbol{m}_{\text{ign}}$, Nodes $N_4$ and $N_9$ are associated with links $L_{15}$ and $L_{18}$, respectively. Node $N_{10}$ is classified as a node with an unknown displacement component. This missing component will be computed in the dimensional synthesis stage. Nodes $N_{17}$ and $N_{11}$ also have a black-filled square since they are fixations of the mechanism, and therefore, their positions and displacements are known. The exception to the rule is constituted by the nodes of new created pivots that are considered as known for the decomposition purpose. The remaining nodes represented by white-filled squares are the unknowns of the problem.

In Figure 7.23-b, it can be observed that the calculation of positions and displacements for nodes connected by joints ($N_3$, $N_7$ and $N_6$) effectively influences the behavior of the SOCs decomposition algorithm for the identification of significant dimensions.

The retained decomposition for **Alternative 1**, where the resultant SOCs are drawn, is shown in Figure 7.23-c. The proposed area for the new pivot is also displayed.

The sketches for the other topologies are shown in Figures 7.24 and 7.25.

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 that a new pivot location must be computed.

### Dimensional synthesis

The initial sizing solver was run for the nine feasible alternatives shown above. Note that three constraints must be taken into account. Examples of initial sizing for **Alternatives 1-4** and **9** are shown in Figure 7.26.

Observe that all of them fulfill the *non-inversion of transmission angle* constraint whereas only **Alternatives 1** and **2** satisfy the allowed space constraint for all the configurations. We may also observe that:

- **Alternatives 1** and **2** are Watt II six-bar linkages since 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 since 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.

These alternatives can be analyzed for evaluating further non-kinematic requirements, such as mechanical advantage, power consumption, and others. Interference

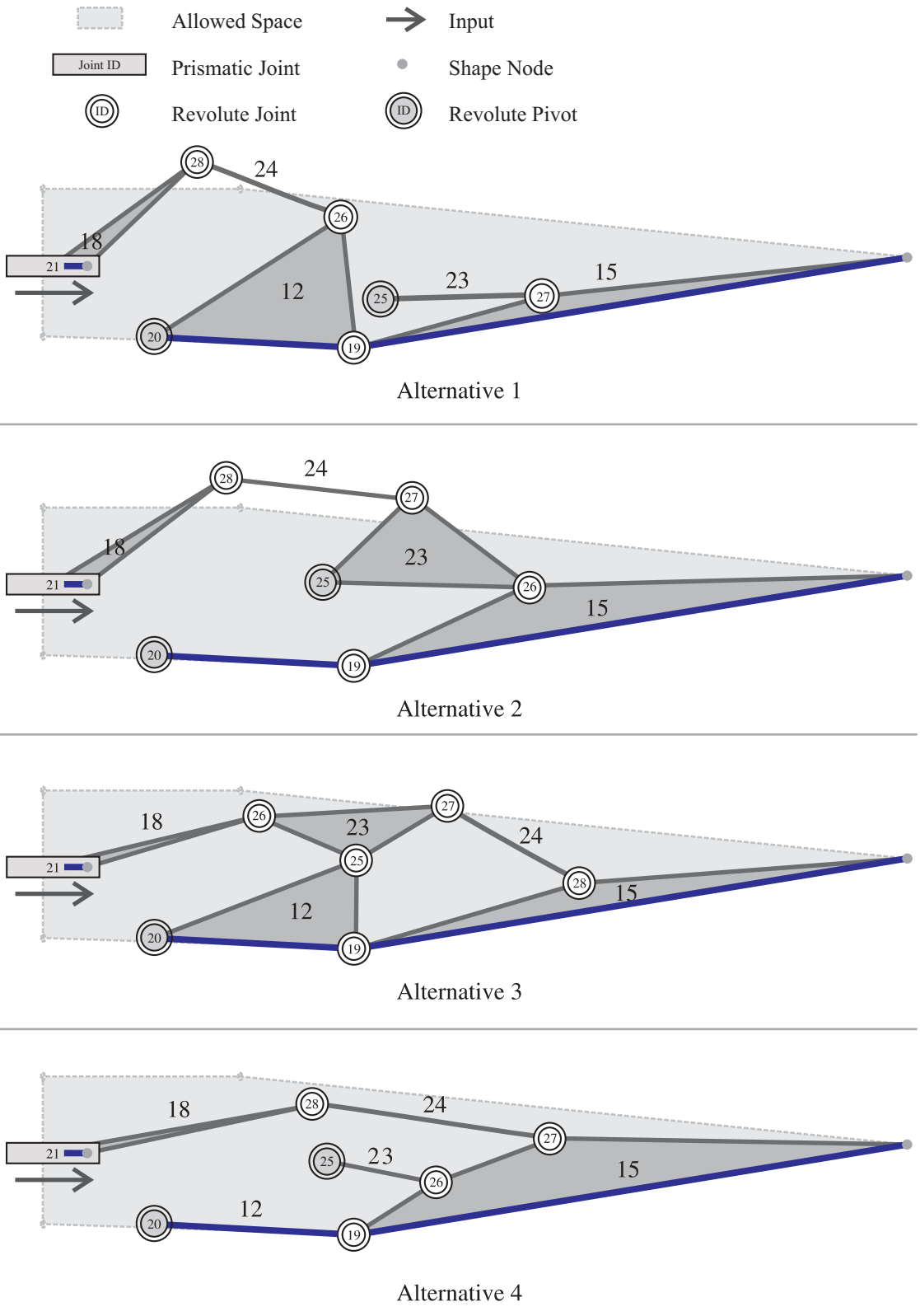



Figure 7.24: Sketches of the first nine alternatives found (continued).

avoidance as well as the automatic qualification of the alternatives will be considered in future research.

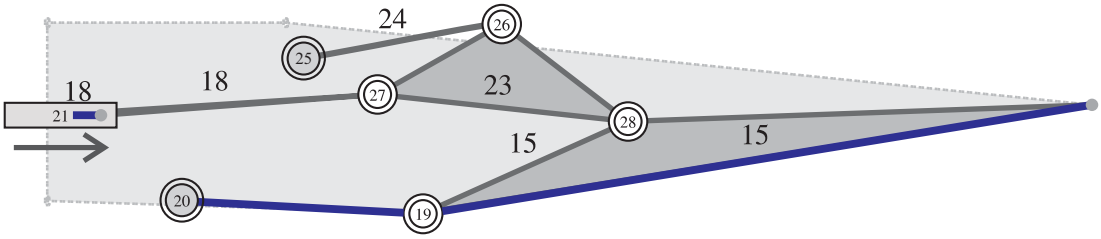
## 7.3. Chapter conclusions

In this chapter, some representative tests were considered to show the utility of the proposal either starting from scratch or for the completion of existing sub-mechanisms. The best solution found for each test was displayed. All the presented tests were rigid tasks, however, it was also shown that some rigid tasks can be stated to guide compliant segments.

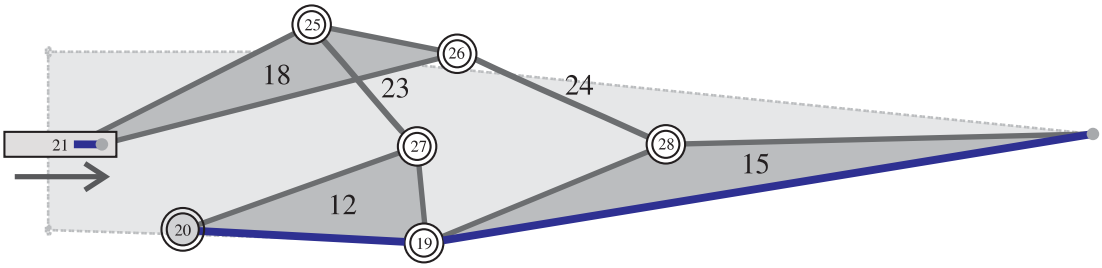
Some details of the CAD environment and data definition for both synthesis stages were given. After the synthesis task, the user can verify the validity of the exported solution (parameterized model) by means of analysis, and the user can even optimize the solution staying within the same topology.

Under the SYNAMEC[1] and SYNCOMECS[2] projects, other tests such as commands of flaps, landing gears, and nozzles of turbines were solved. All of these tasks were smooth and open. Future research will include closed and non-smooth tasks.

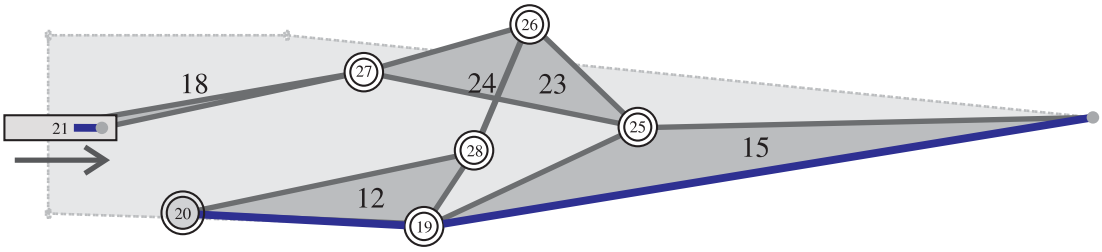
[1] SYNthesis tool for Aeronautical MEChanisms design UE 2001-001-0058 (2002-2005).
[2] SYNthesis of COmpliant MEChanical Systems UE FP6-2003-AERO-1-516183 (2005-2007).



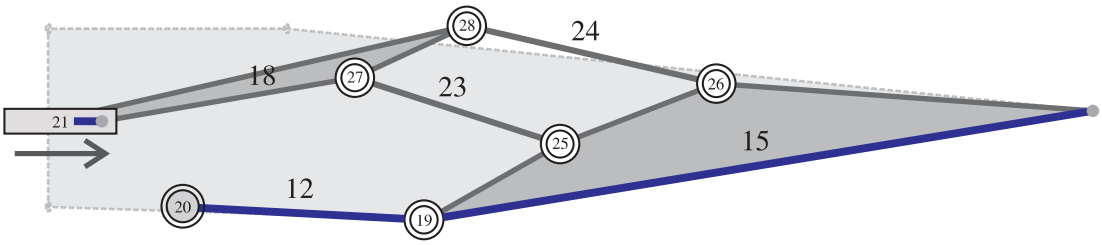

Alternative 5



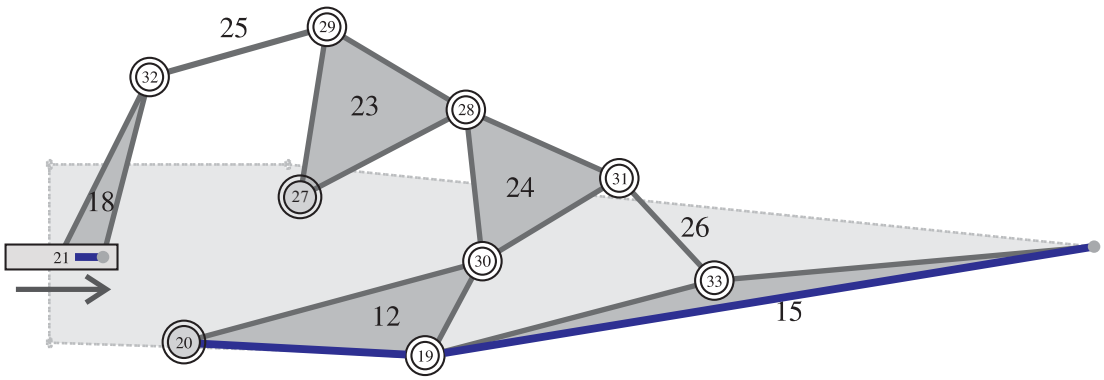

Alternative 6




Alternative 7




Alternative 8




Alternative 9

Figure 7.25: Sketches of the first nine alternatives found.

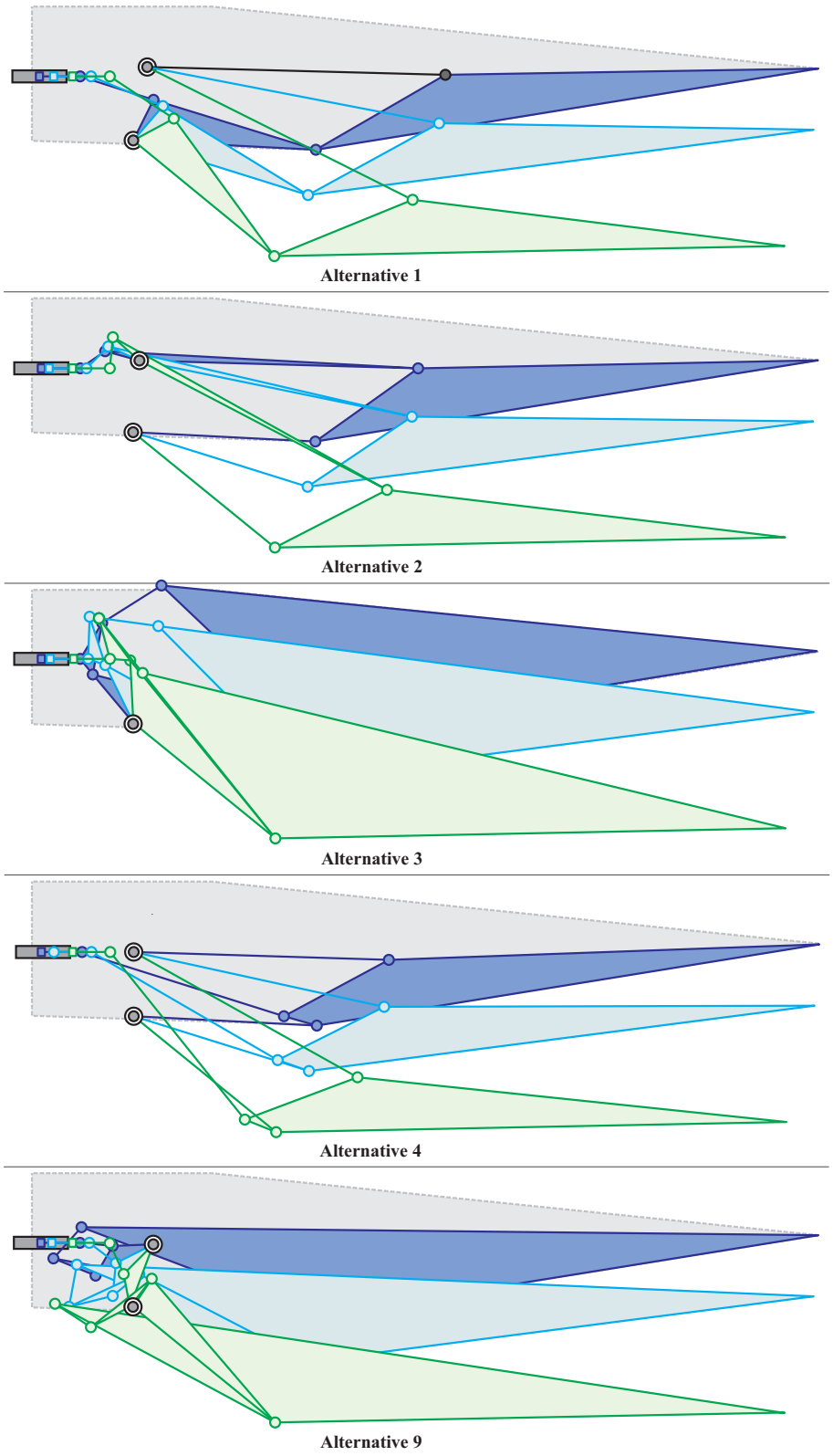



Figure 7.26: Mechanism solutions for **Alternatives 1**, **2**, **3**, **4** and **9**.

# Chapter 8

# Synthesis of flexible mechanisms

The many advantages of compliant mechanisms compared to their rigid-body counterparts have produced a growing interest in compliant synthesis methods. Compliant-mechanism synthesis is currently a challenging area of development that has been addressed in several ways including pseudo-rigid-body models (PRBM) [How01], and shape and topology optimization [LK06, PS07].

In this chapter, preliminary results for the application of a method proposed by Howell [How01, BMH00] to solve the kinematic synthesis of compliant mechanisms by means of *rigid-body replacement* are presented. The approach is useful for designing mechanisms to perform a traditional rigid-body mechanism task –path following, function generation and rigid-body guidance– without concern for the energy storage in the flexible members.

## 8.1. Introduction

Howell [How01] presented the Pseudo-Rigid-Body Model (PRBM) as a very practical tool to simplify the analysis and synthesis of compliant mechanisms. By using rigid-body components, the PRBM allows the designer to model flexible members that undergo large non-linear deflections, see Figure 8.1. There are very simple geometrical relationships to verify between the dimensions of the flexible members and those of their equivalent rigid components. This concept was the main motivation to exploit the presented method as a tool for creative compliant mechanism design.

A beam can be modeled by an articulated rigid-body with the beam *characteristic length* and by torsional springs located on its *characteristic pivots* to emulate the beam stiffness. The *characteristic length* is computed as $\gamma L$ where $\gamma$ is the *characteristic radius factor* which is determined as function of the load case and boundary conditions, and $L$ is the beam length. The spring stiffness $K(EI, L, \gamma)$ is used to reproduce the force-deflection relationship. It depends on the beam material properties $E$, the inertia of the cross-section $I$, the geometry $L$, and load case (direction of the applied force $F$ and existence of end-moment loading $M$) through $\gamma$. The deflection is a function of the so-called the *pseudo-rigid-body angle* $\Theta$. See, for example, the parameterization of a beam of length $L$ subjected to two different conditions in Figure 8.1-**a** and **b**. The boundary conditions as well as the load conditions are different. Both cases consider loads only at the end-points of the beams.

With respect to the kinematic behavior of such simplified model there is an error.

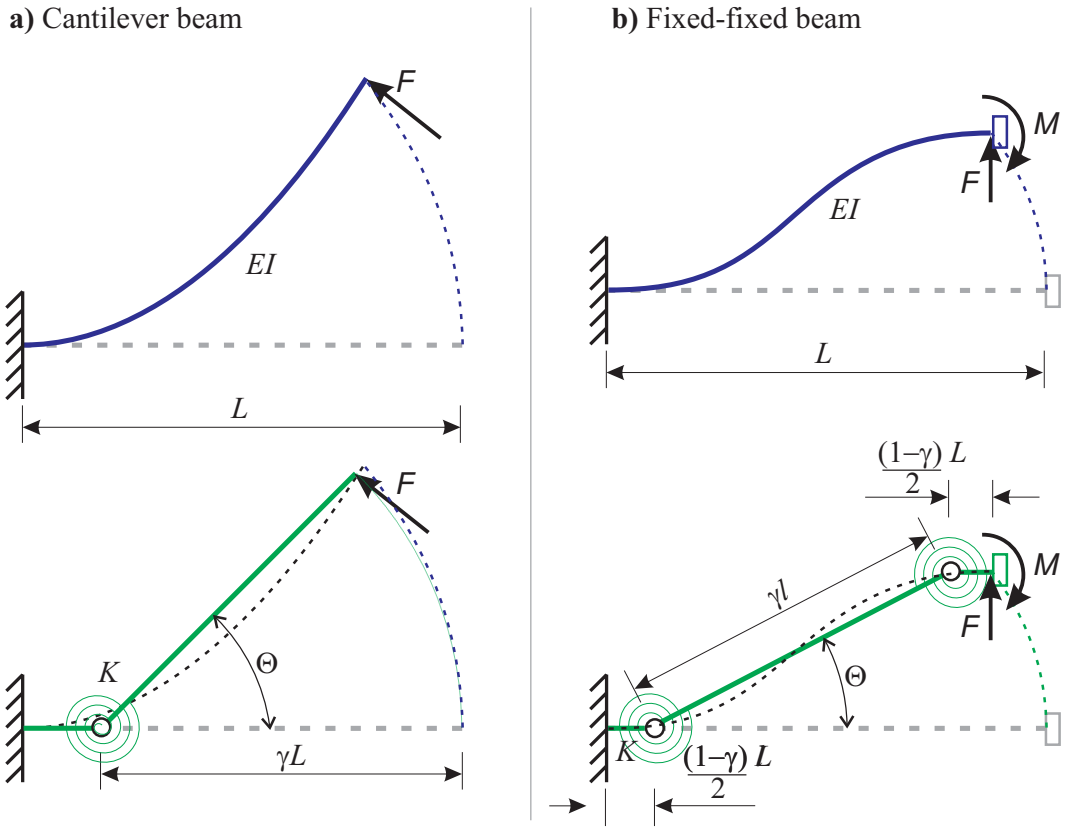



Figure 8.1: PRBM models for two beams [How01].

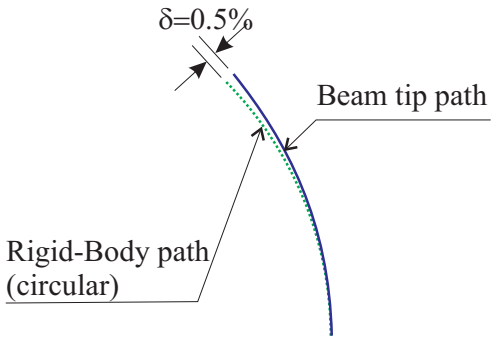



Figure 8.2: Error between tip trajectories: exact beam vs. its PRBM model [How01].

For example, in the cantilever beam, the error between the trajectories is the shown in Figure 8.2. The difference of the trajectories between the exact and the PRBM model increases as the rotated angle $\Theta$ is increased. By defining an admissible path error between the exact beam and the PRBM, Howell computed $\gamma$ that maximized the deflection $\Theta_{\max}$ as function of the angle of the applied load $F$. The example shown in Figure 8.2 corresponds to a load applied perpendicularly to the beam for which a path error of 0.5% was obtained at $\Theta_{\max} = 73°$ and $\gamma = 0.8517$.

The inverse situation, called *Rigid-Body Replacement Synthesis*, is also very easy to make by identifying rigid-body components as the PRBM of flexible members to synthesize. A rough calculation can be made by proposing $\gamma = 0.8517$, Young modulus for steel $E$, and heuristically defining $I = \frac{L^4}{10^5}[m^4]$, and $A = \frac{L^2}{100}[m^2]$ with $[L] = m$. Note that a rigid-body does not support any moment at revolute joints,

but it does at the characteristic pivot when the group rigid-link and revolute joint is replaced by a beam. For the chosen beam length and stiffness, the major kinematic error in the path-deflection will arise in the resultant load-case when the beam is assembled with other pieces inside the mechanism. For example, the end-points of the beam would result loaded by traverse forces and end-moments which are different from that pure bending hypothesis considered by $\gamma = 0.8517$. However, Howell also determined that for a wide range of angles of the applied force $F$ ([135°-63°]) the average of $\gamma$ is 0.85.Additionally, for end-moment loading $\gamma_{\text{ave}} = 0.7346$ with $\Theta_{\text{max}} = 124.4°$. For the model shown in Figure 8.1-**b** he found that $\gamma_{\text{ave}} = 0.8517$ with $\Theta_{\text{max}} = 64.3°$.

The method consists of the following stages: (i) convert the kinematic compliant problem into a rigid one defined by precision positions, (ii) apply rigid number synthesis methods to propose a valid topology, (iii) find the initial dimensions (link lengths and pivot positions) using loop-closure techniques [SE84], (iv) wherever possible, identify the resultant parts as the pseudo-rigid-body models of the compliant members to be synthesized. An additional optimization loop of *dimensional synthesis* must be applied to further refine the dimensions for minimizing the kinematic errors.

In the *Type Synthesis solver* no major change is required, since the selection of the proper atlas is an easy step. Since these atlases offer a big number of alternatives, the specification of filters for some link and joint types might be defined to restrict the search.

In the *Initial Sizing solver* an additional procedure is located just after the execution of the loop on the SOC solvers computes the equivalent pseudo-rigid-body models. Then, the function `SelectOptimum` is modified to take into account additional constraints necessary for the compliant mechanism, e.g. bounds for compliant joint rotations. Inside the *non-inversion of transmission angles* restriction, a different treatment is made for compliant joints of flexible and clamped types.

Two potential applications of the rigid synthesis solvers for compliant synthesis are shown: a) Replacement of rigid parts of a mechanism by flexible members to obtain compliant mechanisms (eventually, with new kinematic behaviors), and b) Design of bi-stable mechanisms.

This Chapter is organized as follows: the modification made to rigid solutions is presented in Section 8.2; results are presented in Section 8.3.

## 8.2. Synthesis by Rigid-Body Replacement

The relative rotations between rigid members which make optimal the fulfilment of a rigid kinematic task might lead to incompatibilities either for distributed compliance replacement (flexible segments or thin beams) or concentrated compliance replacement (flexural joints or living hinges). Compliant hinges cannot fully rotate, so practical limits between end-point rotations must be respected. Thus, after applying rigid synthesis, replacement may result in partially compliant mechanisms, i.e. a mix of mechanisms and structures. But, despite kinematic joints or rigid links remaining in the mechanism, the solution may be a useful candidate from a functional point of view.

### 8.2.1. Compliant dimensional synthesis

It is considered that for each feasible closed-loop topology, each link has functionally and structurally equal degree [How01]. This means that structurally binary links are functionally binary, ternary links are functionally ternary, and so on. Additionally, all segments are assumed initially straight. A third assumption is that all links are homogeneous, i.e. rigid links are constituted only by rigid segments, and flexible links, only by flexible segments. Another topological assumption is added: actuated links are considered rigid, and coupler links (floating) with imposed precision positions are also considered rigid. This aspect will be extended in future research for supporting tasks like *flexible segment guidance* [AFPC07] where sets of displacements and/or orientations are prescribed for at least two nodes of a floating segment.

#### Links replacement

After computing the rigid dimensional synthesis, the set of minimal independent loops of a rigid mechanism topology is used to visit the links and analyze their feasibility to be transformed into their flexible equivalents. Links are analyzed from groups of three consecutive links. When a flexible link is found, decisions about changing the coordinates of its end-point nodes are taken considering not only the type of the terminal joints, but also the type of the terminal links. Inside this analysis, ground and rigid links are equally considered as "rigid".

Let $l_i$ be the considered flexible link, and let $l_{i-1}$ and $l_{i+1}$ be the adjacent ones. Also, let $R$ be the length of the rigid link, and $L$ the length of its flexible equivalent counterpart. The *characteristic ratio* $\gamma = R/L$ is taken to be fixed at a value $\gamma = 0.8517$.

As it is shown in Figure 8.3, two typical cases are synthesized in the following way:

a) Fixed-Revolute (Revolute-Fixed) flexible link: the fixed node is relocated at a distance

$$L = \frac{R}{\gamma}$$

from the pinned end, measured along the link axis. This modification is applied if the link type of the preceding (or following) link $l_{i-1}$ ($l_{i+1}$) is rigid. The other link can be either of rigid or flexible type.

b) Fixed-Fixed flexible link: both fixed nodes are relocated at a distance

$$L_{\frac{1}{2}} = \pm\frac{R}{2\gamma}$$

from the link midpoint along its axis. This modification is applied if the link types of the preceding and the following links, $l_{i-1}$ and $l_{i+1}$, are both rigid.

With these simple modifications, the characteristic pivots are located at the same location as the revolute joints. Other cases are not replaced. Some cases are for now ignored and still have to be investigated.

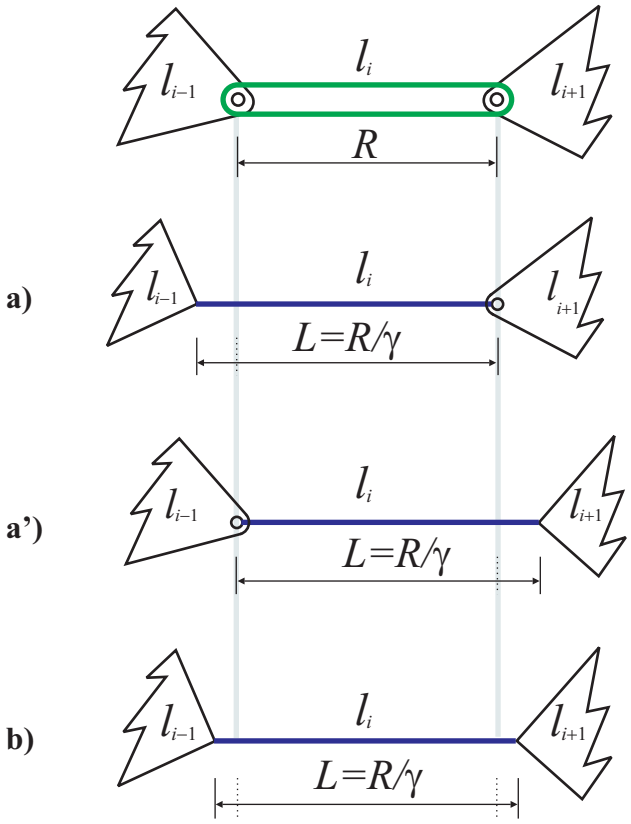



Figure 8.3: Rigid-body Replacements: a) Fixed-Revolute, a') Revolute-Fixed and b) Fixed-Fixed.

