

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/authorsrights>



Contents lists available at SciVerse ScienceDirect

## Mechanism and Machine Theory

journal homepage: [www.elsevier.com/locate/mechmt](http://www.elsevier.com/locate/mechmt)

## Automated sketching of non-fractionated kinematic chains

Martín A. Pucheta<sup>a</sup>, Nicolás E. Ulrich<sup>b</sup>, Alberto Cardona<sup>a,\*</sup><sup>a</sup> Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC-INTEC), Universidad Nacional del Litoral - CONICET, Güemes 3450, 3000 Santa Fe, Argentina<sup>b</sup> Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral, Ciudad Universitaria, Paraje "El Pozo", 3000 Santa Fe, Argentina

## ARTICLE INFO

## Article history:

Received 28 December 2012  
 received in revised form 23 April 2013  
 accepted 29 April 2013  
 Available online xxxx

## Keywords:

Conceptual mechanism design  
 Graph theory  
 Graph layout  
 Force-directed layout  
 Planar linkages

## ABSTRACT

The sketching problem arises frequently in the conceptual design of mechanisms, especially in the enumeration process where a large number of topological solutions automatically generated must be analyzed. This paper presents a new graph layout algorithm to sketch non-fractionated kinematic chains. A combinatorial algorithm based on the independent loops of the graph representation of the kinematic chain, is used to find an adequate initial position of graph vertices with minimal edge crossings; its execution is followed by a force-directed algorithm based on spring repulsion and electrical attraction, including a new concept of edge-to-vertex repulsion to improve esthetics and preserve edge crossings. Both algorithms are used in sequence to generate a representative layout of the graph which optimizes a given quality measure. Finally, standard rules are followed to convert the graph into the sketch, using new heuristic correction rules to avoid newly generated edge crossings. Atlases of complex non-fractionated kinematic chains are used to validate the results. A qualitative comparison with a set of sketches found in the literature is included, showing the advantages of the proposed algorithm.

© 2013 Elsevier Ltd. All rights reserved.

## 1. Introduction

The conceptual design is the earliest stage of linkage mechanisms design where several alternatives are generated and evaluated in order to fulfill functional, structural and other design requirements [1]. The graph representation of mechanisms and Graph Theory based algorithms can be used to computationally solve the enumeration of alternatives satisfying topological constraints of a given problem [2,3,1,4–6]. This enumeration process often leads to a large number of topological solutions, that need to be automatically displayed to the designer in graphical form.

The main objective of graph layout is to obtain a representation to ease the visualization and understanding of the graph. This can be achieved if the main esthetics characteristics of the layout are optimized. Among these characteristics, Purchase [7] established that the most important ones are: to maximize symmetry, to minimize edge crossings, and to minimize bends. Also, she concluded that edge crossing avoidance produces the highest impact on human understanding. A second objective, useful for repeated use of the topological structures [8] and database representation [9], is to develop a *deterministic algorithm* where any randomness is eliminated. In this way, the same layout is obtained for different executions of the algorithm over a given graph.

Three problems of automated representation in the conceptual design stage of mechanisms can be identified: (i) drawing of a graph, (ii) drawing of a kinematic chain, and (iii) drawing of a linkage mechanism including parts to move with few known coordinates (see practical applications in Refs. [6,10]). The first two problems can currently be solved by adapting any computer tool available from the computer science field [11–15]; however, the outputs produced by these algorithms are dependent on the initial representation of the graph, and the esthetics characteristics are not satisfactory when the graphs are non-planar and therefore edge-crossings exist. On the other hand, these problems have a long tradition in the mechanism field [4], with solutions presenting improvements in automation, minimization of edge crossings, and esthetics.

\* Corresponding author. Tel.: +54 342 4511594x1013; fax: +54 342 4511169.

E-mail addresses: [mpucheta@intec.unl.edu.ar](mailto:mpucheta@intec.unl.edu.ar) (M.A. Pucheta), [nikolaseu@gmail.com](mailto:nikolaseu@gmail.com) (N.E. Ulrich), [acardona@intec.unl.edu.ar](mailto:acardona@intec.unl.edu.ar) (A. Cardona).

The first sketching algorithms for graphs of kinematic chains were proposed by Dobrjanskyj & Freudenstein in 1967 [2], introducing the use of the minimal independent loops of the graph of the kinematic chain to layout the graph, without considering edge crossing avoidance. Based on this approach, Woo [16] proposed heuristic rules, using the longest loop as the peripheral one, reducing but not eliminating the edge crossings. In 1984, Olson et al. [17] extended the Woo approach with new rules to use the loops as concentric circles for locating the joints of the kinematic chains, but still leaving some corrections of link crossings to be made by hand. In 1989, Yan & Hwang [18], and then Belfiore & Pennestri [19], proposed different drawing strategies based on the characteristics of the graphs with a multilevel approach: they first sketched automatically the contracted kinematic chains from their contracted graphs and then, added joints and links following rules for minimizing edge crossings; however, their study was limited to kinematic chains with ten links. In 1990, Chieng & Hoeltzel [20] proposed the first automated method based on the exhaustive use of the independent loops of the graphs, but also including the peripheral loop. This method was limited to planar graphs and was considerably improved later by Mauskar & Krishnamurthy [21]. They used only the minimal set of independent loops to sketch the kinematic chain, including several graph-based rules to reduce the number of repeated drawings and introducing some rules for avoidance of link-crossings occurrence; however, their approach lacked an emphasis on esthetics criteria. In 2003, Mruthyunjaya [4] reviewed the automated sketching methods and considered Ref. [21] as the most promising candidate for being integrated into a computerized synthesis process. Recently, Ding & Huang [9] proposed a unique and simplified representation of the graphs of kinematic chains where most of the vertices are located on a peripheral loop; they emphasized repeatability over esthetics. Following Olson et al., Nie et al. [22] presented a loop approach to the drawing of joints on concentric circles and also described complex algorithms for corrections of branches for improvement of esthetics. Hsieh et al. [23] extended the work initiated by Yan & Hwang [18] integrating their algorithm into a graphical user interface.

In a previous work [24], the authors presented a preliminary version of a new algorithm for sketching of kinematic chains which employs a combinatorial algorithm that uses the minimal independent loops to find an initial graph with minimum edge-crossings, and then uses a force-directed algorithm to improve the esthetics by keeping the number of edge-crossings to a minimum. Also, a new concept of edge-to-vertex repulsion was proposed and incorporated in the force-directed algorithm to avoid the generation of new edge-crossings. Recently, in Ref. [25], the complexity of the combinatorial part of the algorithm was reduced and the formulae for the edge-to-vertex repulsion forces was improved. In this paper, the conversion from graphs to their associated kinematic chains is presented along with corrections for the avoidance of new link-crossings and further improvements for regularization of edges of the peripheral loop and appropriate orientation of the whole drawing. The algorithm is deterministic, without any randomness in its computation steps. Therefore, it is able to find a single representative among all possible layouts.

Several graphs of complex kinematic chains with one to six independent loops are used to illustrate the algorithm, to explain the methodology and its limitations, and to validate the results.

## 2. Basic definitions

The following basic definitions introduce the equivalences between graphs and mechanisms [2,1,4].

An undirected graph, denoted as  $G(V,E)$ , is an algebraic structure composed by a non-empty set of vertices  $V$  and a set of undirected edges  $E \subseteq V \times V$  connecting vertices (hereafter, in the context of this paper, a simple or undirected graph will be called a *graph*). A kinematic chain can be represented by a graph where the  $n$  bodies are represented by vertices of the graph, and the  $j$  joints between bodies are given by edges connecting the associated vertices. Hereafter, vertices and links will be referred to in an homologous way, and the same consideration is valid for edges and joints. The size of the set of vertices  $V$  is therefore denoted by  $v = n = |V|$ , and the size of the set of edges is  $e = j = |E|$ .

The adjacency matrix,  $A_G$ , is one of the possible matrix representations of a graph. It is a  $n \times n$  square matrix indexed by labels of vertices, where an entry  $a_{ij}$  equals 1 if vertices  $i$  and  $j$  are adjacent, or 0 otherwise.

Two simple graphs  $G_1(V_1,E_1)$  and  $G_2(V_2,E_2)$  are isomorphic if there exists a bijective (one-to-one and onto) function  $f$  from  $V_1$  to  $V_2$  with the property that  $u$  and  $v$  are adjacent in  $G_1$  if and only if  $f(u)$  and  $f(v)$  are adjacent in  $G_2$ , for all  $u$  and  $v$  in  $V_1$ . Such a function  $f$  is called an *isomorphism*.

A path is a sequence of vertices and edges where no edge is traversed more than once. A loop, circuit, or cycle is a path where only the starting and ending vertices are repeated. The length of the loop is the number of edges, which is equal to the number of vertices. Euler's equation establishes that a planar graph has  $\tilde{L} = e - v + 2$  loops, including  $L$  independent loops and the peripheral one, thus  $\tilde{L} = L + 1$ ; hence, there are  $L = e - v + 1$  independent loops (also known as fundamental circuits, loop basis, or cycle basis). Equivalently, the kinematic chain has  $\mu = j - n + 1$  independent loops and its mobility can be computed as  $M = j - \lambda\mu$  if all joints are simple (a joint that permits only one degree of relative movement), where  $\lambda$  is the order of the screw system ( $\lambda = 6$  for spatial mechanisms, and  $\lambda = 3$  for planar and spherical mechanisms). It is possible to find one basis of minimal length loops or minimal independent loops with the following characteristics: (a) any other loop of the graph can be spanned by sums or linear combination of the loops of the basis, (b) no loop is contained into another loop.

The graph layout problem is to find the best representation of a graph by drawing the vertices in a plane without edge crossings if the graph is planar, or minimizing edge crossings if it is not planar. A more general definition is: *Given the graph  $G(V,E)$ , find the positions of the vertices in  $V$ , i.e.  $\mathbf{x}_i = (x_i; y_i) = \{\mathbf{x}_i \in \mathbb{R}^2 | \mathbf{v}_i \in V, i = 1, 2, \dots, n\}$ , that maximize the quality or esthetic criteria.* The positions of the vertices can be collected in a unique array as  $\mathbf{X} = \{\mathbf{x}_1 | \mathbf{x}_2 | \dots | \mathbf{x}_n\}$  indexed by vertex identifiers.

The layout of a kinematic chain has a representation of one polygon per link and one vertex per joint. Additional edges can be added to represent the links inside each polygon with more than three joints, but then they are not shown in the final drawing. The graph of a kinematic chain can be graphically obtained by shortening the sides of each polygon to zero and by separating the locations of the circles representing the joints. However, the converse process is not straightforward. Since the number of edges in a graph layout is less than

the number of edges in the layout of its associated kinematic chain, the more rational starting point is to layout the graph firstly and then to transform it into a kinematic chain.

