Mecánica Computacional Vol. XXIII, pp. 3369-3389
G.Buscaglia, E.Dari, O.Zamonsky (Eds.)
Bariloche, Argentina, November 2004

SOFTWARE PARA SÍNTESIS DE MECANISMOS PLANOS

Martín A. Pucheta y Alberto Cardona

Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC)
INTEC (UNL-CONICET)
Güemes 3450
3000 Santa Fe, Argentina
e-mail: acardona@intec.unl.edu.ar, web page: http://www.cimec.org.ar

Palabras Clave: Síntesis de Tipo, Algoritmos Genéticos, Mecanismos Rígidos, Teoría de Grafos.

Abstract. *La síntesis de mecanismos es una tarea perteneciente al diseño conceptual que vincula geometría, cinemática y análisis combinatorio para encontrar un mecanismo adecuado para un movimiento propuesto. Tiene especial aplicación en la industria aeronáutica y mecánica en general, cuando en un marco con restricciones espaciales severas, se debe hallar una alternativa inicial factible para su posterior análisis y optimización.*

En una presentación anterior se desarrolló una metodología para la síntesis de tipo y dimensional de mecanismos utilizando algoritmos genéticos y ecuaciones algebraicas exactas *a partir de partes de cuerpos y movimientos descriptos por el usuario.*[1]

El presente trabajo muestra la evolución del anterior, en cuanto a la incorporación de mayor cantidad de resolución de situaciones y mejora del criterio de optimalidad. El programa desarrollado en C++ se ha integrado como el módulo generador de mecanismos del soft multifísico Oofelie,[2–4] *e interactúa con el CAD para elementos finitos SAMCEF FIELD v5.*[5]

Finalmente, se muestran resultados de tests para problemas del sector aeronáutico.

1. INTRODUCCIÓN

Con la presión del mercado en el costo, calidad, tiempo e innovación de los productos, los diseñadores de hoy necesitan herramientas para concebir mejores productos en sus etapas de síntesis, simulación y validación. La síntesis de mecanismos consiste en hallar el mecanismo adecuado para un movimiento dado. Usualmente, las tareas de síntesis cinemática se clasifican en tres:

- Generación de trayectoria: El objetivo es hallar las dimensiones de los elementos, de manera que uno o más puntos del mecanismo se muevan sobre una curva plana o espacial.
- Guiado de un cuerpo rígido: Uno o más elementos del mecanismo deben moverse a lo largo de una curva plana o espacial conservando además una orientación prescripta.
- Generación de función: El movimiento de un elemento o un punto debe estar funcionalmente coordinado con el movimiento del eslabón de entrada.

En el área de software para síntesis de mecanismos las opciones del mercado son escasas y con desarrollo incipiente. Mencionando a los sistemas de CAD/CAE* más difundidos disponibles para diseño mecánico se tiene a Inventor[6] de Autodesk, CATIA[7] y SolidWorks[8] de Dassault Systèmes, Unigraphics y SolidEdge[9] de EDS, ADAMS de MSC[10] y Pro/ENGINEER[11] de PTC, sólo éste último tiene posibilidad de integrar un módulo para síntesis cinemática denominado SyMech.[12]

El software que presentamos contribuye a la automatización del diseño creativo de mecanismos. Permite la síntesis cinemática de cadenas eslabonadas planas con el requerimiento de tener “un grado de libertad, eslabones de barras y juntas rotoidales y prismáticas” (en inglés, estos mecanismos son denominados *linkages*).

Entre los programas de síntesis que pueden resolver las tareas enumeradas anteriormente para mecanismos de barras, se tiene a Synthetica,[13]WATT,[14] LINCAGES,[15] SAM[16] y el mencionado SyMech. Entre las características más notables que los diferencian, Synthetica resuelve problemas espaciales pero sin restricciones, WATT tiene incorporado algoritmos de optimización con restricciones espaciales para una topología y tipo de mecanismo elegido por el usuario, en LINCAGES y SyMech hay que interactuar manualmente por cada topología elegida para satisfacer las restricciones espaciales y SAM sólo resuelve algunos tipos predefinidos.

Respecto a eśtos, el que presentamos tiene las ventajas de identificar el tipo de tarea, buscar automáticamente topologías factibles realizando una síntesis de tipo y número, y -con el ajuste de pocos parámetros- proveer con un tamaño mínimo todas las soluciones de dichas topologías. Exige entonces, una mínima cantidad de etapas de colaboración del usuario para aplicar su arte y tomar decisiones con lo cual se reduce el tiempo del diseño cinemático.

La especificación de las partes prescriptas (Sección 2.1) y la tarea de síntesis (Sección 2.2) puede realizarse mediante el CAD para elementos finitos *SAMCEF FIELD v5*, cuyo entorno

*siglas en inglés de Diseño Asistido por Computador (Computer-Aided Design) e Ingeniería Asistida por Computador (Computer-Aided Engineering).

se puede observar en la Figura 1, y posteriormente se puede continuar en la misma sesión realizando el análisis cinemático, dinámico y su diseño detallado utilizando elementos finitos.

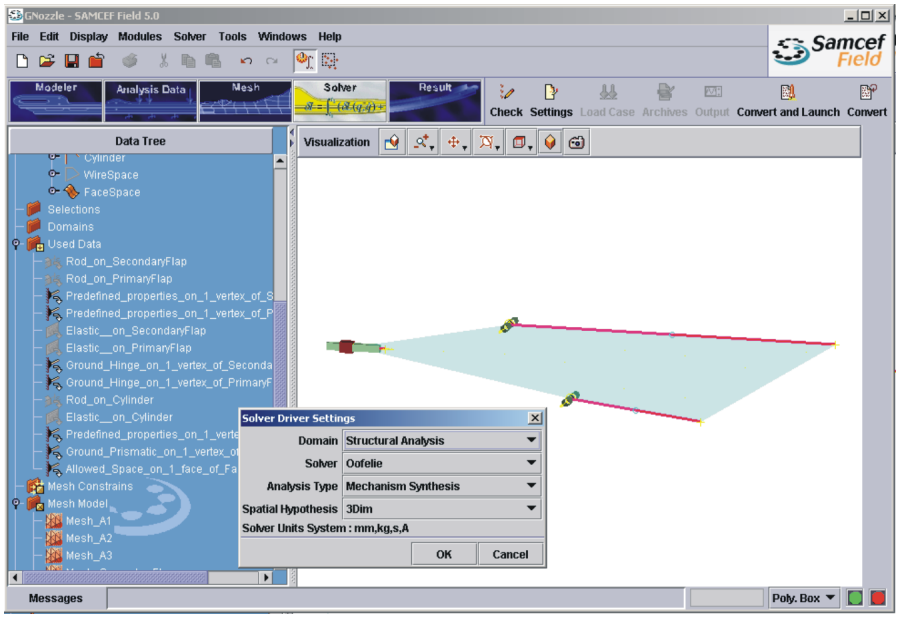

Figura 1: Entorno de CAD de *SAMCEF FIELD v5* y seteo del driver *Oofelie* para síntesis de mecanismos

En el diseño conceptual de mecanismos, la etapa inicial es clasificar las especificaciones requeridas por el cliente, generalmente expresadas mediante documentación escrita y CAD. Se separan los requerimietos necesarios para **generar** alternativas de aquellos necesarios para **evaluarlas**. Una de las potencialidades más importantes que se proponen es la posibilidad de definir una o varias partes de un mecanismo inicial con sus especificaciones y restricciones cinemáticas y solicitarle al programa que lo complete generando una alternativa válida *comenzando por la más simple posible*. Para ello se propone seguir los siguientes pasos correspondientes a la síntesis de tipo:

1. De la información del usuario se extrae una topología inicial que se representa mediante un **grafo inicial** no necesariamente conexo.

2. Para **completar el grafo** se realiza una búsqueda de ocurrencias del grafo inicial o submecanismo en un *atlas de cadenas eslabonadas planas*[17](Ver Anexo A) utilizando teoría de grafos y análisis combinatorio, con lo cual se evita pasar por estructuras bloqueadas sin grados de libertad. Para obtener una mínima cantidad de elementos componentes se busca en el atlas en orden de complejidad creciente.

En el grafo de un mecanismo los vértices se corresponden con los eslabones y las aristas o lados con las juntas. Como convención, la fundación se considera como el vértice cero.

