Geometric Constraint Solving Based on Connectivity of Graph

Gui-Fang Zhang[1] and Xiao-Shan Gao[2]

[1]Tsinghua University, zhanggf@cg.cs.tsinghua.edu
[2] Chinese Academy of Sciences, xgao@mmrc.iss.ac.cn

ABSTRACT

We propose a geometric constraint solving method based on connectivity analysis in graph theory, which can be used to decompose a structurally well-constrained problem in 2D into some smaller ones if possible. We also show how to merge two rigid bodies if they share two or three geometric primitives in a bi-connected or tri-connected graph respectively.

Keywords: Geometric constraint solving, parametric CAD, k-connected graph, separating k-tuple, D-tree decomposition.

1. INTRODUCTION

Geometric constraint solving (GCS) is one of the key techniques in parametric CAD, which allows the user to make modifications to existing designs by changing parametric values. There are four major approaches to geometric constraint solving: the numerical approach [13,22,28], the symbolic computation approach [7,20], the rule-based approach [2,6,21,30] and the graph-based approach [3,4,15,17,24,25,27].

This paper will focus on using graph algorithms to decompose a large constraint problem into smaller ones. In [29], Owen proposed a GCS method based on the tri-connected decomposition of graphs, which may be used to reduce a class of constraint problems into constraint problems consisting of three geometric primitives. In [5,14], Hoffmann et al proposed a method based on cluster formation to solve 2D and 3D constraint problems. In [17], Joan-Arinyo et al proposed an algorithm to decompose a 2D constraint problem into an s-tree. This method is equivalent to Owen's and Hoffmann's methods, but is conceptually simpler.

The above approaches use triangles as basic patterns to solve geometric constraint problems. In [24], Latham and Middleditch proposed an algorithm used to decompose a constraint problem into what we called *general construction sequence* [10,11]. A similar method based on maximal matching of bipartite graphs was proposed in [23]. In [15], Hoffmann et al gave an algorithm to find rigid bodies in a constraint problem. From this, several general approaches to GCS are proposed. In [16], Jermann et al also gave a general approach to GCS based on the method in [15]. In [11], a c-tree decomposition method is proposed to solve both 2D and 3D constraint problems. The method proposed in [15] and the c-tree method [11] can be used to find a decomposition with the smallest controlling problem in certain sense.

In this paper, a method based on connectivity analysis from graph theory is proposed to decompose a constraint graph into a decomposition tree (abbr. D-tree). This method is a natural generalization of the methods in [17,29] which are based on tri-connectivity analysis of the constrained graph, and can solve problems that can be reduced to triangles. On the other hand, our method can be used to deal with general problems. In Section 2, we introduce the concept of connected graph. In Section 3, the method to split constraint graph is proposed. In Section 4, an algorithm to generate the D-tree is proposed. In Section 5, a method of merging bi-connected and tri-connected constraint graphs is proposed, followed by some conclusions.

2. PRELIMINARIES ON CONSTRAINT GRAPHS

The geometric primitives considered in this paper are points and lines in 2D. The geometric constraints considered in this paper include distance constraints between point/point, point/line and angular constraints between line/line in 2D.

We use a constraint graph to represent a constraint problem. The vertices of the graph represent the geometric primitives and the edges represent the constraints.

Let $G=(V,E,□)$ (or $G=(V(G),E(G),□)$ be a geometric constraint graph. For any v in V, $□(v)$ is the weight of vertex v, i.e the number of independent parameters used to determine the vertex, and $□(V)= \Sigma_{v} \in_{V} □(v)$. For instance, the weight of every geometric primitive we consider here in 2D is 2. For any e in E, $□(e)$ presents the weight of edge e, i.e the number of scalar equations to represent the constraint, and $□(E)= \Sigma e \in E$ $□(e)$. For instance, in 2D the weight of the distance constraint between two points is 1 if the distance is not zero, otherwise it is 2.

Definition 1 Let $G=(V,E,□)$ be a geometric constraint graph.

(i) G is structurally over-constrained if there is an induced subgraph H of G satisfying $□(E(H))>□(V(H))$-3.

(ii) G is structurally under-constrained if it is not structurally over-constrained and $0<□(E)<□(V)$-3.

(iii) G is structurally well-constrained if it is neither structurally under-constrained nor structurally over-constrained.

A rigid body is a set of geometric primitives whose position and orientation relative to each other is known [5]. A structurally well-constrained graph defines a rigid body in most cases. But in some special cases the constraint problem represented by a structurally well-constrained graph may have no solutions or an infinite number of solutions. It is obvious that a structurally well-constrained graph is always connected.

Definition 2 An undirected graph $G=(V,E)$ is called *connected* if for every two nodes $x, y \in V$, there exists a path of edges from E joining x and y. A graph is called *disconnected* if it is not connected. A graph is called *k-connected* $(k\geq 2)$ if there does not exist a set of k-1 or fewer nodes $V□\subset V$ such that the removal of all nodes of V' and their incident edges from G results in a disconnected graph.

Definition 3 Two vertices x and y of graph G are said to be *k-connected* if k is the largest integer such that there exist k vertex-disjoint paths from x to y in G. The connectivity of x and y is denoted by $\kappa(x, y)$, which is the maximal number of vertex disjoint paths from x to y in G.

