Tools to Deal with Under-constrained Geometric Constraint Graphs

R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta
Universitat Politècnica de Catalunya
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
Av. Diagonal 647, 8a, E–08028 Barcelona

e-mail: [robert, tonis, sebas]@lsi.upc.es

Abstract

We present tools for handling geometric constraint problems with under-constrained configurations. The basic idea is to add to the given set of constraints a set of extra constraints which can be defined either automatically or drawn from an independently given set of constraints placed on some geometric elements of the problem. In both cases, the resulting completed set of constraints is not over-constrained. If every well-constrained subproblem in the given under-constrained configuration is solvable, the completed constraint problem is also solvable.

Keywords: Constraint solving, geometric constraints, rule-based and graph-based constraint solvers, over- and underconstrained systems.

1 Introduction

Geometric constraint-based design is one of the main features of advanced computer-aided design systems and plays a paramount role as a basic tool to automate the product design cycle. In geometric constraint-based design the user inputs an sketch that approximately defines the shape of the object under design and, at some point, annotates the sketch with a set of constraints that precisely defines the intended object. It is left to the system to elucidate whether the object is well defined and, if so, the system generates some sort of representation that provides an effective way to build it. For a review on geometric constraint solving see [2] and references therein.

Existing geometric constraint solving techniques have been developed under the assumption that problems are well-constrained, that is, that the number of constraints and their placement on the geometric elements define

a problem with a finite number of solutions for non-degenerate configurations.

There are a number of scenarios where the assumption of well-constrained does not apply. An example is the early stages of the design process when only a few parameters are established. The problem then is under-constrained, that is, it has an infinite number of solutions for non-degenerate configurations.

Another situation arises in cooperative design systems. Here, different activities in product design and manufacture examine different subsets of the information in the object's model. The presentation of such an information subset has been called a *view*, [1]. Maintaining views consistently is a central issue. To solve this problem, Hoffmann and Joan-Arinyo introduced in [8] the *constraint schema reconciliation* concept. It is assumed that with each view there is associated a geometric constraint graph (formally defined later). If the graphs are different, they are reconciled by drawing constraints from one of the graphs and adjoining them to the other graph. However, how to actually select the adjoined constraints is not defined.

To automate product design and manufacture, techniques to solve under-constrained problems and the schema reconciliation problem should be devised. If the geometric constraint problem is represented by a geometric constraint graph, solving under-constrained problems and the schema reconciliation problem can be seen as specific instances of the *geometric constraint graph completion* problem or just the *completion* problem. This problem is defined as follows:

Problem 1 (Well-constrained completion) *Given the geometric constraint graph associated to an under-constrained geometric constraint problem, add automatically new edges (constraints) to the graph in such way that the corresponding geometric constraint problem is well-constrained.*

Although not much attention has been paid to the well-constrained completion problem, its solution is not obvious at all. Results from distinct fields, such as combinatorial rigidity theory and matroid theory, must be taken into account to achieve an optimal solution for this problem. In Section 7 we briefly review this results and we discuss how to apply them to the solution of the well-constrained completion problem.

Only complete geometric constraint solving techniques solve any well-constrained geometric constraint graph, see [9]. The generality of complete solvers prevents them from always finding a symbolic construction plan. Then, numerical techniques must be applied to compute the solution. On the other hand, constructive geometric constraint solvers, like those in [4, 5, 15], are incomplete. However, they always compute a symbolic construction plan if the analyzed geometric constraint graph is in the domain of the solver. In this paper we study techniques to complete geometric constraint graphs.

The result must be a graph that can be solved by constructive techniques. This problem, which is the object of this paper, is defined as follows:

Problem 2 (Completion) *Given the geometric constraint graph associated to an under-constrained geometric constraint problem, add automatically new edges (constraints) to the graph in such way that the corresponding geometric constraint problem is solvable.*

We report here on a technique that solves Problem 2 in two different scenarios. In one, the extra constraints are automatically defined without any further condition. In the other one, the extra constraints are drawn from an independently given set of constraints defined on the geometries of the problem at hand. In the first scenario, the technique always generates a solvable problem if the initial problem is *completable*, i.e., all subproblems can be solved by a constructive technique.

In the second scenario, it is not always possible to generate a solvable graph, at least in a first step. This situation arises, for instance, when there are not enough extra constraints. However, the algorithm always computes a completable graph, a graph that can be completed by applying the first technique.

The rest of the paper is organized as follows. Section 2 introduces the concept of tree decomposition of a constraint graph. An algorithm to compute tree decompositions is also presented. In Section 3 we recall previous results that characterize the domain of constructive techniques using tree decompositions. Section 4 gives a definition for completability of underconstrained geometric constraint graphs. Section 5 presents the general algorithm for completing an underconstrained graph with extra constraints with no further conditions. Section 6 presents the algorithms for completing an underconstrained graph by adding constraints drawn from a given set. The well-constrained completion is discussed in Section 7. Finally we offer a brief summary in Section 8.

2 Tree Decomposition

In this section first we define the concepts of *set decomposition* and *tree decomposition*. Then we outline the algorithm used to compute tree decompositions.

2.1 Definitions

The concept of *set decomposition* refers to a way of partitioning a given abstract set. The concept of *tree decomposition* of a graph, is a tool that we will make use of later on.