Posteriormente a la síntesis de tipo, se descompone el mecanismo en cadenas abiertas y se resuelven sus dimensiones para satisfacer un número de posiciones prescriptas. Para el cómputo

dimensional utilizamos procedimientos de síntesis exacta para un número de hasta cuatro posiciones prescriptas.[1,18,19] Los datos que falten para este cálculo son generados por un algoritmo genético[20,21] dentro de dominios que definimos según el *tipo de variable* de que se trate,[1] tal como la posición de un pivote, la orientación prescripta sobre un eslabón o el deslizamiento prescripto sobre una junta prismática vinculada a tierra.

En la Sección 4 se describen los tipos de restricciones tenidas en cuenta y su manejo mediante penalidades. El resultado de este proceso es una serie de mecanismos que *satisfacen los requerimientos cinemáticos*. Por lo tanto las dimensiones son consideradas *iniciales* y posteriormente pueden ser ajustadas mediante iteraciones de análisis y optimización que exceden el alcance de este trabajo y cuyo objetivo es satisfacer requerimientos adicionales como por ejemplo velocidades, aceleraciones y fuerzas impuestas.

En la Sección 5 se muestran resultados de la aplicación del programa a la resolución de problemas planteados por el sector aeronáutico.

2. DATOS DE ENTRADA

En esta sección se explica cómo se realiza el ingreso de un problema en el entorno del intérprete *Oofelie* detallando la sintaxis desarrollada.

2.1. Partes prescriptas

Las partes existentes se declaran utilizando posiciones de nodos que permitirán definir elementos, trayectorias y espacio permitido. La sintaxis para definir posiciones es:

```
PositionSet posiciones_Ejemplo{
 ID_Nodo, x , y ,z ;
};
```

Los tipos de elementos disponibles son cuerpos rígidos (`RigidBody_E`), juntas rotoidales (`Hinge_E`) y juntas prismáticas (`Prismatic_E`). La sintaxis es:

```
ElementSet elementos_Ejemplo() {
 ID_elemento, Tipo_de_elemento, ID_Nodo1, ID_Nodo2,...,ID_Nodo_n;
};
```

Los nodos fijos a la fundación se definen declarando el bloqueo de sus grados de libertad traslacionales (`T_eje`) y rotacionales (`R_eje`):

```
FixationSet fijaciones_Ejemplo()
{
ID_Nodo1 ,TX;
ID_Nodo1 ,TY;
ID_Nodo1 ,TZ;
ID_Nodo1 ,RX;
ID_Nodo1 ,RY;
ID_Nodo1 ,RZ;
};
```

Un ejemplo práctico correspondiente a la Figura 2-b se escribe como:

```
PositionSet posits_Slat() {
{
7 ,11.74 ,2.28   ,0.;
6 ,11.70 ,2.14   ,0.;
5 ,11.84075, 2.40107, 0;
};

ElementSet elems_Slat() {
1, RigidBody_E, 5;
};

FixationSet fixa__Slat ()
{
7 ,TX;
7 ,TY;
7 ,TZ;
7 ,RX;
7 ,RY;
7 ,RZ;
6 ,TX;
6 ,TY;
6 ,TZ;
6 ,RX;
6 ,RY;
6 ,RZ;
};
```

Una vez definidas las posiciones, elementos y fijaciones, se agregan como variables miembro de la clase mecanismo:

```
Mechanism a;    // Instancia un objeto vacio "a" de clase mecanismo
a.add(posiciones_Ejemplo);
a.add(elementos_Ejemplo);
a.add(fijaciones_Ejemplo);
```

Luego, se agrega la información de la restricción espacial definiendo mediante e nodos adicionales una poligonal cerrada convexa cuya sintaxis es:

```
SyAllowedSpace espacio_permitido_Ejemplo(Nodo_n+1,...,Nodo_n+e);
a.add(espacio_permitido_Ejemplo);
```

Después, se agregan dos parámetros que se utilizarán para definir la tarea a resolver, indicando la cantidad de puntos de paso y el número de nodo del mecanismo que deberá seguir la trayectoria.

```
a.put_number_of_ppoints(num_passing_points);
a.put_traj_node(ID_Nodo_traj);
```

Finalmente, se construye el mecanismo.

```
a.build();
```

Continuando con el ejemplo anterior se declara:

```
Mechanism a;
a.add(posits_Slat);
a.add(elems_Slat);
a.add(fixations_Slat);
a.put_number_of_ppoints(3);
a.put_traj_node(5);
a.build();
```

2.2. Definición de la tarea de síntesis

El tipo de tarea de síntesis cinemática se identifica según los datos que ingresa el usuario quedando definida implícitamente. La metodología de resolución que se presentará en las secciones siguientes es general y no requiere tratamiento especial para cada tarea.

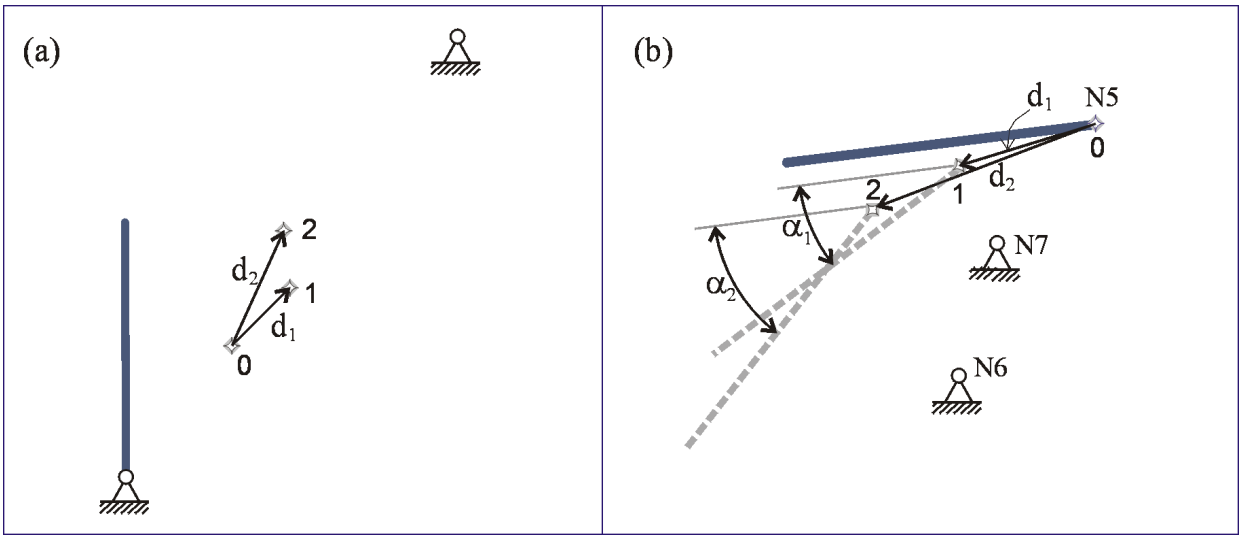

Figura 2: Tareas de generación de trayectoria (a) y guiado de cuerpo rígido (b).

La sintaxis para definir la trayectoria de un punto del mecanismo inicial a se hace indicando el identificador del nodo, el número del punto de paso y las coordenadas x e y relativas a la posición inicial del nodo:

```
a[SYNODESET_PO].put_ppoint(ID_Nodo, nro_punto_de_paso, x_relat , y_relat);
```

y para definir las rotaciones de un elemento se realiza indicando el identificador del elemento, el número de punto de función y la rotación en *radianes* relativas a la orientación inicial del elemento:

```
a[SYLINKSET_PO].put_ppoint(ID_Elemento, nro_punto_de_funcion, angulo_rad);.
```

Si sobre un nodo se especifica una secuencia de posiciones precisas, la tarea es *generación de trayectoria*, este nodo puede definirse aislado o puede pertenecer a un eslabón (Figura 2-a). Si sobre esta secuencia se especifica además de las coordenadas cartesianas, el ángulo de orientación del nodo en cada posición, se trata de *guiado de cuerpo rígido*, por ejemplo, para la Figura 2-b se especifica la trayectoria del nodo 5 mediante:

```
a[SYNODESET_PO].put_ppoint(5,0,0,0,0);
a[SYNODESET_PO].put_ppoint(5,1,-0.14076,-0.04024,0);
a[SYNODESET_PO].put_ppoint(5,2,-0.22766,-0.08580,0);
```

y, agregando la condición de que el elemento `1` rote 30 grados en la posición precisa `1` y 45 grados en la `2` se especifica su guiado:

```
a[SYLINKSET_PO].put_ppoint(1,0,0);
a[SYLINKSET_PO].put_ppoint(1,1,0.523);
a[SYLINKSET_PO].put_ppoint(1,2,0.7854);
```

Si se especifican dos trayectorias sobre dos nodos de un mismo eslabón los datos deben adaptarse a este mismo caso.