Theorem 4 (**Theorem of Whitney**) A graph $G=(V,E)$ is k-connected if and only if $\kappa (x, y)\geq k$ for any two vertices x and y of G, that is,

$$\kappa (G)=\min \{\kappa (x, y): x, y \in V\}.$$

Definition 2, Definition 3 and Theorem 4 are available in [1]. The complexity of the algorithm to calculate the connectivity of a connected graph $G=(V, E)$ is $O(|V|^{\frac{1}{2}}|E|^2)$ [1].

Theorem 5 Let G be a structurally well-constrained graph. We have $\kappa(G)\leq 3$ in 2D.

Proof: For a graph $G=(V,E)$, it is known that $\kappa(G)\leq \frac{2|E|}{|V|}$ from [26]. Then for a structurally well-constrained constraint graph G, we can obtain the bound of κ (G) explicitly. Because the constraint problem is in 2D, $|E|=2|V|$-3 and $\frac{2|E|}{|V|} = \frac{2(2|V|-3)}{|V|}<4$. Thus $\kappa (G) \leq 3$.

Let $G=(V,E)$ be a connected undirected graph. A vertex $v \in V$ is a separating vertex for G, if the subgraph induced by V-$\{v\}$ is not connected. G is bi-connected if it contains no separating vertex.

A pair of vertices v_1, $v_2 \in V$ is a *separating pair* for G if the subgraph induced by V-$\{v_1,v_2\}$ is not connected. G is tri-connected if it contains no separating vertices and pairs [12].

A triplet $\{v_1,v_2,v_3\}$ of distinct vertices in V is a *separating triplet* of a tri-connected graph if the subgraph induced by V-$\{v_1,v_2,v_3\}$ is not connected. G is 4-connected if it contains no separating vertices, pairs and triplets [19].

A tuple $\{v_1,v_2,\ldots,v_k\}$ of distinct vertices in V is a *separating k-tuple* of a k-connected graph if the subgraph induced by V-$\{v_1,v_2,\ldots,v_k\}$ is not connected. G is (k+1)-connected if it contains no separating i-tuple $\{v_1,v_2, \ldots,v_i\}$ $(i\leq k)$.

3.DECOMPOEITION OF K-CONNECTED GRAPH

In this section, let $G=(V,E,□)$ be a geometric constraint graph. Assuming that G is k-connected and $V_s=\{v_1,v_2,\ldots,v_k\}$ is a separating k-tuple of G. The subgraph of G induced by V_s is $G_s=(V_s, E_s, □)$ and G_s is called a *cut graph*. A k-connected graph can be split into separating components $C^1,C^2,\ldots,C^n$ by splitting it at the separating k-tuple V_s. Assuming that $1\leq m\leq n$ and let

$$C_1=\bigcup_{i=1}^{m} C^i,\ C_2=\bigcup_{i=m+1}^{n} C^i,$$

such that $|V(C_1)| \geq 1$ and $|V(C_2)| \geq 1$. Thus we refer to the graphs

$$G^1=(V(C_1) \cup V_s,\ E(C_1) \cup \{(v_1,v_2)|v_1 \in V(C_1),v_2 \in V_s\})$$

$$G^2=(V(C_2) \cup V_s,\ E(C_2) \cup \{(v_1,v_2)|v_1 \in V(C_2),v_2 \in V_s\})$$

as the *separating graphs* of G. The graphs $G_1=(V(G^1),E(G^1) \cup E_s))$ and $G_2=(V(G^2),E(G^2) \cup E_s))$ are called *split graphs* of G. The relation of split graphs and the cut graph is shown in Fig.1

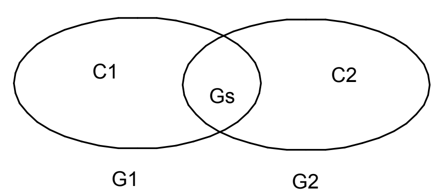

Fig. 1. The relation of split graphs and the cut graph

Joan-Arinyo et al defined a deficit function *deficit*=$2|V|-|E|-3$ to compute the difference between the number of weight of edges for a constraint graph $G=(V,E)$ to be structurally well-constrained and its actual number of weight $|E|$ in 2D [17]. Here we generalize the deficit function to more general cases.

Definition 6 Let $G=(V,E,\square)$ be a geometric constraint graph. We can define the deficit function associated with G as $deficit(G)=\square(V)-\square(E)-3$.

If G is a structurally well-constrained problem, $deficit(G)=0$. If G is not structurally over-constrained, $deficit(G) \geq 0$.

In 2D, if G is a structurally well-constrained graph and the primitives are points and lines, for every edge $e=(u,v)$ in $E(G)$, $\square(e)=1$.

Theorem 7 Let $G=(V,E,\square)$ be a geometric constraint graph in 2D. If G is structurally well-constrained, there is no separating vertex in G.

Proof: Assuming that there is a separating vertex v, such that we can split graph G into two split graphs G_1 and G_2 by the vertex.

Because G is structurally well-constrained, $deficit(G)=0$, $deficit(G_1) \geq 0$ and $deficit(G_2) \geq 0$.