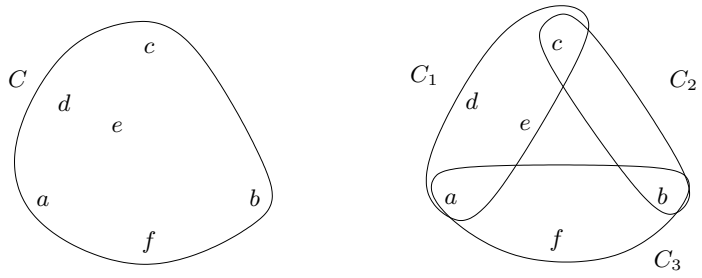

Figure 1: A set C (left) and a set decomposition of C (right).

Definition 1 *Let C be a set with, at least, three different members, say a, b, c. Let $\{C_1, C_2, C_3\}$ be three subsets of C. We say that $\{C_1, C_2, C_3\}$ is a* set decomposition *of C if 1) $C_1 \cup C_2 \cup C_3 = C$, 2) $C_1 \cap C_2 = \{a\}$, 3) $C_2 \cap C_3 = \{b\}$ and 4) $C_1 \cap C_3 = \{c\}$.*

Figure 1 shows a set and a possible set decomposition.

Next we define the concept of set decomposition of a graph.

Definition 2 *Let $G = (V, E)$ be a geometric constraint graph, let $\{V_1, V_2, V_3\}$ be three subsets of V, and let E_1, E_2 and E_3 be the subsets of E corresponding to the subgraphs of G induced by V_1, V_2 and V_3 respectively. $\{V_1, V_2, V_3\}$ is a* set decomposition *of G if it is a set decomposition of V, and E_1, E_2 and E_3 are a partition of E.*

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. Figure 2 illustrates this concept.

The concept of *tree decomposition* of a graph is defined as follows.

Definition 3 *Let $G = (V, E)$ be a geometric constraint graph. A 3-ary tree T is a* tree decomposition *of G if*

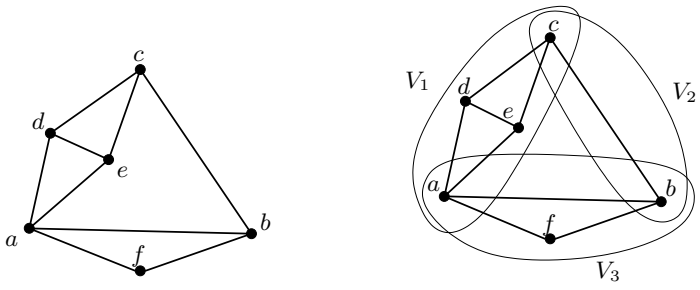

Figure 2: A graph G (left) and a set decomposition of G (right).

1. *V is the root of T,*

2. *Each internal node $V' \subseteq V$ 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.*

We will refer to a geometric constraint graph for which there is a tree decomposition as a *tree decomposable* graph. In what follows the terms well-constrained, over-constrained and under-constrained applied co consttaint graphs refer to the definition given by Fudos and Hoffmann in [5]. Tree decomposable graphs are not over-constrained. This is stated in the following lemma (Proofs of lemmas and propositions are given in [13]).

Lemma 1 *Let $G = (V, E)$ be a geometric constraint graph. If G is tree decomposable, then G is not over-constrained.*

Figure 7 left shows an example of an under-constrained graph and Figure 9 shows a tree decomposition of this graph. Notice that all the edges of the graph correspond to a leaf in the tree decomposition. The following proposition generalizes this observation.

Proposition 1 *Let $G = (V, E)$ be a geometric constraint graph and T be a tree decomposition of G. For each edge $(a, b) \in E$, there is a leaf $\{a, b\}$ in T.*

Well-constrained graphs that are tree decomposable fulfill an additional property: there is a one-to-one correspondence between the edges in the graph and the leaves in the tree.

Definition 4 *Let $G = (V, E)$ be a geometric constraint graph. T, a tree decomposition of G, is a* full tree decomposition *of G if there is a one-to-one correspondence between the leaves of T and the edges of G.*

We will refer to a graph for which there is a full tree decomposition as a *full tree decomposable* graph.

Lemma 2 *Let $G = (V, E)$ be a geometric constraint graph. G is well-constrained and tree decomposable if and only if G is full tree decomposable.*

If a graph is tree decomposable, then any subgraph obtained by removing some of its edges is also tree decomposable. This is a useful property of tree decompositions which we will make use of later on.

Proposition 2 *Let $G = (V, E)$ be a tree decomposable graph. For all $E' \subseteq E$, $G' = (V, E')$, the subgraph of G with the set of edges E', is tree decomposable.*

2.2 Computing a Tree Decomposition

First we introduce the concept of *superblocks* defined in a geometric constraint graph with respect to an articulation point. Then we give an efficient algorithm to compute a set decomposition of a graph. A recursive application of this algorithm yields a tree decomposition of a graph.

Definition 5 *Let $G = (V, E)$ be a constraint graph. Let $a \in V$ be an aticulation vertex of G. The* superblocks *of G are the graphs induced by the equivalence classes defined in E by the following property. Two edges in E, say e_i and e_j, belong to the same class if and only if there is a path including e_i and e_j where vertex a does not occur except as an endpoint.*

When needed, if $G = (V, E)$ is the constraint graph, we will denote V by $V(G)$ and E by $E(G)$.

Assume that the following procedures on a geometric constraint graph G are defined, [7].

- `ConnectedComponents (`G`)` computes the set of connected components.
- `ArticulationVertex (`G`)` computes an articulation vertex.
- `FundamentalCycles` (G) computes the fundamental cycles.

