Geometric Constraint Graphs Decomposition Based on Computing Graph Circuits

Robert Joan-Arinyo, & Marta Tarrés-Puertas & Sebastià Vila-Marta [1]

[1]Grup d'Informàtica a l'Enginyeria, Universitat Politècnica de Catalunya, Barcelona, Catalunya

Resumen

Geometric constraint solving is a growing field which plays a paramount role in industrial applications and that is deeply rooted in automated deduction in geometry. In this work we report on an algorithm to solve geometric constraint-based problems by decomposing biconnected graphs. The algorithm is based on recursively splitting the graph through sets with three vertices located on fundamental circuits of the graph. Preliminary practical experiments suggest that the algorithm runtime is at worst quadratic with the total number of vertices in the graph.

Categorías y Descriptores (de acuerdo con ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—subcategorias

1. Introduction

Constraint-based parametric geometric models are data structures designed to represent and describe objects by encoding geometric shape and topological properties. They are at the core of a number of paramount industrial applications, say computer-aided design, robot path planning, molecular design, user interaction with virtual reality systems.

A central issue found in parametric geometric modeling is the constraint solving problem which can be roughly summarized as follows: Given a set of geometric elements and a set of constraints between them, place each geometric element in such a way that the constraints are fulfilled.

In this work, we consider 2D geometric constraint problems defined by a set of geometric elements like points, lines, line segments, circles and circular arcs, along with a set of constraints like distance, angle, incidence and tangency between any two geometric elements. The algorithms that solve geometric constraint problems are named solvers. The reader is referred to the work in [Dur98, HJA05, JASV03, SR98] for an extensive review on geometric constraint solving algorithms.

Among the existing solving methods we focus on constructive techniques. In these techniques the input is a geometric constraint problem represented as a geometric constraint graph. The output is a constructive plan, that is, a sequence of basic steps that describe how to build a solution to the constraint-based geometric problem. Basic steps correspond to elemental operations which are solved with dedicated algorithms.

In this paper we introduce a new algorithm based on the tree decomposition technique reported in [JASRVMVP04]. The algorithm directly computes a graph decomposition from which a constructive plan can be easily derived.

In what follows we assume the reader is familiar with basic terminology of graph theory, the concept of geometric constraint graph associated to a geometric problem defined by constraints, and some definitions related to geometric constraint graphs. For more information we refer the reader to the works by Even, [Eve79], Gao et al., [GLZ06], Hoffmann et al., [HJA05], Joan-Arinyo et al., [JASRTPVM07, JASRVMVP04],Owen [Owe91], Thulasiraman and Swamy, [TS92]. and Whitney, [Whi31],

2. Previous work

Many attempts to provide general, powerful and efficient constructive graph-based techniques to solve geometric constraint problems have been reported in the literature. For an extensive review see the works in [FH97, HLS01a, HLS01b, Owe91]. These techniques decompose the geometric constraint graph into a set of basic subgraphs where each basic subgraph represents a standard problem which can be solved by a fixed algorithm or an equational solver.

In [Owe91] it is described a top-down algorithm for computing a decomposition of an arbitrary constraint graph. The algorithm recursively splits the graph into split components. The algorithm terminates when the graphs cannot be split further. At the end of the analysis the original graph has been decomposed into a set of basic subgraphs. The algorithm closely follows the triconnected components decomposition of [HT73]. In [JASRVMVP04], it is proved that the worst case running time complexity of this algorithm is $O(n^2)$.

[FH97] reported on two graph-based constructive approaches to solve systems of geometric constraints. The top-down method is roughly equivalent to the method by [Owe91]. The bottom-up method, named reduction analysis, begins by computing a set S of basic subgraphs. Then graphs in S are iteratively merged until a unique graph which contains all the geometric elements in the problem is obtained. Fudos and Hoffmann claim the algorithm to have a $O(n^2)$ runtime complexity in the worst case.

[HLS01a, HLS01b] described a flow-based method for decomposing the graph of a geometric constraint problem based on degree of freedom calculations. The method first introduces *dense graphs* which are considered the basic graphs that will not be further split. The algorithm decomposes a given constraint graph into a set of dense subgraphs using a network flow algorithm. This approach is general however, operations needed to solve dense graphs can be arbitrarily complex and not necessarily have geometric meaning.