Because $\square(V(G))= \square(V(G_1))+ \square(V(G_2))- \square(v)$

and $\square(E(G))= \square(E(G_1))+ \square(E(G_2))$,

we have

$\square(V(G_1))+ \square(V(G_2))- \square(v)-(\square(E(G_1))+ \square(E(G_2)))-3=0$,

i.e. $deficit(G_1)+ deficit(G_2)+(3-\square(v))=0$.

Because $deficit(G_1) \geq 0$ and $deficit(G_2) \geq 0$, it's obvious that $3-\square(v)\leq 0$. But for every primitive v we consider here $\square(v)=2$, so $3-\square(v)>0$ and there is no separating vertex in a structurally well-constrained graph.

Theorem 8 Let $G_s=(V_s,E_s,\square)$ be the cut graph induced by a separating k-tuple ($k\geq 2$) of a k-connected structurally well-constrained graph $G=(V,E,\square)$ and the split graphs $G_1=(V(G_1),E(G_1),\square)$ and $G_2=(V(G_2),E(G_2),\square)$. We have

$$deficit(G_1)+ deficit(G_2)= deficit(G_s).$$

Proof: Since graph G is structurally well-constrained, $deficit(G_1) \geq 0$, $deficit(G_2) \geq 0$ and $deficit(G_s) \geq 0$.

So $\square(V_s)-\square(E_s)-3\geq 0$, $\square(V(G_1))-\square(E(G_1))- 3\geq 0$

and $\square(V(G_2))-\square(E(G_2))- 3\geq 0$.

Because $\square(E(G_1))+\square(E(G_2))-\square(E_s)=\square(V)- 3$,

and $\square(V) -3=\square(V(G_1))+\square(V(G_2)) -\square(V_s)$.

Thus $(\square(V(G_1)) -\square(E(G_1))- 3)+\square(V(G_2)) -\square(E(G_2))- 3$
$=\square(V_s)-\square(E_s)- 3$.

So $deficit(G_1)+ deficit(G_2)= deficit(G_s)$.

Corollary 9 Let $G=(V,E,\square)$ be a structurally well-constrained k-connected graph, $G_s=(V_s,E_s,\square)$ the cut graph of G induced by a separating k-tuple $V_s=\{v_1,v_2,\ldots,v_k\}$, G_1 and G_2 the split graphs of graph G. G_1 and G_2 are structurally well-constrained if and only if G_s is structurally well-constrained.

Proof: Because G_1 and G_2 are not over-constrained, $deficit(G_1) \geq 0$ and $deficit(G_2) \geq 0$. Then $deficit(G_s)=0$ if and only if $deficit(G_1)=0$ and $deficit(G_2)=0$, according to Theorem 8.

Corollary 10 Let $G=(V,E,\square)$ be a structurally well-constrained k-connected graph, $G_s=(V_s,E_s,\square)$ the cut graph of G induced by a separating k-tuple $V_s=\{v_1,v_2,\ldots,v_k\}$, G_1 and G_2 the split graphs of graph G. If $deficit(G_s)=1$, only one of the split graph is structurally well-constrained.

Proof: Because G_1 and G_2 are not over-constrained, $deficit(G_1) \geq 0$ and $deficit(G_2) \geq 0$. $deficit(G_s)=1$ if and only if $deficit(G_1)=1$ and $deficit(G_2)=0$, or $deficit(G_1)=0$ and $deficit(G_2)=1$ according to Theorem 8.

Corollary 11 Let G_s be the graph induced by a separating k-tuple of a k-connected structurally well-constrained graph G, G_1 and G_2 the split graphs. If G_s is not structurally well-constrained and G_1 is structurally well-constrained, then

$$deficit(G_s)= deficit(G_2).$$

Proof: This is a direct consequence of Theorem 8.

Corollary 12 For a structurally well-constrained bi-connected constraint graph, at least one of the split graphs is a structurally well-constrained graph in 2D.

Proof: Let $\{a,b\}$ be the separating pair of a structurally well-constraint graph $G=(V,E,\square)$, thus $deficit(G_s)$ is 0 or 1. Then the conclusion is obvious according to the proofs of corollary 9 and corollary 10.

This conclusion is the same as that in [17,29], based on which Owen gave an efficient algorithm and Joan-Arinyo et al gave an improved algorithm in 2D.

The examples shown in Fig.2 and Fig. 3 are the case of $deficit(G_s) = 0$ and $deficit(G_s) =1$ respectively.

In the following figures, diagrams (a), (b) and (c) represent original constraint graph, the split graphs G_1 and G_2 respectively.

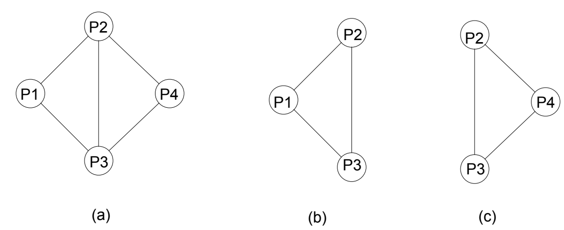

Fig. 2. Separating pair is p_2, p_3, and $deficit(G_s) = 0$.

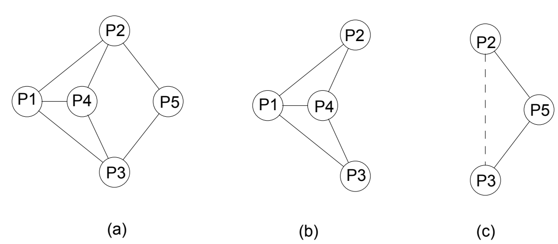

Fig. 3. Separating pair is p_2, p_3, and $deficit(G_s) =1$.