### 3. Graph layout algorithms

In this section, a new graph layout algorithm which takes the advantages of both combinatorial and force-directed algorithms, with the necessary modifications to avoid their drawbacks, is proposed.

#### 3.1. Combinatorial algorithm based on a set of independent loops of the graph

This algorithm is based on the exhaustive drawing of independent loops, and uses concepts proposed by Mauskar & Krishnamurthy [20,21]. The layout consists on locating the vertices of the first loop over a circle, followed by distributing all successive loop vertices on arcs joined to the previous drawing: each arc starts from a vertex in common with a previous loop and ends in another common vertex. The radius of arcs are drawn in increasing size, e.g. doubling the radius of the previous loop (note that, in this way, minimal chances for edge-crossing are produced). The line connecting the starting and ending points of the arcs divides the plane in two semiplanes; then, each loop can be drawn in two opposite directions: to the left or to the right of this line.

Given the graph of the mechanism in the form of its adjacency matrix, the steps of the loop-based layout algorithm are summarized in Algorithm 1. Let us consider, for instance, a graph with three loops with the following basis  $\mathbf{L} = \{L_1, L_2, L_3\}$ , where:

$$\begin{aligned} L_1 &= \{0, 3, 2, 4\} \\ L_2 &= \{0, 1, 6, 2, 3\} \\ L_3 &= \{0, 3, 2, 6, 1, 7, 5\}. \end{aligned} \quad (1)$$

The first call to function  $C_{\text{IRCLES\_LAYOUT}}$  in Algorithm 1 is made for the ordering  $P = \{0, 1, 2\}$  and directions values  $S = \{1, 1\}$ . This function, detailed in Algorithm 1, performs the following steps. A base circle is drawn using the loop  $L_1$ , see Fig. 1(a). Next, two vertices

---

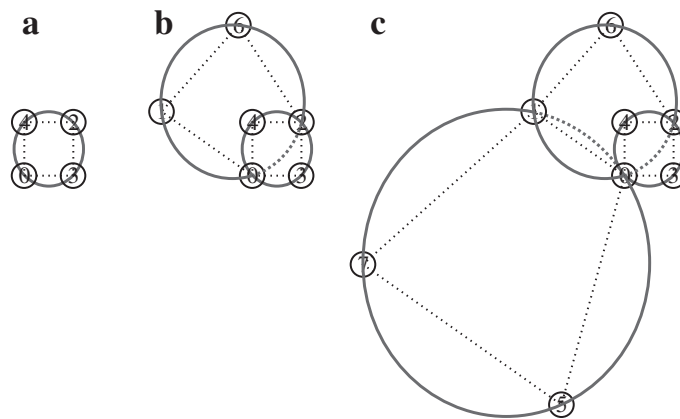
**Algorithm 1**  $\mathbf{X} := \text{LOOPBASEDLAYOUT}(A_G)$

---

**Input:** Adjacency matrix  $A_G$

**Output:** Positions  $\mathbf{X}$  of the set of vertices  $V$  (with minimal edge crossings)

- 1: Compute the number  $\mu$  of independent loops, and obtain a basis of independent loops  
 $\mathbf{L} = \{L_1, L_2, \dots, L_\mu\}$
  - 2: Generate all the sequences,  $\mathbf{S}$ , of  $\mu - 1$  directions stored as +1 or -1. The first loop has no direction; in this way, for each base circle there are  $2^{\mu-1}$  possibilities for drawing the arcs.
  - 3: **for**  $p := 1$  to  $\mu!$  **do** {Loop on loop permutations}
  - 4:   Take a permutation  $P$  of the  $\mu$  independent loops
  - 5:   **for**  $s := 1$  to  $2^{\mu-1}$  **do** {Loop on arc directions}
  - 6:     Take a sequence of arc directions  $S := S[s]$
  - 7:      $\mathbf{X}_{PS} := \text{CIRCLES\_LAYOUT}(\mathbf{L}, P, S)$
  - 8:     **if**  $\mathbf{X}_{PS}$  “better layout than”  $\mathbf{X}$  **then** {Apply suitable criterion}
  - 9:        $\mathbf{X} := \mathbf{X}_{PS}$
  - 10:   **end if**
  - 11:   **end for**
  - 12: **end for**
- 



**Fig. 1.** Combinatorial algorithm based on independent loops.

in common with the next loop  $L_2$  are searched: vertices 0 and 2 are selected as starting and ending points of the arc. Since vertex 3 was already drawn in the previous loop, it is ignored and the first arc includes the list of vertices  $\{2,6,1,0\}$ , see Fig. 1(b). Then, the loop  $L_2$  is processed. The first two vertices in common with the set of the already drawn vertices are vertices 0 and 1, since vertices  $\{3,2,6\}$  were already included in the layout; the new arc will include vertices 7 and 5, Fig. 1(c). The graph layout is obtained by connecting vertices of the base circle and those of the successive arcs as shown by the dashed straight lines in Fig. 1.

---

**Algorithm 2**  $X := \text{CIRCLES\_LAYOUT}(L, P, S)$

---

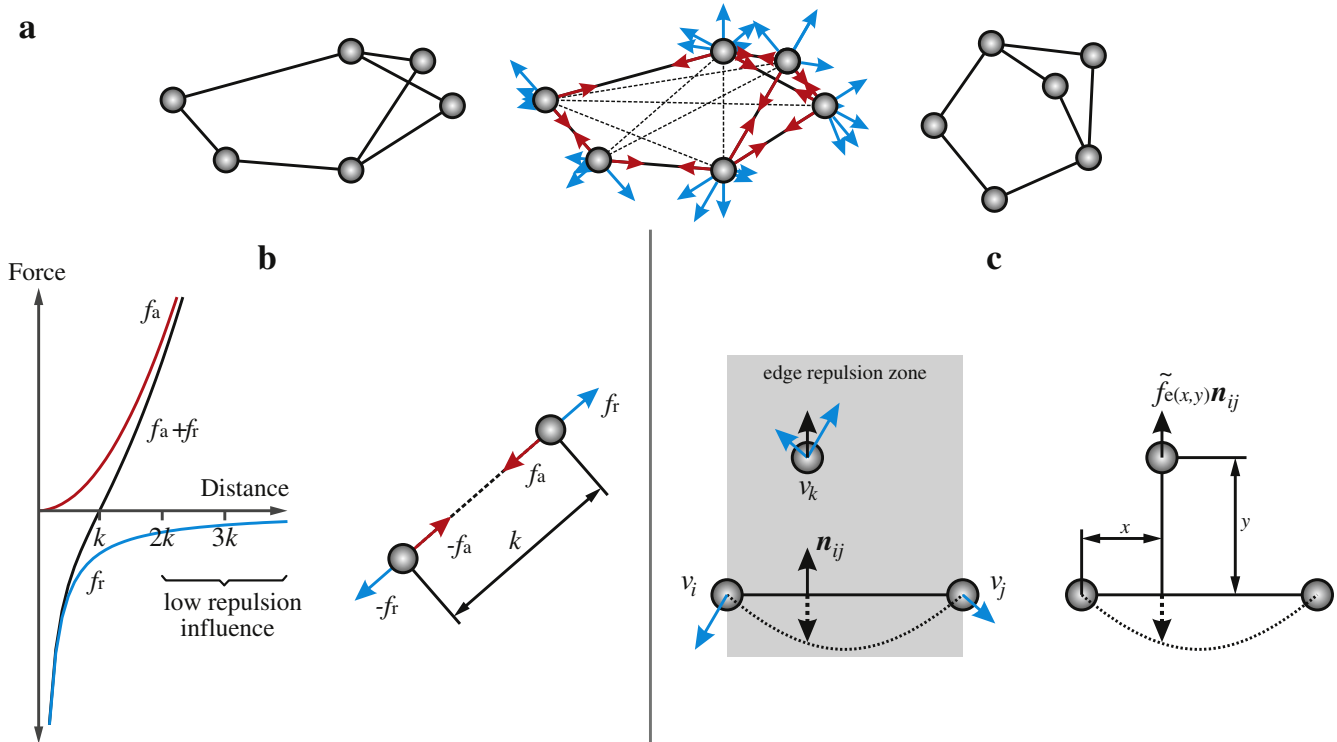
**Input:** Basis of independent loops  $L$ , permutation  $P$ , arc directions  $S$

**Output:** Positions  $X$  of vertices for layout based on circles and graph loops

- 1: Draw the vertices of the first loop  $L_{\text{ini}} := L[P[1]]$  over a circle with radius  $r$ .
  - 2: **for**  $n := 2$  to  $\mu$  **do** {Loop on number of loops}
  - 3: Use the order given by the permutation and take the next loop  $L_n := L[P[n]]$ : Find two vertices,  $v_s$  and  $v_e$ , in common between the loop  $L_n$  and the already drawn vertices. Vertices in common can coincide. If vertices in common do not exist, the drawing is stopped and the function is exit (the next permutation needs to be explored).
  - 4: Compute the arc radius. It is greater than the previous one in a given factor, for instance, 2.
  - 5: Use the direction assigned by the sequence  $S[n - 1]$  and uniformly distribute the free vertices of the loop  $L_n$  over an arc which starts at  $v_s$  and ends in  $v_e$ .
  - 6: **end for**
- 

The computational complexity of this algorithm clearly is  $\mathcal{O}(\mu!2^{\mu-1})$ , dominated by the number of permutations and arc directions.

Note that for complex planar graphs, the rectification of arcs into straight lines can produce new edge crossings that were not present in the layouts drawn using arcs and circles. The layouts generated from a basis of minimal independent loops can be grouped in families which start from a circle that represents the initial loop of the basis; then, for each of them, the arcs are generated depending on the other loops. However, a given graph can have more than one basis of minimal independent loops, and each basis can give different layouts. Therefore, the search of an optimal layout with minimal edge crossings should in fact be performed for the full set of bases  $\mathcal{L} = \{L_1, L_2, \dots, L_b\}$  of minimal independent loops. The final algorithm, which is implemented in Algorithm 4, takes into account this fact and spans the complete solution space.



**Fig. 2.** Example of forces computed in a force-directed algorithm (a); vertex-to-vertex repulsion and attraction force laws [13,24] (b); edge-to-vertex repulsion force between a vertex  $v_k$  and an edge  $e_{ij}$ .

### 3.2. Edge-crossing preserving force-directed algorithm

Fruchterman & Reingold [13] proposed to plot a graph based on an associated pseudo physical or mechanical system with vertices connected by traction springs together with a set of repulsion forces acting between any pair of vertices, see Fig. 2(a). The algorithm starts from a randomly chosen configuration of the vertices in the work plane and then iteratively moves them evolving towards a minimal energy configuration.