### Joint constraints

The angle rotation in the neighborhood of clamped joints must not surpass certain limits. After rigid synthesis, the information about rigid links rotations for a number of precision positions is available, thus the maximum rotation between two links can be measured on the revolute joint which will be the characteristic pivot of the PRBM of the flexible link. This does not exactly reflect what occurs in flexible links, but it is useful to develop a new constraint: *limit angle for clamped joints* denoted as $\Theta_{\max}$.

Figure 8.4-a shows the initial and final positions for a rigid link to be replaced. The maximum rotation, $\theta_{\text{rigid}}$, is measured on the revolute joint. True rotations at the clamped end of the flexible link, $\theta_{\text{clamped}}$, and the desired maximum angle, $\Theta_{\max}$, are shown in Figure 8.4-b. Note that the maximum angle performed by the rigid link in the rigid mechanism is a rough but useful approximation of the rotations of the flexible link end-points in the transformed mechanism.

To compute the angle violation on the $k$-th joint, the angular position $\psi_k^0$ between the connected links is taken as reference, see Figure 8.5. Unlike its rigid counterpart, no constraint is considered for angle $\psi_k^0$. Then, this angular reference plus the limit angle $\Theta_{\max}$ configure a lower and an upper limit for the movement

$$l_{\min} = \psi_k^0 - \frac{\Theta_{\max}}{2} \quad l_{\max} = \psi_k^0 + \frac{\Theta_{\max}}{2}$$

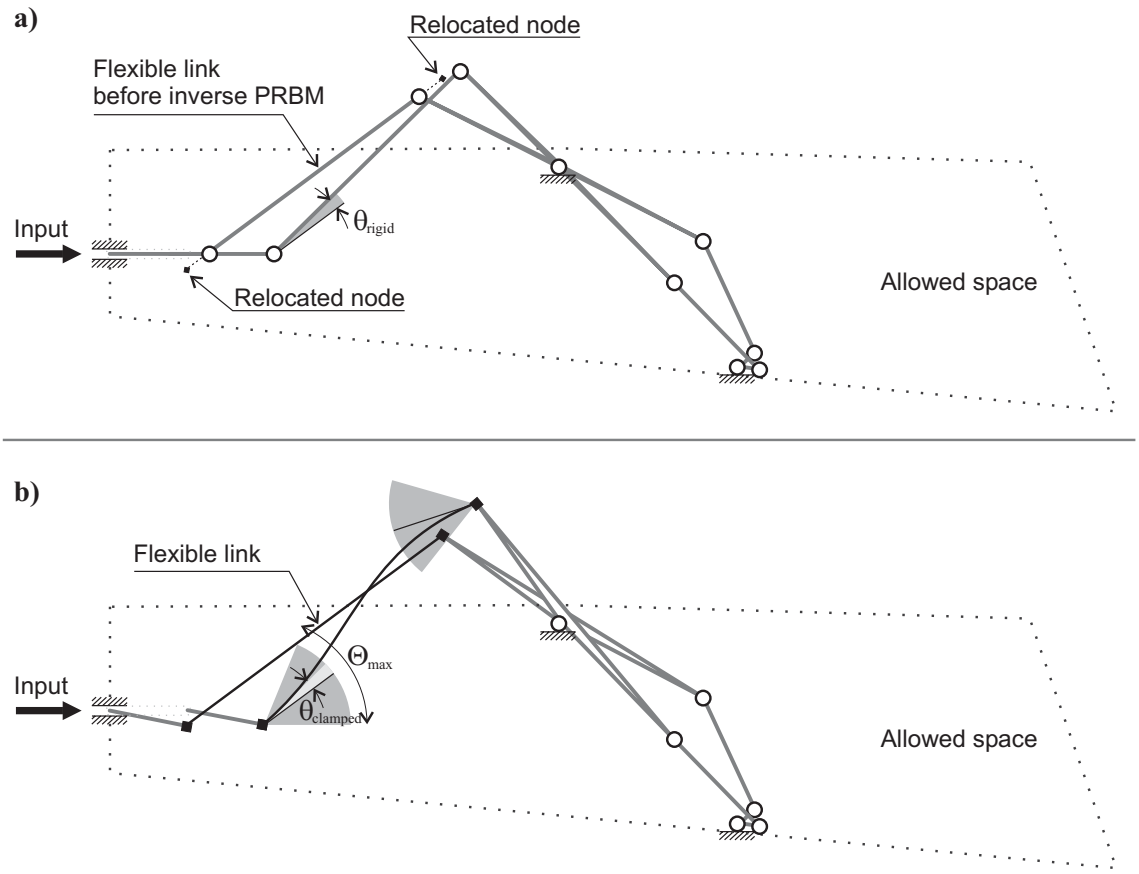



Figure 8.4: Limit angle for clamped ends of flexible segments: a) Rigid mechanism, b) Replacement.

in such a way, if a following position $\psi_k^j$ $(j = 1, \ldots, n_{pp} - 1)$ falls outside the region $[l_{\min}, l_{\max}]$, the constraint violation computed as the difference

$$\Delta\psi_k^j = \min\{ \mod(\psi_k^j - l_{\max}, 2\pi), \mod(l_{\min} - \psi_k^j, 2\pi) \} \tag{8.1}$$

is accumulated.

An example of such violation occurs in the third position $\psi_k^3$ depicted in Figure 8.5 where its contribution $\Delta\psi_k^3$ is shown.

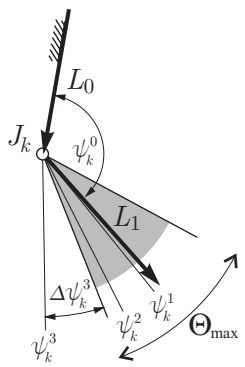



Figure 8.5: Example of violation for the limit angle for clamped and flexible joints.

For convenience, two groups of practical values for $\Theta_{\max}$ are taken (i) a value $\Theta^F_{\max} = \pi$ for ***F**lexible joints*, and (ii) $\Theta^C_{\max} = \pi/2$ for ***C**lamped joints*. The same limits are applied for all joints of the same type. Under the group of flexible joints, *flexural pivots* and *living hinges* which, of course, admit a range of rotation wider than clamped joints are considered.

## 8.3. Flexible double function generation example

The description of this example was illustrated in Figure 2.2 and it is also shown in Figure 8.6. The movement of the linear actuator (whose initial and final positions are 30 and 40mm) must be coordinated with the rotations of two cranks. Between both cranks there is a non-linear law which is also prescribed. The initial graph for such problem is shown in the same figure. The second position of lower crank, $\beta_2$, can take values inside the interval $[-0.79, -0.65]$.

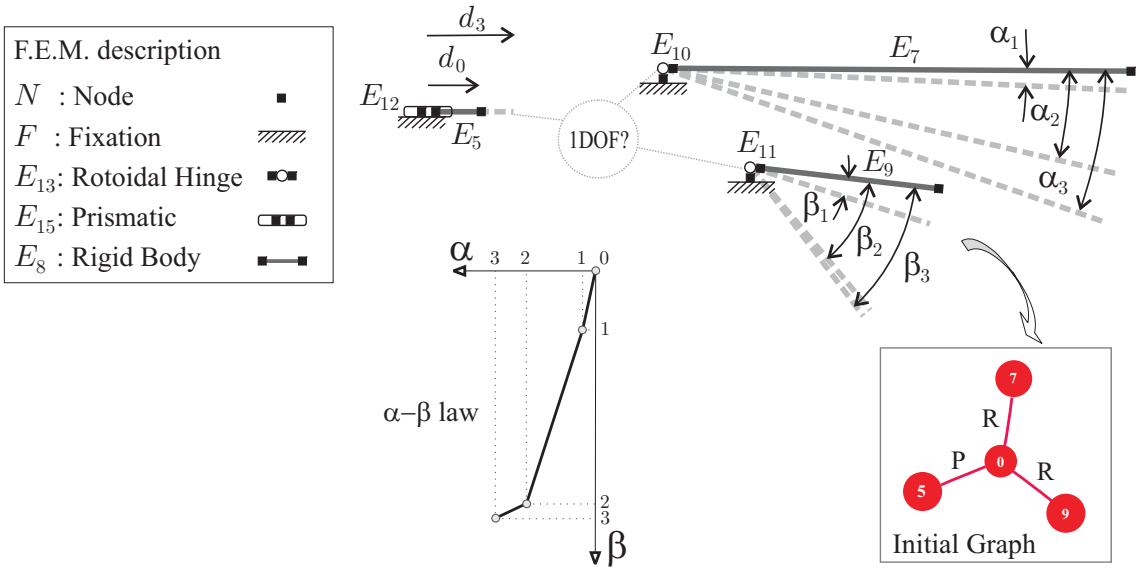



Figure 8.6: Double function generation problem.

The type synthesis solutions for a user's prescription of a maximum of ten alternatives plus the selection of the CompliantOneDofRoneP atlas are shown in Figures 8.7 and 8.8. Solutions **0**, **1** and **2** are fully rigid; the remaining ones have some flexible links connected by revolute joints or by one or two clamped ends connected to other links. Regardless of link and joint types, the **Alternatives 3**, **5**, **7** and **9** have the same topology as **Alternative 0**. They arise from the same Watt-II inversion where link 7 (the upper crank) is ternary. To show the effect of the combinatorial explosion, note that the synthesized flexible link 15 found by the subgraph search, takes all possible combinations of connections: revolute-revolute in alternative **3**, clamped-revolute in **5**, revolute-clamped in **7** and clamped-clamped in **9**.

We also depict three sized solutions in Figures 8.9 (rigid) , 8.10 (partially compliant), and 8.11 (partially compliant).

Finally, in Figure 8.12 the behavior for all alternatives after the initial sizing is shown. For each kinematic analysis the prismatic input had the same prescribed

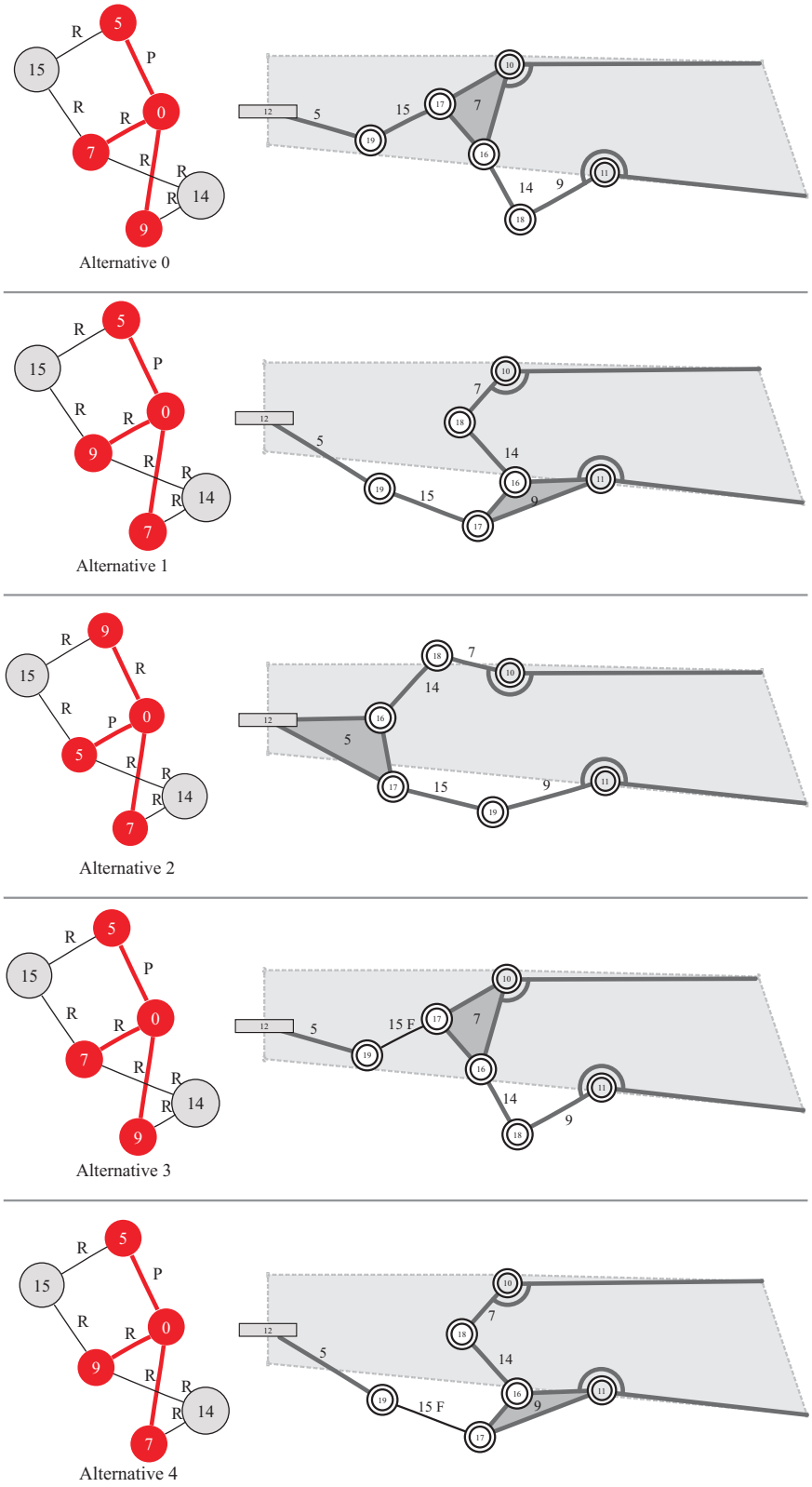



Figure 8.7: Outputs of the type synthesis solver and their corresponding physical sketches for **Alternatives 0** to **4**. References for joint types in graphs: R=revolute, P=prismatic, C=clamped. In sketches, flexible links have a letter "F" (Continued on Figure 8.8).

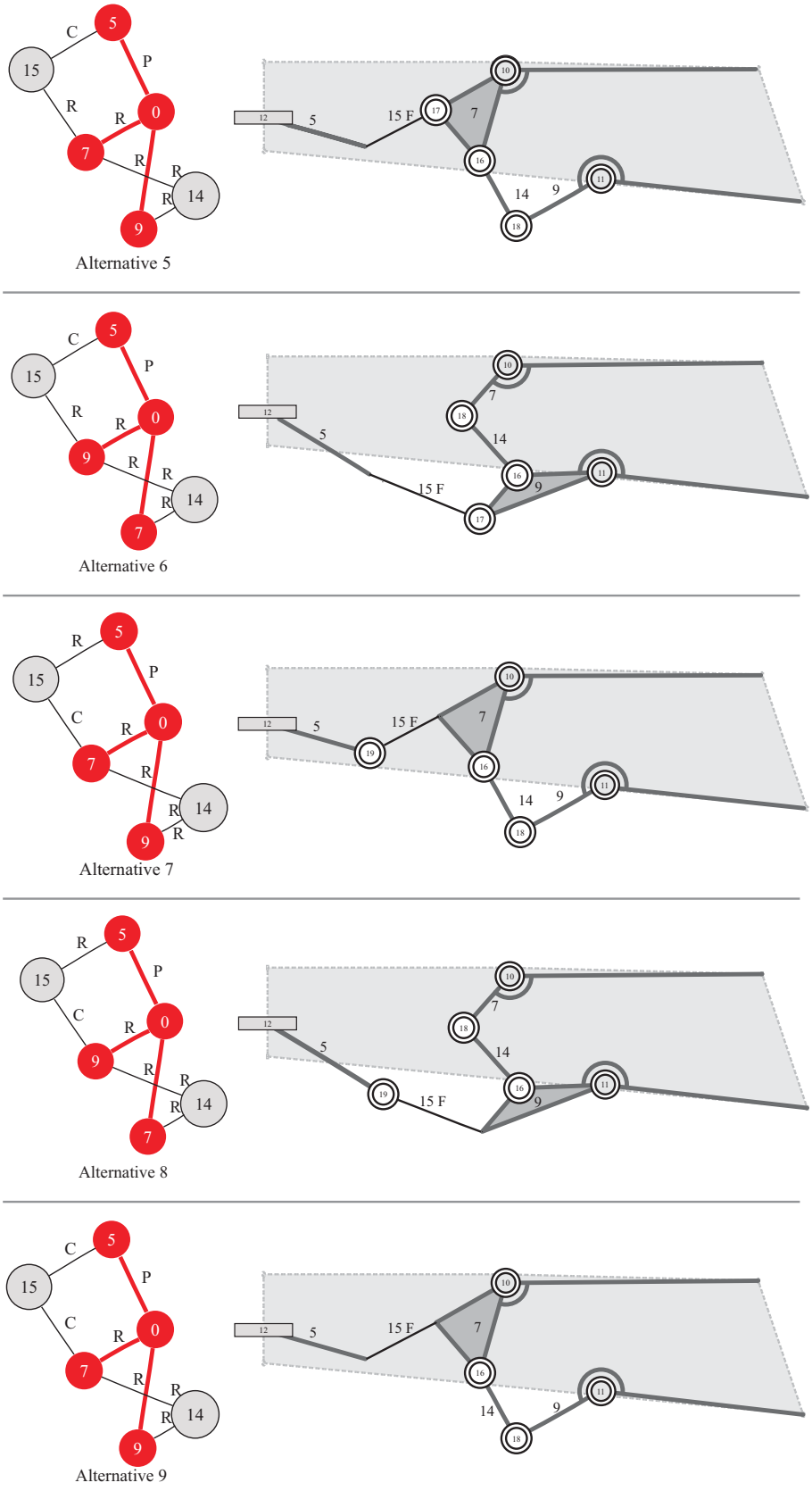



Figure 8.8: (Continued from Figure 8.7) Outputs of the type synthesis solver and their corresponding physical sketches for **Alternatives 5** to **9**.

value (0 to 0.030 m). Among the rigid alternatives (**0-2**), only **Alternative 2** passed trough the four required points. If we have a look at the partially-flexible alternatives (**3-9**), we find that **Alternative 9** also fulfils the required task; since it has a beam with two clamped ends, it helps to reduce the number of kinematic pairs in two. Although it is not shown here, there is an alternative which has two beams with clamped ends (bodies $L_{14}$ and $L_{15}$), which reduces the kinematic pairs in four.

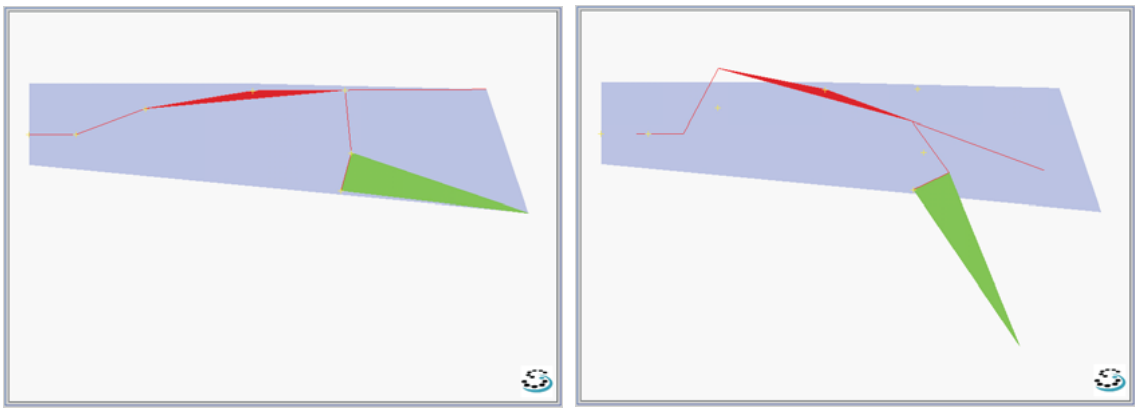


Figure 8.9: **Alternative 0** at the initial (left) and final (right) positions.

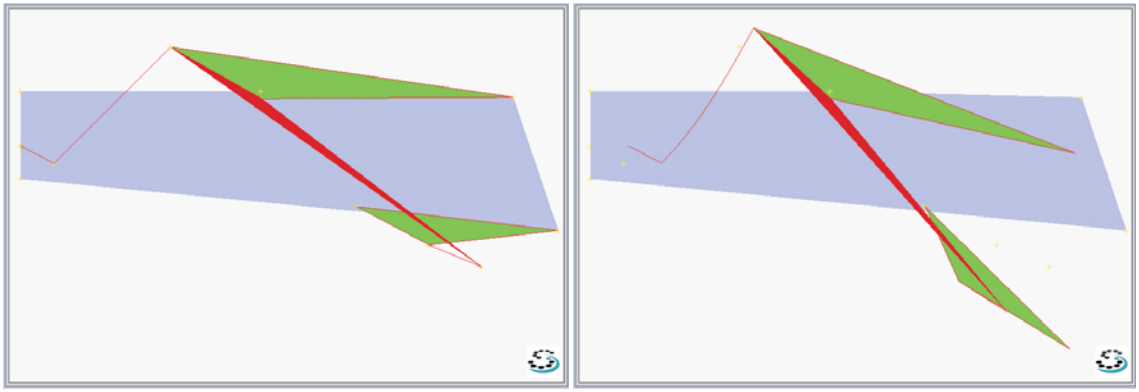


Figure 8.10: **Alternative 5** at the initial (left) and final (right) positions.

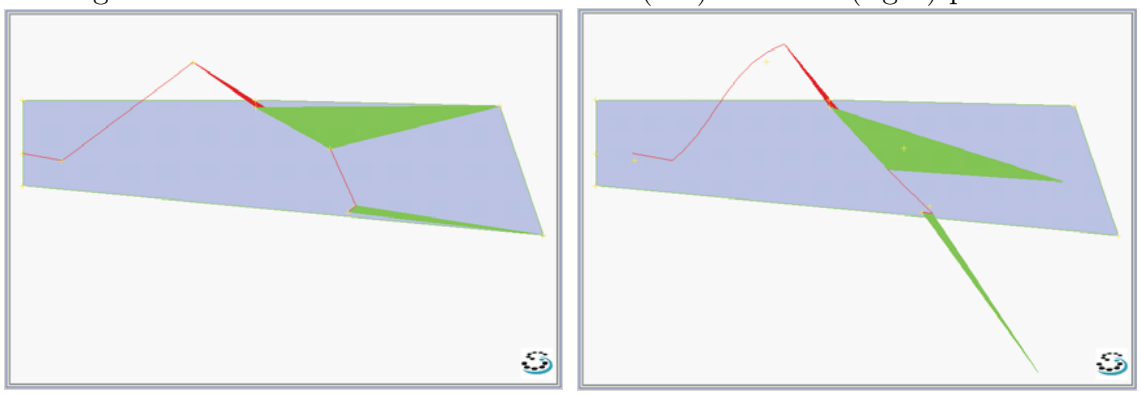


Figure 8.11: **Alternative 9** at the initial (left) and final (right) positions.

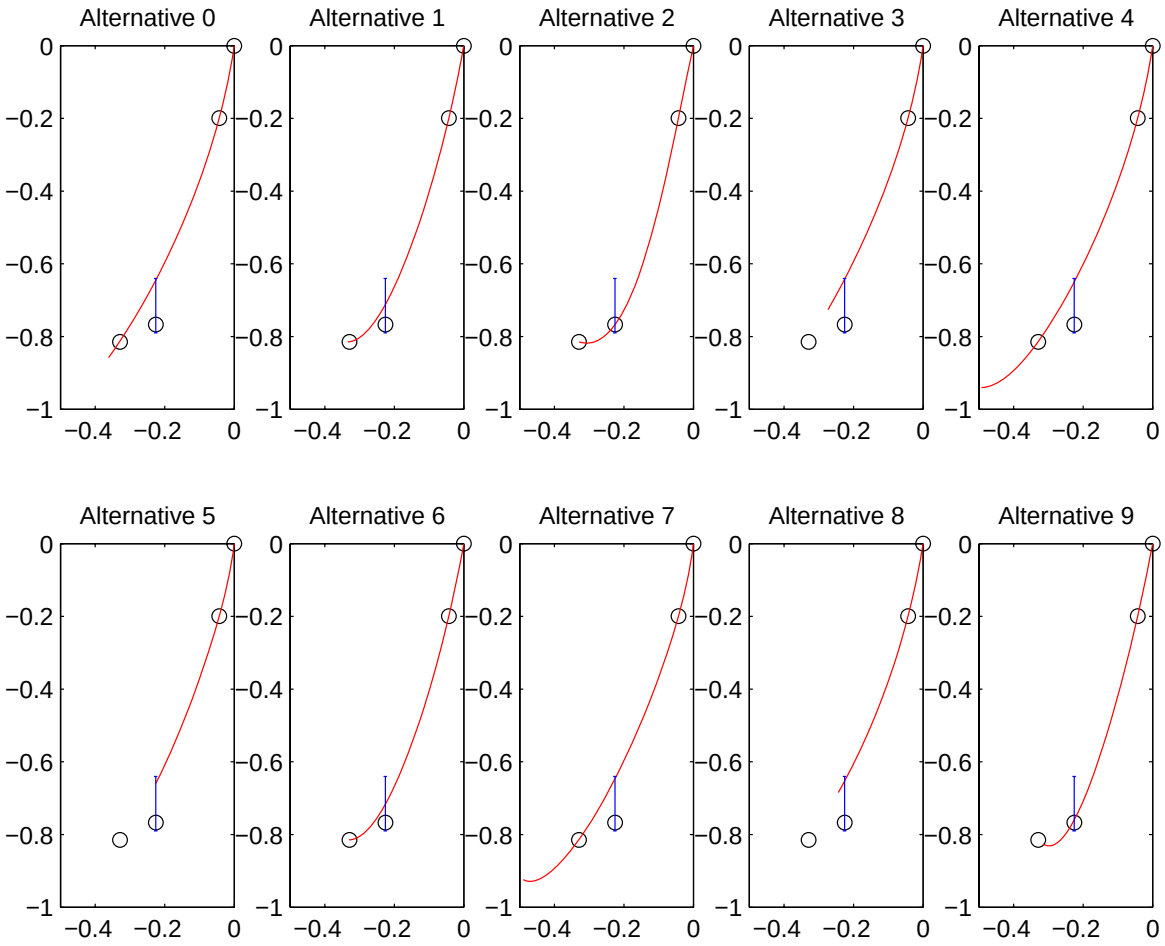



Figure 8.12: Kinematic response for the upper vs. lower crank for the double function generation problem.

## 8.4. Discussion

The main disadvantage when including flexible members (links or joints) in the mechanism is the

↓ Fatigue life reduction.

Among the disadvantages of replacements it can be remarked that:

↓ After links transformation, flexible links are larger than their rigid counterpart, thus they might produce a worse fulfilment of the *allowed space* and *minimal link lengths* constraints (see Figure 8.4).

↓ Links which fully rotate cannot be replaced.

Some advantages of including compliant segments (while replacing kinematic pairs) into traditional mechanisms are well-known. Under the scope of planar linkages, these advantages are:

- ✓ Increase in precision by backlash elimination.
- ✓ Maintenance reduction by elimination of wear and lubrication needs.
- ✓ Reduction of parts compared to rigid designs.
- ✓ Reduction of manufacturing and assembly time costs.
- ✓ Increase in the possibility of miniaturization and one layer manufacturing.

Based on the experience provided by the example presented above, it is worth adding that:

- ✓ Capabilities of non-linear motions similar to that performed by articulated mechanisms.