Then, our algorithm, inspired on the work reported in [14], can be stated as follows.

procedure SetDecompositionOfaGraph
INPUT
$G = (V, E)$: Geometric constraint graph with $|V| \geq 3$.
OUTPUT
success: is **true** if G is set decomposable.
$\{V_1, V_2, V_3\}$: a set decompostion of G .

$\kappa :=$ Connectivity (G)
case
$\kappa = 0$: $\{G_1, G_2, \ldots, G_n\} :=$ ConnectedComponents (G)
$V_1 := V(G_1) \cup V(G_2) \cup \ldots \cup V(G_{n-1})$
Choose $a, b \in V_1$, with $a \neq b$
$V_2 := V(G_n) \cup \{b\}$
Choose $c \in V(G_n)$
$V_3 := \{a, c\}$
success := **true**
$\kappa = 1$: $a :=$ ArticulationVertex (G)
$\{G_1, G_2, \ldots G_n\} :=$ SuperBlocks (G, a)

$V_1 := V(G_1) \cup V(G_2) \cup \ldots \cup V(G_{n-1})$
$V_2 := V(G_n)$
Choose $b \in V_1$, with $b \neq a$
Choose $c \in V_2$, with $c \neq a$
$V_3 := \{b, c\}$
success := **true**

$\kappa = 2$: $\{C_1, C_2, \ldots, C_n\}$:= FundamentalCycles (G)
$i := 1$
DecompositionByCycle (G, C_i, success, V_1, V_2, V_3)
while not success **and** $i < n$ **do**
$i := i + 1$
DecompositionByCycle(G, C_i, success, V_1, V_2, V_3)
endwhile

$\kappa \geq 3$: success := **false**

endcase
endprocedure

Now, the procedure `DecompositionByCycle()` is as follows.

procedure DecompositionByCycle
INPUT
G : Constraint graph.
C : Cycle in G.
OUTPUT
success: **true** if G is set decomposable according to C.
V_1, V_2, V_3 : Set decomposition of G.

$\{B_1, B_2, \ldots, B_n\}$:= Bridges (G, C)
$\{B'_1, B'_2, \ldots, B'_m\}$:= MergeBridges ($B_1, B_2, \ldots, B_n$)
$\{F_1, F_2, \ldots, F_l\}$:= FacesOfPlanarEmbedding ($C, B'_1, B'_2, \ldots, B'_m$)
$i := 1$
ThreeVerticesInCommon (C, F_i, success, a, b, c)
while not success **and** $i < l$ **do**
$i := i + 1$
ThreeVerticesInCommon (C, F_i, success, a, b, c)
endwhile
if success **then**
V_1, V_2, V_3 := SplitGraph (G, a, b, c)
endif
endprocedure

Where `MergeBridges()` merges a set of bridges into a smaller set, and `FacesOfPlanarEmbedding()` computes the faces of a planar embedding of a set of bridges. Definitions for these terms can be found in [3].

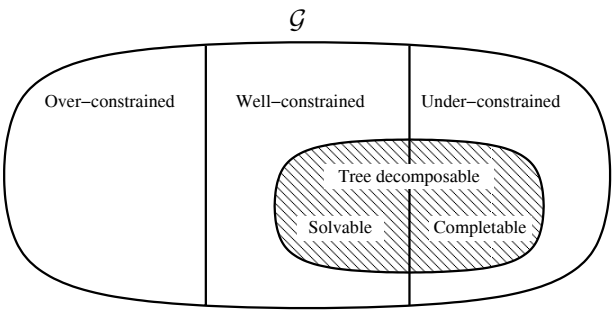

Figure 3: Classification of geometric constraint graphs according to Definition 9 and the set of tree decomposable graphs.

3 The Domain of Some Constructive Techniques

First we recall a previous result that shows that the domain of some known constructive geometric constraint solving techniques is the same. Then we give a constructive procedure that generates the class of graphs in this domain.

3.1 Domain Characterization

In previous papers, [10, 11, 12], we proved that the domain of three constructive geometric constraint solving techniques is the same. Moreover, we proved that the domain of these techniques is the class of full tree decomposable graphs. We recall from [12] the following theorem. In this theorem, we say that a geometric constraint graph G is s-tree decomposable if it can be solved by Owen's algorithm, [15]; we say that G is solvable by reduction analysis if it can be solved by Fudos and Hoffmann's reduction analysis, [4]; and we say that G is solvable by decomposition analysis if it can be solved by Fudos and Hoffmann's decomposition analysis, [5], which was reformulated in [10].

Theorem 1 *Let $G = (V, E)$ be a geometric constraint graph. The following assertions are equivalent:*

1. *G is full tree decomposable.*
2. *G is s-tree decomposable.*
3. *G is solvable by reduction analysis.*
4. *G is solvable by decomposition analysis.*

In what follows we will say that a constraint graph G that fulfills Theorem 1 is a *solvable* graph.

Theorem 1 states that the domain of constructive techniques is the class of solvable graphs. If $\mathcal{G}$ denotes the set of geometric constraint graphs, Figure 3 illustrates the relationship between different subsets of $\mathcal{G}$. The definitions of over-constrained, well-constrained and under-constrained graphs induce a partition on $\mathcal{G}$. By Lemma 1, tree decomposable graphs are a subset of not over-constrained graphs. The class of tree decomposable graphs lays between the set of well-constrained graphs and the set of under-constrained graphs. By Lemma 2, the solvable graphs (or full tree decomposable graphs) are the well-constrained graphs which are tree decomposable.