The force between two vertices  $v_i$  and  $v_j$  is a function of their current distance  $d_{ij}$  and of a proposed ideal length  $k$  for all edges. The Euclidean distance  $d_{ij}$  between vertices  $v_i$  and  $v_j$  is computed from their positions as  $d_{ij} = |\Delta_{ij}|$ , where  $\Delta_{ij} = \mathbf{x}_i - \mathbf{x}_j$ . Attraction forces are applied in the direction of  $\Delta_{ij}$  over  $v_j$  and in the opposite direction over  $v_i$ . The magnitude of the attraction force is computed as  $\tilde{f}_a(v_i, v_j) = f_a(d_{ij}) = C_a d_{ij}^2 / k$ , while the magnitude of the repulsion force is given by  $\tilde{f}_r(v_i, v_j) = f_r(d_{ij}) = -C_{rvv} k^2 / d_{ij}$ . The constants  $C_a$  and  $C_{rvv}$  are the vertex-to-vertex attraction and vertex-to-vertex repulsion coefficients, respectively; hereafter, we will consider  $C_a = C_{rvv} = 1$ . Using these definitions, the equilibrium between forces acting over a pair of vertices  $v_i$  and  $v_j$  is achieved when the distance between them is  $k$ , see Fig. 2(b). In order to reduce the  $\mathcal{O}(n^2)$  complexity of the repulsion forces, Fruchterman & Reingold [13] proposed to assume the influence of vertices negligible if they are separated at a certain distance, see Fig. 2(b); in this way, repulsion forces at each vertex are computed only for vertices inside a given influence zone, for instance, that limited by a radius  $r$  equal to twice the desired length  $k$  of the edge.

Starting from non-coincident randomly generated positions of the vertices, the Algo. 3 executes several iterations until a stopping criteria on the energy  $E$  is reached; the energy accumulates the magnitude of forces over the vertices (see Algo. 3-line 32).

The main disadvantage of methods based only on vertex-to-vertex interactions is that edge crossing cannot be avoided because the energy of the system (or objective function to minimize) does not take into account edge crossings. Besides, the convergence rate of the algorithm is strongly dependent on the initial state, often randomly generated. To overcome this disadvantage and eliminate the randomness, an optimal initial state can be computed using a determinist graph layout algorithm which minimizes edge crossings, and then a modified force-directed algorithm which preserves edge crossings can be used to improve the esthetics.

A rational approach to obtain an edge-crossing preserving algorithm is to include edge-to-vertex repulsion forces as shown in Algo. 3 in lines 22–28. Fig. 2(c) shows a single interaction between an edge and a vertex. The included edge-to-vertex repulsion force between a vertex  $v_k$  and an edge  $e_{ij}$  has a direction  $n_{ij}$  normal to the edge and is directed towards and applied only to vertex  $v_k$  (while no force is applied to the edge). Its magnitude is computed as

$$f_e(v_k, e_{ij}) = \tilde{f}_e(x, y) = g(y)h(x)$$

and depends on:

- (i) The minimal distance  $y$  from the vertex to the edge through an hyperbolic function  $g(y) \approx C_{rev} k^2 / (y + \varepsilon)$  where  $\varepsilon$  is a small quantity that avoids division by zero, and the constant  $C_{rev}$  is the edge-to-vertex repulsion coefficient; and

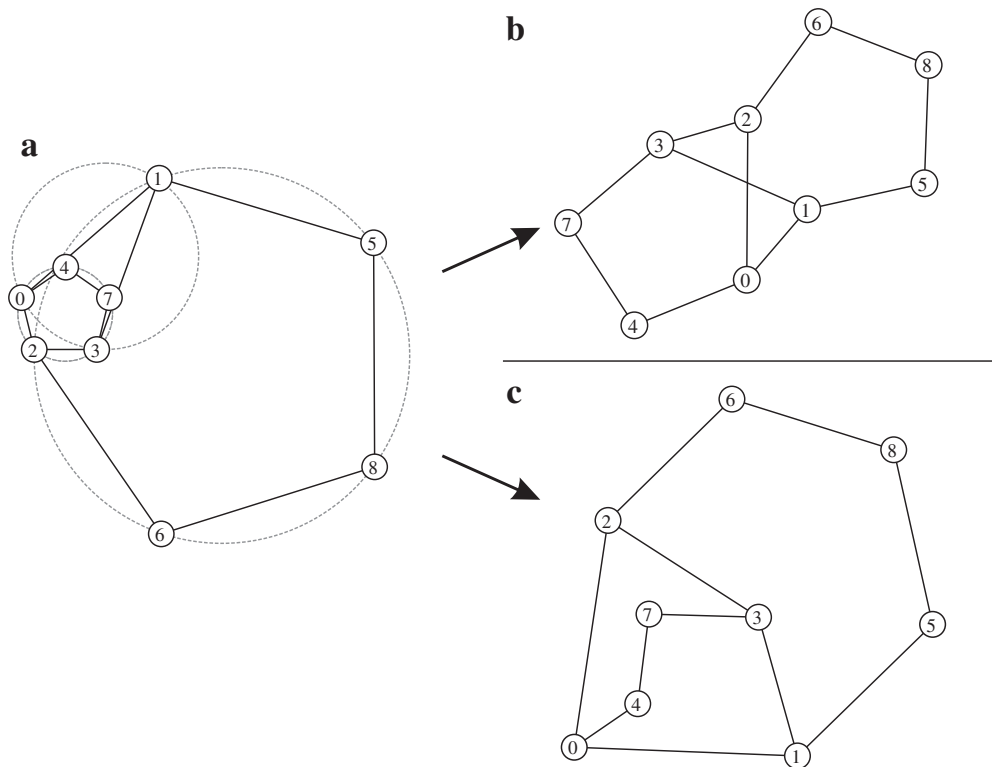


Fig. 3. Loop-based layout (a), followed by a classical force-directed layout (b), and followed by force-directed layout with edge-to-vertex repulsion (c).

(ii) The projection  $x$  of the line joining  $v_j$  and  $v_i$  over the edge  $e_{jk}$  through a parabolic function  $h(x) = 1 - [(2x/d_{jk}) - 1]^2$ .

In this way, the magnitude of the force is increased in a hyperbolic form with  $y$  as the vertex  $v_k$  is near the edge, and varies in parabolic form with the distance  $x$  to the vertices of the edge, reaching a maximum at the mid-point of the edge and getting null values on the vertices  $v_i$  and  $v_j$  (note that repulsion forces between vertices already exist).

Note that a vertex can be projected over more than one edge and all contributions must be computed to get the final repulsion force (interactions can be ignored if the distance from the vertex to the edge exceeds a given value). Then, the computational complexity of the classical force-directed algorithm (see lines 8–20 in Algo. 3) is slightly increased from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(n \times e)$  as shown in Algo. 3 in lines 22–28.

Typical parameters used to update the position of vertices are  $s = k/100$  and  $t = 0.95$ . The parameter  $s$  modifies the position of a vertex in the direction of the resultant of its applied forces (see Algo. 3-line 31). An additional parameter  $b$  is used to update the amount  $s$  of displacement as an adaptive function of the energy evolution. If the energy increases with the iterations, the magnitude  $s$  is decreased to  $s \times t$ , and  $s$  is increased as  $s = s/t$  if the energy is decreased  $b$  times. Hu [14] proposed a value of  $b = 5$ ; this value worked satisfactorily for all the graphs and kinematic chains generated in this work. There are many other improvements to update the positions of vertices inspired in physics which take decisions in terms of the energy evolution [14], or in terms of the velocity and acceleration of the vertices to include damping and inertia to the vertices motion [26]; however, they increase the number of parameters and the computational cost with marginal improvements on the results. Kamada & Kawai [12] also proposed the use of numerical analysis to find efficiently the local minima, but this approach is not easy to apply when there are additional constraints to be computed (e.g., interactions between vertices and other edges or between vertices and a prescribed area).

---

**Algorithm 3**  $X := \text{FORCEDIRECTEDLAYOUT}(G, k, s, t, X)$

---

**Input:** Graph  $G$ , motion parameter  $s$ , relaxation parameter  $t$ , ideal edge size  $k$ , initial positions  $X$

**Output:** Positions  $X$  of the set of vertices  $V$  for a minimal energy configuration

```

1: while the termination criteria is not met (e.g. reduction of  $E$  is not small enough and the
   number of iterations is below the maximum specified) do
2:   Initialize energy and vertex forces
3:   Set  $E := 0$ 
4:   for  $i := 1$  to  $n$  do {Take a vertex  $v_i \in V$ }
5:     Set  $d_i := \mathbf{0}$ 
6:   end for
7:   Vertex-to-vertex repulsion forces
8:   for  $i := 1$  to  $n - 1$  do {Take a vertex  $v_i \in V$ }
9:     for  $j := i + 1$  to  $n$  do {Take a vertex  $v_j \in V$ }
10:      Compute  $\Delta_{ij} := \mathbf{x}_i - \mathbf{x}_j$  and  $d_{ij} := |\Delta_{ij}|$ 
11:       $d_i := d_i - f_r(d_{ij})\Delta_{ij}$ 
12:       $d_j := d_j + f_r(d_{ij})\Delta_{ij}$ 
13:    end for
14:   end for
15:   Vertex-to-vertex attraction forces
16:   for  $k := 1$  to  $e$  do {Take the  $k$ -th edge  $e_{ij} \in E$ }
17:     Compute  $\Delta_{ij} := \mathbf{x}_i - \mathbf{x}_j$  and  $d_{ij} := |\Delta_{ij}|$ 
18:      $d_i := d_i + f_a(d_{ij})\Delta_{ij}$ 
19:      $d_j := d_j - f_a(d_{ij})\Delta_{ij}$ 
20:   end for
21:   Edge-to-vertex repulsion forces
22:   for  $k := 1$  to  $n$  do {Take a vertex  $v_k \in V$ }
23:     for  $w := 1$  to  $e$  do {Take the  $w$ -th edge  $e_{ij} \in E$ }
24:       Compute  $\mathbf{n}_{ij}$  and the relative position of vertex  $v_k$  w.r.t edge  $e_{ij}$ 
25:       Update repulsion force on  $v_i$  as
26:        $d_i := d_i + \sum_{k \neq i, k \neq j, e_{ij} \in E} f_e(v_k, e_{ij})\mathbf{n}_{ij}$ 
27:     end for
28:   end for
29:   Move  $v_i$  in the direction of  $d_i$  an amount  $s$ , and accumulate  $|d_i|$  in the energy  $E$ 
30:   for  $i := 1$  to  $n$  do {Take a vertex  $v_i \in V$ }
31:      $\mathbf{x}_i := \mathbf{x}_i + (d_i/|d_i|) * s$ 
32:      $E := E + |d_i|$ 
33:   end for
34:   Reduce the magnitude  $s$  in a factor  $t$  as the energy increases with the iterations,  $s := s * t$ , or increase the magnitude  $s$  as  $s := s/t$  if the energy is decreased  $b$  times.
35: end while

```

---

### 3.3. Combined algorithm

In order to obtain a robust algorithm which avoids edge crossings, the two algorithms presented in the previous subsections are combined and used in sequence in Algorithm 4: the loop-based combinatorial algorithm is firstly executed followed by the force-directed algorithm based on spring attraction, and vertex-to-vertex and edge-to-vertex repulsion.

