

# A constructive approach to solving 3-D geometric constraint systems using dependence analysis

Yan-Tao Li<sup>\*</sup>, Shi-Min Hu, Jia-Guang Sun

*Department of Computer Science and Technology, Tsinghua University, Beijing, China*

Received 9 June 2000; revised 7 December 2000; accepted 9 December 2000

---

## Abstract

Solving geometric constraint systems in 3-D is much more complicated than that in 2-D because the number of variables is larger and some of the results valid in 2-D cannot be extended for 3-D. In this paper, we propose a new DOF-based graph constructive method to geometric constraint systems solving that can efficiently handle well-, over- and under-constrained systems based on the dependence analysis. The basic idea is that the solutions of some geometric elements depend on some others because of the constraints between them. If some geometric elements depend on each other, they must be solved together. In our approach, we first identify all structurally redundant constraints, then we add some constraints to well constrain the system. And we prove that the order of a constraint system after processing under-constrained cases is not more than that of the original system multiplied by 5. After that, we apply a recursive searching process to identify all the clusters, which is shown to be capable of getting the minimum order-reduction result of a well-constrained system. We also briefly describe the constraint evaluation phase and show the implementation results of our method. © 2001 Elsevier Science Ltd. All rights reserved.

*Keywords:* Geometric constraints; Dependence analysis; Basic clusters; Graph reduction

---

## 1. Introduction

The problem of solving geometric constraint systems is an important topic in developing intelligent or parametric CAD systems [1–3,9–13]. In constraint based CAD applications, the design objects are viewed as a geometric constraint system constituted of various geometric elements and constraints, and the designs are modified by changing the parameter values. Therefore, geometric constraint solver has been considered as an important tool in many applications such as mechanical part design, simulations, kinematics, and assembly design.

There are many attempts in the literature to provide a powerful yet efficient method for solving geometric constraint problems. Among them, we distinguish three major approaches for solving declarative constraint systems: the numerical approach, symbolic approach, and synthetic approach.

In the numerical approach, the geometric constraints are translated into a system of equations. The constraints on simple geometry, such as *distance*, *angle*, and *incidence*,

can be represented by polynomials. Therefore, it can be restricted to algebraic equations and various numerical techniques are used to solve them. Sketchpad [22] and ThingLab [2] used numerical relaxation as the last resort of solving constraint problems. Duff et al. [4] provided a review of algorithms that process systems of linear equations to find sub-sets of equations, which define values for several variables. Light and Gossard [14] used such techniques to partition the Jacobian matrix associated with a set of geometric constraints, and thus improved the efficiency of numerical constraint solution. The main advantage of numerical methods is their generality. However, the shortcomings of numerical methods include slow runtimes, numerical instabilities, and difficulties in handling redundant constraints.

In the symbolic approach, the constraints are also transformed into algebraic equations. Instead of being solved directly, the equations are first changed to new forms using general symbolic methods such as Wu-Ritt's characteristic set method [7] and the Gröbner basis method [15], and then the new equations are solved numerically. The symbolic approach may provide complete methods for many geometric constraint satisfaction problems. If the constraints are used symbolically, the parameterized solutions are produced. These generic solutions can be evaluated

---

<sup>\*</sup> Corresponding author. Tel.: +86-10-62782052; fax: +86-10-62771138.

*E-mail address:* lyt@ncc.cs.tsinghua.edu.cn (Y.-T. Li).

for different constraint assignments. The disadvantage of symbolic solvers is that current methods are still too slow for real time computation.

In the synthetic approach, a planning phase is carried out to transform the constraint problem into a step-by-step constructive form that is easy to compute, then the constraint system can be solved efficiently. Two main techniques used in the synthetic approach are the rule-based search and the graph analysis. In the approach of Borning [2], a constraint was represented as a rule and was assigned a set of methods that could be invoked to satisfy the constraint. Roller [18] and Verroust et al. [23] described another approach of rule-based constraint evaluation based on the earlier works of Sunde [21]. Kramer [16] tried to determine a solution that satisfies the specified constraints by analyzing the degree of freedom using techniques from kinematics. Owen [17] and Bouma et al. [3] presented graph reduction approaches to solve geometric constraint systems. Hoffmann and Vermeer [11] extended the algorithm from Bouma et al. [3] for 3 dimensions. Fudos and Hoffmann [6] presented the correctness proof of a graph-based variational geometric constraint solver. Lee and Kim [13] solved a geometric constraint problem by incrementally identifying a set of constrained geometric elements with 3 DOF as a rigid body. At present, most parametric design systems adopt the graph reduction approach as a fundamental scheme for solving declarative geometric constraint systems.