In general, a structurally well-constrained graph can be decomposed by three ways based on connectivity analysis.

(i) *deficit*(G_s)=0.

Now the split graphs G_1 and G_2 are structurally well-constrained according to Corollary 9. We can solve them separately, and merge them to obtain the solutions to the initial problem. Here the merging step is easy for G_1 and G_2 who share the same cut graph G_s. Fig. 4 is an example of this case. The graph is split into two structurally well-constrained graphs, which are solved explicitly in [10].

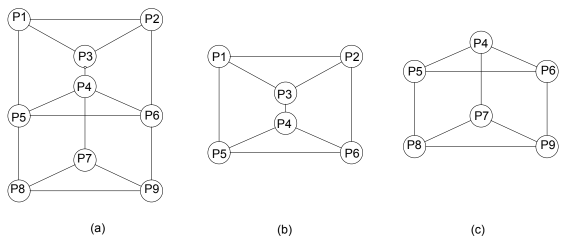

Fig.4. Separating triplet is p_4, p_5, p_6 and *deficit*(G_s)=0

(ii) *deficit*(G_s)=1.

Now one of the split graphs is structurally well-constrained according to Corollary 10. Let G_1 be the well-constrained split graph. We solve G_1 first, and add one auxiliary constraint to G_s so that G_s become a structurally well-constrained problem denoted by G_s'. Since G_s is part of G_2 and *deficit*(G_s)=*deficit*(G_2) by Corollary 10, when replacing G_s with G_s' and adding these auxiliary constraints to G_2, G_2 becomes a structurally well-constrained problem denoted by G_2', and can be solved separately. After getting the solution of G_1 and G_2', we can merge them to obtain the solution to the initial problem. G_2' is called the modified split graph of G with G_1. Fig. 5 is an example of this case. In diagram(c), an auxiliary constraint between p_2 and p_3 is added. This problem is split into two ruler and compass constructible triangles and a basic merging pattern solved analytically in [10].

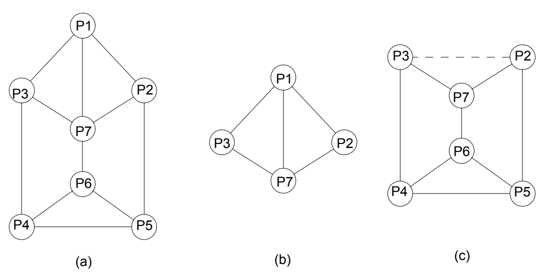

Fig.5. Separating triplet is p_2, p_3, p_7 and *deficit*(G_s) = 1

(iii) *deficit*(G_s)>1.

In a structurally well constrained problem G, if *deficit*(G_s)>1 then the cut graph G_s contains three geometric primitives. By Theorem 5, k(G)≤*3*, which means G_s contains at most three geometric primitives. According to Corollary 12, if |*V*(G_s)|=2, we have either *deficit*(G_s)=1 or *deficit*(G_s)=0. Thus, G_s can only contain three elements.

In Theorem 8, *deficit*(G_1)+*deficit*(G_2)=*deficit*(G_s). So *deficit*(G_1) can be 0,1,…,*deficit*(G_s), and *deficit*(G_2) will be *deficit*(G_s), *deficit*(G_s)-1,…,0 correspondingly. For a structurally well-constrained graph in 2D, we know *deficit*(G_s)≤3.

Fig.6 is an example of the case that *deficit*(G_2)=*deficit*(G_s)=2. Fig. 7 is an example of the case that *deficit*(G_2)=*deficit*(G_s)=3. Fig.8 is an example of the case that *deficit*(G_s)=2 and *deficit*(G_1)=*deficit*(G_2)=1. Fig.9 is an example of the case that *deficit*(G_s)=3 and *deficit*(G_1)=*1*, *deficit*(G2)=2.

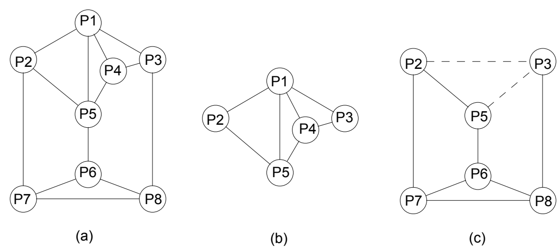

Fig.6. Separating triplet is p_2, p_3, p_5, *deficit*(G_s)= 2.

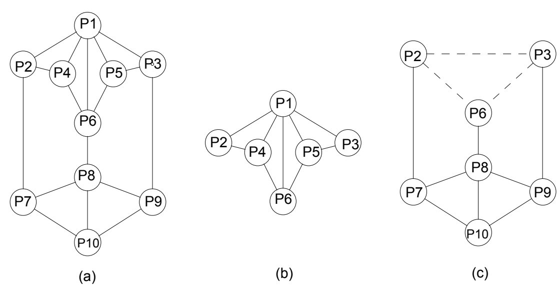

Fig.7. Separating triplet is p_2, p_3, p_6, *deficit*(G_s) =3.