The inclusion of the edge-to-vertex repulsion force avoids modifying any initial situation without edges crossings in the final layout, to one with edges crossings or having more edges crossings than in the original one. For example, the initial situation given in Fig. 3(a) would be guided to the state shown in Fig. 3(b) if only the vertex-to-vertex repulsion forces were used. By including the edge-to-vertex repulsion, the graph shown in Fig. 3(a) is guided to the state shown in Fig. 3(c). At the first iterations, the vertices  $v_4$  and  $v_7$  are strongly repelled by the edges  $e_{01}$  and  $e_{31}$ , respectively.

The loop-based combinatorial algorithm requires to include methods for measuring the quality of the obtained layouts and a selection criteria to retain the best of them among all the loop combinations used to generate the initial graph. This aspect is rarely described in the literature and is not trivial. Two measures are here proposed: an integer index  $I_c$  accounting for the number of link crossings, and a continuous index  $I_e$  which measures the variance of edge sizes, accounting thus for vertices distribution and indirectly, for the symmetries. The computation of these indexes have complexities  $\mathcal{O}(e^2)$  and  $\mathcal{O}(e)$ , respectively. Both indexes,  $I_c$  and  $I_e$ , can be arranged in a vector as  $\mathbf{I} = [I_c, I_e]$ , and comparisons can be made by strong precedence to edge crossings. That is, given  $\mathbf{I}_1$  and  $\mathbf{I}_2$ , then  $\mathbf{I}_1 < \mathbf{I}_2$  either if  $I_c^1 < I_c^2$ , or if  $I_c^1 = I_c^2$  and  $I_e^1 < I_e^2$ .

---

**Algorithm 4**  $\mathbf{X} := \text{COMBINEDLAYOUT}(A_G, k, s, t)$

---

**Input:** Adjacency matrix  $A_G$ , motion parameter  $s$ , relaxation parameter  $t$ , ideal edge size  $k$

**Output:** Positions  $\mathbf{X}$  of the set of vertices  $V$  with minimal edge crossings and good aesthetics

```

1: Set  $\mathbf{I}_{\text{best}} = [\infty, \infty]$  and  $\mathbf{X}_{\text{best}} = \mathbf{X}_{\text{rand}}$ 
2: Generate all loops; compute all bases of minimal independent loops denoted as  $\mathcal{L} = \{\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_b\}$ . Set  $\mu = |\mathbf{L}_i|, i = 1, \dots, b$ .
3: for  $g := 1$  to  $b = |\mathcal{L}|$  do {Loop on loop bases}
4:   Take a basis of minimal independent loops  $\mathbf{L} := \mathcal{L}[g]$ .
5:   Generate all the sequences,  $\mathbf{S}$ , of  $\mu - 1$  directions stored as +1 or -1.
6:   for  $p := 1$  to  $\mu!$  do {Loop on loop permutations}
7:     Take a permutation  $P$  of the  $\mu$  independent loops
8:     for  $a := 1$  to  $2^{\mu-1}$  do {Loop on arc directions}
9:       Take a sequence of arc directions  $\mathbf{S} := \mathbf{S}[a]$ 
10:       $\mathbf{X} := \text{CIRCLES LAYOUT}(\mathbf{L}, P, \mathbf{S})$ 
11:      Compute quality index  $I_c$ 
12:      if  $I_c \leq I_c^{\text{best}}$  then
13:         $\mathbf{X} := \text{FORCE DIRECTED LAYOUT}(G, k, s, t, \mathbf{X})$ 
14:        Compute quality index  $I_e$  and build  $\mathbf{I} := [I_c, I_e]$ 
15:        if  $I_e \leq I_e^{\text{best}}$  then {Criteria to select the best layout}
16:           $\mathbf{I}_{\text{best}} := \mathbf{I}$ 
17:           $\mathbf{X}_{\text{best}} := \mathbf{X}$ 
18:        end if
19:      end if
20:    end for
21:  end for
22: end for
23:  $\mathbf{X} := \text{FINAL ADJUSTMENT}(\mathbf{X}_{\text{best}})$ 

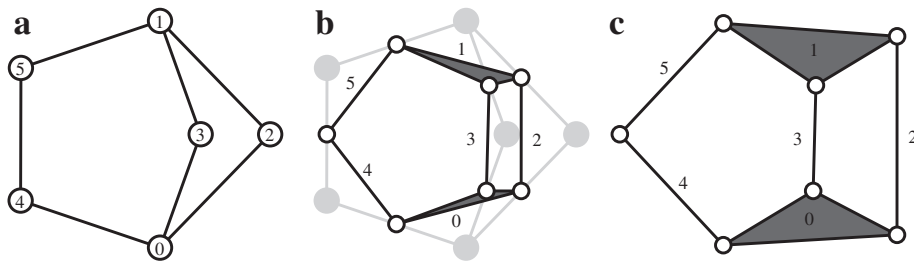
```

---

The total computational complexity of the proposed Algorithm 4 is  $\mathcal{O}(b \times \mu! \times 2^{\mu-1} \times n \times e)$  where  $\mu! \times 2^{\mu-1}$  is the cost of the combinatorial algorithm and  $n \times e$  is the cost of the force directed algorithm which is dependent on the number of vertices  $n$  and on the number of edges  $e$ . Note that this cost would be increased if, instead of plotting the graph, the kinematic chain is drawn directly because it has more edges than the graph (i.e. there are more sides and diagonals of polygonal links than edges in the associated graph).

## 4. Conversion from graph to kinematic chain of a mechanism

A method to convert the graph of the mechanism to its kinematic chain representation is now presented. This problem was called “direct sketching” by Belfiore and Pennestri [19]. The basic procedure for direct sketching of the kinematic chain is merely geometric: starting from the optimal layout of the graph,  $(G, \mathbf{X})$ , each joint of the sketch is located on the mid-point of each edge of the graph; see e.g. Fig. 4(a) and (b). Then, the joints of the sketch are joined by lines: each vertex of the graph is converted into a



**Fig. 4.** Direct sketching of the kinematic chain and refinement using a force-directed algorithm.

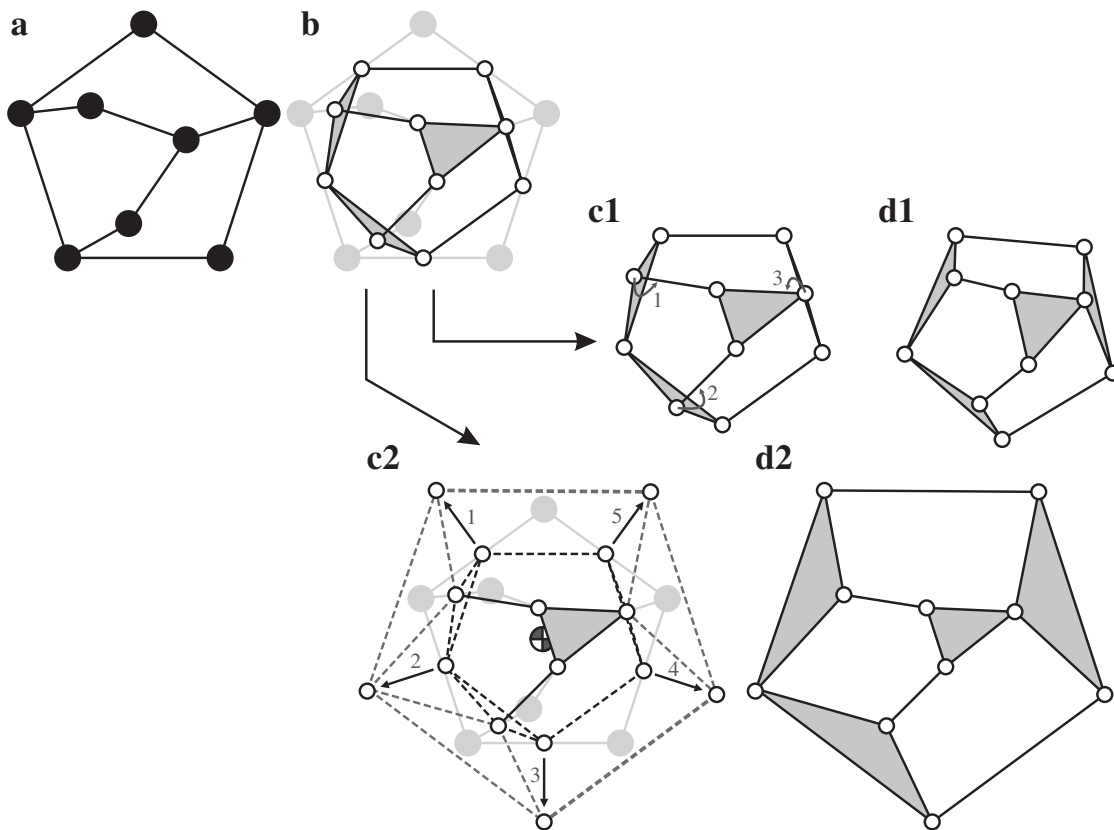
single link (if the vertex degree is 2) or into a polygonal link (if the vertex degree is higher than 2). To obtain regular polygons representing higher order links, one edge is added for each pair of joints. Once the coordinates of the  $s$  joints of the polygon are assigned, a number  $s$  of outer edges are retained in the plot avoiding self-crossings (e.g., a four-sided polygon folded as a bow) and depicted in the sketches, the other  $s(s - 3)/2$  diagonals are not drawn. Finally, the esthetics is improved as shown in Fig. 4(c).

The number of edge crossings in the graph must be equal to the number of link crossings in the kinematic chain. However, new link crossings can be originated in the kinematic chain after the geometric conversion [20,19]. See for example, that the new crossings originated in the conversion from the graph shown in Fig. 5(a) into the kinematic chain shown in Fig. 5(b).

The first correction algorithm was proposed by Chieng and Hoeltzel [20] for link crossings between binary and ternary links, where the joints of triangles like the one shown in Fig. 5(b) are moved as shown in Fig. 5(c1) obtaining Fig. 5(d1).

