Geometric Constraints Solving by a Decomposition-Recombination Method

Samy Ait-Aoudia[1], Rabah Meraihi[2], Adel Moussaoui[3]
[1]INI - Institut National de formation en Informatique, BP 68M - Oued Smar 16270 Algiers Algeria
E-mail : s_ait_aoudia@ini.dz
[2]Computer Science and Networks Department, GET/Télécom Paris (ENST), LTCI-UMR 5141 CNRS
46 rue Barrault, 75634 Cedex Paris France
Email : rabah.meraihi@enst.fr
[3]Université M. Boudiaf, Département Informatique, BP 166, Ichebilia, 28000 Msila Algeria
Email : amoussaoui@univ-msila.dz

Abstract

In computer-aided design, geometric modeling by constraints enables users to describe shapes by relationships called constraints between geometric elements. The goal is to capture the designer intent. Then the solution of the problem (i.e. finding the geometric elements) is to be derived automatically. In this paper, we describe a constraint-based modeler that uses a graph based decomposition-recombination method to solve systems of geometric constraints.

Key Words: Computer aided design, geometric constraints, constraints solving, graph-based solvers.

1. Introduction

In computer aided design, geometric modeling by constraints enables users to describe shapes by specifying a rough sketch and adding to it geometric constraints i.e. a set of required relations between geometric elements. The constraint solver must then derive automatically the correct shape needed.

Many resolution methods have been proposed for solving systems of geometric constraints. We classify the resolution methods in four broad categories: symbolic, numerical, rule-oriented and graph-constructive solvers.

In symbolic methods, the constraints are translated into a system of equations. Methods such as Gröbner bases or elimination with resultants are applied to find symbolic expressions for the solutions. These methods are "extremely" time consuming. They are typically exponential in time and space (see [5,9]). They can be used only for small systems.

Numerical methods (Newton-Raphson's iteration, homotopy, Gaussian elimination and so on) for solving systems of equations are $O(n^3)$ or worse (see [2,11,14]). Most numerical methods have difficulties for handling over- and under-constrained schemes. We can also mention the use of Cayley-Menger determinants to yield simpler algebraic systems (see [12,19]). Others optimize the resolution process by pruning the solutions space (see [15]).

Rule-based solvers rely on the predicates formulation (see [16,17,18]). Although they provide a qualitative study of geometric constraints, the "huge" amount of computations needed (exhaustive searching and matching) make them inappropriate for real world applications.

Graph-constructive solvers are stemming from graph theory. They are based on an analysis of the structure of the constraint graph. The graph constructive approach provides means for developing sound and efficient algorithms (see [4,7,8,13]).

In this paper, we describe a 2D constraint-based modeler that uses a graph-based decomposition-recombination method to solve systems of geometric constraints. The algorithm described is an extension of the method given in [1].

This paper is organized as follows. The graph representation of the constraint problem is explained in section 2. We present in section 3, the core algorithm that handles structurally well-constrained problems. In section 4, we briefly compare our method with Hoffman et al.'s methods described in [8]. Finally, we give conclusions in Section 6.

2. Constraints and graphs

2.1. Graph representation

Geometric modeling by constraints enables users to describe geometric elements such as points, lines, circles, line segments and circular arcs by a set of required relationships of distance, angle, incidence, tangency, parallelism and perpendicularity (see [3]).

In this paper, we will consider geometric elements that have two degrees of freedom and constraints that 'fix' one degree of freedom of each object related.

We use an undirected graph G=(V,E) where |V|=n and |E|=m to represent the constraint problem. The graph nodes represent the geometric elements and the constraints are the graph edges.

2.2. Graph structure analysis

Let us now give some definitions concerning the structural properties of the constraint graph.

Definition 1

A constraint graph G=(V,E) where |V|=n and |E|=m is structurally well-constrained if and only if m=2*n-3 and m' ≤ 2*n'-3 for any induced subgraph G'=(V',E') where |V'|=n' and |E'|=m' (see [10]).

Definition 2

A constraint graph G=(V,E) contains a structurally over-constrained part if there is an induced subgraph G'=(V',E') having more than 2*n'-3 edges.

Definition 3

A constraint graph G=(V,E) is structurally under-constrained if it is not over-constrained and the number of edges is less than 2*n-3.

3. Constraints solving

3.1. Basic clusters forming

After specifying a rough sketch and adding to it geometric constraints, the final solution must be derived automatically. The first phase of the algorithm is the formation of what we call "basic clusters". A cluster is a rigid geometric structure whose elements are known relatively to each other. A basic cluster in our meaning is a minimal dense graph and its sequential extension described by Hoffman et al. (see [6,7,8]).