## 8.5. Chapter conclusion

The first steps for developing a computer tool for conceptual compliant mechanism design was presented. Deliberations about applied forces and moments, energy storage, and bi-stability, were out of the kinematic scope of this work, but they can also be managed by geometrical considerations in the dimensional synthesis solver. Our aim is to study the proper modifications to account for compliant kinematic synthesis tasks, and also, to design the constraints to be incorporated into a fully automated solver for compliant parts replacement. As well as for rigid mechanisms, precision-point synthesis does not assure a good behavior between passing points; therefore, in many cases optimization of kinematic errors must be done. Some advantages, like the low number of variables and the proximity to optimal solutions of the initial guess -mechanism topology with initial dimensions- provided by the precision-position synthesis, may help the convergence of a subsequent optimization.

Further research would incorporate synthesis rules into the designing of non-homogeneous links, initially curved segments, automated segment identification and force-deflection analysis. Specifically related to the latter topic, we are able to do a static analysis of forces and moments after the rigid-dimensional synthesis in order to choose the right pseudo-rigid body models with their proper parameters: characteristics factors, spring constants, etc. as function of load cases.

Part of future research could be the study and comparison of these results with other continuum approaches based on structural optimization.

# Part III

# Closure

# Chapter 9

# Thesis conclusions

A methodology implemented in software that allows us to synthesize planar simple-jointed linkages that respond to kinematic requirements given by the user *starting from scratch*, was developed. Solutions to these kind of inverse problems are far from being unique, either at the structural or the dimensional level.

The main contribution of this thesis is a systematic approach for type synthesis. Three computational tools for type synthesis were developed: a mechanism identifier *Type Adjacency Matrix*, an isomorphism identifier *Diagonally Extended Degree Code computed by Rows*, and a *Synthesis Adjacency Matrix* for identification of functionally different mechanisms and subgraph location. Using these tools it is possible to explore hundreds of potential mechanisms in few minutes and find alternatives which automatically match the given task –without occurrence of repetitions. The enumeration of compliant four-bar mechanisms was validated against the results of the literature. New six-bar-mechanism results were given. The enumeration of mechanisms, as well as the sub-graph search enumeration, is completely exhaustive, deterministic and suitable for the current computational power of modern PC's.

The modular decomposition of each topology was proposed as the key step between the type synthesis stage and a dimensional synthesis stage for applying analytical methods. A Finite Element description of the topology in conjunction with Graph Theory theorems facilitates the identification of the significant dimensions of the mechanism. Several Single-Open Chains modules were used to link the data and unknowns of the problem with the proper Loop-Closure Equations solvers.

These modules allow us to systematize the initial sizing stage where the multiplicity of solutions and free parameters, if they exist, are automatically identified.

An optimality criterion based on *minimal size* subject to several restrictions such as *allowed space*, *singularity of the movement* and *minimum dimensions*, was defined. A simple Genetic Algorithm with a non-stationary penalty strategy for constraint management was used to find the best set of free parameters whose evaluation gives the best satisfaction of an optimality criterion. The algorithm showed robustness to find the best solution using a moderate number of evaluations ($\approx 5000$). However, in some cases it was necessary to tune the right *penalty value for a non-solution occurrence* and re-launch the solver.

A simple path following test was running through the thesis showing the utility for exploring the feasible solution space. Also, problems of synthesis of mechanisms for landing gear retraction, commands of slats, flaps and nozzles of turbines, were solved.

This computational tool pertains to the most primitive stage of design: *Conceptual Design*. It can be used for three tedious activities for which academic and commercial software is scarce: (1) completion of an existent mechanism, (2) redesign of a mechanism, and (3) design of new mechanisms.

It is expected that the presented test problems serve as benchmarks for type and dimensional synthesis of simple-jointed planar linkages.

A software prototype was integrated into a system of analysis of mechanisms and structures by finite elements. The data definition is made in graphic form in the own CAD program of the general system of analysis. After the synthesis task, designers can verify the validity of the proposed solution by means of analysis, and they can even optimize the solution staying within the same topology.

## 9.1. Further research

Several aspects can be improved in order to extend the methodology capabilities.

Among the *input of data*, it is mandatory to allow the definition of (i) continuum tasks; (ii) one *allowed space* for each precision position, more complex polygonal area for pivots location and obstacles; (iii) kinematic tasks for flexible segments; (iv) more complex planar joints like curved-slider[1].

At the *type synthesis* level, the atlas of kinematic chains can be extended to 10, 12 and 14 links, and from these it is further possible to generate atlases for multiple joints. In this way pin-slider joints can be incorporated. An atlas of mechanisms with multiple joints may serve also to generate an atlas of flexible mechanisms where flexible members are only binary. All constraints for the atlas generation may be expressed in terms of Graph Theory formalism. Several researchers had shown the applicability of GT to the enumeration of *cam-* and *gear-linkages*, *gear-trains* and *tridimensional mechanisms*.

The interactivity with this solver can be improved by developing a more user-friendly interface than the primitive script.

The *decomposition algorithm* can be further refined by considering more paths of decomposition than the one defined by the presented loop-by-loop approach.

Developments in *dimensional synthesis* will be dependent on the available atlas of mechanisms. For example, considering topologies with multiple joints (e.g., straight- and curved-pin-slider joints have links with null size), the proper SOC modules and solvers need to be developed. The Precision Point Methods are also available to be computationally implemented for synthesizing cam- and gear-linkages [SE84].

For designing any *optimal mechanism* it is necessary to take into account multiple objectives and constraints. The most useful objectives missing in this thesis were (i) the minimization of the *kinematic error in the continuum task*, (ii) the maximization of the *mechanical advantage*, and (iii) the optimization of the *transmission angle*. With respect to constraints, the most relevant are the (i) elimination of *branch* and *circuit* defects and (ii) the prescription of fully rotatable driver-cranks which finds application in mechanisms for mass production machinery (e.g. textile and food-packaging).

With respect to the user-friendliness of the developed initial sizing software,

[1] All of them are already supported by the analysis program Samcef Field–Implicit Non-Linear, but the capability in the Oofelie solver was not yet developed.

much work will be focused on interactive definitions of bounds variables, drawing of the desired and generated curves, and the minimization of the number of sessions. Currently, three successive interactive sessions are needed: one for defining the kinematic problem from which the type synthesis solver is launched; one for each topology found for which the initial sizing solver is launched; and a final session for animating the solution.

Finally, an ambitious goal is the automatic comparison of all the mechanism solutions obtained from the synthesis stages. Two test problems solved by hand may be taken to validate the method, the design of a *variable-stroke engine* of Freudenstein and Maki [CP05, FM83] and the *casement window operator linkage* [SE84]. Both tests require atlases of pin-slider mechanisms.

# Appendix A

# Resumen extendido (extended abstract in Spanish)

El objetivo del presente trabajo de investigación es el estudio y desarrollo de técnicas para el diseño y síntesis de mecanismos de eslabonamientos planos. La síntesis de mecanismos consiste en hallar el mecanismo adecuado para un movimiento dado. En particular, esta tesis trata el problema de síntesis de mecanismos partiendo desde las especificaciones o requerimientos iniciales de diseño, o sea, partiendo de cero. Por lo tanto, es necesario determinar el número, el tipo de componentes y la conectividad entre ellos (síntesis de tipo); y luego calcular las dimensiones de los componentes, posiciones de pivotes, y los parámetros de control de los pares cinemáticos de entrada del movimiento (síntesis dimensional).

En esta tesis se trata la *síntesis cinemática de posición*, cuyo problema consiste en determinar las dimensiones de un mecanismo que satisfaga un conjunto de *desplazamientos* y *rotaciones* deseados en ciertos puntos de un mecanismo y para ciertos instantes de simultaneidad. Esta especificación se denomina tarea cinemática. El espacio permitido -para el mecanismo solución y el desarrollo de la tarea- es un requerimiento muy común que restringe las soluciones a obtener. El problema es altamente no lineal y al incluir la selección de la topología a dimensionar constituye además un problema discreto de complejidad combinatoria.

Para resolver este difícil problema se propone utilizar una representación del mecanismo basada en el Método de los Elementos Finitos y en la Teoría de Grafos, logrando preservar y unificar ambas representaciones para integrar la síntesis a sus posteriores etapas de análisis detallado y optimización del mecanismo. Se presenta la teoría e implementación de un resolvedor de síntesis que consta de dos etapas: un **generador** de alternativas que resuelve la síntesis de tipo y un **evaluador** que resuelve la síntesis dimensional, utilizando ecuaciones exactas, para cada alternativa provista por el generador.

Como resultado final de la aplicación de esta técnica se obtiene un listado de alternativas que constituyen buenas condiciones iniciales para su posterior optimización mediante el uso de técnicas de gradientes ya disponibles en software comercial.

Durante el desarrollo de la tesis se muestran diversos ejemplos de prueba y validación mostrando la capacidad de la herramienta inventiva desarrollada. Sin embargo, en este resumen en español sólo se detallarán los resultados para un problema de ejemplo: un generador de trayectoria con tiempo prescripto.

## A.1. Conceptos introductorios

Los mecanismos son dispositivos mecánicos de uso intensivo en maquinaria agrícola, industrial, automotríz, aeronáutica, etc., y actualmente, tiene un gran auge su uso en los sistemas micro-electromecánicos (MEMS).

Desde tiempos muy antigüos la intuición técnica y actividad creativa del diseñador han permitido concretar una innumerable cantidad de mecanismos que *transmiten potencia mientras realizan la conversión de movimientos desde unos cuerpos a otros*, en forma repetitiva, en reemplazo y/o en multiplicación del trabajo humano o animal o proveniente de otra fuente de potencia, por ejemplo, del agua de un río, del viento, de un proceso químico, etc.

El diseño de mecanismos es una actividad en la cual el ingeniero se enfrenta a la difícil tarea de aplicar su experiencia e ingeniosidad para combinar una amplia variedad de elementos mecánicos con distintas funciones, buscando satisfacer requerimientos funcionales y severas restricciones de espacio. El proceso de diseño es naturalmente cíclico e iterativo. Entre las primeras etapas del diseño, se utilizan iterativamente la *síntesis* y el *análisis* con el objetivo de alcanzar una –y en lo posible "la mejor"– solución válida que satisfaga los requerimientos. La automatización de estas etapas mediante el auxilio de técnicas computacionales debe conducir al diseñador a: (i) sistematizar la definición del problema y la carga de datos entre las etapas de diseño, (ii) reducir los tiempos de cálculo, y (iii) facilitar la interpretación de resultados, y consecuentemente, permitir *enfocar su labor intelectual y creativa sólo en aquellas decisiones de vital importancia para el éxito del diseño.*

## A.2. Descripción del problema

Debido al desconocimiento y/o complejidad de la herramientas de síntesis, muchos ingenieros diseñan mecanismos mediante el empleo de sucesivos ciclos de análisis y optimizaciones. Aún después de muchos ciclos, en muchos casos casos acontece que no se llega a mejorar el diseño y no se logra el comportamiento especificado. Es en estas situaciones en que se debe recurrir a las etapas tempranas del diseño (rediseño) y estudiar mecanismos con otras configuraciones, revisar otros inventos relacionados, etc.

Hoy en día, el *proceso de diseño computacional de mecanismos* puede resumirse en cuatro etapas:

1. **Identificación de los requerimientos del problema**;
   - Movimiento deseado;
   - Espacio permitido;
   - Mínimo consumo de potencia, y otros;
2. **Síntesis**

   2-I) Síntesis topológica o estructural:

   - *Síntesis de tipo*: Combinación de tipos de mecanismos para la función deseada: levas, trenes de engranajes, eslabonamientos, etc.;

- *Síntesis de Número*: Determinación del número, tipo y conectividad de las partes componentes, para el grado de libertad requerido;

2-II) Síntesis dimensional: Cálculo de las dimensiones significativas de cada miembro componente.

3. **Análisis**;

4. **Optimización, diseño detallado, ensayo y experimentación**.

Las primeras etapas del proceso pertenecen al diseño conceptual y tienen gran importancia, debido a que el tiempo y los costos de diseño para el análisis, la optimización y el ensayo o experimentación son fuertemente dependientes del concepto elegido.

La etapa de síntesis consiste en hallar el mecanismo adecuado para requerimientos dados. Aquí aparece otra dificultad: la multiplicidad de alternativas y la falta de sistematización para modelar matemáticamente los requerimientos y el criterio de selección. Esta necesidad ocupará la mayor atención de la tesis debido a que *la enumeración y selección de topologías para una tarea dada es una actividad que se conoce como un problema abierto del diseño de mecanismos.*

La síntesis de tipo, es atribuida al alemán Franz Reuleaux (1829-1905) quien plantea el problema de la elección del "tipo" de mecanismos como primer instancia de diseño. Una gran cantidad de combinaciones de tipos de mecanismos pueden "potencialmente" satisfacer un movimiento: mecanismos de levas, de poleas y correas, de engranajes, de eslabonamientos, etc. En esta tesis no se resuelve el problema de combinar mecanismos para una especificación dada, conociendo la capacidad de los *eslabonamientos planos* para transformar movimientos, se trabajará en este tipo prestablecido. Aún así de restringido el espacio de soluciones, una gran cantidad de eslabonamientos pueden satisfacer un mismo movimiento o tarea; unos con menor error que los otros, con menor número de partes componentes, etc. La síntesis de número consiste en generar alternativas factibles que satisfagan los requerimientos estructurales de las especificaciones[1].

La síntesis cinemática de posición consiste en hallar mecanismos para *tareas cinemáticas* de *guiado de cuerpos* (ver Figura A.1), *seguidores de trayectorias* y de *generación de funciones* entre dos o más cuerpos (coordinación múltiple). Para una simple barra rígida restringida a rotar en uno de sus puntos, la posición se especifica en términos de ecuaciones trascendentes. Consecuentemente, la síntesis cinemática de posición es un problema no lineal en la dimensión de los eslabones (incógnitas).

El mecanismo de un grado de libertad más simple es el de cuatro barras articuladas por pares rotoidales. En un punto de su *eslabón acoplador* (el eslabón no articulado a tierra) es capaz de desarrollar curvas planas polinómicas de hasta grado seis. De allí su intensiva utilidad pero también su dificultad para el diseño inverso: dada la curva, obtener el mecanismo. Mecanismos con mayor número de eslabones permiten realizar tareas aún más complejas.

Los movimientos o *tarea deseada* pueden ser de variada complejidad (prescripción de posiciones, velocidades, aceleraciones, esfuerzos a transmitir, etc.). Generalmente y en primera instancia, es coherente resolver la *síntesis cinemática de posición* bajo

[1]En las publicaciones actuales sobre eslabonamientos es común el empleo de la denominación *síntesis de tipo* para la etapa completa de síntesis topológica incluyendo la síntesis de número.

la hipótesis de *mecanismo rígido* donde se considera que sus miembros componentes no tienen masas ni propiedades de inercia [HD64, SE84, ES97, Nor95]. Para la síntesis cinemática de posición, el movimiento deseado se especifica ya sea en forma continua o como una *secuencia finita de posiciones deseadas*. Éstas últimas suelen

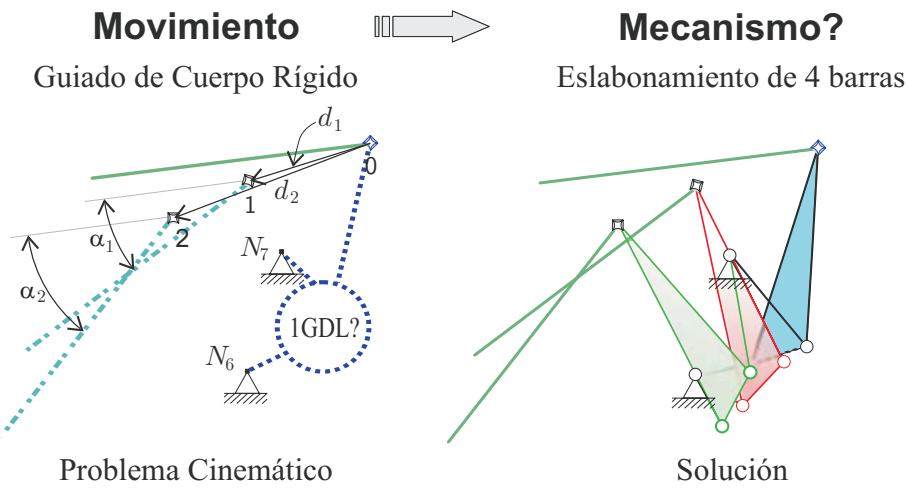



Figura A.1: Síntesis dimensional de posición para un eslabonamiento.

denominarse, posiciones separadas finitamente, posiciones precisas, o también puntos precisos, y la técnica de síntesis dimensional Método de Posiciones Precisas, Método de Puntos Precisos o Síntesis de Burmester. Los métodos de síntesis dimensional se dividen en tres categorías:

- *Gráfica o geométrica*,
- *Analítica, exacta o de Burmester*, y
- *Aproximada*.

Las dos primeras se aplican para posiciones precisas y la solución puede ser única, múltiple o puede no existir. Las expresiones analíticas se obtienen mediante *ecuaciones de lazos cerrados*, las cuales dependiendo del número de posiciones a resolver pueden conducir a sistemas de ecuaciones lineales o no lineales. Para alguno casos no lineales, las expresiones en números complejos facilitan hallar expresiones cerradas. Una solución hallada con estas expresiones es meramente geométrica y puede acontecer (i) que sea cinemáticamente inválida y el mecanismo se bloquee en alguna posición determinada (la inicial inclusive), (ii) que se mueva por otros puntos distintos de los especificados, o en el mejor de los casos, (iii) que el mecanismo solución pase por los puntos especificados pero que se aleje considerablemente del objetivo en porciones intermedias de la tarea. Tal solución requeriría de una optimización posterior.

La *síntesis aproximada* se aplica para casos en que la cantidad de puntos especificados es muy grande, o cuando el comportamiento entre puntos de la tarea es muy importante. La técnica requiere de varios análisis cinemáticos, al menos uno por iteración, donde se compara la *tarea generada con la tarea deseada*. Entre ambas tareas se define un índice de error que debe minimizarse.

En general, una vez que el mecanismo satisface la tarea prescripta, se busca que éste cumpla con otros requerimientos cinemáticos y dinámicos.

## A.3. Hipótesis simplificativas

En el desarrollo de esta tesis se tendrán en cuenta las siguientes hipótesis:

- Se considerará la síntesis de eslabonamientos planos.
- El movimiento se discretiza en 3 ó 4 posiciones precisas.
- Bajo la hipótesis de rigidez, las principales características del mecanismo que pueden diseñarse son:

  **Tarea**: satisfacer la relación *movimiento/s de entrada* vs. *movimiento/s de salida* con el menor error de posible. Por cada miembro se desea minimizar el error entre el *movimiento deseado* y el *movimiento generado*.

  **Movimiento de Entrada**: cuando sólo se especifica el movimiento de salida, parte del problema consiste en determinar los correspondientes estados del resto de los miembros como posibles entradas.

  **Espacio**: el mecanismo, en su posición inicial y durante su movimiento, debe hallarse dentro de un área prescripta.

  **Ángulo de transmisión**: es inminente que el mecanismo no se bloquee durante su movimiento y se halle lo más alejado posible de esa situación.

  **Peso**: con la hipótesis mencionada, el peso queda relacionado con el tamaño de los eslabones y por lo tanto se desea que tengan un tamaño mínimo.

## Antecedentes en herramientas computacionales

Es notable cómo los hechos históricos muestran que las herramientas computacionales se han desarrollado en orden inverso a las etapas del proceso de diseño. Primero se implementó el *análisis* por computador y su empleo como herramienta para agilizar el diseño a prueba y error; luego aparecieron los programas de *síntesis* [HD64, SE84, ES97, Nor95, OER87]. Más tarde, los programas capaces de clasificar y combinar automáticamente mecanismos desde los movimientos deseados, por medio de combinación exhaustiva [CK99, YO05] o basada en reglas dadas por expertos.

En lo que respecta a los programas para el *análisis*, fueron los primeros que se han traducido a sistemas de diseño e ingeniería asistida por computadora de uso industrial, más difundidos como sistemas CAD/CAE (Computer-Aided Design/Computer-Aided Engineering). En cuanto a las técnicas numéricas para el análisis, vale la pena mencionar que en las últimas dos décadas, el modelado y análisis de mecanismos utilizando el Método de los Elementos Finitos ha avanzado mucho lográndose gran exactitud en las simulaciones [WN03]. Cabe mencionar que en la Argentina y desde hace veinte años, el tópico de análisis de mecanismos flexibles es una línea de investigación muy fructífera del grupo CIMEC [Car89, GC01, Len06]. Existen, además, programas comerciales eficientes para la *optimización* de mecanismos, se puede mencionar por ejemplo el software SAMCEF BOSS Quattro que permite iterar con el módulo de análisis no lineal SAMCEF Mecano [Car89, SAM07].

Entre los pioneros de la *síntesis* computacional de mecanismos de mediados del siglo pasado se tienen a Ferdinand Freudenstein (1926-2006) [FS59] y George N. Sandor (1912-1996) [San59] en los Estados Unidos y a N. I. Leviskii y K. K. Shakcvazian

en la Union Soviética [Ang97]. Poco después se destacan Richard Hartemberg (1907-1997) y Jacques Denavit [HD64], Oene Bottema (1901-1992), Kurt Hain (1908-1995), Bernard Roth, Allen S. Hall Jr. [Hal61], en las últimas décadas, Arthur G. Erdman [ES97, SE84], Lung-Wen Tsai (1945-2002) [Tsa01], Robert L. Norton [Nor95], Hon-Sen Yan [Yan98], y J. Michael Mc Carthy [McC00], entre otros. En las investigaciones de estos profesores, puede verse cómo la *cinemática*, la *geometría*, el *álgebra*, las *matemáticas discretas* y el *análisis combinatorio*, los *métodos numéricos* y las *técnicas de optimización*, han mostrado su intersección con la *ciencia de los mecanismos* impartiéndole un carácter multidisciplinario. La ciencia adquiere su verdadera amplitud de nuestros días si consideramos a los mecanismos como partes interactuantes en sistemas más complejos, por ejemplo, sistemas electrónicos, de control, robóticos y mecatrónicos.

Desde hace más de tres décadas, existen programas académicos que resuelven satisfactoriamente algunos problemas de síntesis dimensional. Refiriéndonos a mecanismos planos, entre los pioneros se puede mencionar a KINSYN [Kau71], LINCAGES [EG77], y RECSYN [WS81]. Luego surgió SYNMECH [PK97], y posteriormente otros que actualmente continúan en desarrollo tales como LINCAGES 2000 [Erd, YEB02], SAM [Ran07], TADSOL [CKv], SYNTHETICA [McC], WATT [DK07], y SyMech [Coo]. El uso de estos programas requiere una amplia experiencia como cinematicista, lo cual es simplificado por el desarrollo de interfaces de usuario amigables. Algunos de estos programas poseen capacidad de exportación a formatos de archivos industriales para el análisis posterior de las soluciones. Los programas comerciales de CAD/CAE más difundidos en la industria carecen de módulos propios para síntesis de mecanismos. Pocos han logrado la integración, un caso concreto es el de SyMech como "Add-In" de Pro/E [CO02], el estudio de factibilidad de incorporación de SYNMECH en ADAMS [PVW97], y recientemente, el método de Kinzel *et al.* utilizando el modo de "sketching" y el manejo de restricciones de los CADs comerciales, en una técnica denominada Programación de Restricciones Geométricas (Geometric Constraint Programming) [KSP06]. Puede afirmarse que la categoría *Diseño de Eslabonamientos Asistido por Computador ("Computer-Aided Linkage Design")* es relativamente joven[1].

Cabe destacar que la bibliografía relacionada a tópicos de síntesis de tipo era hasta hace poco tiempo escasa y relegada a comunicaciones cortas en revistas especializadas. Curiosamente un libro con capítulos muy valiosos dedicados a este campo fue publicado en idioma español por el profesor Justo Nieto Nieto [Nie77] en 1977. Dos décadas después fueron publicados los libros de los profesores Yan [Yan98] y Tsai [Tsa01] con un enfoque más moderno, ambos provistos de algoritmos y conceptos que han sido empleados en la presente tesis.

## A.4. Metodologías actuales y estado del arte

Lung-Wen Tsai [Tsa01] propone una metodología sistemática para el diseño de mecanismos y la resume en lo siguiente:

1. "Identificar los requerimientos funcionales, basados en los requerimientos del

[1] Sesión Especial sobre *Computer-Aided Linkage Design* en archivos del *DECT'02 ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Montreal, Canadá, 29 de septiembre al 2 de octubre de 2002

comitente, para la clase de mecanismo de interés.

2. Determinar la naturaleza del movimiento (plano, esférico, o espacial), grados de libertad, tipo, y complejidad de los mecanismos.

3. Identificar las *características estructurales* asociadas con algunos de los requerimientos funcionales.

4. **Enumerar** todas las estructuras cinemáticas posibles que satisfacen las características estructurales usando Teoría de Grafos y análisis combinatorio.

5. Esquematizar el mecanismo correspondiente y **evaluar** cada uno de ellos cualitativamente en términos de su capacidad de satisfacer los requerimientos funcionales restantes. Este proceso conduce a obtener un conjunto de mecanismos factibles.

6. **Seleccionar** el mecanismo más prometedor para la síntesis dimensional, optimización del diseño, simulación por computadora, demostración del prototipo, y documentación.

7. Pasar a la fase de producción."

*"Notamos que la metodología consiste de dos motores: el* ***generador*** *y el* ***evaluador****. Algunos de los requerimientos funcionales se transforman en características estructurales y se incorporan al generador como reglas de enumeración. El generador enumera todas las soluciones posibles usando Teoría de Grafos y análisis combinatorio. Los requerimientos funcionales restantes se incorporan en el evaluador como criterios de evaluación para la selección de conceptos. Ello resulta en una clase de mecanismos admisibles. Finalmente, el candidato más prometedor se elige para el diseño del producto. El proceso puede iterarse varias veces hasta llegar al producto final"* [Tsa01].

Tsai basó su metodología de enumeración sistemática en el concepto de Freudenstein y Maki, de "separación" de la estructura cinemática (para generar alternativas) de los requerimientos funcionales (para evaluar las alternativas generadas) [FM79]. Y también remarcó que *"mientras más requerimientos funcionales sean traducidos e incorporados en el generador, menor trabajo se necesitará en el evaluador. Sin embargo, esto puede conducir a que el generador se torne muy complejo de desarrollar. Generalmente, si un requerimiento funcional puede escribirse en una forma matemática, éste debería incluirse en el generador"* [Tsa98, Tsa01].

Sardain estableció que no se puede juzgar la optimalidad de una topología hasta que se consideren sus dimensiones y lo estudia en un caso de ejemplo en la referencia [Sar97]. Esto significa que el evaluador no sólo debe incluir cuestiones topológicas, sino que también debe incluir las dimensionales y todas aquellas mediciones del comportamiento que puedan construirse desde las mismas.

Existen varias metodologías para la enumeración de topologías, pero pocas estan integradas a la síntesis dimensional. En el método propuesto por Tsai algunos aspectos que él resolvió manualmente podrían ser implementados computacionalmente. Sin embargo aplicó su método con éxito en la enumeración de trenes de engranajes, manipuladores paralelos y mecanismos de eslabones [Tsa87, Tsa98, Tsa01]. Recientemente, Liu y Mc Phee argumentaron que la etapa de síntesis de tipo es un problema

abierto y hacen referencia a una frase de Arthur Erdman de 1995: "La etapa más crítica en todo proceso de diseño... es elegir la mejor topología para una tarea dada" [LM05]. Estos investigadores aplican Algoritmos Genéticos (AGs) para encontrar una topología que satisface ciertos requerimientos topológicos. También utilizando AGs, en una línea similar, Sedlaczek *et al.* [SGE05] proponen incorporar la síntesis dimensional en la evaluación de cada topología.