In this paper, we are concerned with the class of under-constrained graphs which are tree decomposable. We call the graphs in this class *completable* graphs. Sections 5 and 6 report on techniques to get solvable graphs from completable ones.

The proof of Theorem 1 is constructive, [10, 11, 12], in the sense that algorithms to compute tree decompositions can be derived from it. These algorithms compute a tree decomposition in polynomial time on the number of vertices of the graph. This time bound is justified because tree decompositions are computed in polynomial time from the output of the algorithms described in [4, 5, 15] which are polynomial.

From now on, the solutions to the completion problem will be described in terms of tree decompositions. The main advantage of this strategy is that the proposed solutions do not depend on any particular algorithm used to solve the geometric constraint problem.

3.2 The Set of Solvable Graphs

Let us show how the graphs in the domain of the constructive techniques considered above can be generated.

If (e, f) is an edge in $E(G)$, we will denote the graph $G = (V, E \setminus \{(e, f)\})$ by $G - (e, f)$.

First we recall the vertex amalgamation of graphs defined by Gross and Yellen, [7], illustrated in Figure 4.

Definition 6 *Let G and H be two disjoint graphs with $u_1, u_2 \in V(G)$ and $v_1, v_2 \in V(H)$. The* two vertex amalgamation $(G \cup H)/\{u_1 = v_1, u_2 = v_2\}$ *is the graph obtained from the union $G \cup H$ by merging the vertex u_1 of G and v_1 of H in a single vertex named u_1 and merging the vertex u_2 of G and v_2 of H in a single vertex named u_2.*

The following definition refers to Figure 5. The pair of vertices and the edge that define the amalgamation are depicted respectively as hollow circles and a thick line.

Definition 7 *Let G and H be two disjoint graphs with $u, v \in V(G)$ and $(e, f) \in E(H)$. We define the* substitution of (e, f) in H by G modulo

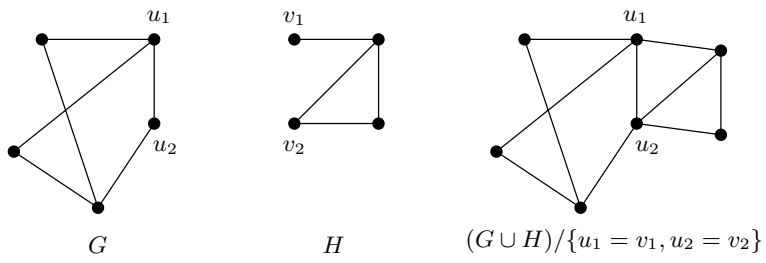

Figure 4: Two vertex amalgamation $(G \cup H)/\{u_1 = v_1, u_2 = v_2\}$.

$\{u, v\}$ as the new graph obtained by deleting the edge (e, f) from H and amalgamating this with G. That is $(G \cup (H - (e, f)))/\{u = e, v = f\}$.

Definition 8 *The set of graphs $\mathcal{T}$ is defined as follows:*

1. *If G has three nodes pairwise connected by edges, in short a* triangle*, then $G \in \mathcal{T}$.*
2. *Let $G, H \in \mathcal{T}$ be arbitrary graphs not necessarily distinct and let $(e, f) \in E(H)$ and $\{u, v\} \in V(G)$. Then the graph $(G \cup (H - (e, f)))/\{u = e, v = f\} \in \mathcal{T}$.*
3. *No other graph belongs to $\mathcal{T}$.*

Notice that the definition of $\mathcal{T}$ gives a way to actually build elements in it. Figure 6 depicts some graphs of $\mathcal{T}$ and their constructive definition.

To prove that $\mathcal{T}$ is the class of solvable graphs, we need the following results. For a proof see [19].

Lemma 3 *Let G be a graph in $\mathcal{T}$. Then G is solvable.*

Lemma 4 *Let $G = (V, E)$ be an arbitrary geometric constraint graph with $|V| \geq 3$. If G is solvable, then $G \in \mathcal{T}$.*

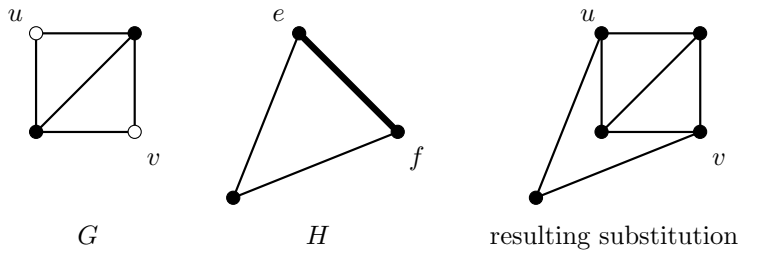

Figure 5: Substitution of (e, f) in H by G modulo $\{u, v\}$.