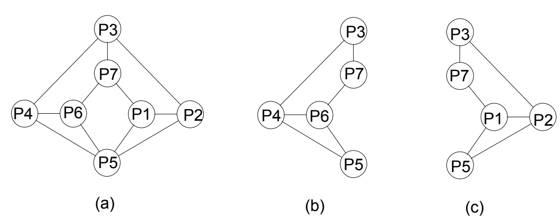

Fig.8. Separating triplet is p_3, p_5, p_7, *deficit*(G_s)= 2.

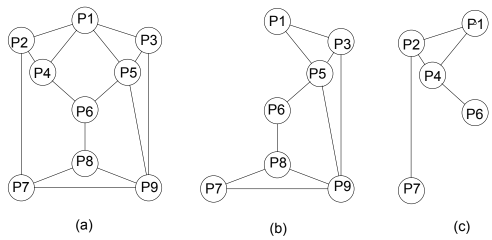

Fig.9. Separating triplet is p_1, p_6, p_7, *deficit(G_s)*=3.

If either G_1 or G_2 is structurally well-constrained, we may solve the problem as case (ii) above. For example, in diagram(c) of Fig. 6, two auxiliary constraints between p_2/p_3 and p_3/p_5 are added; in diagram(c) of Fig. 7, three auxiliary constraints between p_2/p_3, p_2/p_6 and p_3/p_6 are added.

If neither G_1 nor G_2 is structurally well-constrained, we can make the following choices:

(a) Select another separating k-tuple and re-decompose the constraint graph G. For example, to the problem in Fig.9, if the triplet is {p_1,p_5,p_9}, *deficit(G_s)*=1, the problem can be solved similar to (ii) as shown in Fig.10.

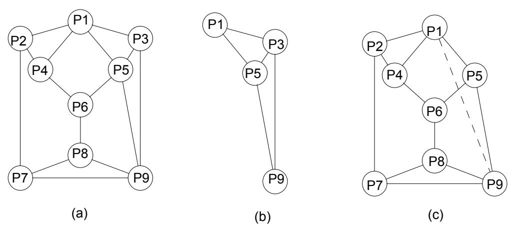

Fig.10. Separating triplet is p_1, p_5, p_9, *deficit(G_s)*=1.

(b) Computing the solution to G_1 or G_2 is feasible in certain cases. For example, to the constraint problem in Fig.8, the initial graph is split into two parts, which can be treated as two four-bar linkages. If we take p_3 and p_7 as the fixed points, p_4 and p_2 as the driving points respectively, the intersection of the coupler loci of the two four-bar linkages is the solution to p_5 [9]. We can also use LIMd method [8] to solve this problem too. After removing the distance constraint $|p_3p_4|=d$ between points p_3 and p_4, we can take points p_2 and p_3 as the fixed points and point p_7 as a driving point. Then we can get the locus of point p_7. The intersection of this locus and the circle whose center is point p_3 and radii is d is the solution of point p_7. If there is no geometric solution, we may use numerical techniques to solve the problem.

In the cases *deficit(G_s)*>0, we need to add auxiliary edges to make the cut graph G_s and the split graph G_2 structurally well-constrained. Latham et al presented a method to detect whether the constraint graph is structurally under-constrained and decide how to add constraints if the graph is structurally under-constrained [24]. Joan-Arinyo et al also proposed an algorithm used to get a well-constrained problem from an under-constrained problem [18]. But the type of the constraints added to G_s should be based on the shape of the split graph G_1 assuming that G_1 is a rigid body while G_2 is not.

4. A DCOMPOSTITION ALGORITHM

We will introduce a new decomposition tree, D-tree, which can be used to simplify a structurally well-constrained problem.

Definition 13 A D-tree for a structurally well-constrained k-connected graph $G=(V,E,\square)$ is a binary tree.

(i) The root of the tree is the graph G.

(ii) For each node N in the tree, its left child L is the split graph of N which is either a triangle or a structurally well-constrained tri-connected subgraph of N, and the right child R is the (modified) split graph of N with L which is either a triangle or a structurally well-constrained tri-connected graph.

(iii) Every leaf is either a triangle or a tri-connected structurally well-constrained graph that can not be split into smaller well-constrained graph further.

Algorithm 1 The input is a structurally well-constrained graph $G=(V,E)$. The output is a D-tree for G. Let $T=G$ as the initial value; S_k the set of separating k-tuples in $V(T)$.

S1 Calculate connectivity k of the structurally well-constrained connected graph T. If $k\geq|V(T)|$-2, the algorithm terminates; else $S_k \leftarrow \emptyset$, goto step ***S2***.

S2 (i) If k=2, find all the separating pairs with Hopcroft and Tarjon method [12]. Then add these separating pairs to S_k, goto step ***S3***.

(ii) If k=3, find all the separating triplets with Kanevsky and Ramachandran method [19]. Then add these separating pairs to S_k, goto step ***S3***.

S3 If $S_k \neq \emptyset$, taking a separating pair or triplet S in S_k, $S_k \leftarrow S_k$-{S}, goto ***S4***. Otherwise, the algorithm terminates.

S4 If the deficit function of the cut graph induced by S is 0, goto ***S5***; else goto ***S6***.