Finalmente, si se especifican dos secuencias de orientaciones u desplazamientos en dos eslabones, el caso se conoce como *generación de función* y se desea cumplir una ley $\beta = f(\alpha)$ como puede verse en la Figura 3-a.

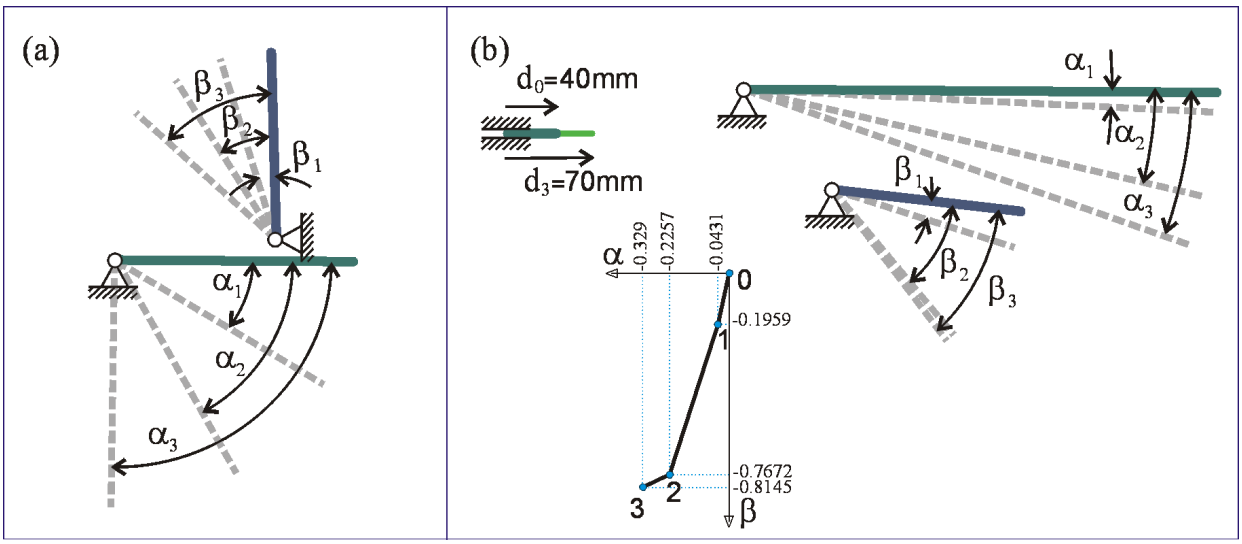

Figura 3: Tarea de generación de función (a) y de un caso combinado (b).

También es posible resolver problemas combinados como el de la Figura 3-b donde la coordinación entre dos eslabones está definida por una función $\beta = f(\alpha)$ y debe hallarse una tercer entrada de movimiento en el mismo mecanismo. Se la podría denominar *doble generación de función*.

3. ELECCIÓN DEL TIPO Y DESCOMPOSICION EN CADENAS ABIERTAS

Al construir el mecanismo `a` se identifican los datos para armar el grafo inicial G_a el cual tendrá n_a vértices. A continuación se declara un nuevo mecanismo `b` para generar una nueva alternativa mediante el comando `lookfor` que trabaja con el siguiente algoritmo:

1. Se elige del primer nivel del atlas de mecanismos un grafo G_{Tsai} con n_{Tsai} vértices.
2. Del grafo elegido se toma un subgrafo H_{Tsai} formado por n_a vértices[**].
3. Si H_{Tsai} es isomorfo a G_a ($H_{Tsai} \equiv G_a$), significa que G_a es subgrafo de G_{Tsai} y se asigna $G_b \leftarrow G_{Tsai}$, sino se sigue eligiendo un nuevo subgrafo H_{Tsai} volviendo al paso 2, y finalmente, si no aparece ninguna vez en el nivel de Tsai elegido se retorna al paso 1 y se elige el próximo más complejo.

Una vez detectado G_a en el atlas se tiene en G_b una topología que hereda los datos de síntesis del mecanismo `a` y G_b tendrá nuevos vértices cuyos nodos tienen posiciones desconocidas. Se procede a descomponer el grafo G_b en cadenas abiertas mediante la función miembro `decompose`. La sintaxis es:

```
int found = 0;
Mechanism b;
found = b.lookfor(a);
if (found) {
   b.decompose();
   b.assemble();
}
```

4. SÍNTESIS DIMENSIONAL RESTRINGIDA

El conjunto de cadenas abiertas se analiza para determinar un orden de evaluación, dado que sobre algunas habrá más datos que en otras. Una vez ordenadas se analiza secuencialmente cada cadena abierta y se resuelven sus dimensiones. Cada cadena puede tener más de una solución, por lo que la secuencia cálculo se puede ramificar.

Por cada cadena se plantea un sistema de ecuaciones de lazos de números complejos con tantas ecuaciones como el máximo número de posiciones precisas n_{pp} que aparezca en la tarea.

$$\boldsymbol{C}^j \boldsymbol{L}^j = \mathbf{0} \quad \boldsymbol{C}, \boldsymbol{L} \in \mathbb{C} \quad j = 1, ..., p, \tag{1}$$

donde p es el número de cadenas abiertas (no hay convención de Einstein). El planteo y resolución de estas ecuaciones aparece en variada bibliografía con el nombre de “síntesis exacta”, o también como “síntesis analítica o algebraica”.[19,22,23]

[**]esto se puede hacer de $\begin{pmatrix} n_{Tsai} \\ n_a \end{pmatrix}$ maneras que recorremos en orden lexicográfico de las etiquetas de los vértices

Del análisis de cada sistema $\boldsymbol{CL} = \boldsymbol{0}$, resultará un número de parámetros libres necesarios para tener solución no trivial,[24] o sea $det(\boldsymbol{C}) = \boldsymbol{0}$. Estos parámetros constituirán las variables genéticas que se arreglan en un vector $\boldsymbol{x}^j$ particular de la cadena j. Las dimensiones de la cadena abierta, serán dependientes de estos parámetros y de los prescriptos por el usuario que se mantendrán constantes y denominaremos $\boldsymbol{p}$.

Como problema de optimización, planteamos como criterio a minimizar la sumatoria de las longitudes L de los eslabones en todas las cadenas en la posición inicial,

$$min\boldsymbol{F}^j(\boldsymbol{x}, \boldsymbol{p}, 0) = min\sum_{j=1}^{p}\sum_{i=1}^{e(j)}|L_i^j(\boldsymbol{x}, \boldsymbol{p}, 0)|, \tag{2}$$

donde $e(j)$ es el número de eslabones de la cadena abierta j, sujeto a restricciones de:

- *mínima longitud de barras*

$$L_{min} - |L_i^j(\boldsymbol{x}, \boldsymbol{p}, 0)| \leq 0 \quad j = 1, ..., p \quad i = 1, ..., e(j), \tag{3}$$

- *no-inversión de ángulos de transmisión*

$$sgn(\psi^j(L_i, L_{i+1}))_0 = sgn(\psi^j(L_i, L_{i+1}))_t \quad j = 1, ..., p \quad i = 1, 2, .., e(j) - 1 \quad t = 1, ..., n_{pp} - 1, \tag{4}$$

  donde la función $\psi^j(L_i, L_{i+1})_0$ devuelve el ángulo de transmisión en la junta que vincula el eslabón L_i y el siguiente L_{i+1} en la posición inicial, cuyo valor debe tener igual signo que en las siguientes posiciones prescriptas;

- restricción de *área permitida*, que resumidamente se puede escribir como

$$N_{0,k} \subset \boldsymbol{A}(P_1, P_2, ..., P_s) \quad k = 1, ..., r \tag{5}$$

  donde $N_{0,k}$ es cada nodo del mecanismo en la posición inicial $t = 0$, r es la cantidad de nodos del mismo y s es la cantidad de puntos P que definen la poligonal convexa que constituyen la restricción espacial;

- y finalmente, se deben satisfacer las igualdades de lazos en números complejos de los sistemas

$$\boldsymbol{C}^j\boldsymbol{L}^j = \boldsymbol{0} \quad j = 1, ..., p. \tag{6}$$