En un intento de automatizar la propuesta de Tsai, desde 2004, Pucheta y Cardona [PC04, PC05a, PC05b, PC06, PC07] proponen (i) resolver la síntesis de tipo incluyendo las partes prescriptas en la enumeración de mecanismos a través del uso de Teoría de Grafos y (ii) la posterior utilización de AGs en la síntesis dimensional analítica de cada topología. Un prototipo, desarrollado en colaboración con la compañía SAMTECH [SAM07] permitió a estos investigadores usar la interfaz del CAD Samcef Field para el pre y postproceso de los problemas de síntesis para continuar luego con las etapas de diseño detallado y optimización dentro del mismo entorno [CCSP07].

En el 2005, Chen y Pai [CP05, CFH+06] presentaron un enfoque metodológico exhaustivo para resolver la síntesis de tipo desde una clasificación de las especificaciones de diseño y resuelven un problema propuesto por Freudenstein y Maki.

# Motivación

El problema de síntesis es un problema de complejidad combinatoria. La velocidad actual de los procesadores para ejecutar *casi en tiempo real* complejos algoritmos combinatorios es una de las principales motivaciones para orientar esta tesis hacia la enumeración exhaustiva, tanto de las soluciones de la síntesis de tipo como de la dimensional (combinaciones de la múltiples soluciones geométricas). Hace diez años atrás hubiera sido impensable ejecutar estos algoritmos sin desviar la atención del usuario del problema de diseño.

# Objetivo

El **objetivo computacional** es el desarrollo de aplicaciones para el diseño "desde el inicio", o sea, partiendo de las especificaciones y restricciones del movimiento impuestas sobre las partes existentes que se desean mover, y lograr enumerar todos los posibles eslabonamientos hasta cierta complejidad, en forma ordenada y sin repetición.

La selección desde una enumeración ordenada de todas las posibles soluciones constituye una tarea de decisión muy compleja. Aunque es también un aspecto automatizable del proceso de diseño, en esta tesis se asume que la selección se realiza manualmente.

Con la revisión hecha arriba acerca de los programas existentes, fue de especial interés incorporar los resultados de la tesis a un programa capaz de interactuar con un sistema de ingeniería asistido por computadora CAD/CAE cuyo uso minimice los tiempos de capacitación del diseñador y además minimice el tiempo insumido en entregar los resultados.

## A.5. Automatización propuesta

Para cumplir los objetivos mencionados se propone una representación conjunta de los mecanismos y de las partes prescriptas basada en Teoría de Grafos y en el Método de los Elementos Finitos. Se propone hacer uso de una u otra representación convenientemente dependiendo del algoritmo que se desea implementar.

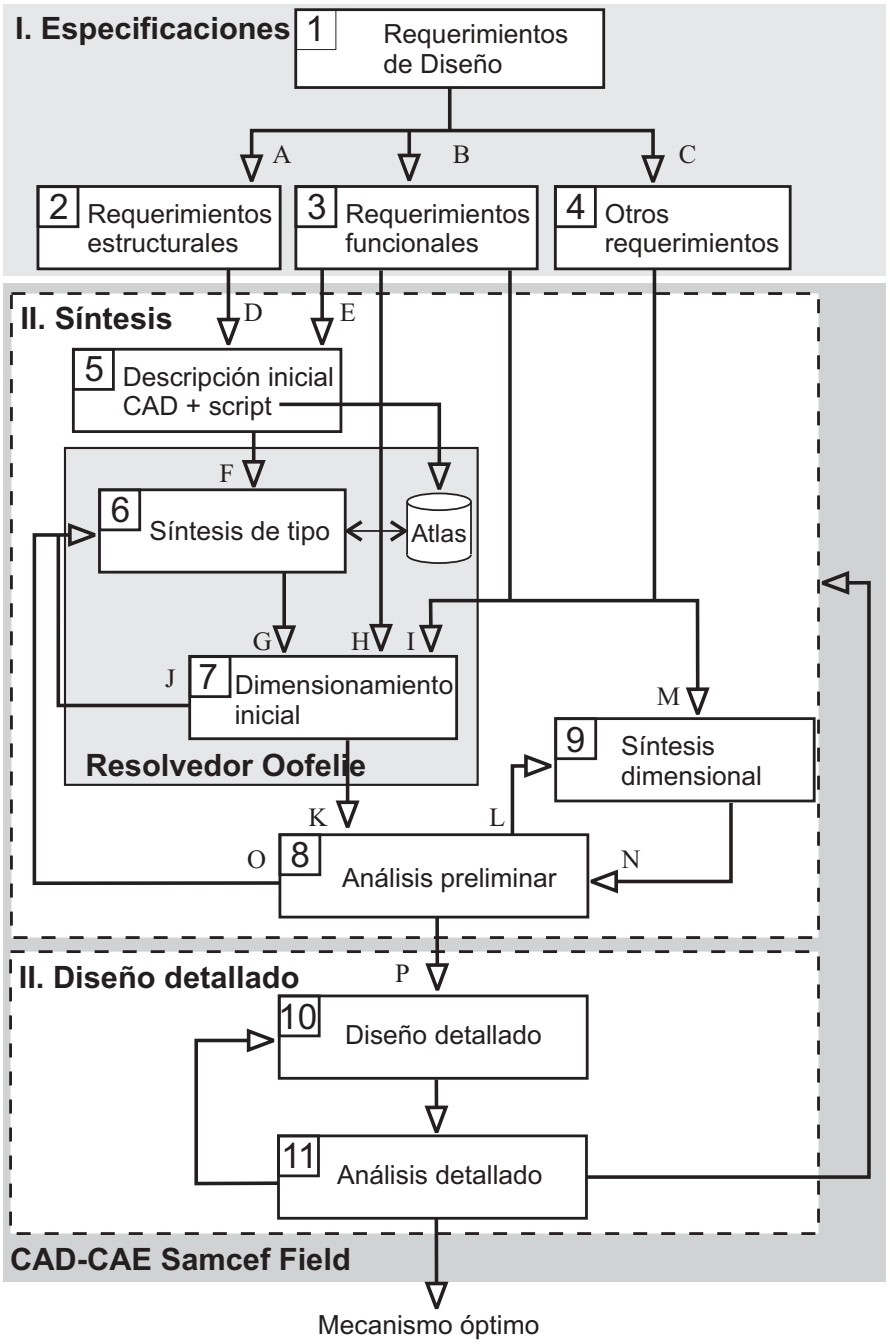



Figura A.2: Ubicación del módulo de síntesis en el diseño computacional de mecanismos.

En la Figura A.2 puede verse un diagrama en bloques de las etapas de diseño de un mecanismo utilizando herramientas computacionales, y especialmente se destaca al módulo de síntesis, el cual se presenta con más detalle en la Figura A.3.

Para la *Síntesis de **T**ipo* se propone [PC07]:

**T1**: Representar el submecanismo o partes prescriptas y la tarea cinemática utilizando un CAD para elementos finitos.

**T2**: Convertir el problema cinemático en un grafo denominado *Grafo Inicial*, lo cual da un modelo matemático para las partes prescriptas y restricciones estructurales.

**T3**: Generar varios atlas de mecanismos con diferentes tipos de eslabones y uniones utilizando una codificación unívoca y eficiente.

**T4**: Resolver la síntesis de tipo utilizando la búsqueda del *Grafo Inicial* como subgrafo de cada mecanismo del atlas seleccionado e identificando todas las ocurrencias no isomorfas.

**T5**: Realizar un diagrama esquemático para cada alternativa.

Para la *Síntesis **D**imensional* se propone utilizar métodos geométricos exactos (ya existentes) para dar dimensiones a cada topología abstracta resultante de la *síntesis de tipo* desarrollando computacionalmente estos pasos:

**D1**: Discretizar la tarea en posiciones precisas.

**D2**: Descomponer la topología en cadenas abiertas [PC06].

**D3**: Buscar el ordenamiento óptimo de las cadenas [PC06].

**D4**: Resolver analíticamente las cadenas utilizando números complejos para representar los eslabones [HD64, SE84].

**D5**: Reensamblar las cadenas para reconstruir la topología.

**D6**: Calificar el cumplimiento de las restricciones.

Los pasos están íntimamente vinculados dado que la resolución de una cadena es dependiente de las restricciones que le impone una previa [SE84, ES97]. La resolución resulta ser jerárquica y las múltiples soluciones hacen que el reensamble también lo sea. En las etapas **D4**-**D6** se utilizan Algoritmos Genéticos para los casos en que hay parámetros libres [PC05a].

## A.6. Síntesis de tipo

Se desarrollan dos herramientas computacionales, una representación de grafo/matriz del mecanismo y un identificador de mecanismos, para cubrir dos aspectos fundamentales de la síntesis topológica:

1. la enumeración, identificación, almacenamiento y restauración de mecanismos, y

2. la acción de evitar las ocurrencias isomorfas del grafo inicial dentro de cada mecanismo del atlas.

Aunque el primer desarrollo es una ligera variación del Código de Grado ("Degree Code") existente [TL93], se presenta por primera vez una discusión sobre las modificaciones necesarias en el mismo para tener en cuenta distintos tipos de eslabones y uniones.

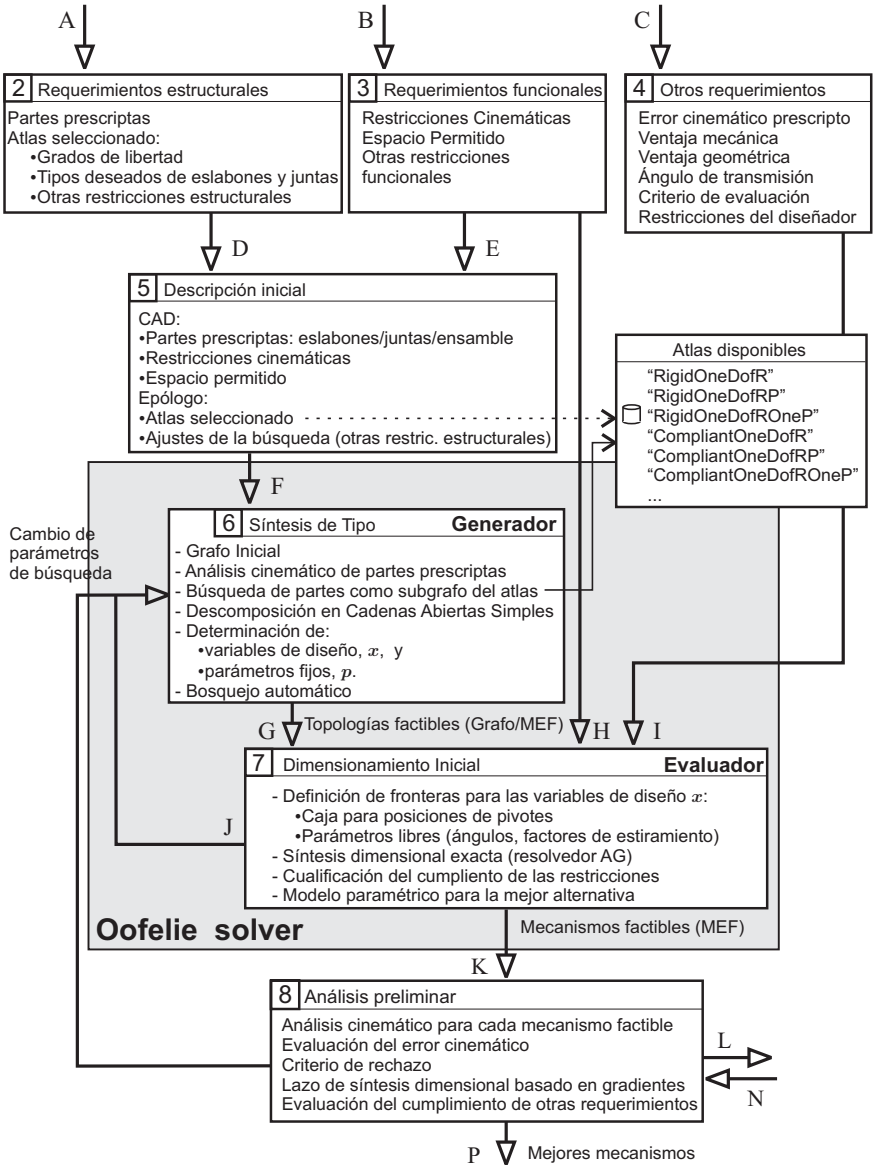



Figura A.3: Detalles de los procedimientos de síntesis.

### Representación de mecanismos utilizando grafos

El *grafo* $G(V, E)$ de una cadena cinemática se obtiene representando cada *eslabón* por un *vértice* $v_i$ y cada *unión cinemática* por una *arista* $e_{ij}$ que conecta los vértices correspondientes $\{v_i, v_j\}$. El tamaño del conjunto de vértices se denota por $n = |V|$ y el tamaño del conjunto de aristas por $j = |E|$. El conjunto de vértices sin etiquetas $V = \{v_0, v_1, ..., v_{n-1}\}$ se numera en base cero, esto es, $V = \{0, 1, ..., n-1\}$.

Adicionalmente al grafo $G$, se definen dos niveles de información necesarios para definir un mecanismo:

- ***Etiquetas*** identificando el *significado funcional de cada eslabón o unión.*
- ***Tipos*** identificando el *tipo estructural (ó comportamiento físico) de cada eslabón o unión.*

Una vez que las etiquetas y los tipos son asignados sobre el grafo, la *estructura*

*topológica de un mecanismo* queda completamente definida. El modelo del mecanismo se completa con la ayuda de una representación matricial.

Un grafo tiene varias representaciones matriciales que expresan cómo los vértices están conectados mediante aristas. Una representación es la *Matriz de Adyacencia* vértice a vértice. La matriz de adyacencia $A$ de un grafo $G$ es una matriz de tamaño $n$ por $n$ en la cual la entrada $a_{ij}$ es el número de aristas en $G$ con vértices extremos $\{v_i, v_j\}$ y $a_{ii} = 0$.

Una *Matriz de Adyacencia de Tipos* $\boldsymbol{T}$ se define como sigue:

$$T_{ii}(v_i) = \begin{cases} 0 & \text{si } v_i \text{ es la fundación,} \\ 1 & \text{si } v_i \text{ es un eslabón rígido,} \\ 2 & \text{si } v_i \text{ es un eslabón flexible,} \end{cases}$$

$$T_{ij}(e_{ij})_{i<j} = \begin{cases} 0 & \text{sin conección,} \\ 1 & \text{si } e_{ij} \text{ es una unión rotoidal,} \\ 2 & \text{si } e_{ij} \text{ es una unión prismática,} \\ 3 & \text{si } e_{ij} \text{ es una unión flexible,} \\ 4 & \text{si } e_{ij} \text{ es una unión fija.} \end{cases} \tag{A.1}$$

Nótese que esta definición se puede extender fácilmente a un mayor número de tipos de eslabones y uniones. Además, las entradas se definen intencionalmente como *números enteros positivos* en acuerdo con el identificador de isomorfismos que se presenta a continuación.

### Detección de isomorfismos

Las permutaciones de las etiquetas cambian la matriz de adyacencia $A$. Un reetiquetado sobre el conjunto de vértices etiquetados $\mathcal{V}(V)$ es equivalente a la transformación congruente $A^p = PAP^T$ sobre la matriz de adyacencia, donde la matriz $P$ es un reordenamiento de filas y columnas de la matriz identidad por medio un vector de índices permutados $p$.

*Dos grafos son isomorfos si existe una biyección uno a uno desde el conjunto de vértices de un grafo hacia el otro y las aristas preservan incidencia. Además, dos grafos son isomorfos si comparten la misma matriz de adyacencia.*

Basado en este hecho, el isomorfismo puede detectarse comparando las matrices de adyacencias, realizando $n!$ permutaciones. Esto se vuelve rapidamente infactible conforme crece el número de vértices. Otros identificadores de isomorfismos permiten reducir esta complejidad de $\mathcal{O}(n!)$. Cualquier identificador debe gozar las propiedades de *unicidad*, *eficiencia* y *decodificabilidad*. Los identificadores basados en códigos gozan de estas propiedades. Éstos consisten de un certificado $C$ y un grupo de permutaciones $\Pi$ para las cuales se calcula el certificado, se guarda el certificado de mayor (ó menor) valor $C_{\text{máx}}$, de modo que:

1. $C_{\text{máx}}$ es *único*: Dos grafos etiquetados coloreados con el mismo $C_{\text{máx}}$ son isomorfos:
$$C_{\text{máx}}(G_1) = C_{\text{máx}}(G_2) \Leftrightarrow G_1 \cong G_2;$$
2. $C_{\text{máx}}$ es *eficiente*: El costo computacional crece en un orden menor que factorial.

3. $C_{\text{máx}}$ es *decodificable*: Dado el $C_{\text{máx}}$, se puede reconstruir el grafo.

Por ejemplo, Tang y Liu diseñaron el *Código de Grado* ("Degree Code"), donde el certificado es una cadena binaria obtenida de concatenar, fila por fila, las entradas de la submatriz diagonal superior de $A$ excluyendo la diagonal, y las permutaciones se realizan entre los vértices de igual grado.

En forma similar, en esta tesis se diseñó un identificador coherente con la definición (A.1), denominado *Código de Grado Extendido a la Diagonal*, $DC_b^d$, donde se incluye la diagonal y una base adecuada de codificación $b$. Una versión mejorada es el calculado por filas $DC_b^r$ en donde el certificado es un vector conteniendo en cada entrada la fila correspondiente codificada en base $b$. Un ejemplo de certificado es el siguiente:

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

Las comparaciones entre códigos para obtener el Código de Grado se hacen en orden lexicográfico: dados $C = \{a_0, a_1, \ldots, a_{n-1}\}$ y $C' = \{b_0, b_1, \ldots, b_{n-1}\}$, $C' > (<)\ C$ si $b_k > (<)\ a_k$, donde $k$ es el primer índice en $\{0, 1, \ldots, n-1\}$ para el cual $a_k \neq b_k$. En el ejemplo de arriba, el Código de Grado por filas se obtiene para la permutación de $A''$ con $p = \{3, 2, 0, 1\}$:

$$DC_3^r(\boldsymbol{T}) = C_3^r(p(\boldsymbol{T})) = \begin{bmatrix} 48 \\ 10 \\ 4 \\ 0 \end{bmatrix}$$

Nótese que se requiere de sólo de una comparación ($48 > 12$) para determinar que $DC_3^r(\boldsymbol{T}) > C_3^r(\boldsymbol{T})$.

Este identificador se utiliza para enumerar atlas de mecanismos, y para localizar las ocurrencias no isomorfas de los subgrafos de las partes iniciales en cada grafo tomado del atlas.

### A.6.1. Enumeración de atlas de mecanismos

Para enumerar mecanismos se sigue una metodología similar a la presentada por Murphy *et al.* [MMH96], donde se realiza la *asignación de tipos de eslabones y uniones sobre una cadena cinemática en todos los modos posibles, reteniendo los no isomorfos*. Este proceso se conoce como *especialización de mecanismos* [YH91]. El principal aporte respecto a estos autores es el de utilizar una nueva representación del mecanismo $\boldsymbol{T}$ adecuada al identificador de isomorfismo $DC_b^r(\boldsymbol{T})$. Esto permitió enumerar atlas con miles de mecanismos. El procedimiento utilizado se detalla en los párrafos siguientes.

Desde un listado de cadenas cinemáticas almacenadas por sus Códigos de Grado, se toma una y luego se decodifica su matriz de adyacencia $A$, obteniéndose un grafo de $n$ vértices y $j$ aristas con etiquetado de Código de Grado. Los tipos de eslabones a asignar sobre la matriz de adyacencia se arreglan en un *Vector de Tipos de Eslabones* $\boldsymbol{t}_L$ con $n$ entradas. Los tipos de uniones se arreglan en un *Vector de Tipos de Uniones* $\boldsymbol{t}_J$ con $j$ entradas. Luego, la cadena cinemática se especializa en dos pasos:

**Especialización de Eslabones:** genere todos los vectores de tipos de eslabón $\boldsymbol{t}_L$ para un alfabeto dado (por ejemplo, {0=fundación,1=rígido}) satisfaciendo apropiadamente las restricciones relativas a eslabones; asigne secuencialmente cada $\boldsymbol{t}_L$ sobre las entradas de la diagonal de la matriz de adyacencia $A$ construyendo la matriz $\boldsymbol{T}'$, calcule su $DC_b^r$ y salve aquellas que son diferentes. Este procedimiento puede resultar en muchas $\boldsymbol{T}$s no isomorfas. Cada una de ellas se usa en el segundo paso.

**Especialización de Uniones:** genere todos los vectores de tipos de uniones $\boldsymbol{t}_J$ para un alfabeto dado (por ejemplo, {1=rotoidal,2=prismática}); asigne cada $\boldsymbol{t}_J$ sobre los elementos fuera de la diagonal correspondientes de una matriz elegida $\boldsymbol{T}'$ configurando la matriz $\boldsymbol{T}$, verifique las restricciones de diseño para uniones, si son satisfechas calcule su $DC_b^r$; si éste no fue almacenado aún, será un nuevo mecanismo del atlas.

Un ejemplo del primer paso para una cadena cinemática de cuatro barras podría ser:

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

donde el símbolo "$\wedge$" denota la operación de asignar tipos de eslabones a la matriz de adyacencia. Luego, para la matriz calculada $\boldsymbol{T}'$ el segundo paso es:

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

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

Nótese que para ambos pasos, después de calcular $DC_b^r(\boldsymbol{T})$ se realiza la verificación de isomorfismos mediante la comparación de dicho código con los almacenados. Por ejemplo, en el segundo paso del ejemplo de arriba, el vector de tipos de uniones $\boldsymbol{t}_J = [2,1,1,4]$ produce el mismo $DC_5^r(\boldsymbol{T})$ que el vector $\boldsymbol{t}_J = [1,2,4,1]$, de modo que ambas soluciones son isomorfas.

Este procedimiento de asignación sujeto a restricciones de diseño apropiadas permitió generar diversos atlas de mecanismos rígidos y flexibles para cadenas cinemáticas de cuatro, seis y ocho barras, todas ellas de un grado de libertad. El mismo procedimiento se utiliza para cadenas cinemáticas de dos y tres grados de libertad, pero los resultados no se mostraron en la tesis.

### A.6.2. Representación de problemas cinemáticos utilizando grafos

Utilizando una descripción de Elementos Finitos, se definen las partes iniciales de un mecanismo (ver Figura A.4 izquierda). Sobre estas partes se declaran diversas

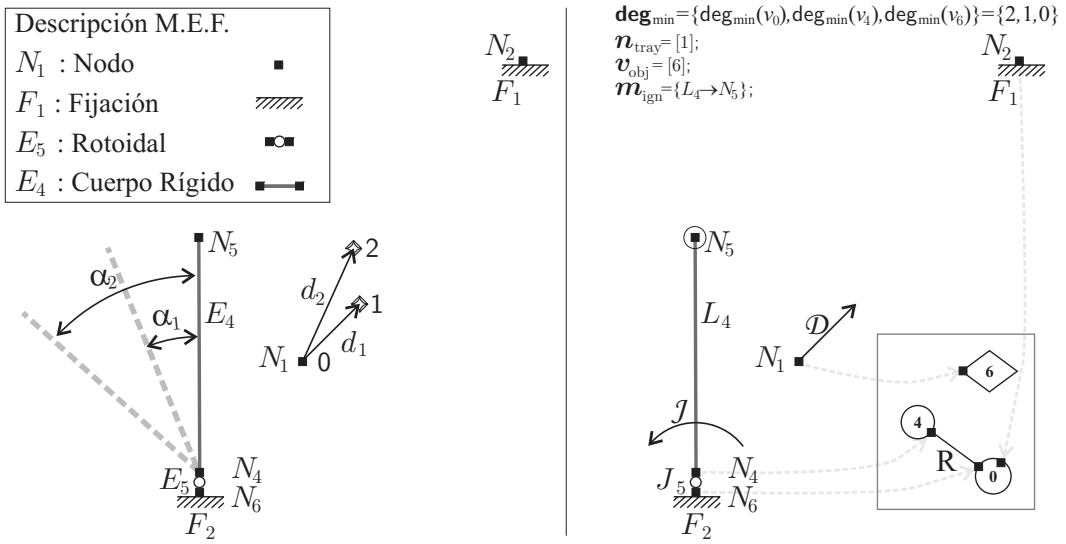



Figura A.4: Tarea de seguimiento de trayectoria.

restricciones del movimiento en forma discreta utilizando conjuntos de desplazamientos sobre un nodo dado $\mathcal{D}$, rotaciones de eslabones $\mathcal{L}$ y restricciones del movimiento sobre uniones $\mathcal{J}$, o sea, rotaciones sobre uniones rotoidales ó deslizamientos sobre uniones prismáticas. Por ejemplo, un conjunto de desplazamientos se define por la tripleta de datos

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

donde $ID$ es el identificador de nodo (ó elemento para los casos de $\mathcal{L}$ y $\mathcal{J}$), $j$ es número de punto de paso en la secuencia de posiciones precisas, y $(d_x, d_y)_j$ son los desplazamientos (o bien rotaciones $\alpha_j$ para los casos de $\mathcal{L}$ y $\mathcal{J}$) para el punto de paso $j$ expresados en coordenadas relativas a la posición inicial del nodo.

Para la tarea de **seguimiento de trayectoria** de la Figura A.4 se desea que el nodo $N_1$ de un mecanismo desconocido pase por tres posiciones precisas de una trayectoria. Estos desplazamientos se declaran como:

$$\mathcal{D} = \{N_1, 0, (0,0);\ N_1, 1, \boldsymbol{d}_1;\ N_1, 2, \boldsymbol{d}_2\}.$$

Si la simultaneidad con la actuación es prescripta ("prescribed timing"), los parámetros de la unión pueden declararse como

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

En base a la descripción MEF y las restricciones $\mathcal{D}$, $\mathcal{L}$ y $\mathcal{J}$, se utilizan reglas (que se enuncian por vez primera en esta tesis) para construir un grafo coloreado etiquetado denominado *grafo inicial*. Este grafo modela en forma implícita diversas restricciones estructurales. Otras restricciones y características útiles para las etapas subsiguientes se almacenan en estructuras de datos auxiliares, éstas son:

*Nodos de Trayectoria:* Cada nodo de trayectoria se almacena en un vector denotado $\boldsymbol{n}_{\text{tray}}$ y su vértice objetivo asignado se almacena en un vector separado denominado $\boldsymbol{v}_{\text{obj}}$.

*Nodos Ignorados:* Estos nodos son ignorados para los propósitos de la síntesis de tipo. Éstos se almacenan en un multimapa auxiliar $\boldsymbol{m}_{\text{ign}}$ que permite mapear

múltiples nodos a un mismo eslabón. Estos nodos que no pertenecen a ninguna unión cinemática y no tienen movimientos prescriptos se muestran rodeados con un círculo; véase, por ejemplo, la Figura A.4.

*Grado de Conectividad de los eslabones prescriptos:* Después de la construcción del grafo inicial, se llena un vector llamando *grado mínimo de vértices* $\mathbf{deg}_{\text{mín}}$ que se utilizará como restricción para acelerar y/o personalizar la búsqueda de subgrafos.

Como se observa a la derecha de la Figura A.4, las partes iniciales tienen una representación de grafo inicial que puede, inclusive, ser disconexo. Para este ejemplo, en la matriz de adyacencia del grafo se considera una fila y una columna nula para el vértice objetivo:

$$
\begin{aligned}
V_{\text{ini}} &= \{\ 0 \quad 1 \quad 2\ \}, \quad E_{\text{ini}} = \{(0,4)\} \\
\mathcal{V}(V_{\text{ini}}) &= \{\ 0 \quad 4 \quad 6\}\ , \quad \mathcal{E}(E_{\text{ini}}) = \{(0,4) \to 5\} \\
A_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\
\mathcal{T}_L(\mathcal{V}(V_{\text{ini}})) &= \{0 \to 0, 4 \to 1, 6 \to 1\}, \\
\mathcal{T}_J(\mathcal{V}(V_{\text{ini}}) \times \mathcal{V}(V_{\text{ini}})) &= \{(0,4) \to 1\}, \\
\boldsymbol{T}_{\text{ini}} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},
\end{aligned}
$$

donde $\mathcal{V}$ es una función de etiquetado de eslabones, $\mathcal{E}$ es una función de etiquetado de uniones (utiliza pares de eslabones etiquetados), $A$ es la matriz de adyacencia del grafo, $\mathcal{T}_L$ es un mapeo de los eslabones etiquetados hacia los tipos de eslabón y $\mathcal{T}_J$ es un mapeo de las uniones etiquetadas hacia los tipos de unión. Con $A$, $\mathcal{T}_L$ y $\mathcal{T}_J$, la matriz de tipos $\boldsymbol{T}$ queda definida implícitamente. Además, $\mathcal{T}_L$ y $\mathcal{T}_J$ son invariantes ante las permutaciones de $A$ (permutando los eslabones etiquetados), lo cual facilita el chequeo del isomorfismo.