[JASRVMVP04] defined the tree decomposition of a constraint graph. We review this concept in Section 3. Tree decompositions has been specially useful from a theoretical point of view. Moreover, they are also a suitable representation for constructive plans. Tree decomposition is the technique underlying the graph-based geometric constraint solving framework SolBCN, [SRVMSF07], that has been used to develop the technique reported here.

In general, the graph decomposition is carried out by splitting it into three subsets of vertices that pairwise share just one element. We denote these shared vertices as *hinges*. Since current algorithms identify hinges by an almost exhaustive search in graphs, it is a time consuming computation. Therefore, devising efficient algorithms based on direct computation would be a great accomplishment.

In this paper we present a new algorithm to compute the tree decomposition of a constraint graph. The algorithm is based on the decomposition of a graph in the set of bridges induced by a fundamental circuit. It is inspired in the work of [MR92], developed with the aim of finding the set of triconnected components of a given graph.

3. Preliminaries

In this section we recall basic terminology of graph theory, the concept of geometric constraint graph associated to a geometric problem defined by constraints, and some definitions related to geometric constraint graphs. For more information we refer the reader to the works by Hoffmann et al., [HJA05], Joan-Arinyo et al., [JASRVMVP04, JASRTPVM07], Whitney, [Whi31], Even, [Eve79], and Thulasiraman and Swamy, [TS92].

3.1. Graph concepts

A graph $G=(V,E)$ consists of a set of vertices V, also called nodes, and a set of edges E. An edge $e \in E$ is a pair of vertices $e=(v_i,v_j)$ such that $v_i,v_j \in V$. Vertices v_i and v_j associated with and edge e are called the *end vertices* of e. In general, $V(G)$ will denote the set of vertices and $E(G)$ the set of edges of a graph G. A graph can be represented by a diagram in which a vertex is symbolized by a dot and an edge by a line segment connecting two dots.

The number of edges incident on a vertex v is called the *degree* of the vertex, and is denoted by $d(v)$.

A walk in a graph $G=(V,E)$ is a finite alternating sequence of vertices and edges $[v_0,e_1,v_1,e_2,\ldots,v_{k-1},e_k,v_k]$ beginning and ending with vertices such that v_{i-1} and v_i are the end vertices of edge e_i, $1 \leq i \leq k$. A walk is a trail if all its edges are distinct. Note that any trail is itself a graph. A trail is open if its end vertices are distinct. An open trail is a *path* if all its vertices are distinct. A closed trail is a *circuit* if all its vertices except the end vertices are distinct.

A graph $G=(V,E)$ is *connected* if there exists a path between every pair of vertices in G, otherwise G is *disconnected*. The maximal connected subgraphs of a disconnected graph G are the *connected components* of G.

Let $G=(V,E)$ be a connected graph. According to [TS92], we say that a vertex $v \in V$ is an *articulation vertex* if the subgraph induced in G by $\{V(G)-v\}$ is disconnected. A vertex $v \in V$ is an articulation vertex if and only if there are vertices $u,w \in V$, with $u \neq v$ and $w \neq v$ such that v is on every $u-w$ path.

A *non-separable* or *biconnected* graph $G=(V,E)$ has no articulation vertices, otherwise it is *separable*. See [Whi31].

A *biconnected component* of a connected graph G is a maximal biconnected subgraph of G. A connected graph can be decomposed into biconnected components. For any biconnected graph $G=(V,E)$, given a pair of vertices $u,v \in V$ with $u \neq v$, there are, at least, two disjoint paths $u-v$.

The *connectivity* of a graph G is the minimum number k of vertices that must be removed to disconnect G. If the connectivity of G is k, we write $\kappa(G)=k$. For a disconnected graph G, $\kappa(G)=0$. For a connected graph G, we have $\kappa(G) \geq 1$. A separable graph G has $\kappa(G)=1$. For a biconnected graph G has $\kappa(G) \geq 2$. A graph G with $\kappa(G) \geq 3$ is called *triconnected*. Biconnected graphs can be decomposed into triconnected components.

A graph is said to be *acyclic* if it has no circuits. A *tree* of a graph G is a connected acyclic subgraph of G. A *spanning tree* T for a graph G is a tree that connects all the vertices in V. The edges of a spanning tree T are called the *branches* of T. The edges of G that are not in T are called the *chords*.

Let $G = (V, E)$ be a graph with $|V| = n$ and let G' be a graph such that $G' \subset G$. Following [TS92], G' is said to be a *spanning tree* of G if and only if G' is acyclic, connected, and has $n-1$ edges.

Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$. Let T be a *spanning tree* to G with $b_1, \ldots, b_{n-1}$ *branches* and $c_1, \ldots, c_{m-n+1}$ *chords* of T. The graph resulting from adding to T the chord c_i contains exactly one circuit C which consists of the chord c_i and those branches of T that lie in the unique path in T between the end vertices of c_i. The circuit C is *a fundamental circuit* of G with respect to the chord c_i of the spanning tree T.

The $m-n+1$ possible fundamental circuits $C_1, \ldots, C_{m-n+1}$ of G with respect to the chords of the spanning tree T of G is known as a *set of fundamental circuits*.

Consider a graph $G = (V, E)$. We say that G is *embeddable in a surface S* if G can be drawn in S in such a way that: (1) each vertex $v \in V$ is represented by a point in S, (2) each edge $e \in E$ is represented by a continuous curve $c \in S$ connecting the two points which represent its end vertices, and (3) no two curves share any point but the vertices.

Such a drawing is called an *embedding of G in S*. A graph G embedded in the Euclidean plane is said to be *planar*.

Let G be a graph, S the Euclidean plane and D an embedding of G in S. A *face F* of D is a maximal region of S bounded by edges of D such that for any pair of points (x, y) in F, there is a continuous curve c that connects x to y with $c \in F$.

3.2. The basic constraint problem

In this paper we focus on *the basic constraint problem* defined as follows. Given a set of geometric elements and a set of constraints between them, place each geometric element in such a way that the constraints are fulfilled. We consider 2D geometric elements like points, lines, line segments or circles, along with constraints, like distance, incidence and tangency between any two geometric elements.

The geometric constraint problem can be represented by means of a *geometric constraint graph* $G = (V, E)$, where the nodes in V are geometric elements with two degrees of freedom and the edges in E are geometric constraints such that each of them cancels one degree of freedom.

Once a geometric constraint problem has been translated into a geometric constraint graph, solving the geometric constraint problem amounts to decompose the graph until basic configurations, are found to which standard equational solvers are applied. Therefore, devising feasible algorithms that efficiently decompose constraint graphs is paramount.

As reported by [JASRTPVM07], the concept of *set decomposition* refers to a way of partitioning a given abstract set. Let S be a set with at least three different members, say a, b, c. Let $S_1, S_2, S_3 \subset S$. We say that $\{S_1, S_2, S_3\}$ is a *set decomposition* of S if $S_1 \cup S_2 \cup S_3 = S$ and $S_1 \cap S_2 = \{a\}$ and $S_2 \cap S_3 = \{b\}$ and $S_1 \cap S_3 = \{c\}$.

We say that vertices a, b, c are the *hinges* of the set decomposition, and S_1, S_2 and S_3 are *clusters*. Notice that a set decomposition is not necessarily unique.

In an analogous way, let $G = (V, E)$ be a graph and $V_1, V_2, V_3 \subseteq V$. Then V_1, V_2 and V_3 is a *set decomposition* of G if it is a set decomposition of V and for every edge $e \in E$, $V(e) \subseteq V_i$ for some $i, 1 \leq i \leq 3$.

Roughly speaking, a *set decomposition of a graph $G = (V, E)$*, is a set decomposition of the set of vertices V such that does not break any edge in E.

Finally, we define the concept of *tree decomposition* of a graph. Let $G = (V, E)$ be a graph. A 3-ary tree T is a *tree decomposition* of G if: (1) V is the root of T, (2) each node $V' \subset V$ of T is the father of exactly three nodes, say $\{V'_1, V'_2, V'_3\}$, which are a set decomposition of the subgraph of G induced by V', and (3) each leaf node contains exactly two vertices of V.

Geometric constraint graphs for which there is a tree decomposition will be called *tree decomposable*, that is, the associated geometric constraint problem is solvable by the tree decomposition approach.

4. The Algorithm

Let $G = (V, E)$ be a geometric constraint graph where V represents the set of geoms and E the set of constraints defined between them. Given a set of hinges $\{a, b, c\} \subseteq V$, a set decomposition of G can be trivially computed. Moreover, a recursive application of set decompositions yields a tree decomposition of G.

The goal is now to compute a set of hinges of a constraint graph G. We consider two distinct cases according the connectivity of G. For 0 and 1 connected graphs, there is a smooth approach. We refer to [JASRTPVM07] for the details. For biconnected graphs, the approach is far more difficult. In what follows we focus on this class of graphs. Figure 1 outlines our algorithm.