The running time of the algorithm to find all the basic clusters is in O(n(n+m)). This algorithm is network flow based degree of freedom analysis.

3.2. Constraint graph skeleton

When the first phase terminates, we must have a method to assemble all the basic clusters to form the final solution. For this purpose, we construct in the second phase of the algorithm the "skeleton" of the constraint graph. The skeleton is a graph Gs=(Vs,Es) which is obtained by the algorithm given below.

Algorithm :

1. The nodes Vs of the skeleton are the nodes of the constraint graph that belong to several basic clusters (these nodes are marked with at least two basic clusters names).
2. The edges Es of the skeleton are obtained as follows :

   Es=∅;

   For each basic cluster Ci

   Do pick the nodes {N1,N2,…,Nj} of Ci that belong to Vs;

   triangulate the set {N1,N2,…,Nj} by doing: Add* edge (N1,N2) to Es;

   For k=3 to j

   Do Add* edges (Nk,Nk-1) and (Nk,Nk-2) to Es;

If an added ***edge (s,t)*** to ***Es*** is not a member of the initial set of constraints then we insert a "virtual" constraint whose value is easily determined because the nodes ***s*** and ***t*** belong to a rigid structure (a basic cluster). The goal of the triangulation is to have a rigid structure for the skeleton. The "skeletonization" phase is computed in linear time.

We can note that the skeleton can be empty if the constraint graph consist of only one basic cluster (the final scheme is then directly obtained).

After constructing the skeleton, we form its basic clusters. If the skeleton consists of a single basic cluster, all its nodes are placed in the plane.

Once the nodes of the skeleton are placed (computed), each cluster is positioned relatively to the skeleton by computing the three required parameters (two transitional and one rotational). Note that each basic cluster shares at least two nodes with the skeleton. The final scheme is then obtained.

3.3. Recursive skeletonization

Sometimes the skeleton graph consists of several basic clusters. In this is the case, we can't directly assemble the initial basic clusters. We do then a recursive skeletonization (skeletonization of the skeleton). This process is repeated until the final skeleton is formed by one basic cluster. The recursive skeletonization is computed in quadratic time. This final skeleton is placed in the plane. All the basic clusters are placed relatively to the related skeletons in the reverse order of the recursive "skeletoniztion". The solution is obtained when the basic clusters of the initial graph are positioned. The following algorithm abstracts the resolution process.

Algorithm

Solve (G)

```
{  Find the basic clusters of G ;
   if (number of clusters) > 1
   then
      begin
      G' := Skeleton of G ;
      Solve (G') ;
      Place the basic clusters of G relatively
      to the skeleton G' ;
      end   }
```

4. Comparison with other methods

In this section we will compare our proposed algorithm for solving constraint graphs (where each vertex have two degrees of freedom and each edge fix one degree of freedom of each object related to) with the two algorithm proposed by Hoffman et al. (see [8]) called CA (Condensed Algorithm) and MFA (Modified Frontier Algorithm). We have restricted ourselves to these algorithms beacause they operate in an analoguous manner than ours.

4.1. Comparison with CA method

We will summarize here the CA method. The algorithm called Condensing Algorithm is applied repeatedly to constraint graphs to find minimal dense sub-graphs or clusters. A minimal dense cluster can be sequentially extended by adding more geometric objects one at a time, which are rigidly fixed with respect to the cluster. After a cluster has been thus extended, it is then simplified into a single geometric object, and the rest of the constraint graph is searched for another minimal dense sub-graph. The simplification of an extended cluster is done as follows : an extended cluster ***C*** is replaced by a vertex ***v*** of weight ***3*** ; all edges from vertices in ***C*** to a vertex ***w*** outside ***C*** are combined into one edge (***v***,***w***), and the weight of this induced edge is the sum of the weights of combined edges. After the simplification, another solvable sub-graph is found, and the process is continued until the entire graph is simplified into a single vertex.

The CA method has a major drawback. It is not solvability preserving. Consider the constraint graph of figure 1. The vertex weights are 2 (each vertex has two degrees of freedom), the edge weights are 1 (each edge-constraint fixes one degree of freedom). The graphs ABCDNO, EFHGID and NMKLIJ are all dense/solvable. Suppose the cluster S_1=ABCDNO was found and simplified into one vertex S of weight 3. Now the graph is not dense/solvable any more (see page 414 in [8]).