Esta nueva definición de la matriz de adyacencia es el aspecto clave para tratar con un grafo general y, particularmente, con el problema de isomorfismo de subgrafos. Para restringir la búsqueda de subgrafos se definen dos herramientas que relacionan el grafo inicial $G_{\text{ini}}$ y uno tomado del atlas $G_A$:

- **Distancia desde la fundación hasta el vértice objetivo**: para tareas de seguimiento de trayectoria o guiado de cuerpo rígido, el vértice objetivo $v_{\text{obj}}$, que contiene el nodo que desarrolla la tarea, se elige de modo tal que posea cierta distancia desde la fundación $v_0$. Esta distancia se define como el mínimo número de vértices necesarios para ir desde la fundación al vértice objetivo: $\text{mín}\, d(v_0, v_{\text{obj}})$. La cota inferior se fija en 2 mientras que la superior es especificada por el usuario o, por defecto, calculada como el número de puntos de paso de la tarea $n_{pp}$ menos uno:

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

- **Matriz de Adyacencia de Síntesis**: utilizada para considerar a cada parte del grafo inicial, que aparece como subgrafo del grafo del mecanismo, como

*funcionalmente diferente*:

$$S_{ii}(v_i) = \begin{cases} T_{ii} & \text{si } v_i \text{ es un vértice sintetizado,} \\ k & \text{si } v_i \text{ es un vértice prescripto,} \end{cases}$$
$$S_{ij}(e_{ij}) = T_{ij} \quad \forall\ e_{ij}$$

donde $k = b + j$; $j = 0, 1, ..., n_{\text{ini}} - 1$, con $n_{\text{ini}} = |V_{\text{ini}}|$ la cardinalidad del conjunto de vértices prescripto y $b$ el número de colores en $\boldsymbol{T}(G_A)$. El número de colores para codificar $\boldsymbol{S}$ es el número de eslabones más el número de colores de los grafos tomados del atlas, o sea, $n_{\text{ini}} + b$.

### A.6.3. Síntesis de número mediante una búsqueda de subgrafos restringida

El grafo inicial representa la situación inicial. Para obtener un mecanismo que se corresponda con esta situación, el grafo inicial debe ser un subgrafo de cualquier mecanismo válido del atlas de mecanismo seleccionado. Las repeticiones se evitan mediante el agregado de restricciones a cumplir.

El problema consiste en buscar el mecanismo más simple en el atlas para el cual el grafo inicial es un subgrafo:

$$G_{\text{ini}} \subseteq G_A \tag{A.3}$$

con $G_A$ un grafo del atlas. Sin embargo, también deben satisfacerse las siguientes restricciones:

- la restricción de igualdad

  $$\boldsymbol{T}(G_{\text{ini}}) = \boldsymbol{T}(H_A), \quad G_{\text{ini}} \cong H_A, \quad H_A \subseteq G_A, \tag{A.4}$$

  esto es, *los tipos de eslabones y uniones en* $G_{\text{ini}}$ *deben corresponderse exactamente con los de un subgrafo* $H_A$ en $G_A$;

- la *restricción de distancia* para cada vértice objetivo dada por la ecuación (A.2);

- la *restricción de isomorfismo*, requiere que $DC_b^d(\boldsymbol{S}(G_{\text{ini}}, G_{\text{A}}))$ debe ser diferente de todas las respuestas previas; y

- la *restricción de pseudo isomorfismo*; ninguna solución tiene a una previa como subgrafo.

Se ejecutó una búsqueda de subgrafos en el atlas de mecanismos rígidos RigidOneDofR, con 77 candidatos a explorar, para resolver el problema mostrado en la Figura A.4. En las Figuras A.5 y A.6 se pueden observar las primeras siete soluciones al problema. La segunda figura muestra más claramente el significado físico.

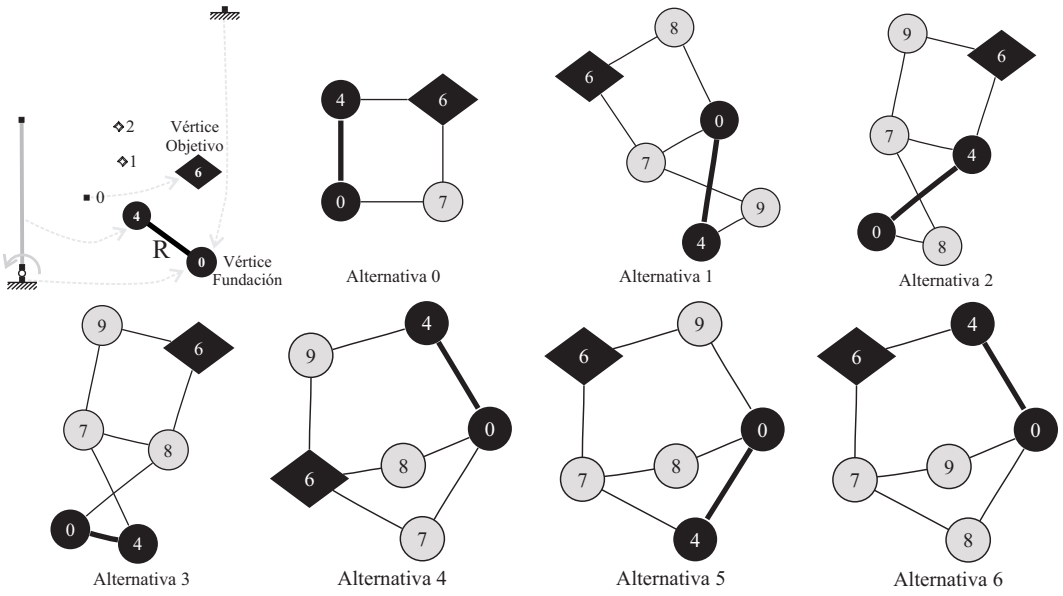



Figura A.5: Primeras 7 ocurrencias no isomorfas del grafo inicial

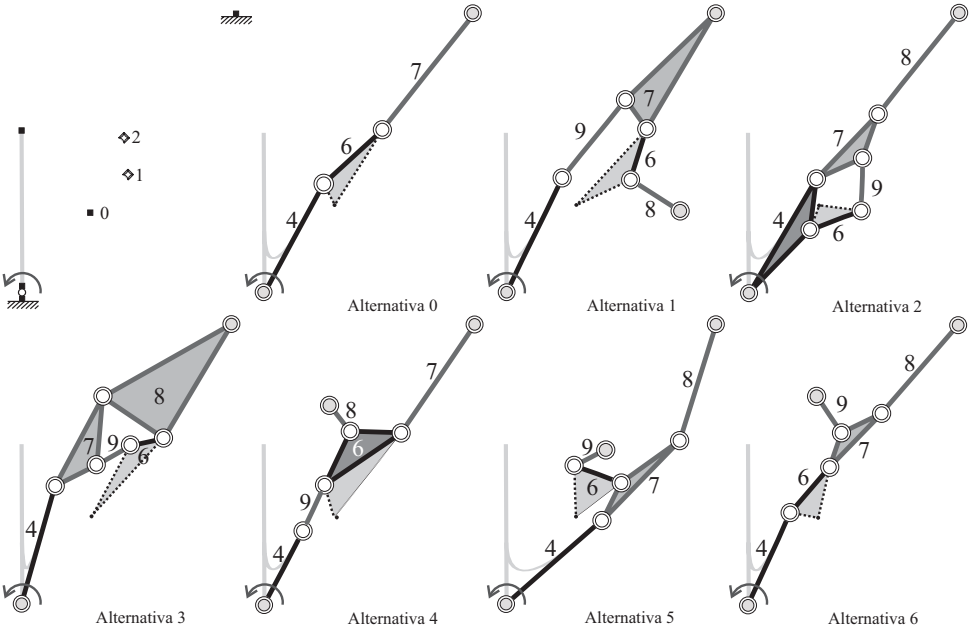



Figura A.6: Esquemas físicos para una tarea de seguimiento de trayectoria.

## A.7. Dimensionado inicial

La estrategia de dimensionado inicial explota el uso de métodos analíticos para la síntesis de cadenas abiertas. Existen varias formas de las ecuaciones de síntesis, debido a ello se presenta una revisión de las dos más importantes y se las compara para justificar la elección de la más adecuada para la sistematización del diseño. Se introducen algunas convenciones acerca de los datos necesarios y las condiciones de resolubilidad usadas en el criterio de descomposición en cadenas abiertas.

### A.7.1. Síntesis analítica

La síntesis analítica se utiliza para dimensionar un mecanismo por medio de ecuaciones de forma cerrada. El método es aplicable cuando (i) la topología puede ser descompuesta en subsistemas más simples denominados *Cadenas Abiertas Simples* ("Single Open Chains (SOCs)") y (ii) la *tarea es simplificada* en un número de desplazamientos y/o orientaciones llamadas *posiciones precisas*. El método para este problema de síntesis simplificado se conoce como el Método de Puntos Precisos ("Precision-Point Method (PPM)"). Para la síntesis de cadenas abiertas no se tiene en cuenta la función (unión motorizada, eslabón flotante, eslabón conductor, etc.) que cumple cada eslabón o unión en el mecanismo.

Para cadenas abiertas planas, la unión que precede a cada eslabón permite, ya sea, la *rotación* de un eslabón respecto de un eje perpendicular al plano o un *deslizamiento* en el plano, de un eslabón respecto al previo. Ambos movimientos se modelan fácilmente en términos de números complejos mediante el *operador de Euler* y un *factor de estiramiento*, respectivamente. Usando estas herramientas, las *Ecuaciones de Lazos Cerrados* pueden plantearse para un número posiciones precisas dado. Luego, dependiendo del número de eslabones en la cadena y del número de posiciones del problema, el sistema de ecuaciones resultante puede ser lineal o no lineal en las dimensiones de los eslabones (incógnitas). Afortunadamente, algunas ecuaciones no lineales pueden ser resueltas manipulando las condiciones de compatibilidad y sus relaciones geométricas con los datos, mediante el empleo de números complejos [HD64, SE84, ES97, LEJ96].

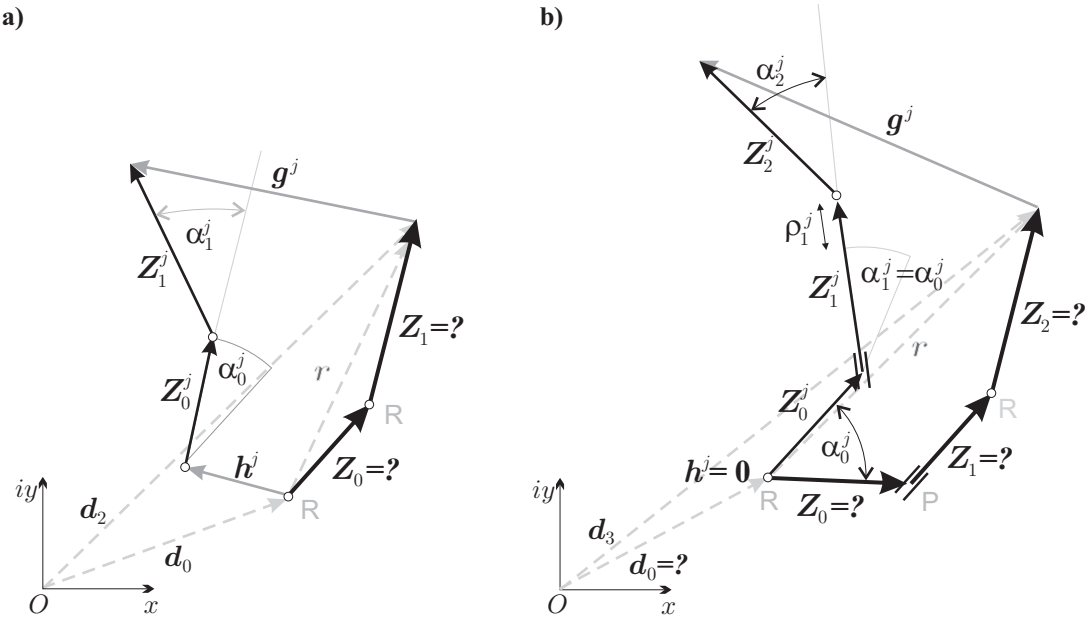



Figura A.7: Ejemplos de datos modelados por números complejos: (a) para una Díada RR y (b) para una Tríada RPR. Se muestran las posiciones inicial y la $j$-ésima.

Cuando la cadena es movida a la $j$-ésima posición precisa, la configuración que se obtiene es caracterizada por la naturaleza de cada unión, es decir, uniones rotoidales permiten rotaciones del eslabón $\alpha_l^j$ (ver por ejemplo, el primer eslabón de la Figura A.7-a donde $\boldsymbol{Z}_0^j = \boldsymbol{Z}_0 e^{i\alpha_0^j}$), y las uniones prismáticas permiten estiramientos

$\rho_l^j$ a lo largo de la dirección del eslabón siguiente de la unión preservando un ángulo fijo con el eslabón previo; véase, por ejemplo, el segundo eslabón de la Figura A.7-b donde $\boldsymbol{Z}_1^j = \rho_1^j \boldsymbol{Z}_1 e^{i\alpha_0^j}$. Los desplazamientos de los puntos extremos de la cadena, $\boldsymbol{h}^j$ en la cola y $\boldsymbol{g}^j$ en la punta, así como las posiciones, $\boldsymbol{d}_0$ y $\boldsymbol{d}_{n_L}$, respectivamente, también se modelan convenientemente mediante números complejos. Si ambas posiciones de inicio y fin de la cadena, $\boldsymbol{d}_0$ y $\boldsymbol{d}_{n_L}$, son conocidas, se dice que la *separación*, $\boldsymbol{r} = \boldsymbol{d}_{n_L} - \boldsymbol{d}_0$ ("offset"), es impuesta. Una estrategia propuesta en esta tesis es la de utilizar ecuaciones que suponen que *esta separación siempre es conocida.*

Para la díada pasando por tres posiciones con separación impuesta de la Figura A.7-a, se tiene el siguiente sistema de ecuaciones de lazos cerrados:

$$\begin{cases} \boldsymbol{Z}_0(e^{i\alpha_0^j} - 1) + \boldsymbol{Z}_1(e^{i\alpha_1^j} - 1) = \boldsymbol{\delta}^j, \\ \boldsymbol{Z}_0 + \boldsymbol{Z}_1 = \boldsymbol{r}. \end{cases} \quad j = 1, 2 \tag{A.5}$$

donde $\boldsymbol{\delta}^j = \boldsymbol{g}^j - \boldsymbol{h}^j$ y $\boldsymbol{r} = \boldsymbol{d}_{n_L} - \boldsymbol{d}_0$. Esta ecuación puede escribirse en forma matricial como:

$$\begin{bmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \boldsymbol{r} \end{bmatrix}, \tag{A.6}$$

o abreviadamente como:

$$\mathbb{C}^{\text{off}}\mathbb{Z} = \mathbb{D}^{\text{off}}. \tag{A.7}$$

Este sistema tendrá solución no trivial si los coeficientes de la matriz no cuadrada en (A.7) es rango($\mathbb{C}^{\text{off}}$) = 2, esto es

$$\det \mathbb{C}_a^{\text{off}} = \begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \end{vmatrix} \neq 0 \wedge \det \mathbb{C}_b^{\text{off}} = \begin{vmatrix} (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) \\ 1 & 1 \end{vmatrix} \neq 0, \tag{A.8}$$

y el determinante característico de tercer orden de su sistema aumentado es cero,

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

$$\begin{vmatrix} (e^{i\alpha_0^1} - 1) & (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_0^2} - 1) & (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ 1 & 1 & \boldsymbol{r} \end{vmatrix} = 0. \tag{A.10}$$

Nótese que la solución trivial para esta condición se halla proponiendo $\alpha_1^1 = \alpha_0^1$ y $\alpha_1^2 = \alpha_0^2$ en $\mathbb{C}^{\text{off}}$ pero ello contradice las ecuaciones (A.8). La tercer columna en la ecuación (A.10) incluye los datos conocidos, $\boldsymbol{\delta}^j$ y $\boldsymbol{r}$, y una de las primeras columnas deben ser completamente desconocidas. Luego, podemos expandir el determinante por la columna desconocida para encontrar relaciones geométricas entre los parámetros. Suponiendo, por ejemplo, que $\alpha_1^1$ y $\alpha_1^2$ son restricciones del movimiento definidos sobre el segundo eslabón, los datos conocidos quedan en la segunda y última columna. Desarrollando el determinante por cofactores de la primer columna se obtiene la *ecuación de compatibilidad* para el sistema (A.7):

$$(e^{i\alpha_0^1} - 1)\underbrace{\begin{vmatrix} (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \\ 1 & \boldsymbol{r} \end{vmatrix}}_{\Delta_1} + (e^{i\alpha_0^2} - 1)\underbrace{\left(-\begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ 1 & \boldsymbol{r} \end{vmatrix}\right)}_{\Delta_2} + \underbrace{\begin{vmatrix} (e^{i\alpha_1^1} - 1) & \boldsymbol{\delta}^1 \\ (e^{i\alpha_1^2} - 1) & \boldsymbol{\delta}^2 \end{vmatrix}}_{\Delta_3} = 0$$

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 \underbrace{-\Delta_1 - \Delta_2 + \Delta_3}_{\Delta_0} = 0 \tag{A.11}$$

$$e^{i\alpha_0^1}\Delta_1 + e^{i\alpha_0^2}\Delta_2 = -\Delta_0. \tag{A.12}$$

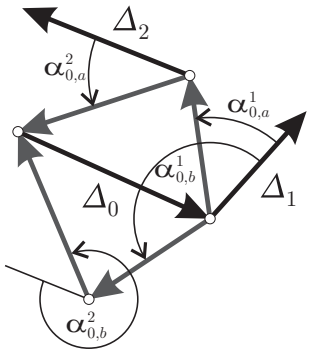



Figura A.8: *Estructura Solución* en sus dos inversiones geométricas.

La Figura A.8 muestra el triángulo de la estructura solución correspondiente a la ecuación (A.12) para obtener dos conjuntos de $\alpha_0^1$ y $\alpha_0^2$. Reemplazando cada par en los coeficientes de $\mathbb{C}_a^{\text{off}}$ (ó $\mathbb{C}_b^{\text{off}}$), se pueden calcular dos conjuntos de eslabones. Se dice entonces que existe multiplicidad 2, y el número de soluciones es finito. Si faltase por ejemplo el dato $\alpha_0^2$, éste se toma como parámetro libre y se dice que el problema tiene $2(\infty)$ soluciones posibles. Existen otras condiciones de compatibilidad más complejas que "tienen forma de eslabonamientos" con diversos grados de libertad (coincidentes con los parámetros libres). Proponiendo dichos grados de libertad se obtienen estructuras solución con fácil eliminación geométrica de las variables. Este procedimiento de construir el *Eslabonamiento de Compatibilidad* y resolver luego la *Estructura Solución* fue introducido por Sandor y Freudenstein [San59], Denavit y Hartenberg [HD64], Sandor y Erdman [ES97], y luego extendido por Lin *et al.* [LEJ96] en un modo sistemático para díadas pasando por 4 y 5 posiciones, tríadas de 5 hasta 7 posiciones, cuadríadas de 6 hasta 9 posiciones [LEJ96].

El planteo general de las ecuaciones de lazos cerrados podría clasificarse en dos categorías –estándar y con separación impuesta– donde la diferencia principal es que la *separación* impone una ecuación adicional:

**Síntesis estándar.** Las ecuaciones de lazos cerrados son:

$$\boldsymbol{Z}_0(\lambda_0^j - 1) + \boldsymbol{Z}_1(\lambda_1^j - 1) + \cdots + \boldsymbol{Z}_{n_L-1}(\lambda_{n_L-1}^j - 1) = \boldsymbol{\delta}^j, \quad j = 1, \ldots, n_{pp} - 1. \tag{A.13}$$

donde $\lambda_l^j = e^{\alpha_l^j}$ si la unión que el precede al eslabón $l$ es de tipo rotoidal ó $\lambda_l^j = \rho_l^j$ para tipo prismática.

- Una de las posiciones de los extremos de la cadena, $\boldsymbol{d}_0$ ó $\boldsymbol{d}_{n_L}$, es conocido. Los desplazamientos asociados al otro extremo (pivote) son nulos.
- Si $n_{pp} = n_L + 1$ el sistema tiene solución lineal directa.

- Para $n_{pp} > n_L + 1$ pueden utilizarse los eslabonamientos de compatibilidad. Entre otras propiedades, Lin [LEJ96] encontró que el número de grados de libertad (número de parámetros libres) de un eslabonamiento de compatibilidad es $F = 2n_L - n_{pp} + 1$.
- Después de resolver el sistema de ecuaciones, se calcula la posición del pivote para el extremo con posición desconocida.

**Síntesis con separación impuesta.** Se tiene en cuenta una ecuación adicional:

$$\begin{cases} \boldsymbol{Z}_0(\lambda_0^j - 1) + \boldsymbol{Z}_1(\lambda_1^j - 1) + \cdots + \boldsymbol{Z}_{n_L-1}(\lambda_{n_L-1}^j - 1) = \boldsymbol{\delta}^j \\ j = 1, \ldots, n_{pp} - 1, \\ \boldsymbol{Z}_0 + \boldsymbol{Z}_1 + \cdots + \boldsymbol{Z}_{n_L-1} = \boldsymbol{r}, \end{cases} \tag{A.14}$$

- Ambas posiciones de puntos extremos, $\boldsymbol{d}_0$ y $\boldsymbol{d}_{n_L}$, son conocidos.
- Si $n_{pp} = n_L$ el sistema tiene solución lineal directa.
- Para $n_{pp} > n_L$ pueden utilizarse los eslabonamientos de compatibilidad en un modo similar al caso sin separación.

Para ambas categorías:

- Los conjuntos de desplazamientos de los puntos extremos, $\boldsymbol{h}^j$ y $\boldsymbol{g}^j$, se consideran como datos conocidos.
- Las restricciones del movimiento que pueden imponerse resultan sujetas a:
  - El número de ecuaciones. Éste coincide con el número de posiciones precisas $n_{pp}$ en la forma estándar (A.13), y es $n_{pp} + 1$ para separación impuesta (A.14).
  - El número de incógnitas (número de eslabones).

  Si el número de ecuaciones excede el número de incógnitas, el sistema resulta sobre determinado (deficiente de rango y no lineal). Para estos casos, la condición necesaria que deben cumplir los movimientos impuestos es que *debe haber un eslabón completamente libre de restricciones para poder desarrollar el determinante característico de* $(n_L + 1)$*-ésimo orden*. Si el número de ecuaciones sobrepasa en 2 al número de eslabones, la condición de compatibilidad resulta en un sistema de determinantes característicos acoplados.

Esta última condición se utiliza en el algoritmo de descomposición en cadenas.

En la ecuaciones con separación impuesta, la ventaja de considerar siempre $\boldsymbol{d}_0$ y $\boldsymbol{d}_{n_L}$ como datos conocidos es la de tener un control sobre el lugar en donde se localizan los pivotes sintetizados. Esto permite una estrategia de diseño más robusta, de modo tal que, si un pivote es sintetizado por el resolvedor de síntesis de tipo, luego: (i) su posición se considera conocida para los propósitos de la descomposición; (ii) los desplazamientos se imponen como nulos; (iii) el resolvedor de síntesis dimensional requiere del usuario la definición de un área para la localización del pivote (definida como fronteras de tipo caja dentro del espacio permitido); y (vi), cuando se ejecuta el resolvedor de síntesis dimensional, la posición de tal pivot, $\boldsymbol{d}_0$ ó $\boldsymbol{d}_{n_L}$, se genera como dato para el correspondiente módulo resolvedor de la cadena.

Por otro lado, si el pivote fuese calculado con las ecuaciones estándar, tal pivote resultante podría caer lejos del espacio permitido para un amplio rango de los parámetros libres. Las ecuaciones estándar sólo admiten la definición de parámetros libres en la forma de restricciones de movimiento (ángulos y deslizamientos). La definición de fronteras para localización de pivotes resulta entonces, más intuitiva que para restricciones de movimientos.

### Módulos programados

Un *módulo SOC* se define siguiendo un *camino de nodos* involucrando ambos, eslabones (vértices) y uniones (aristas). En la Figura A.9 podemos ver los módulos disponibles para analizar y resolver las SOCs. Usando la descripción MEF, un módulo comienza en un nodo y termina en un nodo, y entre dos eslabones siempre existe un par de nodos restringidos por una unión. Para los nodos internos, el símbolo □ significa que las posiciones y los conjuntos de desplazamientos son desconocidos. Para los nodos de los extremos, el símbolo ■ significa que las posiciones y los conjuntos de desplazamientos son conocidos.

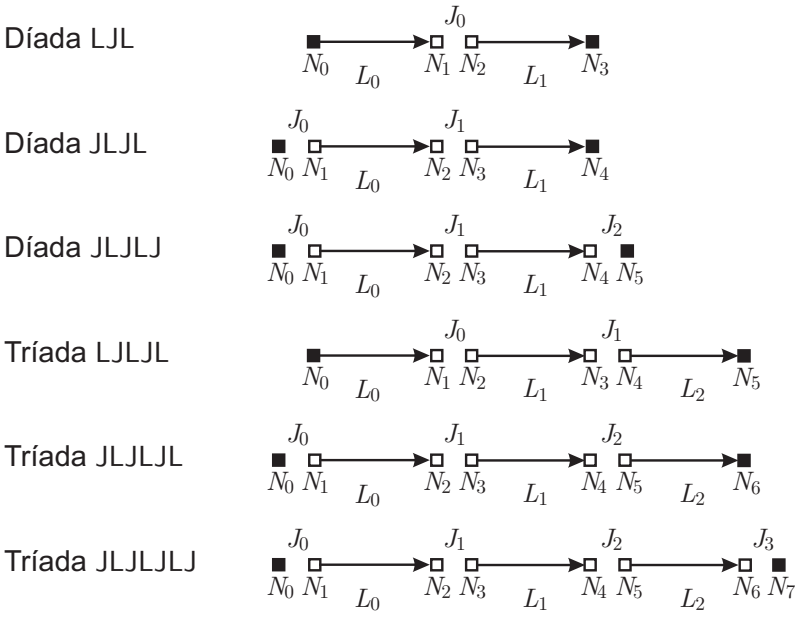



Figura A.9: Módulos para resolver SOCs. La letra `L` denota al eslabón y `J` denota a una unión.

Dados $n_{pp}$ posiciones y los movimientos impuestos, un módulo puede ejecutar dos funciones:

`Assemble`: Devuelve la cantidad de parámetros libres necesarios y su multiplicidad de soluciones.

`Solve`: Resuelve el sistema de Ecuaciones de Lazos Cerrados para un conjunto de parámetros y una multiplicidad elegida externamente. La función puede retornar una respuesta exitosa ó que no existe solución.

Estas funciones serán manejadas automáticamente por un optimizador. Para una SOC dada, la exploración de los parámetros libres dentro de sus rangos (por ejemplo,

$[0, 2\pi]$ para un parámetro rotacional) así como de sus multiplicidades resultará en un número finito o infinito de soluciones que se ensamblarán con soluciones de otras SOCs para formar mecanismos.