Para resolver este problema, o sea (2) sujeto a (3-6), trabajamos individualmente con cada cadena j analizando el determinante del sistema $\boldsymbol{C}^j\boldsymbol{L}^j = \boldsymbol{0}$ correspondiente, aquí pueden aparecer tres casos:

1. El sistema es indeterminado,

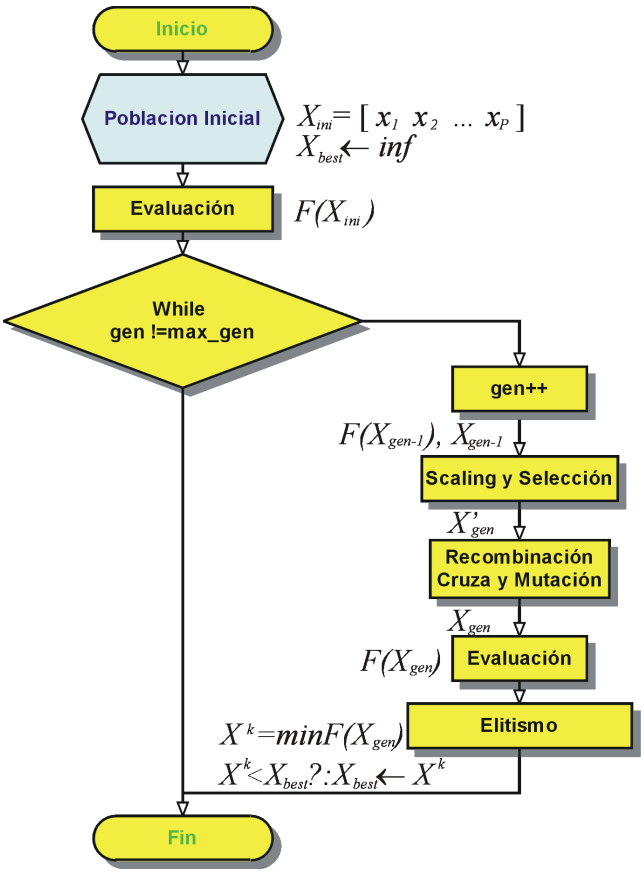

Figura 4: Algoritmo genético clásico para un problema de minimización

2. tiene solucion única (o con multiplicidad finita) definida geométricamente
3. o tiene infinitas soluciones.

Para este último caso, dado que es un problema no lineal y con dominios donde puede o no haber solución (con huecos), utilizamos algoritmos genéticos cuyo esquema se puede ver en la Figura 4. De la condición $det\boldsymbol{C}^j = \mathbf{0}$ se extraen los parámetros libres $\boldsymbol{x}^j$ que lo hacen compatible y definiendo sus fronteras según el tipo de variable, generamos una población

$$\boldsymbol{X}^j = [\boldsymbol{x}_1^j \quad \boldsymbol{x}_2^j \quad ... \quad \boldsymbol{x}_P^j]^T$$

de P individuos, convirtiendo el problema en uno de minimizar la función de aptitud (*fitness*) que diseñamos como:

$$min\boldsymbol{F}^{*j}(\boldsymbol{x}^j, \boldsymbol{p}^j, t) = \sum_{i=1}^{e} |L_i^j(\boldsymbol{x}^j, \boldsymbol{p}^j, 0)| \tag{7}$$

y teniendo en cuenta las restricciones hacemos

$$min\boldsymbol{F}^j(\boldsymbol{x}^j, \boldsymbol{p}^j, t) = \lambda_0 \sum_{i=1}^{e} |L_i^j(\boldsymbol{x}^j, \boldsymbol{p}^j, 0)| + Q(\boldsymbol{x}^j, \boldsymbol{p}^j, t) \qquad t = 0, 1, ..., n_{pp}, \tag{8}$$

donde e es el número de eslabones de la cadena abierta j, y t indica la posición precisa. Las variables $\boldsymbol{x}^j$ y los parámetros fijos $\boldsymbol{p}^j$ van cambiando en la secuencia de resolución de las cadenas. El factor λ_0 es 1 si el sistema es determinado y adquiere un valor de penalidad "suficientemente grande" λ_{max} si para algún valor de una de las variables el sistema es singular. El término $Q(\boldsymbol{x}^j, \boldsymbol{p}^j, t)$ tiene en cuenta las restricciones (3), (4) y (5) que se deben satisfacer para todas las posiciones precisas. Éste posee tres términos afectados de constantes λ que hemos ajustado empíricamente y también adquieren el valor λ_{max} si el problema queda indeterminado.

A continuación explicaremos el manejo de las restricciones. Para reducir notación consideraremos la cadena j y los parámetros fijos $\boldsymbol{p}^j$ correspondientes

$$Q(\boldsymbol{x}, t) = \lambda_1 q_L(\boldsymbol{x}, 0) + \lambda_2 q_T(\boldsymbol{x}, t) + \lambda_3 q_A(\boldsymbol{x}, 0) \tag{9}$$

donde

$$q_L(\boldsymbol{x}, 0) = \sum_{i=1}^{e} \delta(L_{min} - |L_i(\boldsymbol{x}, 0)|) \quad \text{con} \quad \delta = \begin{cases} 1 & \text{si} \quad |L_i| \geq L_{min} \\ 0 & \text{de otro modo} \end{cases}, \tag{10}$$

$$q_T(\boldsymbol{x}, t) = \sum_{r=1}^{e-1} \delta|\psi_{0,r}(\boldsymbol{x}, t)| \quad \text{con} \quad \delta = \begin{cases} 1 & \text{si} \quad sgn(\psi_{0,r}(\boldsymbol{x}, t)) \neq sgn(\psi_{t,r}(\boldsymbol{x}, t)) \\ 0 & \text{de otro modo} \end{cases}, \tag{11}$$

$$q_A(\boldsymbol{x}, 0) = max(0, d(N_{0,k} \not\subset \boldsymbol{A}(P_1, P_2, ..., P_s))) \quad k = 1, ..., r, \tag{12}$$

son las maneras en que calculamos las violaciones. En las ecuaciones (10) y (11), la δ es 1 si la restricción es violada y cero si no. La función $\psi_{t,r}(\bullet)$ devuelve para la posición precisa t, el ángulo de transmisión en la junta r si difiere en su signo con el ángulo de transmisión en la junta r en la posición inicial $t = 0$. Cabe aclarar que la cadena de e eslabones tiene $e - 1$ juntas de chequeo. En la ecuación (12) los argumentos de la función $d(\bullet)$ se definen igual que

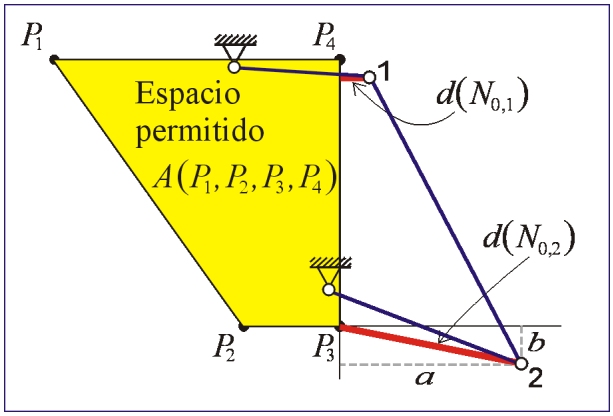

Figura 5: Ejemplo de violación al espacio permitido

para la ecuación (5). Como se observa en la Figura 5, esta función devuelve la distancia desde el nodo hasta el área permitida, siempre que el nodo no esté incluido en dicha área. Tomando

como ejemplo al nodo 2 que está violando la restricción, la distancia $d(N_{0,2})$ se calcula como la composición de las distancias *a* y *b* perpendiculares a las aristas $\overline{P_2P_3}$ y $\overline{P_3P_4}$ respectivamente, ya que recorriendo la poligonal en sentido antihorario $\overline{P_iP_{i+1}}$, para ambos casos, el nodo queda en el semiplano derecho resultante de prolongar los segmentos infinitamente.