A second way to overcome this problem was proposed by Belfiore and Pennestri [19], where the joints of the kinematic chain belonging to the convex hull of the graph are moved towards the exterior region over lines perpendicular to the edges of the graph whereas the other joints are fixed; see the correction for the same example in Fig. 5(c2) obtaining Fig. 5(d2). The same result is obtained if the joints of the convex hull of the graph are maintained as fixed and the other joints are contracted by scaling their coordinates from the barycenter of the graph.

These procedures can solve local problems between bodies adjacent to the moved bodies but they are not able to correct all newly generated link crossings. The correction algorithm adopted here is based on the corrections proposed by Chieng and Hoeltzel [20], generalized for any pair of polygonal links.



**Fig. 5.** A kinematic chain (b) obtained through direct sketching conversion from the graph (a) producing new edge crossings; (c) correction actions; (d) corrected, (e) optimized, and rotated kinematic chain.

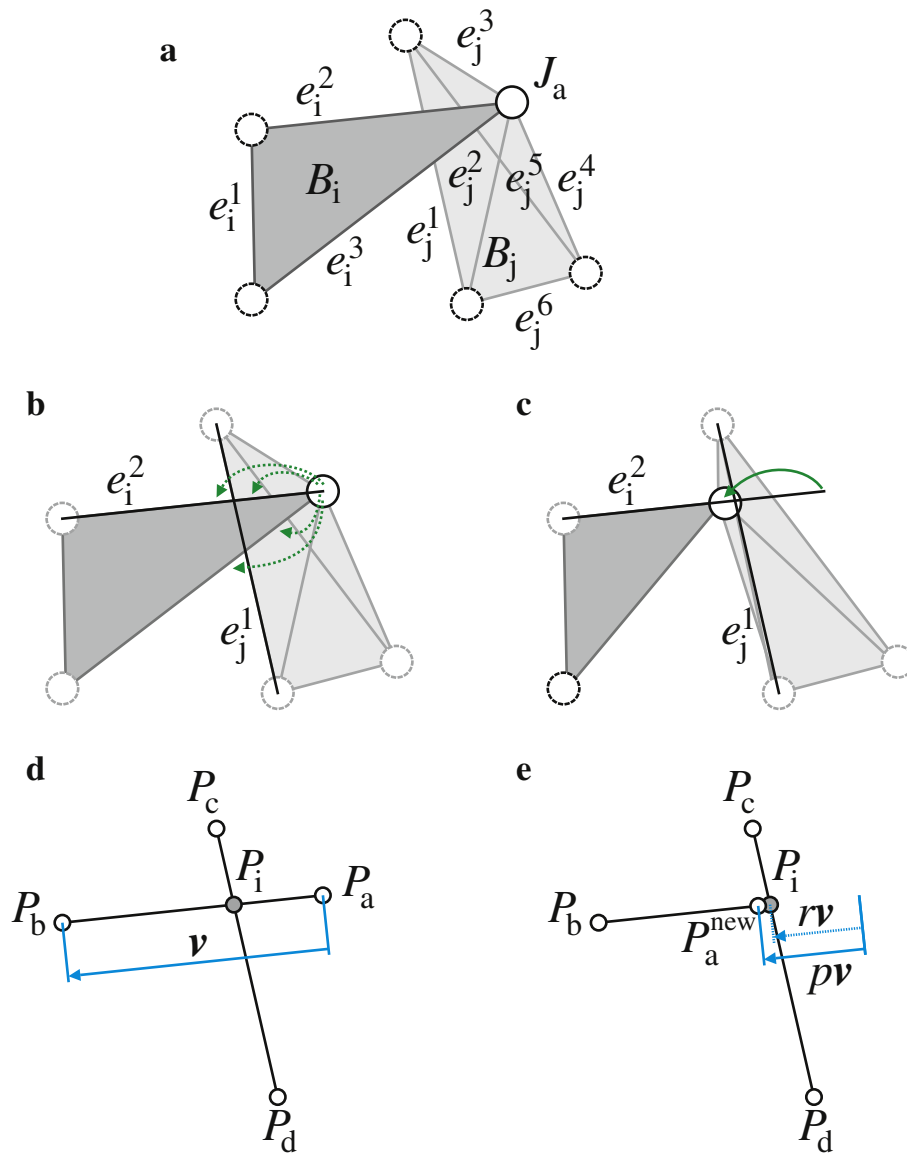


Fig. 6. Two generic links with edge-crossings.

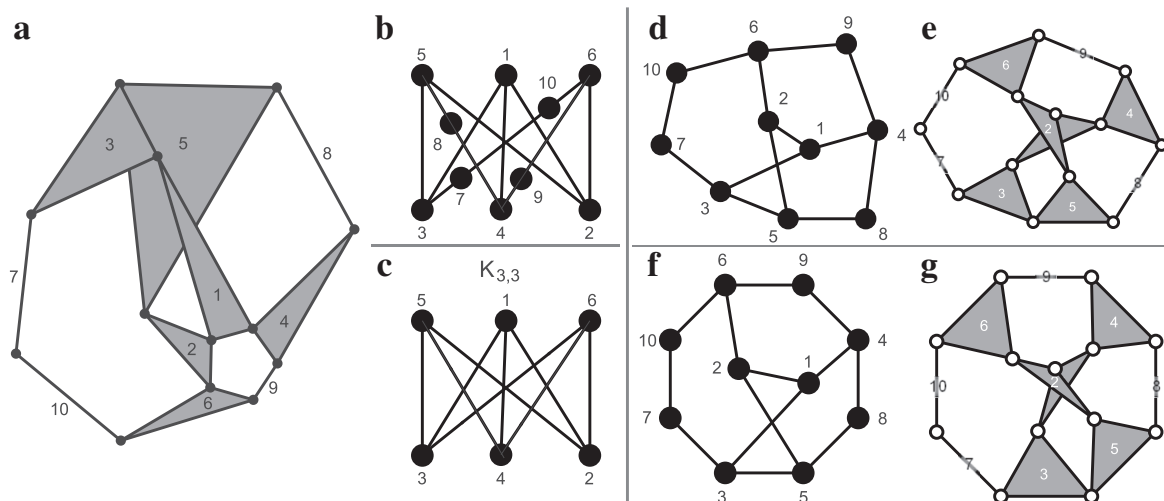


Fig. 7. Graph of a kinematic chain which cannot be drawn without link crossings.

#### 4.1. Proposed correction algorithm

The correction algorithm sequentially verifies all possible edge crossings (here, the word edge includes sides and diagonals of a polygon) between every edge in body  $B_i$  and every edge in an adjacent body  $B_j$ . If necessary, it moves the joint in common to avoid edge crossing. Each individual correction may not avoid the crossings of other pair of edges of the same bodies, but at the end of iterations the process leads to a correction.

Fig. 6(a) shows a pair of overlapped bodies,  $B_i$  and  $B_j$ , connected by a joint  $J_a$  located at position  $P_a$ . The arrows in Fig. 6(b) show that more than one correction can be performed. Fig. 6(b) shows an instance of the algorithm where it is clear that the edge  $e_i^1$  does not cross any edge of link  $B_j$ ; then, the edge  $e_i^2$  is checked against every edge  $e_j^k$ ,  $k = 1, \dots, 6$  of link  $B_j$ . The edge  $e_i^2$  crosses the edge  $e_j^1$ , and therefore, the joint  $J_a$  is moved as shown in Fig. 6(c) using the following simple geometrical correction step.

The intersection point  $P_i$  of the two segments associated to the edges  $e_i$  and  $e_j$ , which belong to adjacent bodies  $B_i$  and  $B_j$  respectively, is computed. The edge  $e_i$  has end-points  $P_a$  and  $P_b$ , where  $P_a$  is associated with the common joint  $J_a$ , and the edge  $e_j$  has end-points  $P_c$  and  $P_d$ . The point  $P_a$  is moved over the segment  $P_cP_d$  to the other side of the intersection. The ratio  $r = P_aP_b/P_aP_i$  is computed, and the joint  $J_a$  is displaced to a new coordinate

$$P_a^{new} = P_a + (P_b - P_a)p$$

where  $p = r + \epsilon$  with  $\epsilon \approx 0.01$ . In this case, this procedure eliminates the penetration of joint  $J_a$  inside body  $B_j$  because there are no further crossings between  $e_i^2$  and edges  $e_j^k$ ,  $k = 2, \dots, 6$  nor between the remaining edge  $e_i^3$  and edges  $e_j^k$ ,  $k = 1, \dots, 6$ .

Finally, the procedure is repeated with the role of  $B_j$  replaced by  $B_i$ , thus eliminating the eventual penetration of joint  $J_a$  inside body  $B_i$ . In this example no crossing appears between edges with the updated position of joint  $J_a$ .

#### 4.2. Final adjustments of the graphs and sketches

Two final adjustments are made on the graphs and sketches.

**Polygonal representation of the peripheral loop:** The representation of the vertices of the peripheral loop as a regular polygon provides a more clear understanding of the sketches and eases the validation with other results, compare Fig. 7(a) and (b) with (f) and (g), respectively; obviously, this conversion is not mandatory. After the relocation of the outer vertices of the graph (or joints of the sketch) on the polygon, a force-directed algorithm with the edge-to-vertex repulsion is used to relocate the inner vertices. Fig. 7(d) and (e) display results obtained without applying this adjustment.

**Rotation:** Each graph is rotated and supported by the lower horizontal edge to give an additional improvement of the esthetics. Once the graph is rotated, the sketch is computed and also rotated a minimal angle in order to preserve the visual correspondence between the homologous vertices and links of the graph and the sketch, respectively.