### A.7.2. Descomposición de la topología en cadenas abiertas

El objetivo que se persigue con la descomposición de una topología en cadenas abiertas es la de hallar una secuencia de cadenas que contengan a todas las dimensiones significativas del mecanismo, sin repetición. Se establece un problema de optimización entera que consiste en hallar, entre todas las descomposiciones factibles, la descomposición que mejor satisface cierto criterio de evaluación. En el criterio de evaluación se propone: (i) considerar las condiciones necesarias para la resolubilidad de cada cadena abierta y (ii) dar prioridad a las cadenas que resuelven un mayor número de restricciones (para reducir al mínimo el número de variables de diseño). El enfoque presentado es algorítmico y también analítico; se presentan las descomposiciones que pueden ser resueltas con los módulos disponibles.

Las *dimensiones significativas* del mecanismo son aquellas que tienen influencia en el comportamiento cinemático del mecanismo. En términos de la descripción MEF del mecanismo, las dimensiones significativas son determinadas por (i) los nodos que están conectados por uniones cinemáticas y (ii) los nodos que deben desarrollar alguna tarea, es decir, los nodos trayectoria. Esto último es particular de los problemas de síntesis. Algunas de las dimensiones significativas son *previamente conocidas*, esto es, pueden calcularse mediante un análisis cinématico de las partes iniciales del problema, en una etapa que denominamos *Cinemática Inicial*. El resto de las dimensiones significativas deben calcularse en la etapa de dimensionado inicial y constituyen el objeto del problema de síntesis. Pueden existir otros nodos que sólo definen la forma de los eslabones y que no tienen movimientos impuestos. Se considera que estos nodos definen *dimensiones obsoletas* y por lo tanto serán ignorados durante el proceso de síntesis dimensional.

El método de descomposición asigna SOCs a las dimensiones significativas de un mecanismo mediante los siguientes pasos:

S1) **Descomposición de la topología:** La cadena cinemática cerrada es descompuesta en un conjunto de lazos cerrados de mínima longitud denominados *base de lazos mínimos*. Para la mayoría de los grafos esta base no es única.

S2) **Descomposición en SOCs:** Por cada base de lazos cerrados, cada lazo es seleccionado en un orden y orientación dados para ser descompuestos en módulos SOCs, o sea, díadas y tríadas, usando restricciones de desplazamientos sobre los nodos.

S3) **Evaluación de SOCs:** Para cada descomposición, se simula la transferencia de datos mientras se genera un índice que evalúa la resolubilidad de cada SOC y el número de restricciones que cada una resuelve.

S4) **Orden de SOCs retenido:** La descomposición en cadenas abiertas mejor valuada se almacena para pasar a la síntesis dimensional.

La base de lazos mínimos se calcula fácilmente utilizando algunas propiedades de los grafos [Tsa01]. Cada lazo se expresa como una secuencia de vértices y aristas,

pero para identificar cadenas se incorpora además el nivel de los nodos mediante una tabla auxiliar. En el caso de que exista un vértice objetivo en el lazo, el lazo es desviado hasta el nodo trayectoria asociado a dicho vértice.

Para los módulos resolvedores de cadenas abiertas basados en ecuaciones con separación impuesta, un lazo cerrado (expresado en términos de nodos) se divide para formar una cadena abierta *comenzando por un nodo con posición y desplazamientos conocidos y terminando en otro nodo con posición y desplazamientos conocidos*. Se exploran las cadenas cinemáticas *lazo por lazo*, en distintos *órdenes*, y además, en distintos *sentidos* para los casos en que existe un vértice objetivo en el lazo. Luego, sobre las cadenas abiertas, descompuestas en una secuencia dada, se realiza la evaluación. Para evaluar las condiciones de resolubilidad, se simulan (i) la transferencia de datos entre posiciones y desplazamientos en los nodos y (ii) la transferencia de rotaciones en los eslabones.

Para cada SOC, se cuenta el *número de eslabones sin restricción de movimiento alguna* denotado como $n_U(\mathbf{SOC})$. Según la síntesis analítica, este número debe ser:

$$n_U(\mathbf{SOC}) \begin{cases} \geq 1 & \text{si } n_{pp} > n_L \text{ y, } d_0 \text{ y } d_{n_L} \text{ son conocidos (separación impuesta),} \\ \geq 0 & \text{si } n_{pp} = n_L \text{ y, } d_0 \text{ y } d_{n_L} \text{ son conocidos (separación impuesta),} \\ \geq 1 & \text{si } n_{pp} > n_L + 1 \text{ y, } d_0 \text{ ó } d_{n_L} \text{ es desconocido,} \\ \geq 0 & \text{si } n_{pp} = n_L + 1 \text{ y, } d_0 \text{ ó } d_{n_L} \text{ es desconocido.} \end{cases} \tag{A.15}$$

Considerando sólo los dos primeros casos, el número de módulos necesarios y, consecuentemente, el número de descomposiciones posibles se reducen.

Las reglas de evaluación se modelan por medio de:

- un vector de Booleanos indexado por las cadenas (SOCs), $\boldsymbol{R}_I$, para el cual una entrada tiene valor verdadero si la cadena correspondiente satisface la condición (A.15), y falso de otro modo.

- un vector de enteros indexado por las cadenas (SOCs),

$$\boldsymbol{R}_{II} = \{n_C(\mathbf{SOC\ 0}), n_C(\mathbf{SOC\ 1}), \ldots, n_C(\mathbf{SOC}\ \boldsymbol{\nu}), \ldots\},$$

  donde la función $n_C(\mathbf{SOC\ k})$ simplemente cuenta el número de restricciones de movimiento resueltas por la $k$-ésima SOC.

Finalmente, un criterio que combina a ambas reglas se arregla en forma de vector de enteros y se lo denota con $\boldsymbol{R}^i = \{\boldsymbol{R}_I, \boldsymbol{R}_{II}\}$, donde el supraíndice $i$ denota el número de descomposición. Las condiciones Booleanas se traducen a enteros: 1 para verdadero y 0 para falso. Dos criterios dados se comparan fácilmente en orden lexicográfico.

Usando este procedimiento se obtuvieron los resultados mostrados en la Figura A.10 donde se dibujan las mejores descomposiciones en SOCs ordenadas, para cada alternativa topológica que es solución del mismo problema.

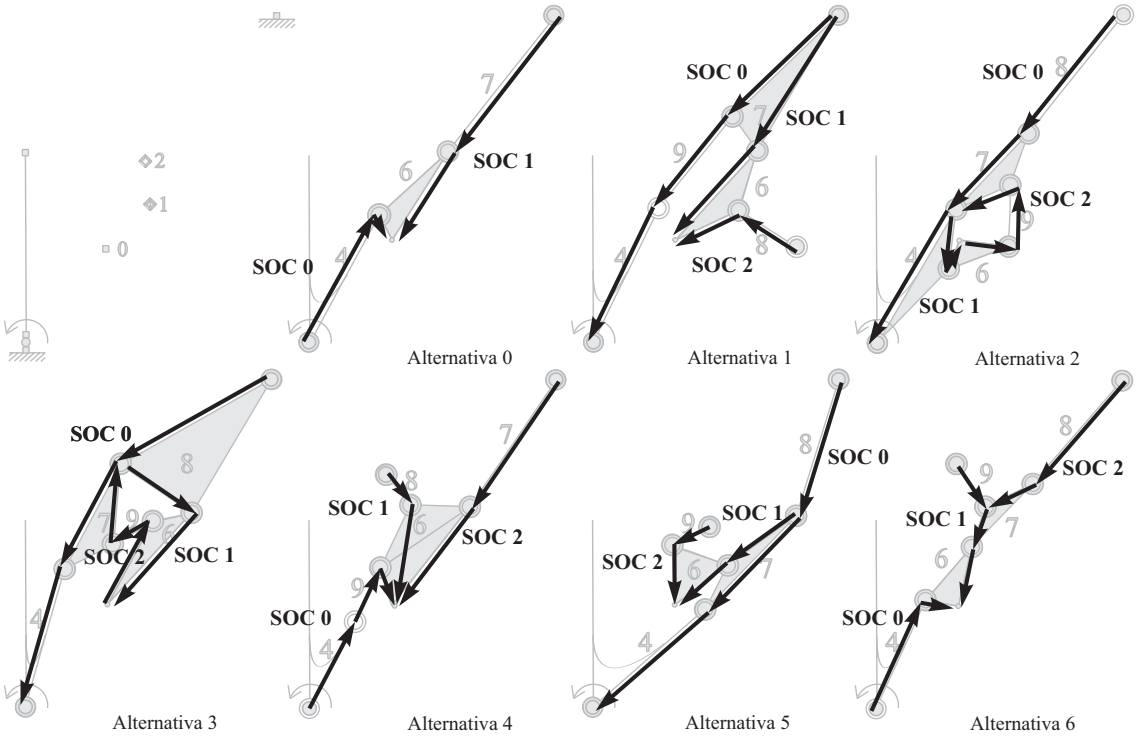



Figura A.10: Mejores descomposiciones para las **Alternativas 0** hasta **6** del problema de seguimiento de trayectoria.

## A.7.3. Estrategia de dimensionado inicial

El objetivo del dimensionado inicial es determinar las dimensiones iniciales del mecanismo utilizando módulos de síntesis exacta. Si es necesaria una posterior optimización entre puntos precisos de la tarea, la solución exacta constituye una buena condición inicial, lo cual es algo muy difícil de hallar para este tipo de problemas. El método propuesto es general y de ejecución casi automática, ya que sólo se deben definir las fronteras de las variables y algunos parámetros del algoritmo optimizador. Sin embargo, los valores por defecto para los mismos se definen en función de los datos del problema aplicando algunas reglas heurísticas.

En la tesis se describen los algoritmos y estructuras de datos relativas a los grados de libertad del problema y se detalla también el manejo de la combinación de soluciones de cadenas abiertas con multiplicidad geométrica. Aquí se presentará una versión resumida.

### Descripción del problema

Las ecuaciones de lazos cerrados para las diversas cadenas abiertas que contienen las dimensiones significativas de un mecanismo, y para un número de posiciones precisas $n_{pp}$ se enuncian como:

$$\mathbb{C}^k\mathbb{Z}^k = \mathbb{D}^k \qquad k = 0, \ldots, s-1 \quad \text{(no implica suma sobre } k\text{)} \tag{A.16}$$

donde $s$ es el número de cadenas abiertas simples en el cual se ha descompuesto al mecanismo; el supraíndice $k$ denota al número de SOC u *orden de cálculo*; $\mathbb{C}^k$ es

una matriz compleja cuyos datos dependen de la tarea y de los tipos de uniones, $\mathbb{D}^k$ es un vector complejo que depende de los desplazamientos de los extremos de la cadena (e implícitamente de las posiciones de pivotes), y $\mathbb{Z}^k$ es un vector complejo que representa a los eslabones de la SOC a calcular, es decir, a las incógnitas del problema. Nótese que usando la descomposición propuesta, todos los vectores $\mathbb{Z}^k$ resultan diferentes.

Previo a la resolución del sistema A.16, se ejecuta un lazo sobre las cadenas resultantes de la descomposición elegida, en el orden de cálculo $k = 0, \ldots, s-1$:

$$\mathbf{SOC}[k].\texttt{Assemble}(\mathsf{SyDOFs}, \boldsymbol{m}_{\text{máx}}[k]).$$

De este modo, se pueden obtener: las multiplicidades de cada cadena almacenadas en un vector $\boldsymbol{m}_{\text{máx}}$, y una tabla de grados de libertad de síntesis SyDOFs desde donde se pueden identificar, si existiesen, los parámetros libres denotados con $\boldsymbol{x}$. Esta ejecución se realiza justo después de que una descomposición se seleccionó como factible. Un ejemplo de parámetros libres puede ser:

$$\boldsymbol{x} = [\underbrace{\alpha_0^1 \quad \alpha_0^2}_{\textbf{SOC 0}} \quad \underbrace{\rho_3^1 \quad d_{x\,N_2}^{\;1}}_{\textbf{SOC 1}} \quad \underbrace{x_{N_5}^0 \quad y_{N_5}^0}_{\textbf{SOC 2}}].$$

Además, según el tipo de variable de que se trate, se definen automáticamente sus fronteras que también se arreglan en forma vectorial $\boldsymbol{x}_{\text{mín}}$ y $\boldsymbol{x}_{\text{máx}}$. Alternativamente, el usuario puede modificarlas en forma manual mediante un archivo de texto denominado *epílogo*.

Una cadena abierta $k$ se corresponde en el orden de cálculo con el $k$-ésimo sistema de ecuaciones en (A.16) y puede tener solución única, una multiplicidad finita de soluciones (por ejemplo 2 o 4), ó, si existiesen parámetros libres, puede tener infinitas o un número múltiplemente infinito de soluciones. Denotemos con $\boldsymbol{p}^k$ al conjunto de parámetros fijos con que se resuelve esta cadena. Cada vez que se tienen soluciones múltiples, el proceso resulta en nuevos subproblemas para las cadenas siguientes –con las cuales, dicha SOC intersecta o comparte partes– con diferentes parámetros fijos $\boldsymbol{p}_m^{k+1}$; el índice $m$ denota la $m$-ésima solución para la SOC $k$.

Utilizando $\boldsymbol{m}_{\text{máx}}$, las diversas combinaciones de soluciones de las SOCs se arreglan convenientemente en una matriz $M$ con vectores de combinación $\boldsymbol{m}$.

Puede ocurrir que el problema no tenga solución y que, por lo tanto, la topología deba descartarse. La principal fuente de interrupción del proceso de síntesis es la inexistencia de solución analítica para algunos de los resolvedores de SOCs en el orden de cálculo. Por otro lado, si la solución analítica existe, el mecanismo podría calificarse como pobre o malo debido a: (i) una marcada violación de las restricciones, y/o (ii) que la solución es inaceptable debido a defectos de rama y circuito [BC02a]. La inexistencia de solución para un problema de síntesis tan simplificado –3 ó 4 posiciones precisas– es un buen índice de su dificultad. Como este problema es altamente no lineal y no convexo, se emplea un método de búsqueda de orden cero para proponer el conjunto de parámetros libres y así poder calcular los eslabones por medio de la secuencia de SOCs.

Por lo tanto, ya sea con o sin parámetros libres, en el caso de soluciones múltiples se tiene un árbol de soluciones con varias ramas que deben cubrirse sucesivamente. De este modo, la estrategia involucra un proceso ramificado que mezcla técnicas de optimización combinatoria y optimización de orden cero, localizando cada mecanismo factible cerca del óptimo global para el criterio considerado. Aquella solución con el mejor índice de comportamiento se retiene para la etapa de análisis preliminar.

### Función objetivo

En el algoritmo de dimensionado, la función objetivo a minimizar es el tamaño del mecanismo, el cual se define como la sumatoria de los tamaños de los eslabones de la forma siguiente:

$$F^*(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_L - 1} s(L_k) \tag{A.17}$$

donde $n_L$ es el número de eslabones en el mecanismo. Luego, para un eslabón dado $k$, la función $s(\,)$ retorna una medida de su *tamaño* considerando la sumatoria de todas las distancias entre sus $n^k = n_c^k + n_p^k$ nodos, donde $n_c^k$ nodos están conectados por uniones, y eventualmente, $n_p^k$ nodos tienen movimientos prescriptos (nodos trayectoria); véase la Figura A.11. Todas las distancias entre pares de nodos se calculan sin repetición, es decir,

$$s(L_k) = \frac{1}{\binom{n_c^k}{2}} \sum_{i=0}^{\binom{n_c^k}{2}} d(N^k_{(C[i,0])}, N^k_{(C[i,1])}). \tag{A.18}$$

La función $d(\ ,\ )$ toma dos nodos como argumentos y retorna su distancia Euclideana, mientras que la matriz $C_{ij}$ contiene los $\binom{n_c^k}{2}$ pares de combinaciones de nodos arreglados por filas, de modo tal que los nodos sean apropiadamente indexados.

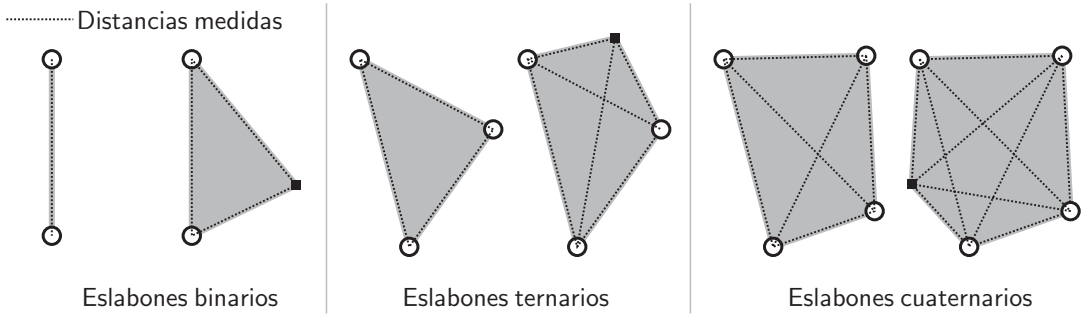



Figura A.11: Distancias a considerar en la *Función Objetivo* y en la restricción *Mínima longitud de eslabones*.

### Evaluación de restricciones

Ya sea con o sin parámetros libres, las restricciones se evalúan después de que las SOCs fueron calculadas exitosamente y luego ensambladas. Se tienen en cuenta tres restricciones:

*Mínima longitud de eslabones* $q_L$: esta restricción se agrega (principalmente debido a razones constructivas) para evitar eslabones con alguna de sus dimensiones demasiado pequeña, mientras que los eslabones con alguna de sus dimensiones demasiado grande se evitan implícitamente mediante la minimización de la función objetivo. Todas las distancias deben ser menores que un parámetro denominado *mínima longitud de eslabón* $L_{\text{mín}}$. Las violaciones de cada distancia de eslabón a esta longitud, se acumulan en la función $s_q(\,)$ para cada

eslabón, y luego se acumulan en la restricción:

$$q_L(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_L-1} s_q(L_k, L_{\text{mín}}). \tag{A.19}$$

La restricción se calcula fácilmente mientras se calculan las distancias (A.18) de la función objetivo.

*Espacio permitido* $q_A$: representa un área restringida donde todos los eslabones y uniones del mecanismo deben estar contenidos en la posición inicial y durante el movimiento del mismo. La violación se calcula como la mayor distancia $d(N_i(j), A)$ desde un nodo $N_i(j)$ que está fuera del espacio permitido $A$ hasta dicha área, y se debe contabilizar para el nodo de cada unión $i$ y cada posición precisa $j$:

$$q_A(\boldsymbol{x}, \boldsymbol{p}) = \max_{\substack{i=0,\dots n_J-1 \\ j=0,\dots n_{pp}-1}} < d(N_i(j), A) >, \quad \forall N_i(j) \subset \mathbb{R}^2 - A \tag{A.20}$$

*No inversión de ángulos de transmisión* $q_T$: esta restricción evita movimientos cinemáticamente inválidos entre dos posiciones precisas (a veces, una solución es geométricamente correcta pero cinemáticamente incorrecta). Se acumulan, para cada unión y cada posición precisa, las violaciones $\Delta\psi_k^j$ del ángulo entre eslabones $\psi_k^j$ a una zona angular permitida $(l_{\text{mín}}, l_{\text{máx}})$ definida por la posición de los eslabones en el instante inicial $\psi_k^0$.

$$q_T(\boldsymbol{x}, \boldsymbol{p}) = \sum_{k=0}^{n_J-1} \sum_{j=0}^{n_{pp}} \Delta\psi_k^j \tag{A.21}$$

La restricción minimiza el *riesgo de atascamiento*, evitando que un par cualquiera de eslabones atraviese una posición de centro muerto ("dead-center position") [Hal61, BC02b].

### Esquema general de resolución

Como se mencionó antes, un lazo sobre las SOCs permite hallar automáticamente la tabla SyDOFs con los datos del problema y el vector de multiplicidad $\boldsymbol{m}_{\text{máx}}$. Para resolver este problema utilizando un Algoritmo Genético (AG) simple [Gol89, Mic97], la *función de aptitud* es la función objetivo expresada en la ecuación (A.17) sujeta a las restricciones (A.19), (A.20) y (A.21) que se tienen en cuenta en forma de penalización:

$$F(\boldsymbol{x}, \boldsymbol{p}) = F^*(\boldsymbol{x}, \boldsymbol{p}) + Q(\boldsymbol{x}, \boldsymbol{p}), \tag{A.22}$$

donde el término $Q(\boldsymbol{x}, \boldsymbol{p})$ es la contribución de las restricciones a la función objetivo. Luego, *mejor aptitud* significa *tamaño mínimo de mecanismo con mejor cumplimiento de las restricciones*.

Como estrategia de manejo de las restricciones se utiliza la llamada *función de penalidad no estacionaria* de Joines y Houck [JH94] donde las restricciones son dinámicamente dependientes de la longitud de la búsqueda, esto es, del número de generación $t$:

$$Q(\boldsymbol{x}, \boldsymbol{p}) = (C \times t)^\alpha \sum_{k=0}^{n_q-1} q_k^\beta, \tag{A.23}$$

donde $C$, $\alpha$ y $\beta$ son constantes (de modo tal que la función $(C \times t)^{\alpha}$ es monotónicamente creciente en valor con $t$) y $n_q$ es el número de restricciones. Para usar esta estrategia se definen

$$q_0 = \lambda_L q_L, \quad q_1 = \lambda_T q_T, \quad \text{y} \quad q_2 = \lambda_A q_A.$$

Donde los valores por defecto de los parámetros de balanceo entre restricciones $\lambda_i$ se ajustan en términos de la *longitud característica*[1] $D$, y un valor de penalidad definido empíricamente por el usuario $\lambda_{\text{máx}}$. Por defecto, estos parámetros se fijan heurísticamente como sigue:

$$L_{\text{mín}} = \frac{D}{15}, \quad \lambda_L = \frac{\lambda_{\text{máx}0}}{2D}, \quad \lambda_T = \frac{\lambda_{\text{máx}1}}{2} \quad \text{y} \quad \lambda_A = \frac{\lambda_{\text{máx}2}}{D},$$

$$\text{con} \quad \lambda_{\text{máx}0} = \lambda_{\text{máx}1} = \lambda_{\text{máx}2} = \lambda_{\text{máx}} = 1000,$$

pero pueden ser cambiados por el usuario.

La evaluación de la función objetivo así como de las restricciones para un individuo, pueden arreglarse en forma vectorial como:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{m}) = [F^* \quad \lambda_L q_L \quad \lambda_T q_T \quad \lambda_A q_A]^T.$$

Nótese que la evaluación se realiza para un conjunto de soluciones elegido mediante $\boldsymbol{m}$.

Adicionalmente, si no existiera solución analítica para los parámetros libres propuestos, los resolvedores de SOCs están programados de modo tal que responden con

$$\boldsymbol{f}_{\text{non\_sol}}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{m}) = [\lambda_{\text{máx}} \quad \lambda_{\text{máx}0} \quad \lambda_{\text{máx}1} \quad \lambda_{\text{máx}2}]^T$$

ya que ni la función objetivo ni las restricciones podrían ser calculados.

La evaluación de la función objetivo y las restricciones de toda la población pueden escribirse en forma matricial como:

$$\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{m}) = [\boldsymbol{f}_1 \quad \boldsymbol{f}_2 \quad \dots \quad \boldsymbol{f}_i \quad \dots \quad \boldsymbol{f}_P]^T.$$

Usando esta forma matricial, la evaluación de la aptitud de un simple individuo $\boldsymbol{x}_i$ se hace como:

$$F(\boldsymbol{x}_i, \boldsymbol{m}) = \begin{cases} F^*(\boldsymbol{x}_i, \boldsymbol{p}) & \text{si } \boldsymbol{x}_i \text{ tiene una solución factible} \\ F^*(\boldsymbol{x}_i, \boldsymbol{p}) + Q(\boldsymbol{x}_i, \boldsymbol{p}) & \text{si } \boldsymbol{x}_i \text{ tiene una solución infactible} \\ Q_{\text{máx}} = \lambda_{\text{máx}} + \sum_{k=0}^{n_q - 1} \lambda_{\text{máx}k} & \text{si } \boldsymbol{x}_i \text{ no tiene solución.} \end{cases} \tag{A.24}$$

El procedimiento de dimensionado inicial se muestra en el Algoritmo 2. En primer lugar, se realiza un lazo externo sobre las combinaciones de soluciones, ver líneas 2 y 33. Luego, una distinción entre un problema sin y con parámetros libres bifurca el algoritmo a dos procedimientos distintos:

[1] La *longitud característica* se calcula como la diagonal de un rectángulo evolvente ("bounding box") de los datos geométricos, o sea, de todas las posiciones de nodos definidos en las partes iniciales.

**Algorithm 2** Algoritmo de dimensionado inicial

```
1: bool foundSol=false
2: while nextMCombination(m,m_máx,s) do
3:   if x = [∅] then {—Caso sin parámetros libres—}
4:     SyDOFsTMP := SyDOFs
5:     bool found = true
6:     for k = 0 to s − 1 do {Lazo sobre SOCs}
7:       found = found and SOC[k].Solve(SyDOFsTMP,m)
8:     end for
9:     if found=true then {Existe solución analítica}
10:      foundSol=SelectOptimum(SyDOFsTMP,f,m,f_best,m_best)
11:    end if
12:  else {—Caso con parámetros libres—}
13:    for t = 0 to t_máx do {Lazo sobre generaciones}
14:      if t=0 then
15:        X^t ← initializeGA(n_vars, n_bits, P, x_mín, x_máx, n_constr, f_goal, p_cross, p_mut)
16:      else
17:        X^t ← evolveGA(t, X^{t−1}, F(X^{t−1}, m))
18:      end if
19:      for i = 0 to P do {Lazo sobre la evaluación de la aptitud de los individuos}
20:        SyDOFsTMPi:=SyDOFs
21:        x_i = X^t(i, :)
22:        bool found = true
23:        for k = 0 to s − 1 do
24:          found = found and SOC[k].Solve(SyDOFsTMPi,m,x_i)
25:        end for
26:        if found=true then
27:          foundSol=SelectOptimum(SyDOFsTMPi,x_i,f_i,m, x_best,f_best,m_best)
28:        end if
29:        F^t(i, :) = f_i {Esta es la matriz de evaluaciones F(X^t, m)}
30:      end for individuos
31:    end for generaciones
32:  end if
33: end while
34: if foundSol then
35:   Salvar, ejecutar el análisis cinemático, exportar y mostrar resultados.
36: else
37:   No se halló solución.
38: end if
```

- El primer procedimiento (líneas 4-11) consiste de un lazo que resuelve SOCs. Si la solución analítica existe, una función `SelectOptimum` evalúa la función objetivo y las restricciones, y luego compara su aptitud $\boldsymbol{f}$ con el mejor ya guardado $\boldsymbol{f}_{\text{best}}$. Nótese que el vector de elección de soluciones $\boldsymbol{m}_{\text{best}}$ también es guardado.

- Un segundo procedimiento (líneas 13-31) asume la existencia de parámetros li-

bres, para los cuales se genera, en forma aleatoria, una población de individuos utilizando la función `initializeGA`. Luego, se ejecuta iterativamente el algoritmo genético mediante la función `evolveGA` hasta que se alcance un número máximo de generaciones. La función `SelectOptimum` evalúa cada individuo y guarda aquel con mejor aptitud $\{\boldsymbol{x}_{\text{best}}, \boldsymbol{f}_{\text{best}}, \boldsymbol{m}_{\text{best}}\}$.

Luego de analizar todas las combinaciones, si se halló solución, en la línea 35 se realiza una evaluación final de la mejor solución y luego se muestran los índices de comportamiento, o sea, los valores de la función objetivo y de cada restricción.