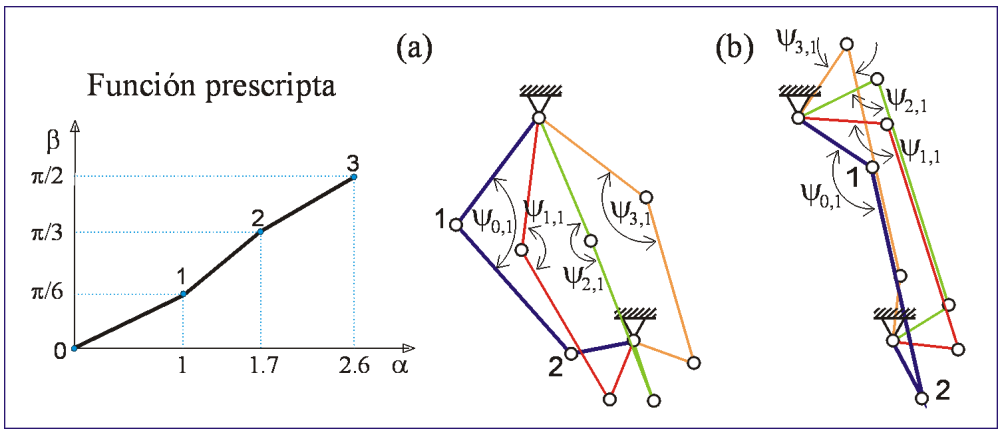

Figura 6: Efecto de la restricción en el ángulo de transmisión para un problema de generación de función

De todas las restricciones, la de efecto menos intuitivo es la de inversión del ángulo de transmisión. Para ello las Figuras (6-a) y (6-b) son ilustrativas, ambas resuelven *geométricamente* el mismo problema de generación de función, pero la primera tiene inversiones en ambas juntas ya que no están penalizadas. Por ejemplo, en la junta 1 los ángulos $\psi_{0,1}$ y $\psi_{1,1}$ son positivos en la primer y segunda posición precisa mientras que son negativas en el resto. Con el mecanismo en movimiento, al llegar a la posición 2 puede bloquearse. Agregando la restricción (11) la solución obtenida es como la de la Figura (6-b). Notar que el mismo fenómeno ocurre en la junta 2.

Volviendo al intérprete, esto está implementado al ejecutar `b.assemble()`. Durante la ejecución se puede observar la evolución del algoritmo genético para cada cadena al mostrarse un vector de aptitud

$$\boldsymbol{F}^j = \left[\begin{array}{cccc} \boldsymbol{F}^{*j} & q_L^j & q_T^j & q_A^j \end{array} \right].$$

Como resultado el programa retorna como topología óptima *la combinación de cadenas con mejores aptitudes*. Para monitorear si el sistema es difícil de resolver se observa si el sistema se hace singular evolucionando a

$$\boldsymbol{F}^j = \left[\begin{array}{cccc} \lambda_{max} & \lambda_{max} & \lambda_{max} & \lambda_{max} \end{array} \right].$$

Para setear valores por defecto de los parámetros intervinientes en la optimización, hacemos uso de la diagonal D de la caja limitante (bounding box) de las coordenadas de los datos, incluyendo la tarea, los pivotes y los nodos de las partes prescriptas. Y experimentalmente hemos relacionado $Lmin = D/15$, $\lambda_1 = \lambda_{max}/D$, $\lambda_2 = \lambda_{max}$, y $\lambda_3 = \lambda_{max}/D$, con $\lambda_{max} = 1000$ con valor por defecto. De este modo, adquirieren órdenes de magnitud adecuados en forma automática. Además, el usuario puede ingresar el tamaño de la población P del algoritmo genético

(por defecto es 10), el número de generaciones máximo (por defecto es 60), el factor de penalidad λ_{max} y la restricción de mínima longitud de barras $Lmin$.

```
if (found) {
 b.decompose();
 b.set_population (20);
 b.set_maxgenerations (100);
 b.set_penalty(10000);
 b.set_minlength (0.02);
 b.assemble();
};
```

Como valores adecuados a los tipos de problemas estudiados, hemos fijado a las probabilidades de cruza y mutación en valores de 0.5 y 0.01 respectivamente y no quedan abiertos al usuario.

Finalmente el resultado se exporta a un formato de texto importable desde *SAMCEF FIELD v5*.

```
b.toSamcefField("C:\DATA\mpucheta\ENIEF04\slat_0");
```

5. RESULTADOS DE PROBLEMAS DE PRUEBA

En trabajos anteriores hemos presentado la metodología mostrando ejemplos de mecanismos de cuatro barras para generación de línea recta.[1] Ahora presentaremos guiado de cuerpo rígido y generación de función.

5.1. Guiado de cuerpo rígido

El archivo de texto que se venía utilizando como tutorial de la sintaxis corresponde a un test industrial de un *slat* de ala de avión. Funcionalmente, su movimiento se encarga de cambiar el ángulo de ataque del ala para realizar maniobras.

En la Figura 7-a puede verse el movimiento requerido cuyos valores se usaron de ejemplo en la Sección 2.2.

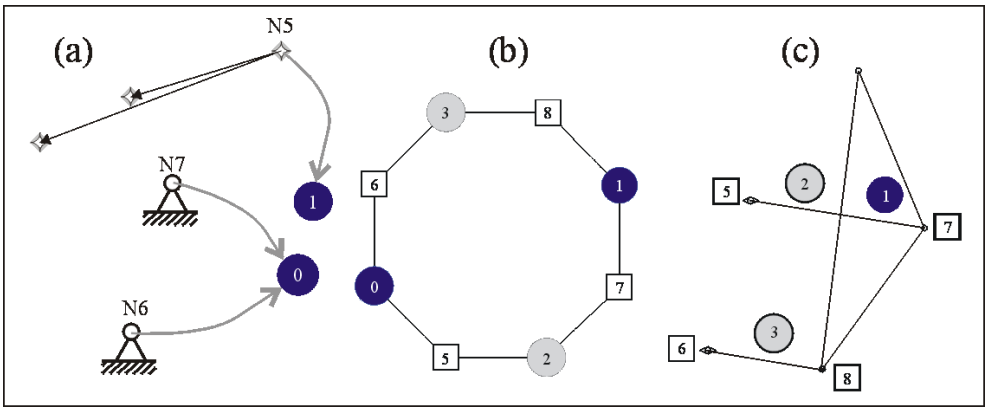

Figura 7: Mecanismo de 4 barras para accionamiento de un *slat* de ala de avión.

Luego de ingresar los datos de la forma en que se mostró en la Sección 2.1 se construye el mecanismo dato a cuya información es la siguiente:

```
>>a.print;
:                   PositionSet [size:   7]:
:                    ElementSet [size:   1]:
:                    VirtualSet [size:  12]:
  Node    5  Link    1  Joint   0  Position 11.8408 2.40107
PPoint displ 0  0  PPoint displ -0.14076  -0.04024  PPoint displ -0.22766  -0.0858
  Node    6  Link    0  Joint   6  Position 11.7   2.14
PPoint displ 0  0  PPoint displ 0  0  PPoint displ 0  0
  Node    7  Link    0  Joint   5  Position 11.74  2.28
PPoint displ 0  0  PPoint displ 0  0  PPoint displ 0  0
  Link         0             |  Nodes   7  6
  Link         1             |  Nodes   5
PPoint displ 0  PPoint displ 0.523  PPoint displ 0.7854
>>
```

El grafo del mecanismo inicial se muestra en la Figura 7-a. Notar que el eslabón 1 consta solamente del nodo 5, y la fundación es representada en el grafo por un vértice binario. Luego de buscar en el atlas se obtiene un grafo con cuatro vértices (Fig. 7-b), donde los eslabones 3 y 2 son nuevos. La Figura 7-c es un esbozo de la cadena cinemática que se desea calcular donde se han superpuesto las etiquetas de su grafo para facilitar su asociación.

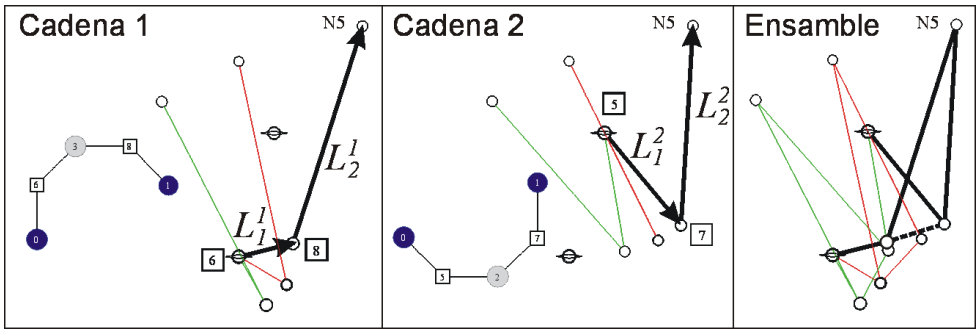

Figura 8: Descomposición en cadenas abiertas y representación con números complejos.