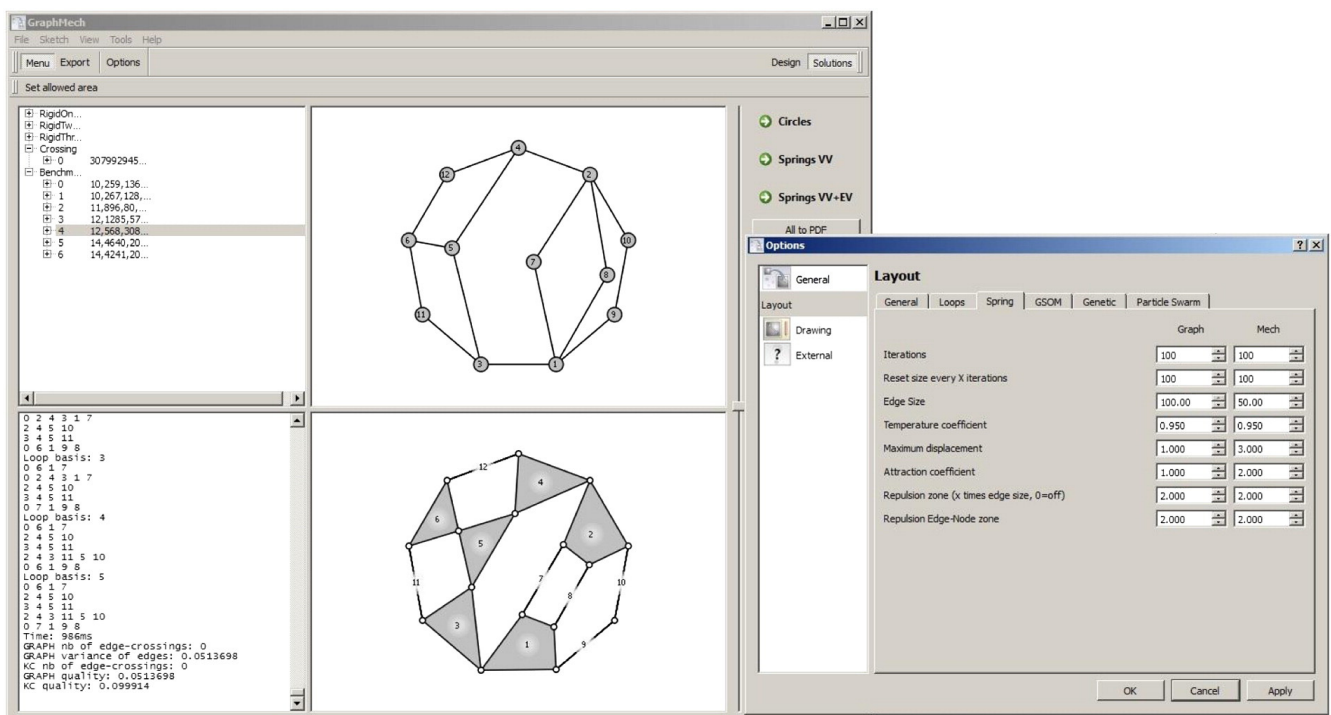


Fig. 8. Graphical user interface for layout of graphs and mechanisms.

The methodology gives a clear representation even in the presence of edge/link crossings. For instance, the sketch of the kinematic chain with a non-planar graph presented by Mauskar & Krishnamurthy ([21], Fig. 12, p.436) is plotted in Fig. 7(a). This is a 10-links 13-joints non-fractionated 1-DOF linkage. Its associated graph, shown in Fig. 7(b), is homeomorphic to the non-planar bipartite graph  $K_{3,3}$  shown in Fig. 7(c); by applying Kuratowski's Theorem [27], it can be shown that it is non-planar and therefore at least one edge-crossing cannot be avoided. The graph  $K_{3,3}$  is the contracted graph (see ([28], Tab. 2, graph V)) of a family of graphs of kinematic chains obtained by addition of binary links to the edges of the contracted graph, one result is the graph shown Fig. 7(b).

The sketches shown in Fig. 7(e) and (g) are obtained for this graph, and they are both more clear than the one shown in Fig. 7(a). The sketch of Fig. 7(g) has the best esthetics and was obtained when vertices of the peripheral loop are distributed on a regular polygon.

## 5. Results

In order to test the presented algorithms, a software was written in C++ language and developed under the Qt environment [29]; see Fig. 8. The reported execution times were measured on a PC Intel® Core i7 950@3.07 GHz 8 GbRAM and exclude the time needed for the construction of the graphical outputs.

The methodology for graph layout and direct sketching of kinematic chains was applied to several atlases of non-fractionated kinematic chains of planar mechanisms ( $\lambda = 3$ ). In all cases, the following parameters were used for the force directed algorithm: ideal edge-size  $k = 100$ , initial step  $s = k/100 = 1$ , and attraction coefficient  $C_a = 1$  for the graphs, whereas the values  $k = 50$ ,  $s = 3$ , and  $C_a = 2$  were used for the sketches. The following parameters were common for graphs and sketches:  $t = 0.95$ ,  $C_{rvv} = C_{rev} = 1$ , the repulsion zones for vertex-to-vertex and edge-to-vertex were limited to  $2k$ , and a maximum number of 100 iterations were used for each execution.

### 5.1. Examples of sketches

The results for an atlas of one-DOF non-fractionated kinematic chains with up to three independent loops [1, App. D] are shown in Fig. 9 and were computed in 608 ms. The results obtained for an atlas of two-DOF non-fractionated kinematic chains [1, App. D] are shown in Fig. 10 and were computed in 1.14 s. The computation time of the 4-loop kinematic chain with 8 bases of minimal independent loops shown in Fig. 7(g) was 0.75 s.

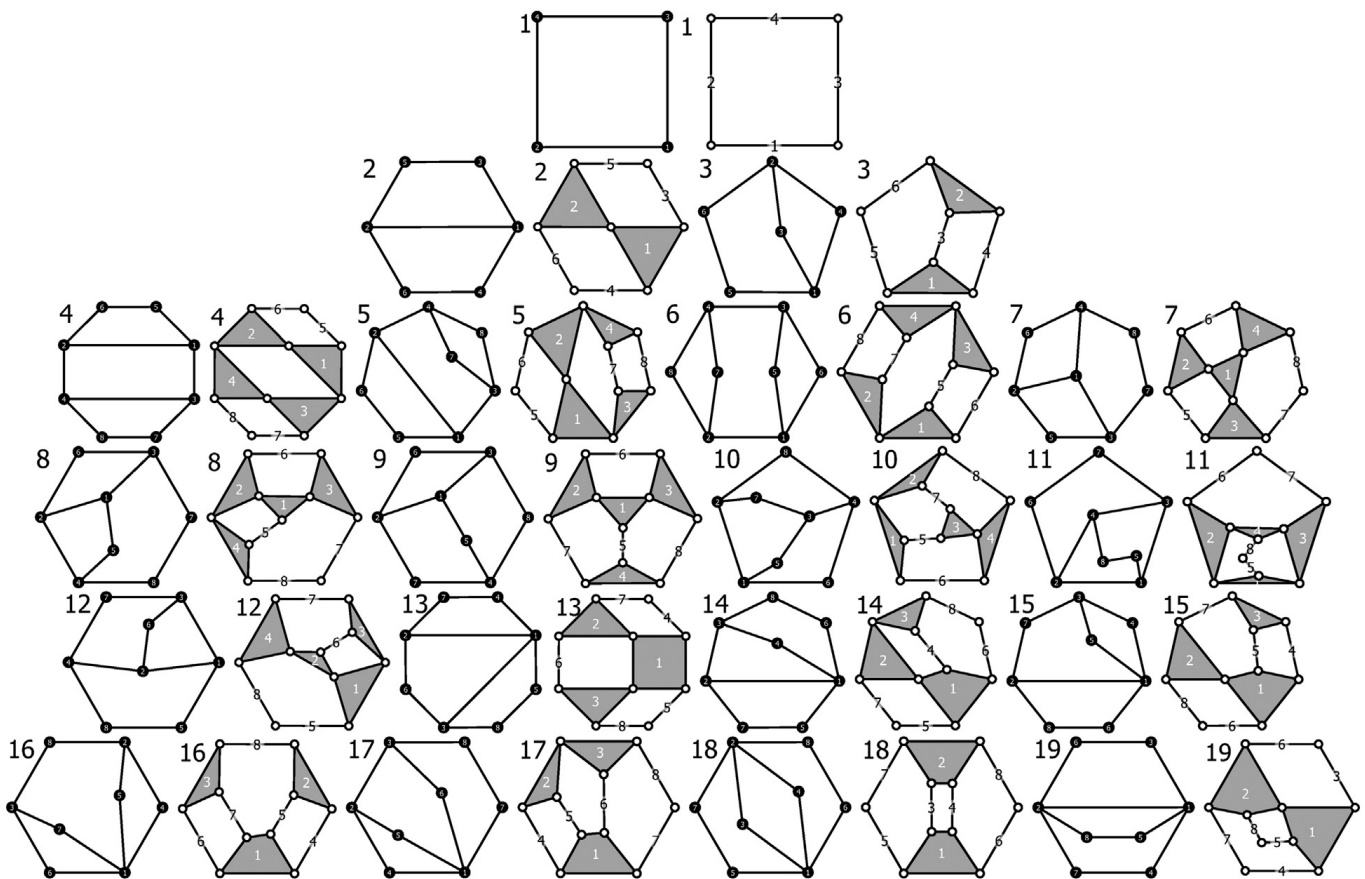


Fig. 9. Atlas of 1-DOF non-fractionated kinematic chains up to 8 links 10 joints.