### Caso sin parámetros libres

Continuando con el problema de seguimiento de trayectoria, la primer topología sintetizada, la **Alternativa 0**, es un mecanismo de cuatro barras. Previo a la ejecución del resolvedor y mediante la función `Assemble` de cada SOC, se obtiene su tabla SyDOFs su vector de multiplicidades $\boldsymbol{m}_{\text{máx}}$. Ya que la tabla SyDOFs no presenta parámetro libre alguno, se resuelve en forma analítica para las distintas combinaciones de soluciones. Como se observa en la Figura A.10, la topología se descompone en dos díadas pasando por tres posiciones, que, como se mostró en el ejemplo de la Sección A.7.1, tendrá dos soluciones. Luego, el vector de multiplicidades $\boldsymbol{m}_{\text{máx}} = [2, 2]$ permite generar cuatro subproblemas, a saber:

$$\boldsymbol{M}(\boldsymbol{m}_{\text{máx}}) = \begin{array}{c} \begin{array}{cc} \textbf{SOC 0} & \textbf{SOC 1} \end{array} \\ \left[\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array}\right] \end{array}. \tag{A.25}$$

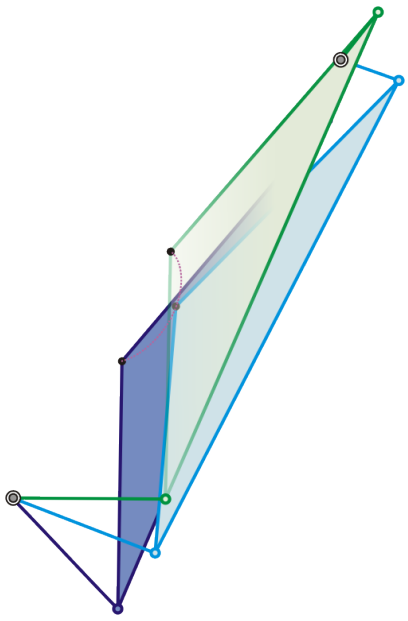


Figura A.12: Problema de seguimiento de trayectoria: solución para el dimensionado inicial de la **Alternativa 0**. Se muestra la curva generada.

El mejor resultado se muestra en la Figura A.12. Por claridad del dibujo, la manivela original no se muestra.

### Caso con parámetros libres

La siguiente solución topológica, la **Alternativa 1** mostrada en la Figura A.10, tiene una topología de tipo Watt-II con una fundación ternaria y consecuentemente con un nuevo pivote. En esta misma figura se muestra que su descomposición resulta en tres SOCs.

La tabla SyDOFs es algo grande para ser mostrada aquí, pero desde ella se identifican seis parámetros libres. Se deben ajustar cuatro parámetros libres en dos eslabones de la primer tríada, la SOC 0. Luego, se necesitan dos parámetros libres adicionales para definir una caja que será frontera del nuevo pivote. Este pivote es calculado por el último módulo, la SOC 2. El Cuadro A.1 presenta la descripción de los parámetros libres y fronteras por defecto para los mismos.

| | SyDOF | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parám. libre | Tipo | ID | Fís. | Pos. Prec. | SOC | mín | máx | mejor |
| 0 | Elem | 7 | L | 1 | 0 | -1 | 1 | -0.54872 |
| 1 | Elem | 7 | L | 2 | 0 | -2 | 2 | -0.90696 |
| 2 | Elem | 9 | L | 1 | 0 | -1 | 1 | 0.052503 |
| 3 | Elem | 9 | L | 2 | 0 | -2 | 2 | -0.36484 |
| 4 | Nodo | 7 | x | 0 | 2 | 0 | 1.2 | 0.20923 |
| 5 | Nodo | 7 | y | 0 | 2 | 0 | 1.6 | 0.72283 |

Cuadro A.1: Parámetros libres para la **Alternativa 1** del problema de seguimiento de trayectoria.

Para el cálculo de este problema, se utilizaron las fronteras por defecto mostradas en el Cuadro A.1, donde las rotaciones son dadas en radianes. Para la ejecución del resolvedor, los parámetros de optimización utilizados fueron:

**Restricciones**: mínima longitud de eslabón $L_{\text{mín}} = 0{,}20$; tolerancia del ángulo de transmisión $\delta = 10°$;

**Resolvedor AG**: tamaño de población $P = 70$, máximo número de generaciones $t_{\text{máx}} = 90$, probabilidad de cruza $p_{\text{cross}} = 0{,}5$, probabilidad de mutación $p_{\text{mut}} = 0{,}01$, número de bits $n_{\text{bits}} = 12$, penalidad $\lambda_{\text{máx}} = 1000$.

Los resultados muestran un cumplimiento completo de las restricciones (mínima longitud de eslabones y no inversión de ángulos de transmisión, ambas nulas; no hay restricción de espacio), y un valor de 0,058002 alcanzado por la función objetivo. Los valores de las variables para la mejor solución se muestran en la última columna del Cuadro A.1 y su mecanismo correspondiente en la Figura A.13.

En la Figura A.14 se compara, para cada subproblema, la evolución del mejor individuo teniendo en cuenta las restricciones. El mínimo se retiene desde la última combinación.

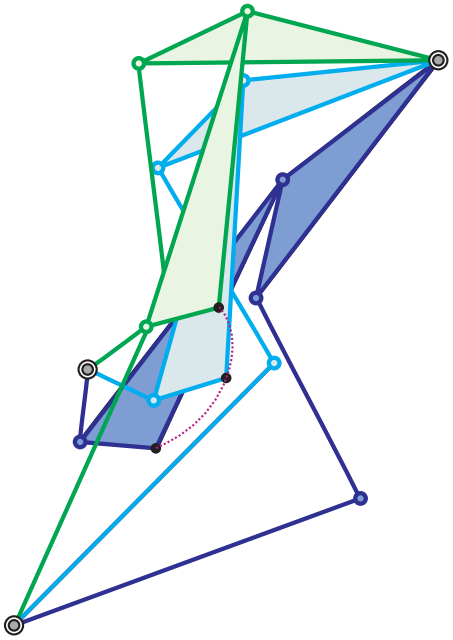


Figura A.13: Problema de seguimiento de trayectoria: solución para el dimensionado inicial de la **Alternativa 1**. Se muestra la curva generada.

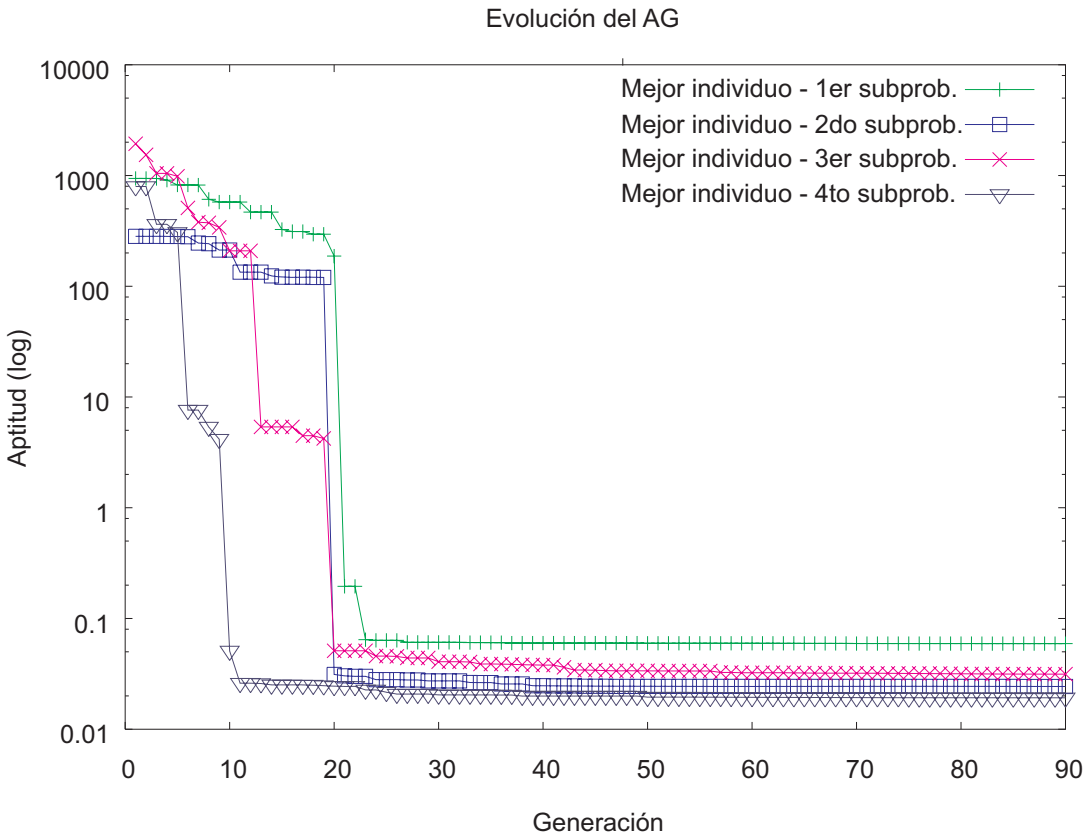



Figura A.14: Comparación entre las evoluciones de los mejores individuos para cada combinación de soluciones (cuatro subproblemas).

### A.7.4. Síntesis de mecanismos flexibles

La síntesis de mecanismos flexibles se presenta como una de las posibles extensiones de la metodología presentada. Para la síntesis dimensional se utiliza en forma inversa el Modelo de Cuerpo Pseudorígido ("Pseudo Rigid-Body Model (PRBM)") presentado por Howell [How01] para simular, en forma simplificada, grandes deflexiones de vigas usadas como miembros componentes de mecanismos flexibles.

La síntesis de tipo para mecanismos con miembros flexibles es descripta en el apartado de enumeración de la tesis. Particularmente, un atlas de mecanismos flexibles se caracteriza por poseer una gran variedad de alternativas. Por ejemplo, un atlas muy útil resulta de la asignación no isomorfa de los siguientes elementos a una cadena cinemática:

- Asignar desde el alfabeto $\{0 = \text{fundación}, 1 = \text{rígido}, 2 = \text{flexible}\}$ los tipos de eslabones, restringidos a que siempre exista la fundación, y
- Asignar desde el alfabeto $\{1 = \text{rotoidal}, 3 = \text{flexible}, 4 = \text{fija}\}$ los tipos de uniones, restringidos a que una unión fija no puede conectar dos cuerpos rígidos, la fundación inclusive.

La asignación en una topología de cuatro barras resulta en 211 mecanismos y más de 100.000 en cada topología de seis barras (incluyendo los rígidos).

Para la síntesis dimensional se aprovechó el resolvedor de mecanismos rígidos y se aplicó convenientemente la técnica de Reemplazo de Cuerpo Rígido ("Rigid-Body Replacement" [How01]) para sintetizar miembros flexibles binarios con el agregado de restricciones para limitar la rotación entre cuerpos conectados por uniones fijas y flexibles.

#### Síntesis mediante Reemplazo de Cuerpo Rígido

Después de calcular la síntesis dimensional, el conjunto de lazos mínimos independientes de la topología del mecanismo rígido se utiliza para visitar eslabones y analizar su factibilidad de ser transformados en sus equivalentes flexibles. Los eslabones se analizan por grupos, de a tres consecutivos. Cuando se encuentra a un eslabón flexible, se toman la decisiones acerca de cambiar las coordenadas de sus puntos extremos considerando no sólo el tipo de unión en que termina, sino también el tipo de eslabón al que se conecta. Dentro de este análisis, la fundación y los eslabones rígidos son igualmente considerados como "rígidos".

Sea $l_i$ el eslabón flexible considerado, y sean $l_{i-1}$ y $l_{i+1}$ los eslabones adyacentes. Además, sea $R$ la longitud del eslabón rígido, y $L$ la longitud de su equivalente flexible. El *radio característico* $\gamma = R/L$ se toma con un valor fijo de $\gamma = 0{,}8517$.

Luego de utilizar el resolvedor de mecanismos rígidos en forma exitosa, es posible sintetizar los miembros binarios flexibles identificando los casos mostrados en la Figura A.15 y asignando propiedades de sección transversal e inercia a los mismos.

En cuanto a los tipos de uniones pertenecientes a mecanismos flexibles, se identifican dos grupos: (a) uniones fijas ("clamped") entre un cuerpo flexible y otro cuerpo, rígido o flexible; y (b) uniones flexibles que desarrollan un par antagonista al giro relativo entre los cuerpos que conecta, por ejemplo, vigas pequeñas ("small-length flexural pivots" y "living hinges" [How01]). Para ambas uniones, se debe limitar el giro relativo entre los cuerpos que conectan. Las violaciones a un ángulo máximo

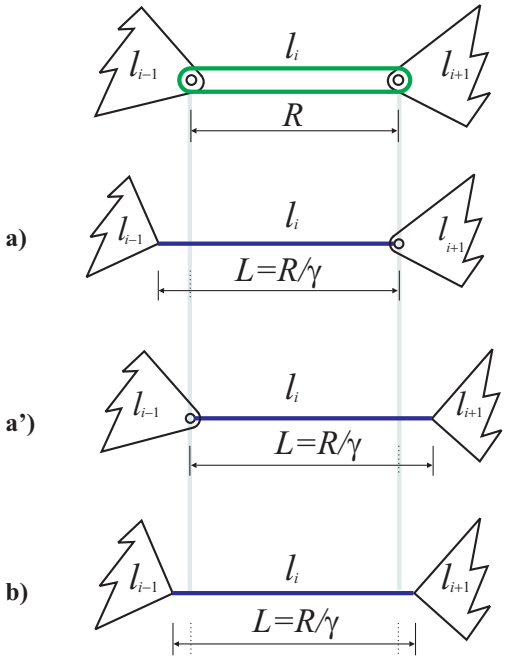



Figura A.15: Reemplazos de Cuerpo Rígido: a) Fija-Rotoidal, a') Rotoidal-Fija, b) Fija-Fija.

admitido se consideran en forma de penalización de la función objetivo de un modo similar que la restricción de no inversión de ángulos de transmisión.

Luego de ejecutar el cálculo de síntesis, se dispone de la información relativa a la rotación de los eslabones para un número de posiciones precisas. De este modo, se puede medir la rotación máxima entre dos eslabones sobre la unión rotoidal que será el pivot característico del PRBM del eslabón flexible. Esta rotación no refleja exactamente lo que ocurre en los eslabones flexibles (compárese $\theta_{\text{rígido}}$ de la Figura A.16-a con $\theta_{\text{fija}}$ de la Figura A.16-b) pero es una aproximación muy útil para desarrollar una nueva restricción: el *ángulo límite para uniones fijas y flexibles* denotada como $\Theta_{\text{máx}}$.

Un ejemplo de violación al ángulo límite ocurre en la tercer posición $\psi_k^3$ que se ilustra en la Figura A.17 donde se muestra la contribución de $\Delta\psi_k^3$ a la restricción.

Entre las desventajas de los reemplazos puede destacarse que:

↓ Después de la transformación de eslabones, los eslabones flexibles son más largos que sus equivalentes rígidos, de este modo pueden producir un peor cumplimiento de las restricciones de *espacio permitido* y *mínima longitud de eslabón* (véase la Figura A.16).

↓ Los eslabones que rotan completamente ($2\pi$) no pueden ser reemplazados.

Algunas ventajas de incluir segmentos flexibles (mientras se reemplazan también uniones cinemáticas) en los mecanismos tradicionales, son bien conocidas. Bajo el alcance de los eslabonamientos planos, estas ventajas son:

✓ Incremento en precisión por eliminación juegos.

✓ Reducción del mantenimiento por la eliminación del desgaste y las necesidades de lubricación.

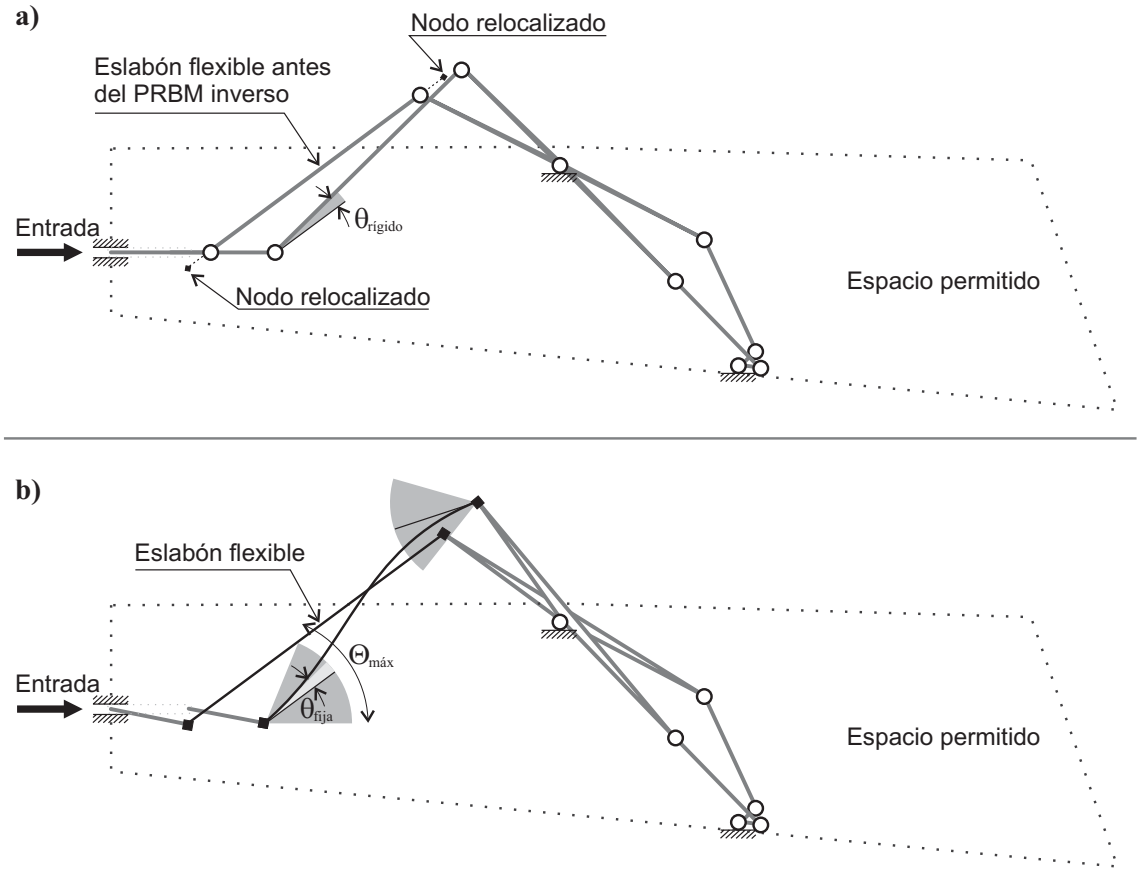



Figura A.16: Ángulo límite para los extremos fijos de segmentos flexibles: a) Mecanismo rígido, b) Reemplazo.

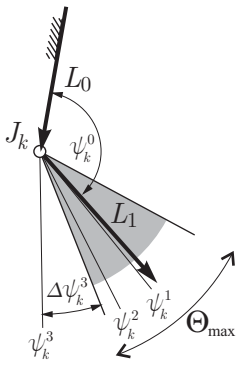



Figura A.17: Ejemplo de violación de ángulo límite para uniones fijas y flexibles.

- ✓ Reducción de partes comparados a los diseños rígidos. Por ejemplo, en el mecanismo rígido mostrado en la Figura A.16 se eliminan dos pares cinemáticos rotoidales.
- ✓ Reducción de costos de manufacturas y del tiempo de ensamblaje.
- ✓ Incremento en la posibilidad de miniaturización y manufactura en una capa.

Basados en la experiencia adquirida con el ejemplo presentado en la tesis, vale la pena agregar la siguiente ventaja:

✓ Capacidad de realizar movimientos no lineales similares a los realizados por los mecanismos articulados.

La principal desventaja de la inclusión de miembros flexibles es la

↓ Reducción de la vida útil por fatiga.

El método propuesto tiene dos potenciales aplicaciones: a) Reemplazo de partes rígidas de un mecanismo por miembros flexibles para obtener mecanismos flexibles (eventualmente, con nuevos comportamientos cinemáticos) y b) Diseño de mecanismos biestables.

## A.8. Contribuciones de la tesis

Los aspectos teóricos originales que se presentan en esta tesis son:

- Desarrollo de un *identificador de isomorfismo de mecanismos* y su utilización en la enumeración de cadenas cinemáticas y diversos atlas de mecanismos.
- Enumeración exhaustiva de topologías utilizando búsqueda de subgrafos para satisfacer requerimientos estructurales desde el principio del proceso de diseño.
- Descomposición automática de las topologías cerradas en cadenas abiertas para resolver la síntesis dimensional de las mismas utilizando expresiones analíticas.
- Para la síntesis dimensional, se calcula la síntesis de las cadenas abiertas utilizando números complejos. En los casos en que existen parámetros libres se aplica un esquema de optimización de orden cero basado en Algoritmos Genéticos con penalización de restricciones para hallar el conjunto de valores de dichos parámetros que hacen mínimo el tamaño de los eslabones, sujeto a restricciones que fuerzan el cumplimiento de un tamaño mínimo de eslabón, espacio permitido, y singularidad de movimiento.
- Se desarrollan y analizan las modificaciones necesarias para extender la metodología para diseñar mecanismos flexibles utilizando métodos de Reemplazo de Cuerpo Rígido.

## A.9. Conclusiones

Se desarrolló una metodología implementada en un software que permite sintetizar mecanismos de eslabonamientos planos que responden a requerimientos cinemáticos definidos por el usuario *comenzando desde cero*. Ya sea en el nivel estructural como dimensional, las soluciones a esta clase de problemas inversos distan de ser únicas.

La principal contribución de esta tesis es el enfoque sistemático para la síntesis. Con respecto a la **síntesis de tipo**, se han desarrollado tres herramientas computacionales: un identificador de mecanismos denominado *Matriz de Adyacencia de Tipo* ("Type Adjacency Matrix"); un identificador de isomorfismos denominado *Código de Grado Extendido a la Diagonal calculado por Filas* ("Diagonally Extended Degree

Code computed by Rows"), y una *Matriz de Adyacencia de Síntesis* ("Synthesis Adjacency Matrix") para la identificación de mecanismos funcionalmente diferentes y la localización del subgrafo de las partes iniciales en un mecanismo tomado del atlas.

Estas herramientas facilitaron el desarrollo de los dos procedimientos primordiales de la síntesis de tipo: la enumeración de cadenas cinemáticas y diversos atlas de mecanismos, y la enumeración exhaustiva de topologías utilizando búsqueda de subgrafos para satisfacer requerimientos estructurales desde el principio del proceso de diseño.

Usando esta herramienta computacional fue posible explorar cientos de potenciales mecanismos en pocos minutos y encontrar alternativas que automáticamente se corresponden con la tarea –sin incurrir en repeticiones.

La enumeración de mecanismos flexibles de cuatro barras fue validada con los resultados de la literatura y se hallaron nuevos resultados para los mecanismos de seis barras. La enumeración de mecanismos, así como la enumeración en la búsqueda de subgrafos, es completamente exhaustiva, determinística y adecuada para la potencia actual de las computadoras personales modernas.

En cuanto a la **síntesis dimensional** se propuso la descomposición modular de cada topología como el paso clave entre las etapas de síntesis de tipo y dimensional para aplicar métodos analíticos. Una descripción de Elementos Finitos de la topología en conjunto con teoremas de Teoría de Grafos facilitaron la identificación de las dimensiones significativas del mecanismo. Se utilizaron varios módulos de *Cadenas Abiertas Simples* para vincular los datos y las incógnitas del problema con sus correspondientes resolvedores de *Ecuaciones de Lazos Cerrados*.

Estos módulos permitieron sistematizar la etapa de dimensionado inicial donde la multiplicidad de soluciones y los parámetros libres son identificados automáticamente.

Se definió un criterio de optimalidad del mecanismo basado en el *tamaño mínimo* sujeto a distintas restricciones para evitar *dimensiones mínimas*, *singularidad del movimiento* y violación del *espacio permitido*. Un Algoritmo Genético simple con una estrategia de penalidad no estacionaria para el manejo de restricciones se utilizó para encontrar el mejor conjunto de parámetros libres cuya evaluación resulte en el mejor cumplimiento del criterio de optimalidad. El algoritmo mostró robustez para encontrar la mejor solución utilizando un número moderado de evaluaciones ($\approx 5000$). Sin embargo, en algunos casos fue necesario ajustar el valor adecuado de penalidad para una ocurrencia sin solución y volver a ejecutar el resolvedor.

Un ejemplo de prueba simple consistente de un generador de trayectoria fue ejecutado a lo largo de la tesis mostrando la utilidad para explorar el espacio factible de soluciones. Además, se resolvieron problemas de síntesis de mecanismos para la retracción de un tren de aterrizaje, comando de alerones de alas de avión y aletas de toberas de turbinas.

Esta herramienta computacional pertenece a la etapa de diseño más primitiva: el *Diseño Conceptual*. También puede ser usada para tres actividades tediosas, para las cuales el software académico y comercial es escaso: (1) completar un mecanismo existente, para realizar alguna tarea adicional, (2) rediseño de un mecanismo, y (3) diseño de nuevos mecanismos.

Se espera que los problemas de prueba presentados sirvan de *benchmarks* para la síntesis de tipo y dimensional de mecanismos de eslabonamientos planos con uniones simples.

Un prototipo de software fue integrado a un sistema de análisis de mecanismos y estructuras por elementos finitos. La definición de datos se hace en forma gráfica en el propio programa CAD del sistema general de análisis. Después de la tarea de síntesis, los diseñadores pueden verificar la validez de la solución por medio del *análisis* y pueden, además, *optimizar* la solución manteniendo la misma topología.

### A.9.1. Futuras investigaciones

Para extender las capacidades de la metodología propuesta pueden mejorarse distintos aspectos.

En lo referente a los *datos de entrada*, es crucial permitir la definición de (i) tareas continuas; (ii) un *espacio permitido* para cada posición precisa y, en particular, poder definir áreas poligonales complejas para localización de pivotes y obstáculos; (iii) tareas cinemáticas para segmentos flexibles; (iv) uniones cinemáticas más complejas, por ejemplo, deslizadores curvos[1].

En el nivel del resolvedor de *síntesis de tipo*, el atlas de cadenas cinemáticas puede ser extendido a 10, 12 y 14 eslabones, y desde éstos es también posible generar atlas para uniones múltiples. De este modo, se podrían incorporar uniones de deslizador ("pin-slider joints"). Un atlas de mecanismos con uniones múltiples puede ser útil, además, para generar otros atlas de mecanismos sólo con miembros flexibles binarios. Todas las restricciones para la generación de los atlas pueden expresarse en términos del formalismo de la Teoría de Grafos. Varios investigadores han mostrado la aplicabilidad de TG para la enumeración de mecanismos de *eslabonamientos con levas* ("cam-linkages"), *eslabonamientos con engranajes* ("gear-linkages"), *trenes de engranajes* y *mecanismos tridimensionales*.

La interactividad con este resolvedor también puede mejorarse mediante el desarrollo de interfases de usuario más amigables que el *script* primitivo.

El *algoritmo de descomposición* puede refinarse aún más, considerando más caminos de descomposición que los que se definieron en el enfoque lazo por lazo que se presentó.

Los desarrollos en la *síntesis dimensional* serán dependientes de los atlas de mecanismos disponibles. Por ejemplo, considerando topologías con uniones múltiples (los deslizadores rectos y curvos tienen eslabones con tamaños de eslabón nulos), se necesitarán desarrollar los módulos de cadenas abiertas y los resolvedores adecuados. Los Métodos de Puntos Precisos también están disponibles para ser implementados computacionalmente para sintetizar eslabonamientos con levas y engranajes [SE84].

Para el diseño de cualquier *mecanismo óptimo* es necesario tomar en cuenta múltiples objetivos y restricciones. Entre los objetivos más útiles faltantes en esta tesis están (i) la minimización del error cinemático en la tarea continua, (ii) la maximización de la *ventaja mecánica*, y (iii) la optimización del *ángulo de transmisión*. Con respecto a las restricciones, los más importantes son (i) la eliminación de defectos de *rama* y *circuito* y (ii) la prescripción de manivelas con rotación completa, lo cual se aplica en mecanismos para maquinaria de producción en masa (por ejemplo, en maquinaria textil o de empaquetamiento de alimentos).