The algorithm proceeds as follows. First a spanning tree for the graph G is computed by applying a depth-first search. Then the associated fundamental circuits $\{C_1, \ldots, C_n\}$ are

INPUT: biconnected constraint graph $G = (V, E)$, $|V| \geq 3$
OUTPUT: a set of hinges $\{v_1, v_2, v_3\} \subseteq V$, if one exists

Compute a spanning tree T of G
Compute the set of fundamental circuits C of G
    according to T
foreach $C_i \in C$ **do**
    Compute the set of bridges B of G with respect to C_i
    Compute the collapsed graph G'
    Compute the merged graph G''
    Compute the planar embedding D of G''
    foreach $F \in D$ **do**
        foreach $\{v_1, v_2, v_3\} \subseteq F$ **do**
            if $\{v_1, v_2, v_3\} \in C_i$ **then**
                return $\{v_1, v_2, v_3\}$
            endif
        endfor
    endfor
endfor
return $\emptyset$

Figura 1: *Decomposition of a biconnected graph.*

identified. From previous work, [JASRTPVM07], we know that any set of hinges of G must be a subset of the vertices of some fundamental circuit C_i of G. Therefore we restrict the search for hinges to the set of fundamental circuits.

The search is performed in a planar embedding D of a graph G' resulting from transforming the given graph G according to the *bridges*, [JASRTPVM07, TS92], defined in G by the fundamental circuit under study. If the algorithm fails finding a fundamental circuit with a set of hinges, the input graph is not decomposable.

5. Experimental Results

To gain insight on the algorithm behavior and to perform a preliminary assessment of the algorithm runtime behavior, we have implemented it in the `SolBCN` framework which can be downloaded under a GNU General Public License (see [SRVMSF07]).

The tests have been conducted on a standard desk computer with a Pentium IV at 3GHz processor and 1GB of core memory. The algorithm is implemented in Java using the Sun JDK. The tests were planned as follows. Using the methodology defined in [JASRVM06] to generate random geometric constraint graphs, two datasets were defined:

1. D_1: A set of 1000 randomly generated geometric constraint graphs with sizes ranging from 3 to 200 vertices. All the graphs were well-constrained but not necessarily tree-decomposable, that is not necessarily solvable by the tree decomposition approach.
2. D_2: A set of 1000 randomly generated of geometric constraint graphs with sizes ranging from 3 to 200 vertices. All the graphs were under-constrained but not necessarily tree-decomposable.

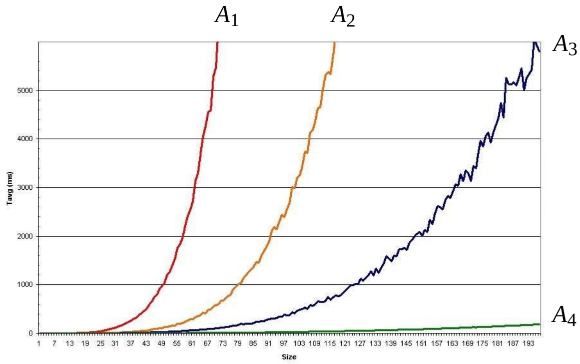

Figura 2: *Behavior of the algorithms A_1, A_2, A_3, and A_4 on the dataset D_1.*

We also defined four versions of the decomposition algorithm:

1. A_1: this is a *brute-force* algorithm that performs an exhaustive search for hinges.
2. A_2: In this version first vertices of degree two are removed. Then the brute-force algorithm is applied.
3. A_3: This version is an improvement of A_2 with specific treatment for 0-connected and 1-connected graphs.
4. A_4: is the algorithm presented in this work.

Let SG_s denote the set of graphs G such that $s = |V(G)|$. Notice that $3 \leq s \leq 200$. We applied each algorithm version to each dataset. For each algorithm and each graph in a data set, we recorded the algorithm runtime $t(G)$. Then for each graph size s, we averaged the runtime values as

$$T(s) = \frac{\sum_{\forall G \in SG_s} t(G)}{s}$$

The results yielded by these tests are represented in Figure 2 for dataset D_1 and in Figure 3 for dataset D_2.

These results show that for both datasets the algorithm A_4 introduced in this paper exhibits a noticeable improved behavior. For graphs G with $|V(G)| \approx 200$, the runtime for the algorithm A_4 is of about 200ms what allows interactive use in the SolBCN framework.