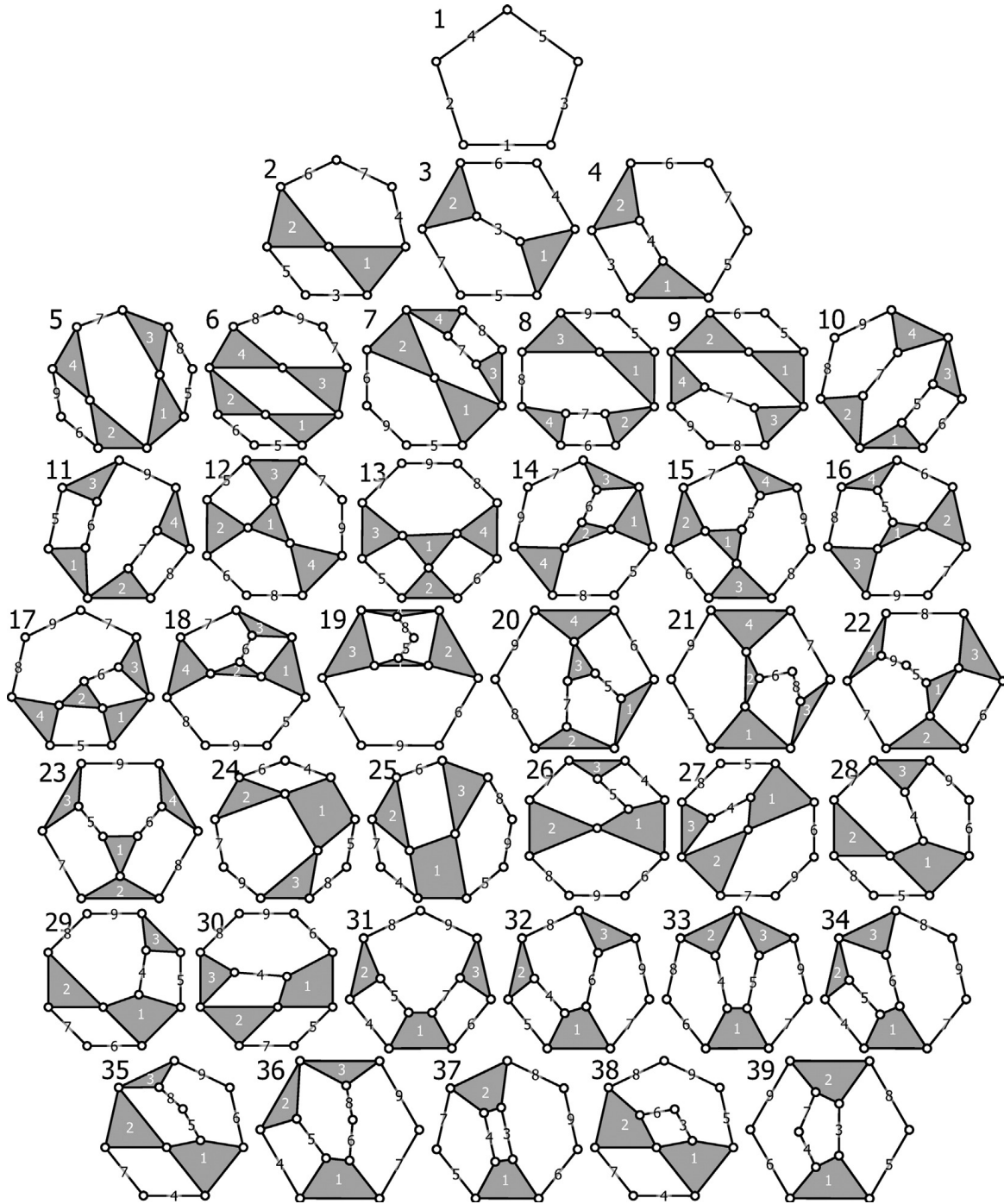


Fig. 10. Atlas of 2-DOF non-fractionated kinematic chains up to 9 links 11 joints.

The 5-loop graph (with 5 bases) shown in Fig. 8 took 0.99 s. This graph is drawn without edge crossings, but appears with edge crossings in Fig. 6 of Ref. [30]. The 5-loop graph shown in Fig. 11(g) has 6 bases and took 2.06 s to be computed. A difficult test is shown in Fig. 11(h); this graph has 6 minimal independent loops and 26 bases, and took 58.06 s to be calculated, with an average time per basis of 2.23 s.

Because of the computational complexity of the algorithm, the CPU times are dominated by the number of minimal independent loops multiplied by number of bases as  $\mathcal{O}(b \times \mu! \times 2^{\mu-1})$ . To the authors knowledge, there is no explicit formula to count the number of bases of minimal independent loops.

## 5.2. Qualitative comparison with other algorithms

Fig. 7 shows the improvements with respect to the approach by Mauskar & Krishnamurthy ([21], Fig. 12,p.436) through the complex 10-bar non-fractionated 1-DOF linkage.

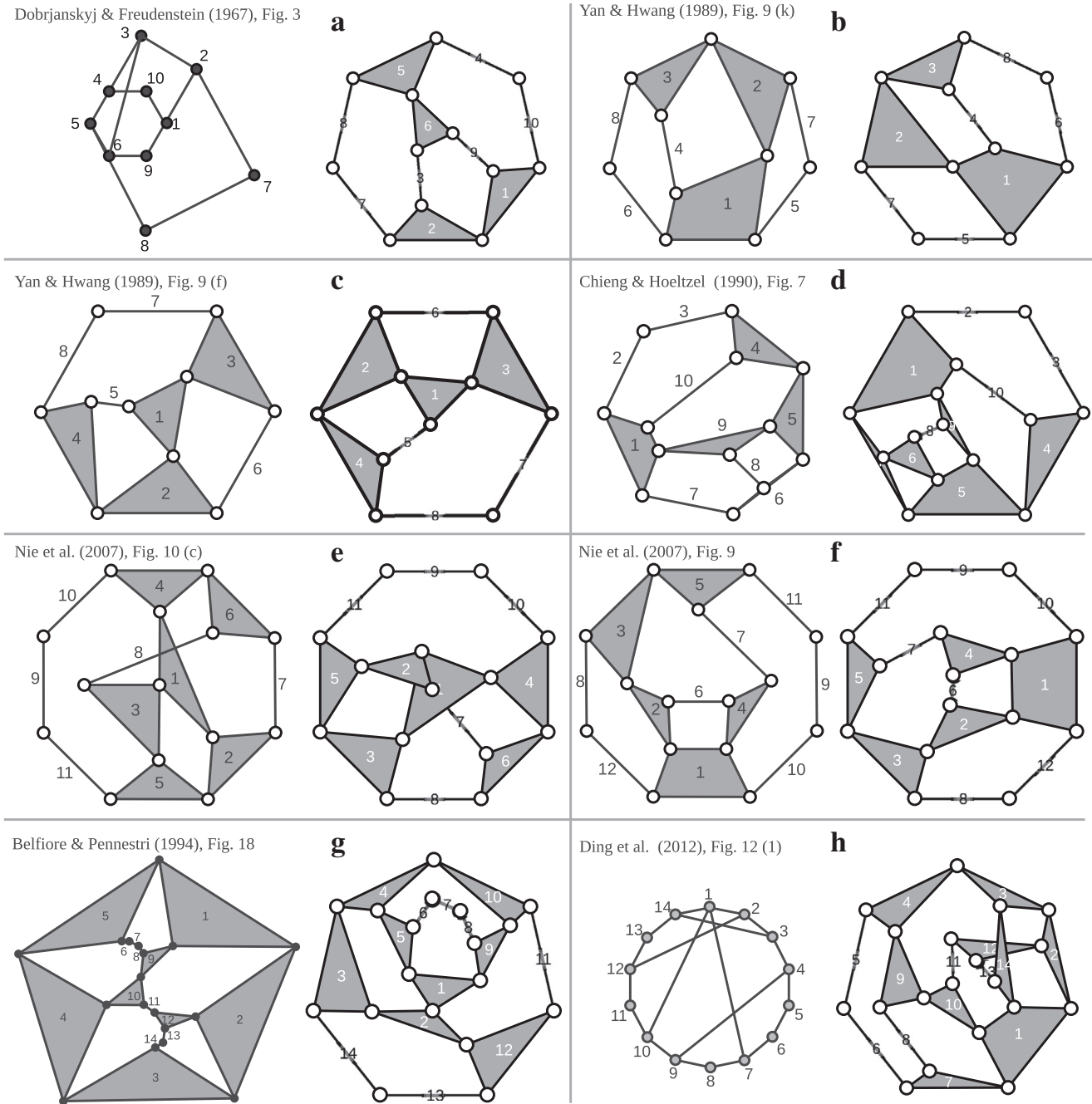
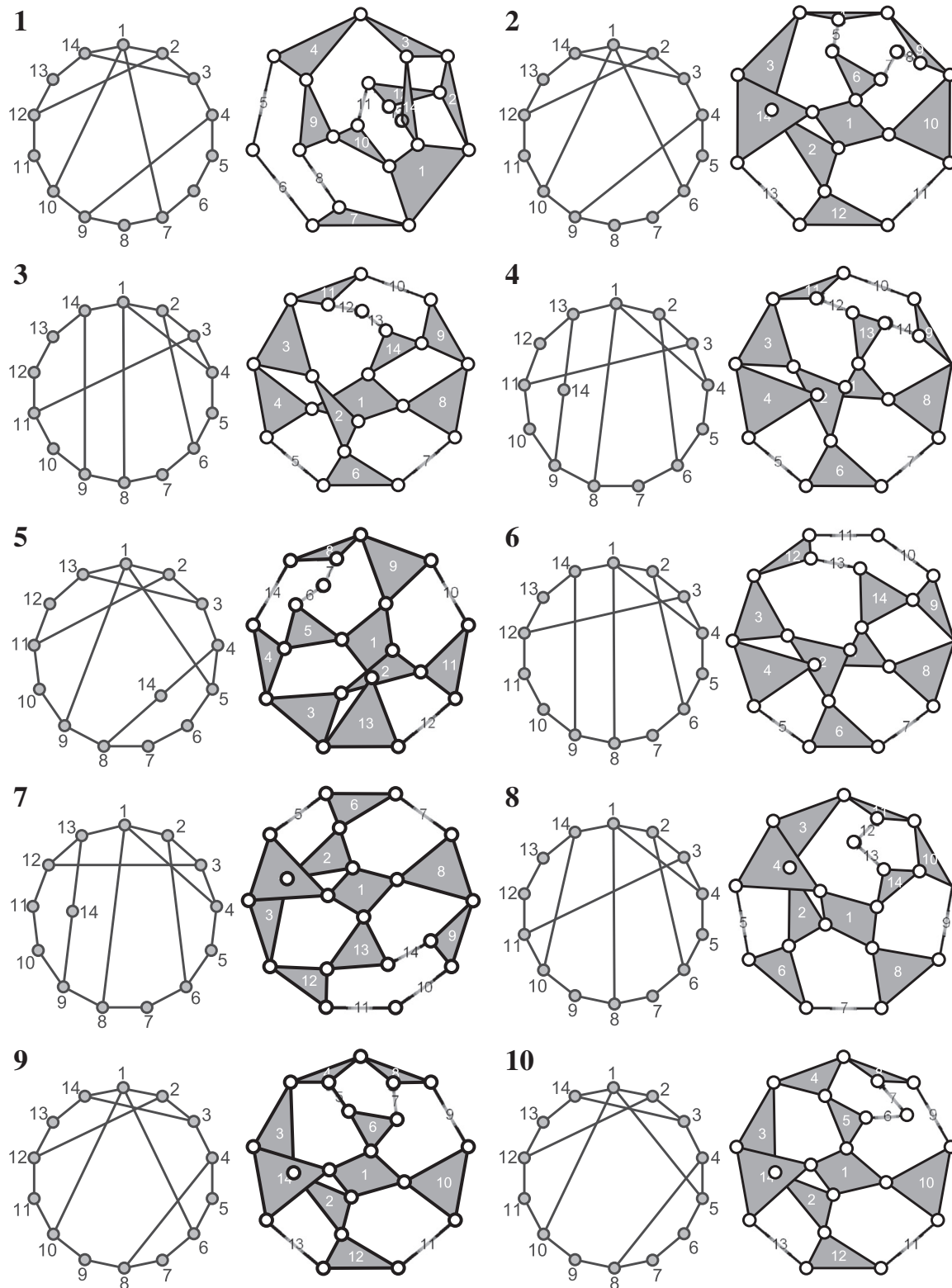


Fig. 11. Comparison with outputs from other algorithms for automated sketching.