S5 Split T by the separating k-tuple S to generate the split graphs G_1 and G_2. If $k<|V(G_1)|$ and $k<|V(G_2)|$, let $L=G_1$ and $R=G_2$; Let $T=L$, operate the algorithm from ***S1*** recursively; Let $T=R$, operate the algorithm from ***S1*** recursively. Otherwise, goto ***S3***.

S6 Split T by the separating k-tuple S to generate the split graphs G_1 and G_2. If one of the split graph is structurally well-constrained, let it be G_1. If $k|<|V(G_1)|$ and $k|<|V(G_2)|$, $L=G_1$. Let $T=L$ and operate the algorithm from ***S1*** recursively. Let R be the modified split graph G_2^m, $T=R$, operate the algorithm recursively. Otherwise, goto ***S3***.

The complexity of step **S1** is $O(|V|^{\frac{1}{2}}|E|^2)$ [1]. When k=2, Hopcroft and Tarjan gave an algorithm for finding separating pairs in a bi-connected graph in $O(|V|+|E|)$ time [12]. When k=3 Kanevsky and Ramachandran gave an algorithm for finding all separating triplets in a tri-connected graph in $O(|V|^2)$ time[19].

Fig. 11 is a 2D constraint problem, which can be decomposed into a D-tree. The angular constraints are $ang(l_2,l_3)$, $ang(l_5,l_6)$ and $ang(l_6,l_7)$. The distance constraints are $dis(p_1p_2)$, $dis(p_2p_3)$, $dis(p_3p_4)$, $dis(p_1,p_4)$, $dis(p_5,p_6)$, $dis(p_6,p_7)$ and $dis(p_7,p_8)$. The incident constraints are *obvious*. The problem can be reduced into solve eleven ruler and compass constructible triangles and a basic merging pattern solved explicitly in [10]. Fig.12 is the D-tree of the problem, where dot lines represent the auxiliary constraints.

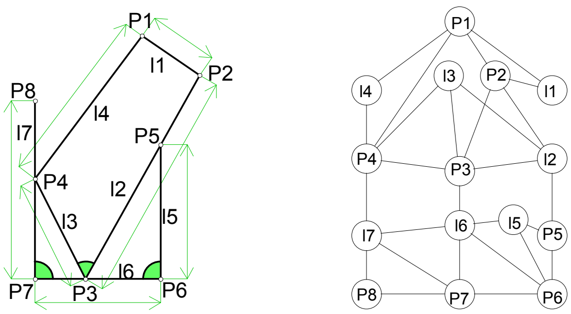

Fig. 11. A geometric constraint problem and its graph

5. MERGE BI-CONNECTED AND TRI-CONNECTED CONSTRAINT GRAPHS

After a D-tree is obtained for a structurally well-constrained geometric constraint problem, we can solve the problem as follows: Do a left to right depth first transversal of the D-tree and solve the constraint problem represented by each node as follows.

(i) If the current node N is a leaf in the tree then it is a structurally well-constrained problem that cannot be decomposed further. Solve N with numerical computation methods [13,22,28].

(ii) Let N be a node with left child L and right child R. This can be done in three steps.

(a) **Solve the left child *L***. L is a structurally well-constrained problem that can be solved recursively.

(b) **Solve the right child *R***. The values for the auxiliary constraints in the right child R can be obtained from L. Now, R is a structurally well-constrained problem that can be solved recursively.

(c) **Merge *L* and R to obtain *N***.

In this section, we will address the problem of merging two rigid bodies. This problem is generally simple because we assume that the two rigid bodies share many geometric primitives. In what below, we will give a detailed analysis of the merging process for bi-connected and tri-connected graphs.

5.1. Merge Bi-connected Constraint Graphs

Let the separating pair be $\{a, b\}$ in a structurally well-constrained bi-connected graph G. The split graphs of G are two rigid bodies R_1 and R_2. Now we show how to assemble R_1 and R_2.

The problem can be classified into the following three cases according to the types of a and b.

(i) The vertices a and b are two points. The relative position of R_1 and R_2 can be fixed. Thus $R_1 \bigcup R_2$ is a rigid body.

(ii) The vertices a and b are a point and a line. The relative position of R_1 and R_2 can be fixed. Thus $R_1 \bigcup R_2$ is a rigid body.

(iii) The vertices a and b are two lines. If a and b are parallel, the relative position of R_1 and R_2 can not be fixed because there is a translation degree of freedom between R_1 and R_2. Thus $R_1 \bigcup R_2$ is not a rigid body although structurally well-constrained. When a and b are not parallel, the relative position of R_1 and R_2 can be fixed. Thus $R_1 \bigcup R_2$ is a rigid body.

We thus proved the following result.

Theorem 14 Let $\{a, b\}$ be the separating pair in a structurally well-constrained bi-connected graph G in 2D, R_1 and R_2 the split subgraphs of G which are two rigid bodies. Then R is a rigid body if and only if $\{a, b\}$ is one of the following three forms: two points; a point and a line; two lines which are not parallel.

5.2. Merge Tri-connected Constraint Graphs

Let the separating triplet be $\{a, b, c\}$ in a structurally well-constrained tri-connected graph G. The split graphs of G are two rigid bodies R_1 and R_2. Now we will try to assemble two rigid bodies R_1 and R_2, i.e. merge the vertices a, b and c in R_1 and R_2.