En la Figura 8, se muestra la descomposición en cadenas abiertas, en este caso, dos díadas. El cálculo de la cadena 1, determina nuevos datos para la cadena 2. Efectivamente, el complejo L_2^2 tiene las rotaciones impuestas por el cálculo previo de L_2^1 (recordar que en L_i^j, i es el número de eslabón y j el número de cadena).

Para tener una idea del conjunto de datos involucrados, se muestra la salida por consola de los nodos, eslabones y juntas del mecanismo b resultante.

```
>>b.print;
:                   PositionSet [size:   7]:
:                    ElementSet [size:   1]:
:                    VirtualSet [size:  12]:
  Node    5  Link    1  Joint   0  Position 11.8408 2.40107
PPoint displ 0  0  PPoint displ -0.14076  -0.04024  PPoint displ -0.22766  -0.0858
  Node    6  Link    0  Joint   6  Position 11.7    2.14
PPoint displ 0  0  PPoint displ 0  0  PPoint displ 0  0
  Node    7  Link    0  Joint   5  Position 11.74   2.28
PPoint displ 0  0  PPoint displ 0  0  PPoint displ 0  0
  Node    8  Link    1  Joint   7  Position 11.827  2.17486
  Node    9  Link    1  Joint   8  Position 11.7616 2.15465
  Node   10  Link    2  Joint   5  Position 11.74   2.28
  Node   11  Link    2  Joint   7  Position 11.827  2.17486
  Node   12  Link    3  Joint   6  Position  11.7   2.14
```

```
  Node   13  Link    3  Joint   8  Position 11.7616 2.15465
  Link         0             |  Nodes  7  6
  Link         1             |  Nodes  8  9  5
PPoint displ 0 PPoint displ 0.523  PPoint displ 0.7854
  Link         2             |  Nodes  10 11
  Link         3             |  Nodes  12 13
  Joint        5   Type  1   |  Nodes  7  10  | Links 0  2
  Joint        6   Type  1   |  Nodes  6  12  | Links 0  3
  Joint        7   Type  1   |  Nodes  8  11  | Links 1  2
  Joint        8   Type  1   |  Nodes  9  13  | Links 1  3
```

Se muestra la primer solución obtenida para lo cual todas las juntas son del tipo rotoidal codificadas como `Type 1` en la salida (y codificamos con `Type 2` a las prismáticas, que en este caso no aparecen).

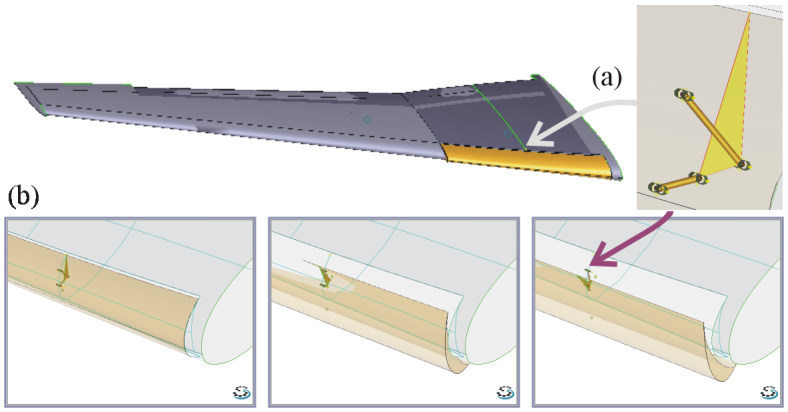

Figura 9: Ubicación del plano de trabajo del mecanismo en el ala (a) y secuencia de resultados para las 3 posiciones prescriptas (b).

En la Figura 9 se muestra el resultado exportado al CAD. En realidad, la curva que describe un punto del slat no es plana, es ligeramente alabeada, pero el resultado logrado en el plano -la síntesis de tipo y dimensionado inicial- es un buen punto de partida para la optimización real en el espacio. En (9-a) se indica el plano en el que va a actuar el mecanismo. Finalmente, en (9-b) se muestran imágenes capturadas de la animación del mecanismo donde se ha fijado rígidamente el eslabón 1 al sólido que representa al *slat*.

5.2. Generación de función

Ahora la tarea es como la de la Figura 3-a aplicada a un mecanismo de retracción del tren de aterrizaje. En la secciones anteriores hemos explicado cómo se especifica la tarea mediante un archivo de texto, ahora mostramos cómo se hace en el CAD. En la Figura 10 pueden verse los pasos necesarios para especificar la secuencia de ángulos sobre el tren. En el contexo de análisis se edita la junta (1), se la motoriza (2), se elige una función lineal a trozos (3), y se carga la tabla de puntos de dicha función (4). De igual modo se procede para la junta del actuador.

El giro requerido del tren es de 90 grados el cual se impone. Como el giro motriz requerido no está impuesto, los valores de sus ángulos se dejan libres. Además, se ingresa la restricción de espacio permitido.

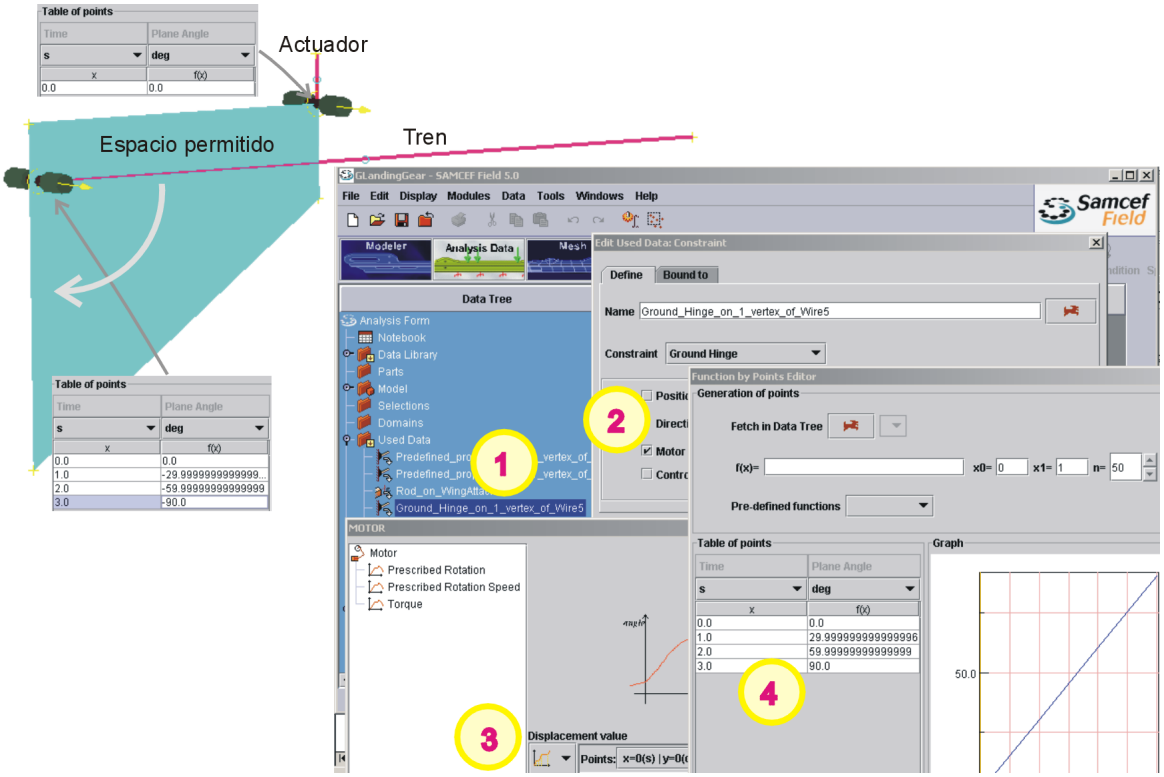

Figura 10: Ingreso mediante CAD de las rotaciones prescriptas sobre las partes del tren de aterrizaje