Some other examples of non-fractionated kinematic chains taken from the literature on automated sketching are compared in Fig. 11. Each figure was redrawn by hand and arbitrarily labeled for the cases in which the authors do not provide the labels. At the right of each figure, the sketches automatically generated and automatically labeled by the proposed algorithm are confronted with:

- (a) The first automated sketch of the graph of a 1DOF (10,13) kinematic chain from by Ref. [2], the obtained sketch clearly shows the planarity of the graph.
- (b) A 1DOF (8,10) kinematic chain with all unsymmetrical links from Ref. [18] seems to be qualitatively similar in terms of regularity of link sizes. These authors presented the atlas of 1DOF non-fractionated kinematic chains in Ref. [18] and the atlas of 1DOF generalized kinematic chains in Ref. [23] and their method deserves further exploration and research.
- (c) Another 1DOF (8,10) kinematic chain from in Ref. [18] where the kinematic chain has all unsymmetrical links, both algorithms converge to two representations with the same regularity of link sizes.



**Fig. 12.** Sketches of the ten first 1-DOF non-fractionated kinematic chains with 6 independent loops synthesized in Ref. [30].

- (d) A 1DOF (10,13) kinematic chain with 5 minimal independent loops (15 bases) from Ref. [20] without link-crossing in both cases.
- (e) A 2DOF (11,14) kinematic chain with 4 minimal independent loops (4 bases) from Ref. [22], where the number of link-crossing is two in both cases.
- (f) A 3DOF (12,15) kinematic chain with 4 minimal independent loops (8 bases) from Ref. [22] without link-crossing in both cases.

- (g) A 3DOF (14,18) kinematic chain with 5 minimal independent loops (6 bases) from Ref. [19]. Note the avoidance in the congestion of joints and the improvement in the regularity of link sizes of the presented approach.
- (h) A graph of a 1DOF (14,19) kinematic chain with 6 minimal independent loops (26 bases) from the data base provided in Ding et al. [30]. Note that the number of link-crossings is reduced to a minimum of one, because the graph of this kinematic chain is homeomorphic to  $K_{3,3}$  by identifying the vertices as partitioned into  $V_1 = \{1,3,5\}$  and  $V_2 = \{8,4,2\}$ . Although the graph and a sketch are not directly comparable, the sketch clearly shows that there exists just one link-crossing. Some more sketches corresponding to the set of 1-DOF non-fractionated kinematic chains with 6 independent loops synthesized by Ding et al., are plotted in Fig. 12.

We remark finally, that the results are, in all cases, better than the layouts obtained with existent software [15] based on force-directed concepts.

## 6. Discussion

The proposed approach has several advantages:

- (i) The approach is systematic and can start from any matrix representation of the kinematic chain;
- (ii) The parameters of the algorithm are simple to understand and define;
- (iii) The addition of loops drawn as circles followed by parts of loops drawn as arcs always enlarges the graph and decreases the esthetics (which is however improved by the force directed algorithm); it is better than those approaches which put the vertices on concentric circles increasing the congestion of vertices and their associated number of rules [17,22];
- (iv) The algorithm can draw topologies with link-crossings;
- (v) The algorithm is parallelizable; a given set of permutations of loops and directions of arcs can be distributed to compute the combined layouts and quality indexes in different processors;
- (vi) The number of heuristic rules is minimal and only employed for the geometric corrections in the conversion from graph to kinematic chain.

The computational complexity of the presented approach is lower than Ref. [20] and similar to Ref. [21]. No rule to detect loop relationships are used (exterior, interior, independent, crossed) [20–22]. The algorithm was tested for kinematic chains with up to 6 minimal independent loops, with reasonable CPU times for practical applications.

No comparison is made with respect to the algorithm by Yan & Hwang [18,23], because its computational complexity is difficult to evaluate and they did not present it. Besides, their method requires to store drawings of contracted graphs and the mapping from each contracted graph to any kinematic chain derived from it, which also limits its application to a just a set of prewired situations.

## 7. Conclusions

We have presented a new algorithm for layout of kinematic chains and associated graphs, which is a combination of a loop-based algorithm and a force-directed algorithm. The loop-based layout algorithm was used to find an initial layout with minimal edge crossings. A force-directed algorithm based on spring attraction and vertex-to-vertex repulsion, which includes a new concept of edge-to-vertex repulsion to avoid the generation of new edge crossings, was also presented. The combined algorithm produced layouts of graphs with good quality in terms of minimization of edge crossings and maximization of esthetic characteristics. These graphs were converted into sketches of kinematic chains through classical conversion methods, which were improved with new correction heuristic rules for avoidance of link-crossing creation.

A qualitative comparison with a set of layouts of graphs and kinematic chains generated by algorithms proposed in the literature was included, showing the advantages of the proposed algorithm. This set of tests can be considered as a benchmark set for automated sketching algorithms.

Further research will include the extension of the presented algorithm to deal with fractionated kinematic chains.

## Acknowledgments

This work has received financial support from: *Consejo Nacional de Investigaciones Científicas y Técnicas* (PIP 2473), *Agencia Nacional de Promoción Científica y Tecnológica* (ANPCyT PICT-2010-1240), and *Universidad Nacional del Litoral: Fellow for research initiation (Cientibeca)* for N. Ulrich and CAI+D2009 PI65-330 project.

## References

- [1] L. Tsai, *Mechanism Design: Enumeration of Kinematic Structures According to Function*, CRC Press, Boca Raton, 2001.
- [2] L. Dobrjanskyj, F. Freudenstein, Some applications of Graph Theory to the structural analysis of mechanisms, *ASME Journal of Engineering for Industry* 89 (1967) 153–158.
- [3] A.G. Erdman, G. Sandor, 3rd edition, *Mechanism Design: Analysis and Synthesis*, vol. 1, Prentice-Hall, New Jersey, 1997.
- [4] T.S. Mruthunjaya, Kinematic structure of mechanisms revisited, *Mechanism and Machine Theory* 38 (4) (2003) 279–320.
- [5] D. Chen, W. Pai, A methodology for conceptual design of mechanisms by parsing design specifications, *ASME Journal of Mechanical Design* 127 (6) (2005) 1039–1044.

- [6] M. Pucheta, Computational methods for design and synthesis of planar mechanisms. , (Ph.D. thesis) Universidad Nacional del Litoral, Santa Fe, Argentina, 2008.
- [7] H. Purchase, Which aesthetic has the greatest effect on human understanding? in: G. DiBattista (Ed.), Graph Drawing, Vol. 1353 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 1997, pp. 248–261.
- [8] M.A. Pucheta, A. Cardona, Automated type and modular dimensional synthesis of planar linkages, Proc. of ASME IDETC/CIE 2010 Conf., Montreal, Quebec, Canada, 2010, (paper DETC2010-28540).
- [9] H. Ding, Z. Huang, A unique representation of the kinematic chain and the atlas database, Mechanism and Machine Theory 42 (6) (2007) 637–651.
- [10] M. Pucheta, A. Butti, V. Tamellini, A. Cardona, L. Ghezzi, Topological synthesis of planar metamorphic mechanisms for low-voltage circuit breakers, Mechanics Based Design of Structures and Machines 40 (4) (2012) 453–468.
- [11] W. Tutte, How to draw a graph, Proceedings of the London Mathematical Society 13 (52) (1967) 743–768.
- [12] T. Kamada, S. Kawai, An algorithm for drawing general undirected graphs, Information Processing Letters 31 (1989) 7–15.
- [13] T.M.J. Fruchterman, E.M. Reingold, Graph drawing by force-directed placement, Software – Practice and Experience 21 (11) (1991) 1129–1164.
- [14] Y. Hu, Efficient, high-quality force-directed graph drawing, The Mathematica Journal 10 (1) (2006) 37–71.
- [15] S.C. North, Drawing graphs with NEATO, NEATO User Manual, AT&T Research, April 2004.
- [16] L. Woo, An algorithm for straight-line representation of simple planar graphs, Journal of the Franklin Institute 287 (3) (1969) 197–208.
- [17] D.G. Olson, T.R. Riley, D.R. Riley, A.G. Erdman, A combinatorial approach for the automatic sketching of planar kinematic chains and epicyclic gear trains, ASME Journal of Mechanisms, Transmissions, and Automation in Design 107 (1) (1985) 106–111.
- [18] H. Yan, Y. Hwang, New algorithm for automatic sketching of kinematic chains, Proc. of the 1989 ASME ICECE Conf., vol. 1, Anaheim, CA, USA, 1989, pp. 245–250.
- [19] N.P. Belfiore, E. Pennestri, Automatic sketching of planar kinematic chains, Mechanism and Machine Theory 29 (1994) 177–193.
- [20] W. Chieng, D. Hoeltzel, A combinatorial approach for the automatic sketching of planar kinematic chains and epicyclic gear trains, ASME Journal of Mechanical Design 112 (1) (1990) 6–15.
- [21] S. Mauskar, S. Krishnamurty, A loop configuration approach to automatic sketching of mechanisms, Mechanism and Machine Theory 31 (1996) 423–437.
- [22] S. Nie, H. Liu, A. Qiu, A maximal loop approach to automatic sketching of mechanisms, Proc. of ASME IMECE 2007 Conf., Seattle, Washington, USA, 2007, pp. 501–509.
- [23] C. Hsieh, Y. Hwang, H. Yan, Generation and sketching of generalized kinematic chains, Proc. of the ASME IDETC/CIE 2008 Conf., vol. 2, 2009, pp. 1337–1346, (Brooklyn, New York, USA pp. DETC2008–49043).
- [24] M. Pucheta, N. Ulrich, A. Cardona, Graph layout algorithms for automated sketching of linkage mechanisms, Proc. of MECOM-CILAMCE 2010 Cong, Mecánica Computacional, vol. XXIX, 2010, pp. 6125–6135, (Buenos Aires, Argentina).
- [25] M.A. Pucheta, N.E. Ulrich, A. Cardona, Combined graph layout algorithms for automated sketching of kinematic chains, Proc. of the ASME IDETC/CIE 2012 Conf., Chicago, Illinois, USA, 2012, (paper DETC2012-70665).
- [26] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing Algorithms for the Visualization of Graphs, Prentice Hall, 1999.
- [27] K.H. Rosen, Discrete Mathematics and Its Applications, 6th edition McGraw-Hill, 2007.
- [28] W.E. Fang, F. Freudenstein, The stratified representation of mechanisms, ASME Journal of Mechanical Design 112 (4) (1990) 514–519.
- [29] J. Blanchette, M. Summerfield, C++ GUI Programming with Qt 4, Prentice Hall Open Source Software Development Series, 2nd edition, 2008. (Westford, Massachusetts).
- [30] H. Ding, F. Hou, A. Kecskeméthy, Z. Huang, Synthesis of the whole family of planar 1-DOF kinematic chains and creation of their atlas database, Mechanism and Machine Theory 47 (1) (2012) 1–15.