6. Acknowledgments

This research has been partially funded by the Spanish Ministerio de Educación y Ciencia and by FEDER under grant TIN2007-67474-C03-02.

Referencias

[Dur98] DURAND C. B.: *Symbolic and numerical techniques for constraint solving*. PhD thesis, Computer science department, Purdue University, 1998. Major Professor-C. M. Hoffmann.

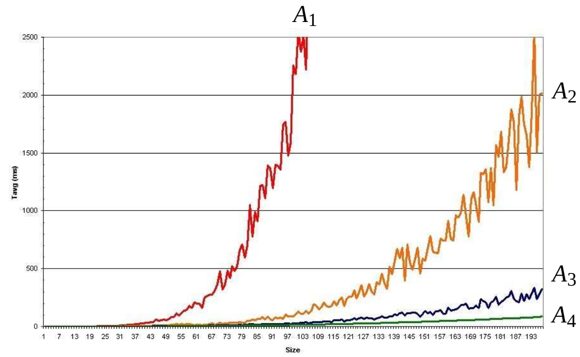

Figura 3: *Behavior of the algorithms A_1, A_2, A_3, and A_4 on the dataset D_2.*

[Eve79] EVEN S.: *Graph Algorithms*. Computer Science Press, Potomac, Md., 1979.

[FH97] FUDOS I., HOFFMANN C. M.: A graph-constructive approach to solving systems of geometric constraints. *ACM Trans. Graph. 16*, 2 (1997), 179–216.

[GLZ06] GAO X., LIN Q., ZHANG G.: A C-tree decomposition algorithm for 2D and 3D geometric constraint solving. *Computer-AIded Design 38*, 1 (2006), 1–13.

[HJA05] HOFFMANN C. M., JOAN-ARINYO R.: A brief on constraint solving. *Computer-Aided Design and Applications 5*, 2 (2005), 655–663.

[HLS01a] HOFFMANN C., LOMONOSOV A., SITHARAM M.: Decomposition plans for geometric constraint systems, Part I: Performance measures for CAD. *Journal of Symbolic Computation 31*, 4 (Apr. 2001), 367–408.

[HLS01b] HOFFMANN C., LOMONOSOV A., SITHARAM M.: Decomposition plans for geometric constraint systems, Part II: New algorithms. *Journal of Symbolic Computation 31*, 4 (Apr. 2001), 409–427.

[HT73] HOPCROFT J., TARJAN R.: Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM 16*, 6 (1973), 372–378.

[JASRTPVM07] JOAN-ARINYO R., SOTO-RIERA A., TARRÉS-PUERTAS M., VILA-MARTA S.: *Decomposition of geometric constraint graphs based on computing fundamental circuits*. Research report LSI-07-31-R, Department de Llenguatges i Sistemes Informatics, Technical University of Catalunya, 2007.

[JASRVM06] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S.: Constraint-based techniques to support collaborative design. *Journal of Computing and Information Science in Engineering 6* (2006), 139–148.

[JASRVMVP04] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S., VILAPLANA-PASTÓ J.: Revisiting decomposition analysis of geometric constraint graphs. *Computer-Aided Design 36* (2004), 123–140.

[JASV03] JOAN-ARINYO R., SOTO A., VILA S.: Resolución de restricciones geométricas. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 20* (2003), 121–136.

[MR92] MILLER G. L., RAMACHANDRAN V.: A new graph triconnectivity algorithm and its parallelization. *Combinatorica 12*, 1 (1992), 53–76.

[Owe91] OWEN J.: Algebraic solution for geometry from dimensional constraints. In *SMA'91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications* (New York, NY, USA, 1991), ACM, ACM Press, pp. 397–407.

[SR98] SOTO-RIERA A.: *Geometric Constraint Solving in 2D*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, UPC, 1998.

[SRVMSF07] SOTO-RIERA A., VILA-MARTA S., SILVA D., FREIXA M.: The SolBCN geometric constraint solver. Source code from `http://floss.lsi.upc.edu`, 2007. Under GNU-GPL license.

[TS92] THULASIRAMAN K., SWAMY N.: *Graphs: Theory and Algorithms*. John Wiley & Sons, 1992.

[Whi31] WHITNEY H.: Non-separable and planar graphs. *Proceedings of the National Academy of Sciences of the United States of America 17*, 2 (February 1931), 125–127.