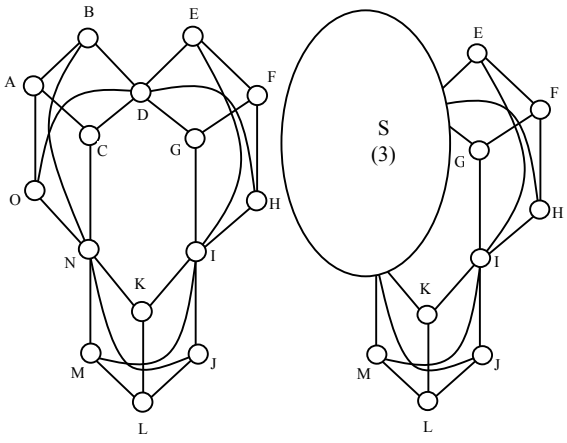

Fig. 1: Original and simplified graph.

The algorithm CA is not strictly solvability preserving is that the removal of the vertices D and N looses valuable information about the structure of the solvable graph. On the other hand, our algorithm solves this case easily. See Figure 2 below.

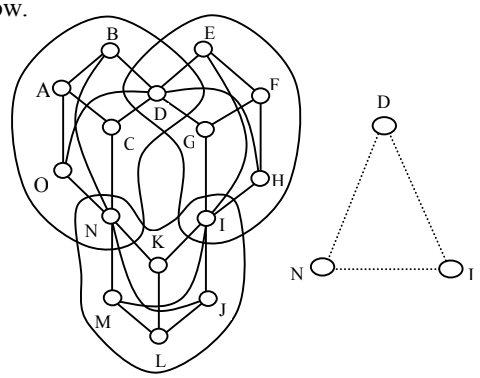

Fig. 2: Basic clusters and the skeleton.

4.2. Comparison with MFA method

We will describe briefly here the MFA method. Once a minimal or extended dense graph ***Si*** is discovered, the sub-graph induced by its internal vertices is contracted into one vertex (the core vertex). This core vertex is connected to each frontier vertex ***v*** of ***Si*** by a combined edge whose weight is the sum of the weights of the original edges connecting internal vertices to ***v***. The frontier vertices, edges connecting them, and their weights remain unchanged. The weight of the core vertex is chosen so that the density of the entire simplified cluster is exactly equal to 3. This process of finding solvable ***Si*** and simplifying them is repeated, until the solvable ***Sm*** found is the entire remaining graph ***Gm***.

This process is illustrated by the sequence of simplification of the constraint graph (see pages 424 and 426 in [8]) shown in figure 3 (left). Initially all vertices have weight 2, all edges have weight 1.

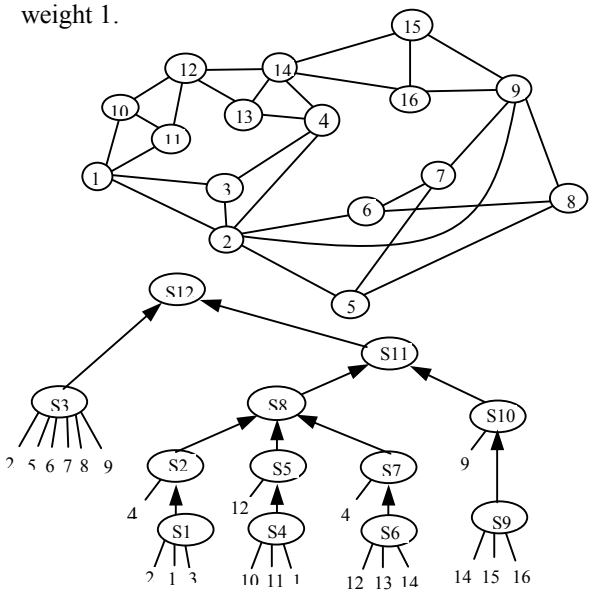

Fig. 3: Constraint graph and partial order of resolution.

Our algorithm proceeds in a simpler manner than the MFA algorithm. First, it never creates additional vertex, as does the MFA method with core vertices. Moreover, the steps to solve a constraint graph are less than the steps needed in MFA method. Figure 4 shows how to solve the previous constraint graph with our method.

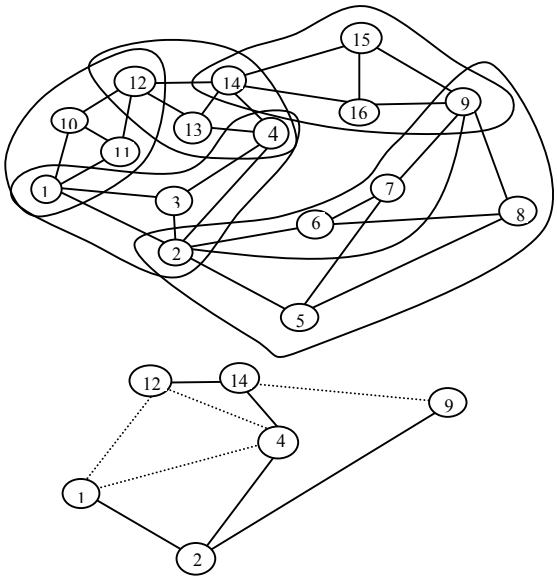

Fig. 4: Basic clusters and the skeleton.