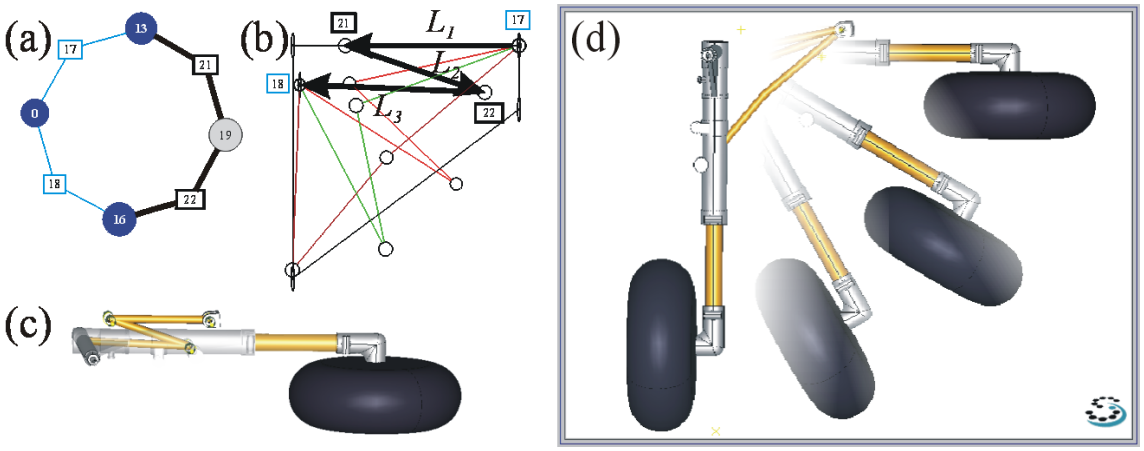

Figura 11: Mecanismo de 4 barras para un tren de aterrizaje

El grafo asociado a las partes tiene 3 vértices, el 0 (fundación), el 13 (actuador) y el 16 (tren). El grafo en que se halló el submecanismo es el de la Figura 11-a, éste se descompone en tan sólo una cadena abierta que se ve resaltada en la misma figura y corresponde a resolver una tríada (Figura 11-b). El nuevo eslabón 19 estará conectado al tren mediante la junta 22 y al eslabón motriz mediante la junta 21. Previo a ejecutar la optimización, ingresamos las fronteras de los

parámetros libres haciendo que la búsqueda de la rotación motriz máxima sea de 0.85 radianes.

```
if (found) {
   b.decompose();
   b.set_bound(1,0.10,0.25);
   b.set_bound(2,0.30,0.45);
   b.set_bound(3,0.50,0.85);
   b.set_population (20);
   b.set_maxgenerations (50);
   b.set_minlength (0.06);
   b.assemble();
}
```

El mecanismo resultante es uno plegadizo de 4 barras, su funcionamiento puede observarse en las Figuras 11-c y 11-d.

5.3. Doble generación de función

Por último se presenta la aplicación a un mecanismo de accionamiento de toberas de un motor de avión a combustión, donde se requiere la coordinación de 3 cuerpos (Figura 12) cuyas especificaciones se mostraron en el ejemplo de la Figura 3-b.

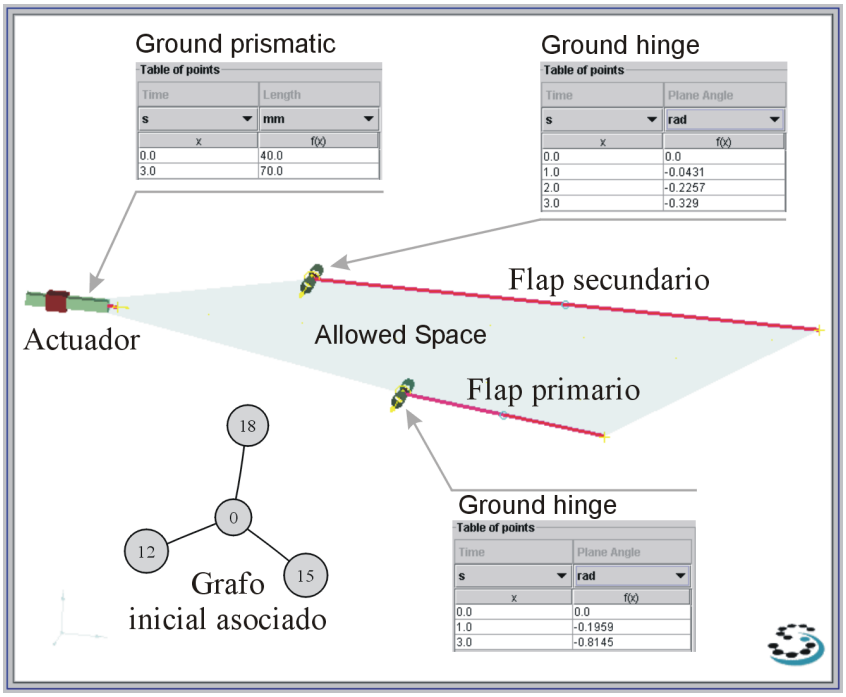

Figura 12: Tobera de turbina

Ambos *flaps* deben girar en sentido horario con el deslizamiento horizontal de un actuador. La especificación original requería un ángulo de -0.7672 radianes para la segunda posición precisa del *flap* primario, pero no tiene solución que satisfaga la no-inversión del ángulo de

transmisión y simultáneamente se halle dentro de las restricciones espaciales. Es por ello que se dejó libre ese dato y se ingresaron las fronteras de búsqueda para esa variable.

En la Figura 12 puede verse que el grafo asociado a las partes prescriptas, tiene un vértice ternario en la fundación por lo que no aparecerá como subgrafo del primer nivel del atlas que tiene 4 vértices binarios, sino que aparece en el segundo nivel del atlas y el grafo resultante tiene 6 vértices (Figura 13-a).

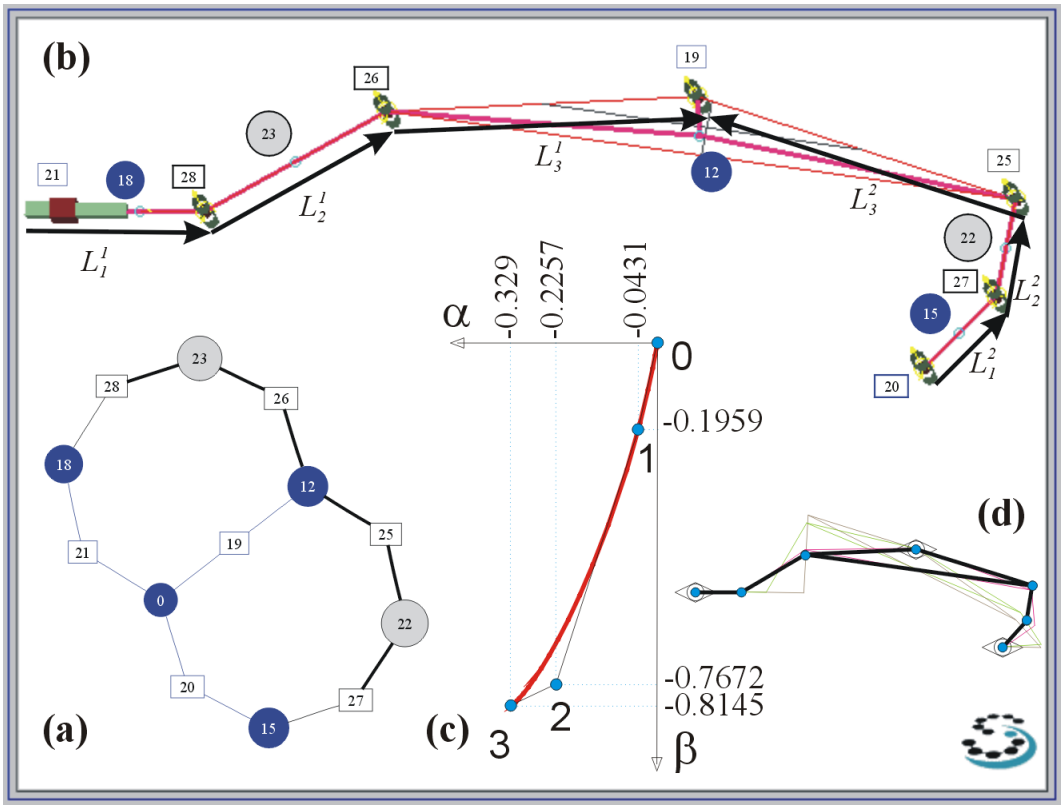

Figura 13: Solución seis barras para coordinar la tobera de turbina

La topología se descompuso en dos cadenas abiertas, una tríada entre el actuador y el *flap* secundario, y una tríada entre ambos *flaps*. La solución ensamblada puede verse en la Figura 13-b donde se referencian las juntas y eslabones sintetizados con los del grafo, se resaltan además, las flechas representativas de los números complejos de las cadenas abiertas (desplazados adrede). Notar que aparecen dos eslabones nuevos, el 22 y el 23, y el cuerpo 12 (*flap* secundario) pasa a ser ternario, debido a las nuevas juntas 25 y 26.