Con respecto a la *amigabilidad* del software desarrollado para el dimensionado inicial, los esfuerzos se enfocarán en la definición interactiva de las fronteras de las

[1] Estas características ya son permitidas por el programa de análisis Samcef Field–Implicit Non-Linear, pero la capacidad en el resolvedor Oofelie no fue desarrollada aún.

variables, el dibujo de las curvas deseadas y generadas, y la minimización del número de sesiones. Actualmente, se necesitan tres sesiones sucesivas: una para definir el problema cinemático desde la cual se ejecuta el resolvedor de *síntesis de tipo*; una para cada topología hallada, para la cual se ejecuta el resolvedor de *dimensionado inicial*; y una sesión final para animar la solución.

Finalmente, una meta ambiciosa es la comparación automática de todos los mecanismos obtenidos desde las etapas de síntesis. Dos problemas de prueba resueltos manualmente se podrían tomar para validar el método, el diseño de un *motor de carrera variable* de Freudenstein y Maki [CP05, FM83] y un *eslabonamiento para operar una ventana de bisagras* [SE84]. Para ambos problemas se requiere de un atlas de mecanismos con uniones de deslizador.

# A. Resumen extendido (extended abstract in Spanish)

# List of Figures

# List of Tables

# Bibliography


[AA86] A. G. Ambekar and V. P. Agrawal. On canonical numbering of kinematic chains and isomorphism problem: max Code. In *ASME Design Engineering Technical Conference*, 1986. 25

[AA87a] A. G. Ambekar and V. P. Agrawal. Canonical numbering of kinematic chains and isomorphism problem: min Code. *Mechanism and Machine Theory*, 22(5):453–461, 1987. 25

[AA87b] A. G. Ambekar and V. P. Agrawal. Identification of kinematic chains, mechanisms, path generators and function generators using min Codes. *Mechanism and Machine Theory*, 22(5):463–471, 1987. 25

[AFPC07] A.E. Albanesi, V.D. Fachinotti, M.A. Pucheta, and A. Cardona. Synthesis of compliant mechanisms for segment-motion generation tasks. In S.A. Elaskar, E.A. Pilotta, and G.A. Torres, editors, *Mecánica Computacional*, volume XXVI, pages 2919–1930, Córdoba, October 2007. XVI Congreso sobre Métodos Numéricos y sus Aplicaciones, ENIEF 2007, I Congreso de Matemática Aplicada, Computacional e Industrial MACI 2007, AMCA. 142

[Ang97] J. Angeles. A fin-de-siecle view of TMM. In *International Conference on Mechanical Transmissions and Mechanisms (MTM'97)*, Tianjin, China, July 1997. 7, 8, 162

[BC02a] S. S. Balli and S. Chand. Defects link mechanisms and solution rectification. *Mechanism and Machine Theory*, 37(9):851–876, 2002. 91, 183

[BC02b] S. S. Balli and S. Chand. Transmission angle in mechanisms (triangle in mech). *Mechanism and Machine Theory*, 37(2):175–195, 2002. 98, 100, 185

[BCCA07] J. S. Bourrelle, C. Chen, S. Caro, and J. Angeles. Graphical user interface to solve the burmester problem. In *12th IFToMM World Congress*, Besaçon, France, June 18–21 2007. 89, 90

[BER01] S. Braun, D. Ewins, and S. S. Rao, editors. *Encyclopedia of Vibration*, volume 2, chapter *Computational Methods: Object Oriented Programming in FE Analysis* by I. Klapka, A. Cardona and P. Devloo, pages 967–966. Academic Press Ltd., September 2001. (3 Volumes). 12

[BF70] F. Buchsbaum and F. Freudenstein. Synthesis of kinematic structures of geared kinematic chains and other mechanisms. *Journal of Mechanisms*, 5(3):357–392, 1970. 25

[BH05] E. A. Butcher and C. Hartman. Efficient enumeration and hierarchical classification of planar simple-jointed kinematic chains: Application to 12- and 14-bar single degree-of-freedom chains. *Mechanism and Machine Theory*, 40(9):1030–1050, September 2005. 31, 32

[BMH00] M.D. Berglund, S.P. Magleby, and L.L. Howell. Design rules for selecting and designing compliant mechanisms for rigid-body replacement synthesis. In *Proceedings of the 2000 ASME Design Engineering Technical Conferences, DETC2000/DAC-14225*, Baltimore, Maryland, September 2000. 139

[Car89] A. Cardona. *An Integrated Approach to Mechanism Analysis*. PhD thesis, Université de Liège, Belgium, 1989. 8, 161

[Car02] A. Cardona. Computational methods for synthesis of mechanisms. Technical report, CIMEC-INTEC, 2002. 52, 90

[CCSP07] F. Cugnon, A. Cardona, A. Selvi, and C. Paleczny. Synthesis and optimization of flexible mechanisms. In C.L. Bottasso, P. Masarati, and L. Trainelli, editors, *Multibody Dynamics 2007*, ECCOMAS Thematic Conference on Multibody Dynamics, pages 156–157, Milan, Italy, June 2007. Politecnico di Milano. 11, 90, 164

[Cec00] M. Ceccarelli, editor. *International Symposium on History of Machines and Mechanisms: Hmm2000*, Dordrecht, January 2000. University of Cassino, Kluwer Academic Pub. 7

[Cec03] M. Ceccarelli. Mechanism classifications over the time: An illustration survey. In *Bernie Roth Symposium*, CA, June 2003. Stanford. 7

[Cec04] M. Ceccarelli. Evolution of TMM (Theory of Machines and Mechanisms) to MMS (Machine and Mechanism Science): An Illustration Survey. In *Keynote Lecture*, Tianjin, China, 2004. 7

[CFG96] E. Ceresole, P. Fanghella, and C. Galletti. Assur's groups, AKCs, basic trusses, SOCs, etc.: Modular kinematics of planar linkages. In *ASME Design Engineering Technical Conferences, DETC/MECH-1027*, Irvine, California, USA, 1996. Paper DETC2002/MECH-34369. 69

[CFH+06] Y. Chen, P. Feng, B. He, Z. Lin, and Y. Xie. Automated conceptual design of mechanisms using improved morphological matrix. *Journal of Mechanical Design*, 128(3):516–526, 2006. 11, 164

[CK99] S.-J. Chiou and S. Kota. Automated conceptual design of mechanisms. *Mechanism and Machine Theory*, 34(3):467–495, 1999. 7, 161

[CKG94] A. Cardona, I. Klapka, and M. Géradin. Design of a new finite element programming environment. *Engineering Computations*, 11(4):365–381, 1994. 12

[CKv] H. A. Crone, A. J. Klein Breteler, and K. van der Werff. *TADSOL Type And Dimension Synthesis Of Linkages*. Delft University of Technology, Netherlands. Web page: http://www.ocp.tudelft.nl/tt/cadom. 8, 162

[CO02] J. B. Cook and D. G. Olson. The design of a 10-bar linkage for four functions using SyMech. In *ASME Design Engineering Technical Conferences 2002*, Montreal, Quebec, Canada, September 2002. Paper DETC2002/MECH-34369. 8, 90, 162

[Col07] J.-F. Collard. *Geometrical and Kinematic Optimization of Closed-Loop Multibody Systems*. PhD thesis, Université Catholique de Louvain - Faculté des Sciences Appliquées, Louvain-la-Neuve, Belgium, 2007. 90

[Coo] J. B. Cook. *SyMech* - synthesis software for *Pro/E*. Web page: http://www.symech.com. 8, 162

[CP05] D.-Z. Chen and W.-M. Pai. A methodology for conceptual design of mechanisms by parsing design specifications. *ASME Journal of Mechanical Design*, 127(6):1039–1044, 2005. 11, 20, 155, 164, 197, 199

[CSP02] J. A. Cabrera, A. Simon, and M. Prado. Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and Machine Theory*, 37(10):1165–1177, 2002. 90

[DF67] L. Dobrjanskyj and F. Freudenstein. Some applications of Graph Theory to the structural analysis of mechanisms. *ASME Journal of Engineering for Industry*, 89:153–158, February 1967. 25

[DK02] H. Draijer and F. Kokkeler. Heron's synthesis engine applied to linkage design the philosophy of WATT software. In *ASME Design Engineering Technical Conferences 2002*, Montreal, Quebec, Canada, September 2002. 90

[DK07] H. Draijer and F. Kokkeler. *WATT Mechanism Suite*, 2007. Heron Technologies. Web page: http://www.heron-technologies.com/watt. 8, 162

[EG77] A. G. Erdman and J. Gustafson. LINCAGES: A Linkage Interactive Computer Analysis and Graphically Enhanced Synthesis Package. *ASME Paper No. 77-DTC-5*, 1977. 8, 162

[Erd] A. G. Erdman. *LINCAGES 2000*. University of Minnesota, USA. Web page: http://www.me.umn.edu/labs/lincages. 8, 162

[ES97] A. G. Erdman and G. N. Sandor. *Mechanism Design: Analysis and Synthesis*, volume 1. Prentice-Hall, New Jersey, 3rd edition, 1997. 3, 7, 8, 15, 57, 62, 63, 89, 160, 161, 162, 166, 175, 177

[FBAAA05] I. Fernández-Bustos, J. Aguirrebeitia, R. Avilés, and C. Angulo. Kinematical synthesis of 1-dof mechanisms using finite elements and genetic algorithms. *Finite Elements in Analysis and Design*, 41(15):1441–1463, 2005. 90

[Fer62] E. S. Ferguson. Kinematics of mechanisms from the time of Watt. *United States National Museum Bulletin*, 228(27):185–230, 1962. 6

[FM79] F. Freudenstein and E. R. Maki. Creation of mechanisms according to kinematic structure and function. *Journal of Environmental and Planning B*, 6:375–391, 1979. 10, 19, 20, 34, 163

[FM83] F. Freudenstein and E. R. Maki. Development of an optimum variable-stroke internal-combustion engine mechanism from the viewpoint of kinematic structure. *ASME Journal of Mechanical Design*, 105:259–267, 1983. 155, 197

[FS59] F. Freudenstein and G. N. Sandor. Synthesis of path-generating mechanisms by means of a programmed digital computer. *ASME Journal of Engineering for Industry*, 81(2):159–168, 1959. 8, 57, 90, 161

[GC01] M. Geradin and A. Cardona. *Flexible Multi-Body Dynamics. A Finite Element Approach.* John Wiley & Sons, 2001. 8, 37, 161

[Gol89] D. E. Goldberg. *Genetic Algorithms in search, optimization, and machine learning.* Addison-Wesley, 1989. 101, 104, 185

[Hal61] A. S. Hall (Jr.). *Kinematics and Linkage Design.* Waveland Press, U.S.A., 1961. 8, 57, 98, 100, 162, 185

[Han93] M. R. Hansen. A multi level approach to synthesis of planar mechanisms. In *Computer Aided Analysis of Rigid and Flexible Mechanical Systems*, Troia, Portugal, 1993. 90

[Har69] F. Harary. *Graph Theory.* Addison-Wesley Series in mathematics, 1969. 72, 74

[HCE00] J. E. Holte, T. R. Chase, and A. G. Erdman. Mixed exact-approximate position synthesis of planar mechanisms. *ASME Journal of Mechanical Design*, 122(3):278–286, September 2000. 5

[HD64] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages.* McGraw-Hill, New York, 1964. 3, 6, 7, 8, 14, 57, 62, 160, 161, 162, 166, 175, 177

[HH92] W.-M. Hwang and Y.-W. Hwang. Computer-aided structural synthesis of planar kinematic chains with simple joints. *Mechanism and Machine Theory*, 27(2):189–199, 1992. 31

[How01] L.L. Howell. *Compliant Mechanisms.* John Wiley & Sons, New York, 2001. 19, 24, 36, 139, 140, 142, 191, 201

[Hsi92] H. I. Hsieh. Sistematic methodologies for the automatic enumeration of topological structures of mechanisms. Master's thesis, University of Maryland, USA, 1992. 31

[HZLW82] P. R. He, W. J. Zhang, Q. Li, and F. X. Wu. A new method for detection of graph isomorphism based on the quadratic form. *ASME Journal of Mechanical Design*, 123(3):640–642, 1982. 25

[JÁCC97] J. M. Jiménez, G. Álvarez, J. Cardenal, and J. Cuadrado. A simple and general method for kinematic synthesis of spatial mechanisms. *Mechanism and Machine Theory*, 32(4):323–341, 1997. 52

[JH94] J. A. Joines and C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In *International Conference on Evolutionary Computation*, pages 579–584, 1994. 102, 104, 185

[Kau71] R. E. Kaufman. KINSYN - An interactive system for the kinematic synthesis of mechanisms. In *Third World Congress on Theory of Machines and Mechanisms*, pages 13–20, Dubrovnik, Yugoslavia, september 1971. 8, 162

[KC93] S. Kota and S.-J. Chiou. Use of orthogonal arrays in mechanism synthesis. *Mechanism and Machine Theory*, 28(6):777–794, 1993. 90

[KCG98] I. Klapka, A. Cardona, and M. Géradin. An object-oriented implementation of the finite element method for coupled problems. *Rev. Europ. Éléments Finis*, 7(5), 1998. 12

[KK95] A. Kunjur and S. Krihnamurty. Genetic algorithms in mechanism synthesis. In *Fourth Applied Mechanisms and Robotics Conference*, 1995. 90

[KKH97] A. Kecskeméthy, T. Krupp, and M. Hiller. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics*, 1(1):23–45, 1997. 69, 72

[KLZ99] F. G. Kong, Q. Li, and W. J. Zhang. An artificial neural network approach to mechanism kinematic chain isomorphism identification. *Mechanism and Machine Theory*, 34(2):271–283, 1999. 25

[KS75] S. N. Kramer and G. N. Sandor. Selective precision synthesis - a general method of optimization for planar mechanisms. *Journal of Engineering for Industry, Series B*, 97(2):689–701, 1975. 5

[KS99] D. L. Kreher and D. R. Stinson. *Combinatorial algorithms: generation, enumeration and search*. CRC Press, 1999. 30

[KSP06] E. C. Kinzel, J. P. Schmiedeler, and G. R. Pennock. Kinematic synthesis for finitely separated positions using geometric constraint programming. *ASME Journal of Mechanical Design*, 128:1070–1079, 2006. 8, 162

[LCL00] M. Da Lio, V. Cossalter, and R. Lot. On the use of natural coordinates in optimal synthesis of mechanisms. *Mechanism and Machine Theory*, 35(10):1367–1389, 2000. 52, 90

[LEJ96] C. S. Lin, A. G. Erdman, and B. P. Jia. Use of compatibility linkages and solution structures in the dimensional synthesis of mechanism components. *Mechanism and Machine Theory*, 31(5):619–635, 1996. 57, 62, 64, 69, 175, 177, 178

[Len06] E. V. Lens. *Energy Preserving/Decaying Time Integration Schemes for Multibody Systems Dynamics*. PhD thesis, Universidad Nacional del Litoral, Argentina, 2006. 8, 161

[LK06] K. J. Lu and S. Kota. Topology and dimensional synthesis of compliant mechanisms using discrete optimization. *Journal of Mechanical Design*, 128(5):1080–1091, 2006. 139

[LM05] Y. Liu and J. McPhee. Automated type synthesis of planar mechanisms using numeric optimization with genetic algorithms. *ASME Journal of Mechanical Design*, 127(5):910–916, September 2005. 11, 164

[Map79] R. Mapstone. Oral history interview with Antonín Svoboda, November 1979. Repository: Charles Babbage Institute, University of Minnesota, Minneapolis. 7

[McC] J. M. McCarthy. *Synthetica*. Laboratory for the Analysis and Synthesis of Spatial Movement. University of California, USA. Web page: http://synthetica.eng.uci.edu/~software. 8, 90, 162

[McC00] J. M. McCarthy. *Geometric Design of Linkages*. Springer, 2000. 8, 162

[ME05] J. R. Mlinar and A. G. Erdman. An update on M- and K-Circle Theory for planar dyads and triads. *ASME Journal of Mechanical Design*, 127(3):464–468, May 2005. 89

[MG03] F. T. Sánchez Marín and A. Pérez González. Global optimization in path synthesis based on design space reduction. *Mechanism and Machine Theory*, 38(6):579–594, June 2003. 90

[Mic97] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1997. 101, 104, 185

[MMH96] M.D. Murphy, A. Midha, and L.L. Howell. The topological synthesis of compliant mechanisms. *Mechanism and Machine Theory*, 31(2):185–199, 1996. 19, 24, 26, 32, 35, 169

[Moo03] F. C. Moon. Robert Willis and Franz Reuleaux: Pioneers in the theory of machines. *Notes and Records of the Royal Society of London*, 57(2):209–230, May 2003. 7

[Mru03] T. S. Mruthyunjaya. Kinematic structure of mechanisms revisited. *Mechanism and Mechanism Theory*, 38(4):279–320, 2003. 19, 25, 53

[Nie77] J. Nieto Nieto. *Síntesis de Mecanismos*. Editorial AC, Madrid, 1977. 8, 34, 162

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7, 1965. 90

[Nor95] R. L. Norton. *Diseño de Maquinaria*. McGraw-Hill, 1995. http://www.designofmachinery.com/DOM/. 3, 7, 8, 100, 160, 161, 162

[OER85] D. G. Olson, A. G. Erdman, and D. R. Riley. A systematic procedure for type synthesis of mechanisms with literature review. *Mechanism and Machine Theory*, 20(4):285–295, 1985. 19

[OER87] D. G. Olson, A. G. Erdman, and D. R. Riley. Formulation of dimensional synthesis procedures for complex planar linkages. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 109(3):322–328, 1987. 7, 161

[Ope] Open Engineering S.A. *OOFELIE: Oriented Object Finite Elements Led by Interactive Executor*. University of Liège (Belgium) and INTEC (Argentina). 12

[O'R98] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, United Kingdom, 2nd edition, 1998. 99

[PC04] M.A. Pucheta and A. Cardona. Software para síntesis de mecanismos planos. In *Mecánica Computacional*, volume XXIII of *XIV Congreso sobre Métodos Numéricos y sus Aplicaciones, ENIEF 2004*, pages 3369–3389, Bariloche, Argentina, 2004. 11, 164

[PC05a] M.A. Pucheta and A. Cardona. Type synthesis and initial sizing of planar linkages using graph theory and classic genetic algorithms starting from parts prescribed by user. In J.M. Goicolea, J. Cuadrado, and J.C. García Orden, editors, *Multibody Dynamics 2005*, ECCOMAS Thematic Conference, Madrid, Spain, June 2005. 11, 15, 46, 52, 164, 166

[PC05b] M.A. Pucheta and A. Cardona. Type synthesis of planar linkage mechanisms with rotoidal and prismatic joints. In A. Larrateguy, editor, *Mecánica Computacional*, volume XXVI of *VII Congreso Argentino de Mecánica Computacional, MECOM 2005*, pages 2703–2730, Buenos Aires, Argentina, Noviembre 2005. 11, 164

[PC06] M.A. Pucheta and A. Cardona. A decomposition method for modular dimensional synthesis of planar multi-loop linkage mechanisms. In *Mecánica Computacional*, volume XXVII of *XV Congreso sobre Métodos Numéricos y sus Aplicaciones, ENIEF 2006*, pages 351–373, Santa Fe, Argentina, November 2006. 11, 14, 52, 164, 166

[PC07] M.A. Pucheta and A. Cardona. An automated method for type synthesis of planar linkages based on a constrained subgraph isomorphism detection. *Multibody System Dynamics*, 18(2):233–258, 2007. 11, 12, 164, 166

[PC08] M.A. Pucheta and A. Cardona. Synthesis of planar multi-loop linkages starting from existing parts or mechanisms: Enumeration and initial sizing. In *International Symposium on Multibody Systems and Mechatronics –MuSMe 2008*, San Juan, Argentina, 8-12 April 2008. Paper N.32. 42

[PK97] E. Peisach and A. Kikin. SYNMECH: The computer system on synthesis of planar linkages. In *Seventh IFToMM International Symposium on Linkages and Computer Aided Design Methods*, pages 227–234, Bucharest, Romania, 1997. 8, 162

[PS07] S. Pellegrino and M. J. Santer. Topology optimization of adaptive compliant aircraft wing leading edge. In *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Honolulu, Hawaii, 2007. 139

[PVW97] E. E. Peisach, R. Vogel, and J. Wei$\beta$. Synthesis of planar linkages and possibilities of integration of synthesis-modules in ADAMS. In *European ADAMS Users' Conference (12th)*, Marburg, Germany, 1997. 8, 162

[Ran07] A. M. Rankers. *SAM: Synthesis and Analysis of Mechanisms*, 2007. ARTAS-Engineering Software. Web page: http://www.artas.nl/. 8, 90, 162

[RF63] B. Roth and F. Freudenstein. Synthesis of path-generating mechanisms by numerical methods. *ASME Journal of Engineering for Industry*, 85:298–306, August 1963. 57, 90

[RR02] Y. Radovcic and A. Remouchamps. Boss quattro: an open system for parametric design. *Journal Structural and Multidisciplinary Optimization*, 23(2):140–152, March 2002. 90

[SAM07] SAMTECH Group. Système dt Analyse des Milieux Continus par Eléments Finis: *SAMCEF Field v6.2, SAMCEF BOSS/Quatro v6.0*, 2007. Web page: http://www.samcef.com. 8, 11, 90, 161, 164

[San59] G. N. Sandor. *A General Complex-Number Method for Plane Kinematic Synthesis with Applications*. PhD thesis, Columbia University, New York, 1959. 8, 62, 69, 161, 177

[Sar97] P. Sardain. Linkage synthesis: Topology selection fixed by dimensional constraints, study of an example. *Mechanism and Machine Theory*, 32(1):91–102, 1997. 10, 34, 89, 163

[SCM02] H. Su, C. Collins, and J. M. McCarthy. An extensible java applet for spatial linkage synthesis. In *ASME Design Engineering Technical Conferences 2002*, September 2002. 90

[SD07] A. Smaili and N. Diab. Optimum synthesis of hybrid-task mechanisms using ant-gradient search method. *Mechanism and Machine Theory*, 42(1):115–130, January 2007. 90

[SE84] G. N. Sandor and A. G. Erdman. *Advanced Mechanism Design: Analysis and Synthesis*, volume 2. Prentice-Hall, New Jersey, 1984. 3, 7, 8, 14, 15, 52, 57, 65, 69, 89, 141, 154, 155, 160, 161, 162, 166, 175, 196, 197

[SGE05] K. Sedlaczek, T. Gaugele, and P. Eberhard. Topology optimized synthesis of planar kinematic rigid body mechanisms. In J.M. Goicolea, J. Cuadrado, and J.C. García Orden, editors, *Multibody Dynamics 2005*, ECCOMAS Thematic Conference, Madrid, Spain, June 2005. 11, 164

[She07] A. Shell-Gellasch, editor. *Hands on History: A Resource for Teaching Mathematics*, volume 72 of *Mathematical Association of America Notes*, chapter 9 - *Historical mechanisms for drawing curves* by D. Taimina, pages 89–104. Cambridge University Press, 2007. 7

[SS06] R. P. Sunkari and L. C. Schmidt. Structural synthesis of planar kinematic chains by adapting a McKay-type algorithm. *Mechanism and Machine Theory*, 41(9):1021–1030, September 2006. 26, 31, 87

[STY00] H.-P. Shen, K.-L. Ting, and T.-L. Yang. Configuration analysis of complex multiloop linkages and manipulators. *Mechanism and Machine Theory*, 35(3):353–362, 2000. 69

[Sun06] R. P. Sunkari. *Structural Synthesis and Analysis of Planar and Spatial Mechanisms Satisfying Gruebler's Degrees of Freedom Equation*. PhD thesis, University of Maryland, College Park, Florida, 2006. 25, 26, 32

[Sut77] G. H. Sutherland. Mixed exact-approximate planar mechanism position synthesis. *Journal of Engineering for Industry, Series B*, 99(2):434–439, 1977. 5

[SVGF04] R. Sancibrián, F. Viadero, P. García, and A. Fernández. Gradient-based optimization of path synthesis problems in planar mechanisms. *Mechanism and Machine Theory*, 39(8):839–856, 2004. 90

[SWM+] J. Saylor, K. Walker, F. C. Moon, D. W. Henderson, D. Taimina, and H. Lipson. Cornell University Digital Library of Kinematics Models. http://kmoddl.library.cornell.edu. 6

[TL93] C. S. Tang and T. Liu. The degree code - a new mechanism identifier. *ASME Journal of Mechanical Design*, 115:627–630, 1993. 19, 25, 26, 166

[Tsa87] L.-W. Tsai. An application of the linkage characteristic polynomial to the topological synthesis of epicyclic gear trains. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 109(3):329–336, 1987. 11, 25, 163

[Tsa98] L.-W. Tsai. Systematic enumeration of parallel manipulators. Technical report, Institute for Systems Research, College Park, Maryland, USA, 1998. 10, 11, 19, 163

[Tsa01] L.-W. Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function*. CRC Press, Boca Raton, 2001. 8, 9, 10, 11, 20, 25, 26, 31, 32, 72, 74, 162, 163, 180, 199

[Tut96] E. R. Tuttle. Generation of planar kinematic chains. *Mechanism and Machine Theory*, 31(6):729–748, 1996. 26, 31

[UR75] J. J. Uicker Jr. and A. Raicu. A method for the identification of and recognition of equivalence of kinematic chains. *Mechanism and Machine Theory*, 10(5):375–383, 1975. 25

[WN03] T. Wasfy and A. Noor. Computational strategies for flexible multibody systems. *Applied Mechanics Review*, 56:553–613, 2003. 7, 161

[WS81] K. J. Waldron and S. M. Song. Theoretical and numerical improvements to the interactive linkage design program, RECSYN. In *Seventh Applied Mechanisms Conference*, pages 8.1–8.8.7, Kansas City, Missouri, 1981. 8, 162

[Yan98] H.-S. Yan. *Creative Design of Mechanical Devices*. Springer-Verlag, Singapore, 1998. 8, 9, 24, 162, 199

[YEB02] N. Yu, A. Erdman, and B. Byers. LINCAGES 2000: Latest development and case study. In *ASME Design Engineering Technical Conferences 2002*, Montreal, Canada, September 2002. Paper DETC2002/MECH-34375. 8, 89, 90, 162

[YH81] H.-S. Yan and A. S. Hall. Linkage characteristic polynomials: Definition, coefficients by inspection. *ASME Journal of Mechanical Design*, 103:578–584, 1981. 25

[YH82] H.-S. Yan and A. S. Hall. Linkage characteristic polynomials: Assembly theorem, uniqueness. *ASME Journal of Mechanical Design*, 104:11–20, 1982. 25

[YH91] H.-S. Yan and Y.-W. Hwang. The specialization of mechanisms. *Mechanism and Machine Theory*, 26(6):541–551, 1991. 26, 32, 36, 169

[YH06] H.-S. Yan and C.-C. Hung. Identifying and counting the number of mechanisms from kinematic chains subject to design constraints. *ASME Journal of Mechanical Design*, 128(5):1177–1182, September 2006. 26

[YO05] H.-S. Yan and F.-M. Ou. An approach for the enumeration of combined configurations of kinematic building blocks. *Mechanism and Machine Theory*, 40(11):1240–1257, November 2005. 7, 161

[YYZ98a] T.-L. Yang, F.-H. Yao, and M. Zhang. A comparative study on some modular approaches for analysis and synthesis of planar linkages: Part 1 – Modular structural analysis and modular kinematic analysis. In *ASME Design Engineering Technical Conferences, DETC/MECH-5920*, Atlanta, Georgia, USA, 1998. 69, 86

[YYZ98b] T.-L. Yang, F.-H. Yao, and M. Zhang. A comparative study on some modular approaches for analysis and synthesis of planar linkages: Part 2 – Modular dynamic analysis, modular structural synthesis and modular kinematic synthesis. In *ASME Design Engineering Technical Conferences, DETC/MECH-6058*, Atlanta, Georgia, USA, 1998. 69, 86

[ZC02] H. Zhou and E. H. M. Cheung. Analysis and optimal synthesis of adjustable linkages for path generation. *Mechatronics*, 12:949–961, 2002. 90

[ZL99] W. J. Zhang and Q. Li. On a new approach to mechanism topology identification. *ASME Journal of Mechanical Design*, 121(1):57–64, 1999. 25