5. Conclusion

We have described and implemented an algorithm that solves a system of geometric constraints using a graph-based decomposition-recombination method. This method is strictly solvability preserving and can deal with 2D constraint graph where each vertex have two degrees of freedom and each edge fix one degree of freedom of each object related to. Further studies must be conducted to deal with general constraint graphs i.e. where vertices can have more than two degrees of freedom and constraints can 'fix' more than one degree of freedom of each object related.

6. References

[1] S. Ait-Aoudia, H. Brahim, A. Moussaoui , T. Saadi. *Solving Geometric Constraints by a Graph-Constructive Approach.* IEEE International Conf. on Information Visualization , London, pp. 250-255, July 99.

[2] E.L. Allgower and K. Georg. *Continuation and path following*. Acta Numerica, pp. 1-64, 1993.

[3] B. Bettig, J. Shah. *Derivation of a standard set of geometric constraints for parametricmodeling and data exchange.* CAD, 33 (2001), 17-33.

[4] I. Fudos, C.M. Hoffman. *A Graph-Constructive approach to Solving Systems of Geometric Constraints.* ACM Trans. on Graphics, Vol. 16, No. 2, April 1997, 179-216.

[5] X.S. Gao and S.C. Chou. *Solving geometric constraint systems: a symbolic approach and decision of Rc-constructibility.* Computer Aided Design, pp. 115-122. vol. 30, n°3, 1998.

[6] C.M. Hoffman, A. Lomonosov, M. Sitharam. *Finding Solvable Subsets of Constraint Graphs*. In Proc. of Principles and Practice of Constraint Programming, Linz, Austria, LNCS 1330, pp 463-477, Springer 1997.

[7] C.M. Hoffman, A. Lomonosov, M. Sitharam. *Geometric constraint decomposition*. In B. Bruderlin and D. Roller, editors, Geometric constraint solving and applications, pp. 170-195, Springer 1998.

[8] C.M. Hoffman, A. Lomonosov, M. Sitharam. *Decomposition Plans for Geometric Constraint Problems, Part II: New Algorithms. Journal of Symbolic Computations 31*, 2001, 409-428.

[9] K. Kondo. *Algebraic method for manipulation of dimensional relationships in geometric models.* Computer Aided Design, 24 (3), pp. 141-147, March 1992.

[10] G. Laman. *On graphs and rigidity of plane skeletal structures.* Journal of Engineering Mathematics, vol.4, n°4, pp. 331-340, Oct. 1970.

[11] H. Lamure and D. Michelucci. *Solving constraints by homotopy.* Symposium on Solid Modeling Foundations and CAD/CAM Applications. May 95. pp. 263-269.

[12] D. Michelucci, S. Foufou. *Using Cayley Menger determinants for geometric constraints solving.* ACM Symposium on Solid Modeling and Applications (2004), Genoa, Italy.

[13] J.C. Owen. *Algebraic Solution for Geometry from Dimensional Constraints.* Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991, pp. 397-407.

[14] A. Perez and D. Serrano. *Constraint base analysis tools for design.* 2nd Symposium on Solid Modeling Foundations and CAD/CAM Applications, Montreal, May 93. pp. 281-290.

[15] J.M. Porta, L. Ros, F. Thomas, and C. Torras. "A Branch-and-Prune Solver for Distance Constraints," IEEE Trans. on Robotics (2005), to appear.

[16] G. Sunde. *Specification of shape by dimensions and other geometric constraints.* Geometric modeling for CAD applications, pp. 199-213. North-Holland, IFIP, 1988.

[17] H. Suzuki, H. Ando and F. Kimura. *Variation of geometries based on a geometric-reasoning method.* Computer and Graphics, 14(2), pp. 211-224. 1990.

[18] A. Verroust, F. Schonek and D. Roller. *Rule-oriented method for parametrized computer-aided design.* Computer Aided Design, 24 (3), pp. 531-540, Oct. 1992.

[19] L. Yan, L. Wenyin, *Engineering Drawings Recognition Using a Case-based Approach*, Proc. 7th International Conf. on Document Analysis and Recognition, ICDAR 2003.