Finalmente, en la Figura 13-b se muestra la función resultante del análisis cinemático. Como se puede observar, no se satisface la posición 2 a causa dejarlo libre por el problema mencionado, pero esta solución no pasa por ninguna posición de centro muerto (Figura 13-d) donde el ángulo de transmisión cambie de signo.

6. CONCLUSIONES

Se desarrolló un software que permite sintetizar mecanismos planos que responden a requerimientos cinemáticos dados por el usuario, *partiendo de cero*. En este sentido, destacamos que el sistema propone inclusive la topología del mecanismo, no limitándose la búsqueda a variaciones dimensionales.

La solución a este tipo de problema inverso dista de ser única y el programa permite recorrer con facilidad el espacio de soluciones admisibles, buscando aquella que mejor satisface criterios de optimalidad. Al mismo tiempo, verifica que las soluciones propuestas no violen restricciones diversas (espacio, singularidad del movimiento, dimensiones mínimas, etc).

El software se encuentra integrado a un sistema de análisis de mecanismos y estructuras por elementos finitos. La definición de datos se realiza en forma gráfica en un sistema de CAD propio del sistema general de análisis, orientado a problemas de elementos finitos. A posteriori de la tarea de síntesis, el usuario puede verificar la validez de la solución propuesta mediante análisis, e incluso puede optimizar la misma manteniéndose dentro de la misma topología.

Se realizaron una serie de tests en problemas propuestos por empresas del sector aeronáutico, mostrándose la utilidad de la propuesta. En particular, se resolvieron problemas de síntesis de mecanismos para trenes de aterrizaje, comando de slats y flaps y comando de deflectores en toberas de turbinas.

AGRADECIMIENTOS

Este trabajo se desarrolló con financiación de la Agencia Nacional de Promoción Científica y Tecnológica y de la Unión Europea, en el marco del proyecto SYNAMEC (SYNthesis tool for Aeronautical MEChanisms design), proy. UE 2001-001-0058.

A. ATLAS DE MECANISMOS

Las Figuras 14 y 15 muestran los grafos de cadenas cinemáticas planas de un grado de libertad enumerados por Tsai.[17] Los círculos representan eslabones y las etiquetas rectangulares las juntas.

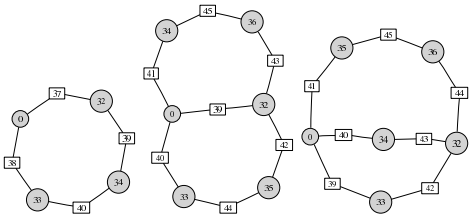

Figura 14: Opciones de 4 y 6 barras

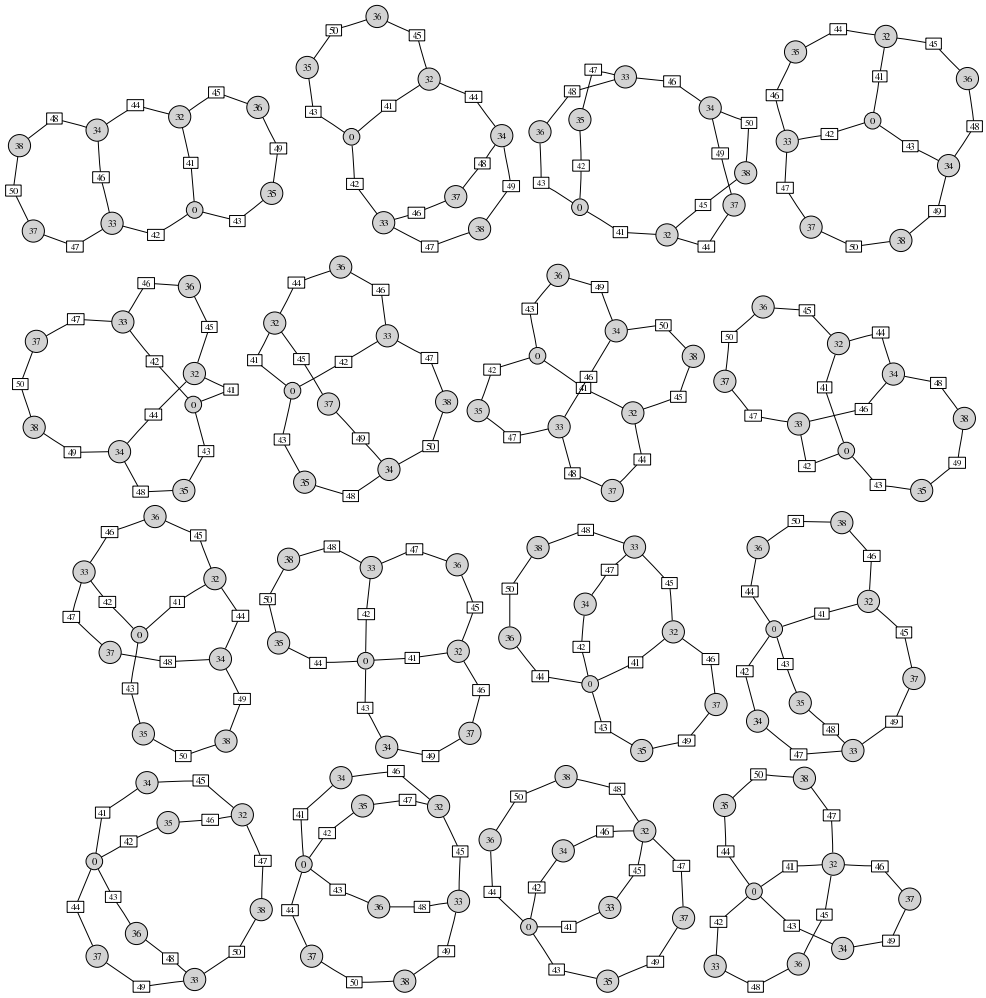

Figura 15: Opciones de 8 barras

REFERENCIAS

[1] M. Pucheta and A. Cardona. Síntesis de tipo y dimensional de mecanismos utilizando algoritmos genéticos y ecuaciones algebraicas exactas. In *Mecánica Computacional*, volume XXII, pages 1200–1216, Bahía Blanca, Argentina, (Noviembre 2003).

[2] Open Engineering S.A. *Oofelie* Oriented Object Finite Elements Led by Interactive Executor. http://www.open-engineering.com.

[3] 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).

[4] A. Cardona, I. Klapka, and M. Géradin. Design of a new finite element programming environment. *Engineering Computations*, **11** (1994).

[5] SAMTECH S.A. *SAMCEF*. http://www.samcef.com.

[6] Autodesk Corporation. *Inventor*. http://autodesk.com.

[7] Dassault Systèmes. *CATIA* DMU Kinematics Simulator 2. http://plm.3ds.com/CATIA.

[8] Dassault Systèmes. *Solid Works* Animator. http://www.solidworks.com.

[9] Electronic Data Systems Corporation. *Solid Edge, Unigraphics Solutions, Parasolid, I-Deas NX*. http://eds.com/plm.

[10] MSC.Software Corporation. *MSC.ADAMS*. http://www.mscsoftware.com.

[11] Parametric Technology Corporation. *Pro/ENGINEER Mechanism Dynamics*. http://www.ptc.com.

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

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

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

[15] A.G. Erdman. *LINCAGES* LInkage INteractive Computer Analysis and Graphically Enhanced Synthesis Package. University of Minnesota, USA. http://www.me.umn.edu/labs/lincages.

[16] A.M. Rankers. *SAM* Synthesis and Analysis of Mechanism. ARTAS-Engineering Software. http://www.artas.nl.

[17] Lung-Wen Tsai. *Mechanism Design: Enumeration of Kinematic Structures According to Function*. CRC Press, Boca Raton, (2001).

[18] G.N. Sandor and A.G. Erdman. *Mechanism Design: Analysis and Synthesis*, volume 2. Prentice-Hall, New Jersey, (1984).

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

[20] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, (1997).

[21] D.E. Goldberg. *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley, (1989).

[22] R.L. Norton. *Diseño de Maquinaria*. McGraw-Hill, (1995).

[23] J.M. McCarthy. *Geometric Design of Linkages*. Springer, (2000).

[24] A. Cardona. *Computational Methods for Synthesis of Mechanisms*. Technical report, CIMEC-INTEC, (2002).