The problem can be classified into the following four cases according to the types of vertices a, b and c.

(i) The vertices a, b and c are three points. The relative position of R_1 and R_2 can be fixed. Thus $R_1 \bigcup R_2$ is a rigid body.

(ii) The vertices a, b and c are three lines. If the three lines are parallel to each other, the relative position of R_1 and R_2 can not be fixed because there is a translation degree of freedom. Otherwise, the relative position of R_1 and R_2 can be fixed and $R_1 \bigcup R_2$ is a rigid body.

(iii) The vertices a, b and c are two points and one line. The relative position of R_1 and R_2 can be fixed and $R_1 \bigcup R_2$ is a rigid body.

(iv) The vertices a, b and c are two lines and a point. The relative position of R_1 and R_2 can be fixed and $R_1 \bigcup R_2$ is a rigid body.

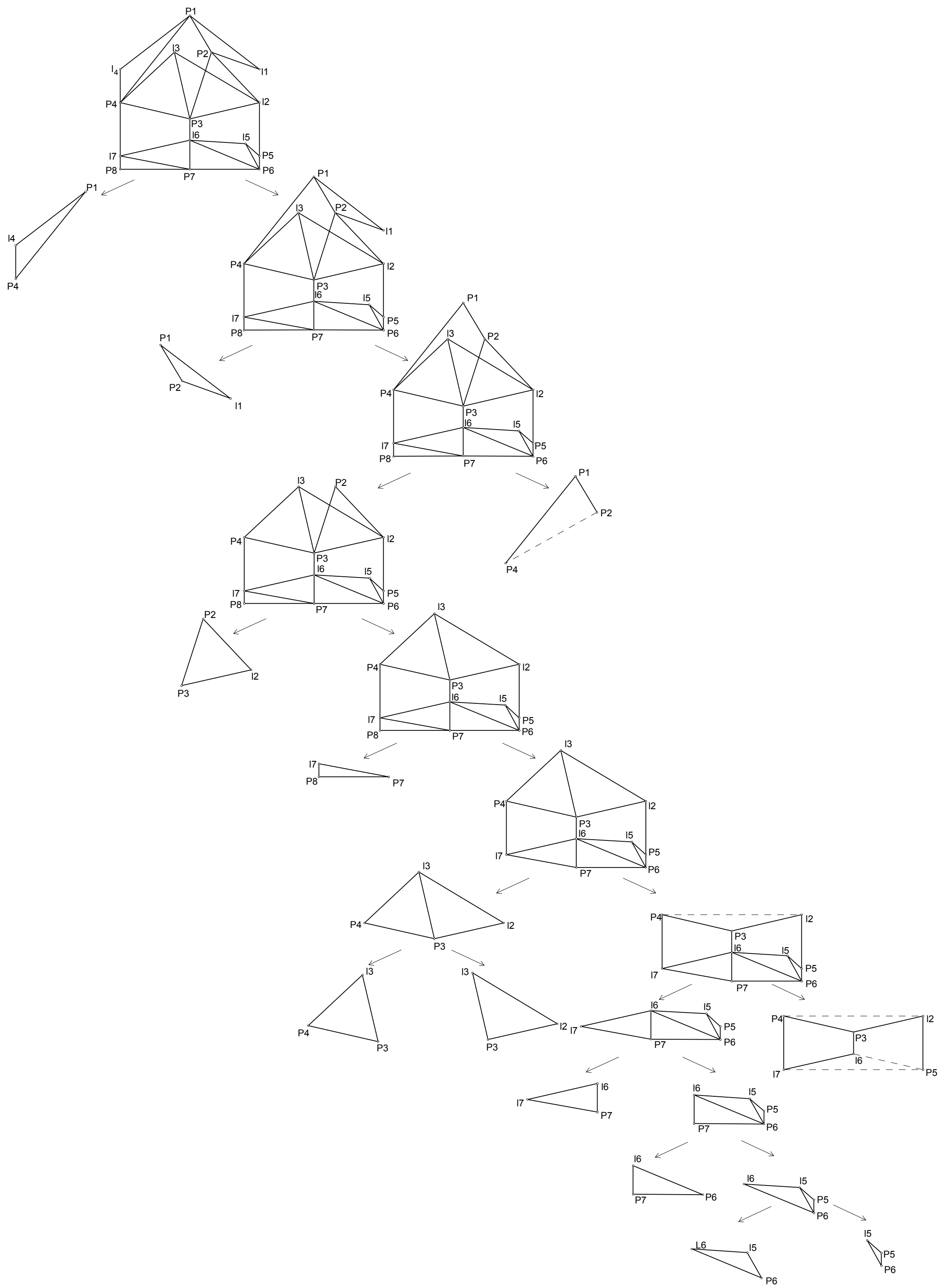

Fig.12 The D-tree of the geometric constraint problem in Fig. 11.

6. REFERENCES

[1] Becker, E., Brostein, M., Cohen, H., Eisenbud, D. and Gilman, R., Algorithms and Computation in Mathematics, Volume 5, 331-351.

[2] Brüderlin, B., Using Geometric Rewriting Rules for Solving Geometric Problems Symbolically, Theoretical Comp. Science, 116, 291-303, 1993.