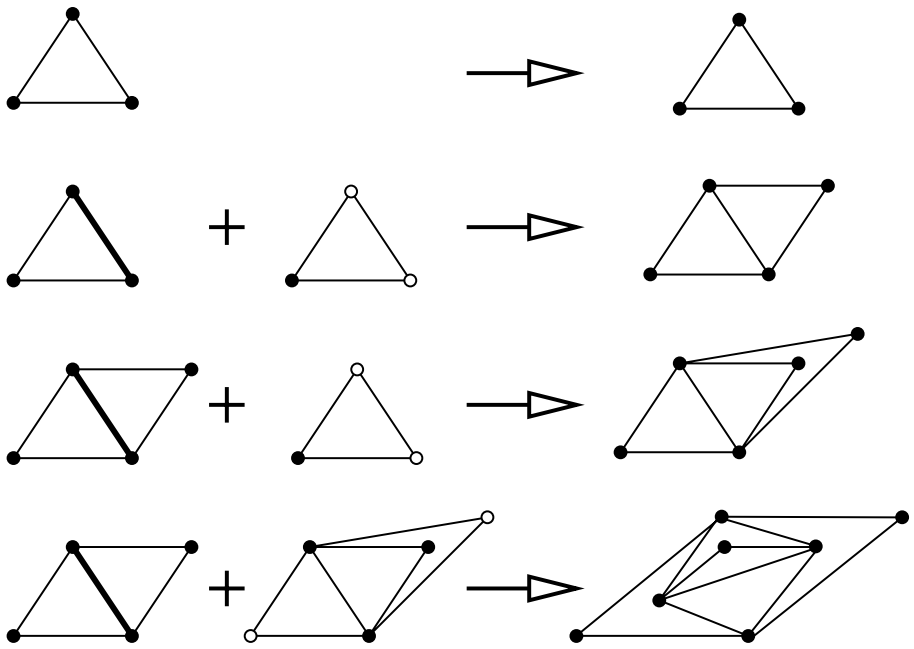

Figure 6: Some elements of $\mathcal{T}$.

Therefore,

Theorem 2 *The members of the set $\mathcal{T}$ are the class of solvable constraint graphs.*

4 Completability of Under-Constrained Graphs

When a geometric constraint problem is represented by a geometric constraint graph we would like to know the graph properties that fulfill the geometric constraint graphs corresponding to well-constrained geometric constraint problems. A survey on the main results on combinatorial characterization of well-constrained graphs can be found in [16]. Unfortunately, no necessary and sufficient combinatorial conditions are known to decide whether or not a geometric constraint graph whose vertices represent points and lines, and whose edges represent distance and angle constraints is well-constrained. Only necessary conditions are known for this kind of geometric constraint graphs. These necessary conditions are collected in the following definition, [5].

Definition 9 *Let $G = (V, E)$ be a geometric constraint graph.*

1. *G is* structurally over-constrained *if there is a vertex-induced subgraph with m vertices, $2 \leq m \leq |V|$, and more than $2m - 3$ edges.*

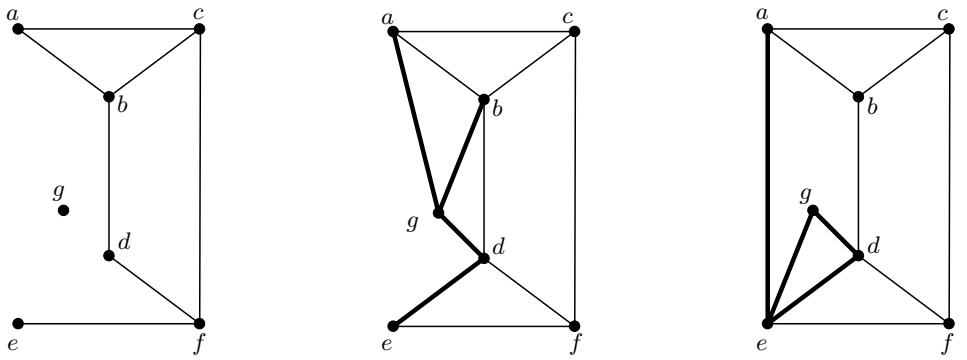

Figure 7: A graph and two different possible completions.

2. *G is* structurally under-constrained *if it is not over-constrained and* $|E| < 2|V| - 3$.

3. *G is* structurally well-constrained *if it is not over-constrained and* $|E| = 2|V| - 3$.

Notice that the structurally over-constraint case handles the situation where the same graph is structurally over-constrained in one part and structurally under-constrained in another at the same time. To make the reading easier, from now on we will drop the word structurally.

Consider an under-constrained tree decomposable geometric constraint graph. To effectively solve it, first we should transform the graph into a solvable one. This amounts to adding new constraints to the graph, that is, adding new edges to the constraint graph.

If $G = (V, E)$ is the geometric constraint graph, Definition 9 gives the number of constraints that must be added to G to transform it into a well-constrained graph. However, deciding which constraints should actually be added to the graph is not an straightforward matter, because it could result in either an over-constrained graph or a well-constrained graph that is not solvable.

We will use the term *completion* to refer to the process of adding new constraints (edges) to an under-constrained graph. The resulting graph will be named the *completed* graph or just *a completion*.

Figure 7 left shows an under-constrained graph. Notice that there is just one edge incident to node e and that node g has no incident edges. Completions shown in the middle and right of Figure 7 are solvable and well-constrained but not solvable graphs respectively.

We are interested in completed graphs that are solvable.

Definition 10 *Let $G = (V, E)$ be an under-constrained geometric constraint graph. We say that G is* completable *if there is a solvable graph $G' =$*

Algorithm FreeCompletion
1. Compute a tree decomposition T of G.
2. Compute the set of edges
$$E' = \{(a, b) \mid \{a, b\} \text{ is a leaf of } T \text{ and } (a, b) \notin E\}$$
3. Complete the graph G as $G' = (V, E \cup E')$.

Figure 8: Free completion algorithm of graph $G = (V, E)$.

$(V, E \cup E')$ *which is a completion of* G *with* $E' \neq \emptyset$.

In the following sections we present two techniques to complete completable graphs: free completion and conditional completion.