[3] Chen, L.P., Wang, X.B., Chen, X.B. and Zhou, J., An Optimal Method of Bipartite Graph Matching for Underconstrained Geometry Solving, Chinese J. Computers, 23(5), 523-530, 2000.

[4] Durand C., and Hoffmann,C. M., A Systematic Framework for Solving Geometric Constraints Analytically, J. of Symbolic Computation, 30(5), 493-529, 2000.

[5] Fudos, I. and Hoffmann, C.M., A Graph-constructive Approach to Solving Systems of Geometric Constraints, ACM Transactions on Graphics, 16(2), 179-216, 1997.

[6] Gao, X.S. and Chou, S.C. Solving Geometric Constraint Systems I. A Global Propagation Approach, Comp.-Aided Des., 30(1), 47-54, 1998.

[7] Gao, X.S. and Chou, S.C., Solving Geometric Constraint Systems II. A Symbolic Approach and Decision of Rc-constructibility, Computer-Aided Design, 30(2), 115-122, 1998.

[8] Gao, X.S., Hoffmann, C.M. and Yang, W.Q., Solving Basic Geometric Constraint Configurations with Locus Intersection, Proc. ACM SM02, 95-104, ACM Press, New York, 2002.

[9] Gao, X.S, Jiang, K. and Zhu, C.C., Geometric Constraint Solving with Conics and Linkages, Computer Aided Design, Vol 34, 421-433, 2002.

[10] Gao, X.S. and Zhang, G.F., Classification and Solving of Merge Patterns in Geometric Constraint Solving, Proc. Shape Modeling and Applications, 89-90, IEEE press, 2003.

[11] Gao, X.S. and Zhang, G.F., Geometric Constraint Solving via C-tree Decomposition, Proc. ACM SM03, 45-55, ACM Press, New York, 2003.

[12] Hopcroft, J.E. and Tarjan, R.E. Dividing a Graph into Triconnected Components, SIAM J. Computing. 135-158, 1973.

[13] Ge, J.X., Chou, S.C. and Gao, X.S. Geometric Constraint Satisfaction Using Optimization Methods, Computer Aided Design, 31(14), 867-879, 2000.

[14] Hoffmann, C.M. and Vermeer, P.J., Geometric Constraint Solving in R^2 and R^3, in Computing in Euclidean Geometry, D. Z. Du and F. Huang (eds), World Scientific, Singapore, 266-298, 1995.

[15] Hoffmann, C.M., Lomonosov, A. and Sitharam, M., Finding Solvable Subsets of Constraint Graphs, LNCS 1330 :163-197, 1997.

[16] Jermann, C., Neveu, B. and Trombettoni, G., A New Structural Rigidity for Geometric Constraint Systems, Winkler. F(Ed.) ADG2002, LNAI 2930: 87-105, 2004.

[17] Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S. and Vilaplan-Pastö, J., Revisiting Decomposition Analysis of Geometric Constraint Graphs, Proc. ACM SM02, 105-115, ACM Press, NY, 2002.

[18] Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S. and Vilaplana-Pastö, J., Transforming an Under-constrained Geometric Constraint Problem into a Well-constrained One, Proc. ACM SM03, 33-44, ACM Press, New York, 2003.

[19] Kanevsky, A. and Ramachandran, V., Improved Algorothms for Graph Four-connectivity, Proc. 28th Ann. IEEE Symp. Foundations of Computer Science, Los Angeles, 252-259, 1987.

[20] Kondo, K., Algebraic Method for Manipulation of Dimensional Relationships in Geometric Models, Computer Aided Design, 24(3), 141-147, 1992.

[21] Kramer, G.A., Solving Geometric Constraints Systems: A Case Study in Kinematics, MIT Press, Cambridge Massachusetts, 1992.

[22] Lamure, H. and Michelucci, D., Solving Geometric Constraints By Homotopy, IEEE Trans on Visualization and Computer Graphics,2(1):28-34,1996.

[23] Lamure, H. and Michelucci, D., Qualitative Study of Geometric Constraints, in Geometric Constraint Solving and Applications, 234-258, Springer, Berlin, 1998.

[24] Latham, R.S. and Middleditch, A.E., Connectivity Analysis: a Tool for Processing Geometric Constraints, Computer Aided Design, 28(11), 917-928, 1994.

[25] Lee, J. Y. and Kim, K., Geometric Reasoning for Knowledge Based Design Using Graph Representation, Computer-Aided Design, 28(10), 831-841, 1996.

[26] Van Leeuwen, J., Handbook of Theoretical Computer Science(Volume A): Algorithms and Complexity, Elsevier Science Publishers B.V. 1990.

[27] Li, Y.T., Hu, S.M. and Sun, J.G., Hybrid Model of Geometric Constraint Satisfaction, Journal of Computer Research and Development, 37(10), 1233-1239, 2000.

[28] Lin, V.C., Gossard, D.C. and Light, R.A., Variational Geometry in Computer-Aided Design, Computer Graphics, 15(3), 171-177, 1981.

[29] Owen J., Algebraic Solution for Geometry from Dimensional Constraints, in ACM Symp., Found of Solid Modeling, ACM Press, NY, 397-407, 1991.

[30] Verroust, A., Schonek, F. and Roller, D., Rule-Oriented Method for Parameterized Computer-Aided Design, Comp.-Aided Des., 24(10),531-540, 1992.