5 Free Completion

Here we present the first technique to complete completable graphs. We are given an under-constrained, completable graph $G = (V, E)$. The goal is to devise algorithms to automatically add new constraints to G to transform it into a solvable graph.

Among the different ways to complete the constraint graph G one can think of, we are interested first in one where the only additional requirement is that edges added to G must be taken from the set

$$E_F = \{(a, b) \mid a, b \in V \text{ and } (a, b) \notin E\}$$

We will call this a *free completion* of graph G.

The free completion problem has been previously addressed in [5]. There, a new method was developed to deal specifically with under-constrained problems. Our approach is based on tree decompositions and, since tree decompositions do not depend on any geometric constraint solving technique, is a general method.

The free completion algorithm is based on the following result. For the proof see [13].

Proposition 3 *Let* $G = (V, E)$ *be an under-constrained geometric constraint graph.* G *is completable if and only if there is a tree decomposition of* G.

The algorithm to carry out the free completion is given in Figure 8.

Consider the constraint graph given in Figure 7 left and assume that step 1 in algorithm `FreeCompletion` computes the tree decomposition T shown in Figure 9. The leaves of T whose nodes do not define an edge

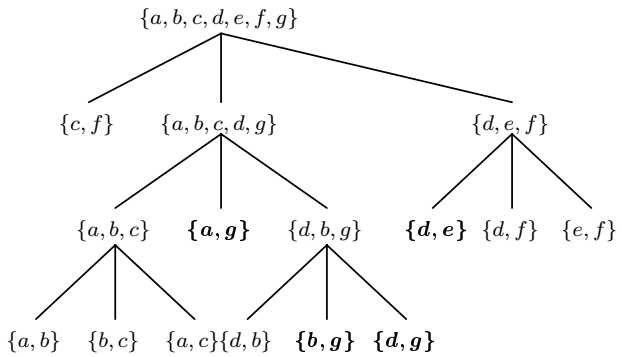

Figure 9: Tree decomposition of the graph in Figure 7 left.

in the graph depicted in Figure 7 left are (a, g), (d, e), (b, g) and (d, g). Therefore, the set of extra edges computed in step 2 is

$$E' = \{(a, g), (d, e), (b, g), (d, g)\}$$

Figure 7 in the middle shows the constraint graph yielded by the algorithm `FreeCompletion`.

Notice that, in general, a tree decomposition T of a constraint graph $G = (V, E)$ is not unique. Therefore, a free completion of a graph G is not necessarily unique.

6 Conditional Completion

Conditional completion is the second technique to complete geometric constraint graphs presented in this paper. The study of the conditional completion was originally motivated by the constraint schema reconciliation problem, [8].

Let $G = (V, E)$ be a completable geometric constraint graph. Let $\widehat{G} = (V, \widehat{E})$ be a geometric constraint graph whose edges $\widehat{E}$ define a set of additional constraints between geometric elements in V with $\widehat{E} \cap E = \emptyset$. Notice that if $\widehat{E} \cap E \neq \emptyset$, we always can consider as additional edges those in the set $\widehat{E}' = \widehat{E} \setminus E$.

Our interest is to build a solvable completion of G by adding edges drawn from $\widehat{E}$. We will refer to this completion as *conditional completion* of graph G from $\widehat{G}$. Formally, we define a conditional completion as follows.

Definition 11 *Let $G = (V, E)$ be a completable geometric constraint graph and let $\widehat{G} = (V, \widehat{E})$ be a geometric constraint graph. $G' = (V, E')$ is a* completion *of G from $\widehat{G}$ if G' is a completable completion of G and $E' = E \cup \widehat{E}'$ with $\widehat{E}' \neq \emptyset$ and $\widehat{E}' \subseteq \widehat{E}$.*

Figure 10: Geometric constraint graphs G (left) and $\widehat{G}$ (right).

We call the graph $\widehat{G}$ the *additional graph*. Figure 10 shows an under-constrained graph G (left) and an additional graph $\widehat{G}$ (right). Figure 11 shows two conditional completions of G from $\widehat{G}$. The graph on the left is completable and the graph on the right is solvable.

Among all the possible conditional completions of an under-constrained graph with respect to a given additional graph, we are interested in those that are maximal in the following sense.

Definition 12 *Let $G' = (V, E')$ be a conditional completion of G from $\widehat{G}$. We say that G' is a* maximal conditional completion *if for any other conditional completion $G'' = (V, E'')$ from $\widehat{G}$, the relation $|E''| \leq |E'|$ holds.*

A maximal conditional completion does not need to result in a solvable graph. Notice that it is not possible to build a solvable completion if the number of edges in $\widehat{E}$ is smaller than the number of edges required by Definition9 to transform G into a well-constrained graph. In these conditions, however, a solvable graph still can be built applying free completion to the graph yielded by the maximal conditional completion.

6.1 Maximal Conditional Completion as a Combinatorial Optimization Problem

Following Papadimitriou *et al.* [18], a *subset system* $S = (\mathcal{E}, \mathcal{S})$ is a finite set $\mathcal{E}$ together with a collection $\mathcal{S}$ of subsets of $\mathcal{E}$ closed under inclusion (that is, if $A \in \mathcal{S}$ and $A' \subseteq A$, then $A' \in \mathcal{S}$). The elements of $\mathcal{S}$ are called *independent*. The *combinatorial optimization problem associated with* a subset system $(\mathcal{E}, \mathcal{S})$ is the following: Given a *weight* $w(e) \geq 0$ for each $e \in \mathcal{E}$, find an independent subset that has the largest possible total weight.

Let $G = (V, E)$ be an under-constrained graph and let $\widehat{G} = (V, \widehat{E})$ be the additional graph. Assume that E and $\widehat{E}$ are disjoint, $E \cap \widehat{E} = \emptyset$. If we define

Figure 11: Conditional completion (left) and maximal conditional completion (right) of G from $\widehat{G}$ in Figure 10.

$\mathcal{E} = E \cup \widehat{E}$ and $\mathcal{S}$ as the subsets of $\mathcal{E}$ such that are tree decomposable, then S is a subset system. We say that a set of edges $E' \subseteq \mathcal{E}$ is tree decomposable if the graph $G' = (V, E')$ is tree decomposable. Notice that by Proposition 2, $\mathcal{S}$ is closed under inclusion.

We define the weight function $w(e)$ over the set of edges $\mathcal{E} = E \cup \widehat{E}$ as

$$w(e) = \begin{cases} 1 & \text{if } e \in \widehat{E} \\ 2 & \text{if } e \in E \end{cases}$$

In these conditions, the problem of computing a maximal conditional completion of G from $\widehat{G}$ is a combinatorial optimization problem associated with the subset system S and the weight function w.

6.2 The Greedy Algorithm

The greedy algorithm is the simplest algorithm to solve a combinatorial optimization problem associated with a subset system. In this section we first describe the greedy algorithm for subset systems. Then we apply it to the maximal conditional completion problem.

Figure 12 shows the greedy algorithm on the subset system $S = (\mathcal{E}, \mathcal{S})$. The algorithm computes I, an independent set in S which we want to have the largest total weight. We recall from [18] that it is not necessary to compute explicitly S to answer the question $I \cup \{e\} \in \mathcal{S}$ in line 2.3 of the algorithm. It suffices to have an algorithm to decide whether the set $I \cup \{e\}$ is in S.

Since we have formalized the maximal conditional completion of a graph G from an additional graph $\widehat{G}$ as a combinatorial optimization problem, we can use the greedy algorithm in Figure 12 as is, without any change. Notice that, for the maximal conditional completion problem, the elements of $\mathcal{S}$ are the subsets of $\mathcal{E}$ such that they are tree decomposable. Therefore,

Algorithm GreedyOptimize
1. $I := \emptyset$
2. while $\mathcal{E} \neq \emptyset$ do
2.1 $e :=$ a maximum weight member of $\mathcal{E}$
2.2 $\mathcal{E} := \mathcal{E} \setminus \{e\}$
2.3 if $I \cup \{e\} \in \mathcal{S}$ then $I := I \cup \{e\}$

Figure 12: The greedy algorithm for subset systems.

deciding whether $I \cup \{e\}$ is in $\mathcal{S}$ is equivalent to decide whether there is a tree decomposition of the set of edges $I \cup \{e\}$.

Unfortunately, the greedy algorithm does not solve the maximal conditional completion problem. Let us show a counterexample. Let G be the completable graph in Figure 10 left and let $\widehat{G}$ be the geometric constraint graph in Figure 10 right. We define the subset system S and the weight function w as in Section 6.1.

The greedy algorithm in Figure 12 first selects the edges in G because their weights are larger than the weights of edges in $\widehat{G}$. Hence

$$I = \{(a,b),(a,c),(b,c),(b,d),(c,f),(d,f),(e,f)\}$$

After that, the edges in $\widehat{G}$ are considered. Because all the edges in $\widehat{G}$ have the same weight, they can be chosen at random. Assume that the sequence of edges chosen is (a,e), (b,g) and (d,g). Then, no more edges can be added to I and the independent set returned by the greedy algorithm is

$$I = \{(a,b),(a,c),(b,c),(b,d),(c,f),(d,f),(e,f),(a,e),(b,g),(d,g)\}$$

See Figure 11 left.

Now assume that the sequence of chosen edges is (e,d), (e,g), (d,g) and (g,b). At this point, no more edges can be added to I and the independent set returned by the greedy algorithm is

$$I = \{(a,b),(a,c),(b,c),(b,d),(c,f),(d,f),(e,f),(e,d),(e,g),(d,g),(g,b)\}$$

See Figure 11 right. Clearly, this independent set includes more edges than the one computed before, that is, it has larger total weight. Therefore, the greedy algorithm does not necessarily yields a maximal conditional completion.

7 Well-Constrained Conditional Completion

So far, we have focused our interest in completing a geometric constraint graph in such a way that the completed graph was solvable by a constructive

technique. Here we relax this requirement and we only expect the completion to be well-constrained; that is, we address Problem 1.

We proceed as in Section 6. First, we define the well-constrained conditional completion. The main difference with respect to Definition 11 is that here the geometric constraint graph is required to be under-constrained instead of completable.

Definition 13 *Let $G = (V, E)$ be an under-constrained geometric constraint graph and let $\widehat{G} = (V, \widehat{E})$ be a geometric constraint graph. $G' = (V, E')$ is a* well-constrained conditional completion *of G from $\widehat{G}$ if G' is a well-constrained completion of G and $E' = E \cup \widehat{E}'$ with $\widehat{E}' \neq \emptyset$ and $\widehat{E}' \subseteq \widehat{E}$.*

Next, we define the maximal well-constrained conditional completion. Notice the analogy with Definition 12.

Definition 14 *Let $G' = (V, E')$ be a well-constrained conditional completion of G from $\widehat{G}$. We say that G' is a* maximal well-constrained conditional completion *if for any other well-constrained conditional completion $G'' = (V, E'')$ from $\widehat{G}$, the relation $|E''| \leq |E'|$ holds.*

Now, we reformulate the maximal well-constrained conditional completion problem as a combinatorial optimization problem associated with a subset system. Let $G = (V, E)$ be an under-constrained graph and let $\widehat{G} = (V, \widehat{E})$ be the additional graph. Assume that E and $\widehat{E}$ are disjoint, $E \cap \widehat{E} = \emptyset$. If we define $\mathcal{E} = E \cup \widehat{E}$ and $\mathcal{S}$ as the subsets E' of $\mathcal{E}$ for which the graph $G' = (V(E'), E')$ is not over-constrained, then S is a subset system. We define the weight function $w(e)$ over the set of edges $\mathcal{E}$ as in Section 6.1. $V(E')$ denotes the set of the endpoints of the edges in E'.

Two well established results are needed to show that the greedy algorithm solves the maximal well-constrained conditional completion problem. These results come from matroids theory and combinatorial rigidity theory, respectively.

Matroids are subset systems which fulfill some additional properties. See [17] for an in depth study of matroids theory. A well known result in combinatorial optimization is that the greedy algorithm solves any combinatorial optimization problem associated with a subset system which is a matroid, [18].

Geometric constraint graphs whose vertices are points in the plane and whose edges are distance constraints has been extensively studied in the context of combinatorial rigidity theory. The result we are interested in is the following, [6].

Theorem 3 *Let $G = (V, E)$ be a geometric constraint graph whose vertices are points in the plane and whose edges are distance constraints. Let $\mathcal{E} = E$ and let $\mathcal{S}$ denote the collection of subsets of $\mathcal{E}$ such that they are not over-constrained. Then the subset system $S = (\mathcal{E}, \mathcal{S})$ is a matroid.*

Combining these results, we can conclude that the greedy algorithm solves the maximal well-constrained conditional completion for graphs whose vertices are points in the plane and whose edges are distance constraints. This result has interesting practical implications. For example, any complete geometric constraint solving technique can benefit from the use of the greedy algorithm to solve the maximal well-constrained conditional completion problem.

8 Summary

In this paper we offer tools to deal with underconstrained geometric constraint graphs. We have given two different techniques: Free completion and conditional completion. In the free completion extra constraints are computed automatically by the algorithm. In conditional completion extra constraints are drawn from a given set of constraints defined on the initial set of geometric elements. In both cases, the resulting completed set of constraints is not over-constrained. If the given set of constraints is solvable, both free and conditional completions generate a completed graph which is also solvable. Maximal conditional completion does not necessarily yield solvable completions. However, further applying a free completion would result in a well-constrained constructively solvable constraint graph.

Acknowledgements

This research has been partially supported by the Ministerio de Ciencia y Tecnologia under the project TIC2001-2099-C03-01 and by the FEDER Agency.

References

[1] K.J. de Kraker, M. Dohmen, and W.F. Bronsvoort. Multiple-way feature conversion to support concurrent engineering. In M. Pratt, R.D. Siriram, and M.J. Wozny, editors, *Product Modeling for Computer Integrated Design and Manufacture*, pages 203–212. Chapman and Hall, London, UK, 1997.

[2] C. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Purdue University, Department of Computer Sciences, December 1998.

[3] S. Even. *Graph Algorithms*. Computer Science Press, Rockville, MD, 1979.

[4] I. Fudos and C.M. Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry & Applications*, 6(4):405–420, 1996.

[5] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[6] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, 1993.

[7] J. Gross and J. Yellen. *Graph Theory and its Applications*. Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, USA, 1999.

[8] C.M. Hoffmann and R. Joan-Arinyo. Distribuited maintenance of multiple product views. *Computer-Aided Design*, 32(7):421–431, June 2000.

[9] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decompostion Plans for Geometric Constraint Systems, Part I: Performance Measurements for CAD. *Journal of Symbolic Computation*, 31:367–408, 2001.

[10] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. On the domain of constructive geometric constraint solving techniques. In R. Ďurikovič and S. Czanner, editors, *Spring Conference on Computer Graphics*, pages 49–54. IEEE Computer Society, 2001.

[11] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Revisiting decomposition analysis of geometric constraint graphs. In K. Lee and M. Patrikalakis, editors, *Seventh ACM Symposium on Solid Modeling and Applications*, pages 105–115, Saarbrüken, Germany, 2002. ACM Press.

[12] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Revisiting decomposition analysis of geometric constraint graphs. *Computer-Aided Design*, 2003. to appear.

[13] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana-Pastó. Transforming an under-constrained geometric constraint problem into a well-constrained one. Research LSI-02-69-R, Dept. Llenguatges i Sistemes Informàtics, UPC, Barcelona, November 2002.

[14] G.L. Miller and V. Ramachandran. A new triconnectivity algorithm and its parallelization. *Combinatorica*, 12:53–76, 1992.

[15] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp Foundations of Solid Modeling*, pages 397–407, Austin, TX, 1991.

[16] J.C. Owen. Constraints on simple geometry in two and three dimensions. *International Journal of Computational Geometry & Applications*, 6(4):421–434, 1996.

[17] J.G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[18] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms And Complexity*. Dover Publications, May 1998.

[19] S. Vila. *Contribution to Geometric Constraint Solving in Concurrent Engineering*. PhD thesis, Dept. LSI, Universitat Politècnica de Catalunya, Barcelona, Catalonia, 2003. (In preparation).