An important concept behind the solvers based on graph reduction is *rigidity*, since there is an intrinsic relationship between well-constrained problems and *generically rigid* constraint graphs [19]. Although the graph reduction methods have acquired great success in solving 2-D geometric constraint systems, finding an algorithm to systematically produce a realization for any 2-D constraint system is still an open problem. In 3 dimensions this problem is even more intricate, since a general characterization for generically rigid 3-D constraint graphs has not been presented.

In this paper, we present a new method of dependence analysis to analyze 3-D geometric constraint systems. The basic idea of *dependence analysis* is that the solutions of some geometric elements depend on some others because of the constraints between them. If some geometric elements depend on each other, they must be solved together. Using bipartite matchings without weights, we analyze the relationships between rigidity of the constraint graph and connectivity of the corresponding dependence graph of a constraint system. The maximum matching method has been considered by, for example, Serrano and Gossard [20], Latham and Middleditch [12], and Hoffmann et al. [10]. As pointed out by Hoffmann et al. [10], the previous methods (1) can not directly identify well-constrained sub-graphs of a constraint system and (2) provide no natural way of finding a minimal well-constrained sub-graph. Using the relationships provided in this paper, we give the conditions when a sub-graph corresponds to a well-constrained one and

prove that our reduction method obtains the minimum order-reduction result of a well-constrained system.

In practice, under- and over-constrained cases frequently exist. For over-constrained cases, we give a rule to check whether a constraint is structurally redundant. After deleting all the redundant constraints, the over-constrained system is translated to structurally under- or well-constrained system. For under-constrained cases, we add some constraints to satisfy the well-constrained requirement. And we show that the order of the new well-constrained system after adding the additional constraints is not more than that of the original system multiplied by 5.

Based on the dependence analysis, a DOF-based graph constructive method capable of efficiently handling well-, over- and under-constrained systems is described. Like most graph-based constraint solvers, the proposed method contains two phases: (1) analysis phase and (2) constraint evaluation phase. In the analysis phase of our approach, after converting the under- and over-constrained cases into well-constrained cases, we apply a recursive searching process to identify all the clusters. Then the whole constraint system is translated into a tree structure such that the none-leaf nodes are basic cluster nodes and the leaf nodes are geometric elements. Our reduction method can handle *any* type of constraint systems. In the constraint evaluation phase, the inner constraints of each cluster are solved together, and then the results are propagated to the whole constraint system. And we also show the implementation results of our method.

The remainder of this paper is organized as follows. Section 2 presents the dependence analysis method. Section 3 briefly describes the constraint evaluation phase. Section 4 shows the implementation results. Finally, Section 5 presents a summary with some remarks.

## 2. Dependence analysis

Dependence analysis is concerned with the associations between the geometric elements and constraints. In this paper, the theory and examples are presented in a 3-dimensional domain. Of course, there are similar conclusions in a 2-D plane.

### 2.1. Definitions and representation

The geometric elements include points, lines, planes, spheres, cylinders and polyhedrons. Each element has corresponding *degree of freedom* that is the minimal number of real-valued parameters required to specify the element in space unambiguously, noted as  $DOF(g)$ . A geometric constraint  $c$  reduces the DOFs of relative geometric elements by a certain number, called the *degree of constraint* of  $c$ , noted as  $DOC(c)$ . If the DOC of a constraint is more than one, that constraint can be substituted by several constraints with one DOC each.

For the convenience of constraint processing, we add a

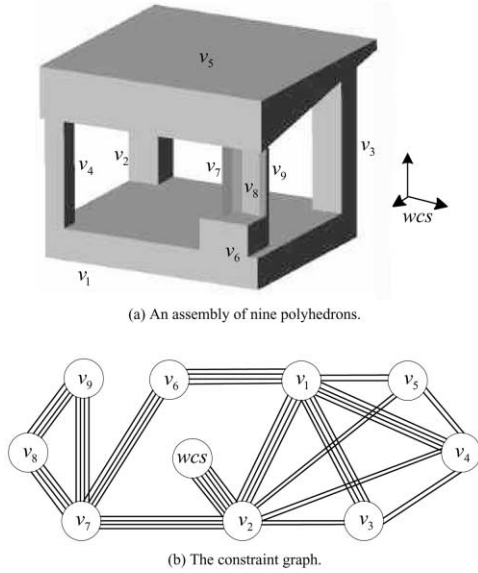


Fig. 1. A constraint system and corresponding constraint graph.

world coordinate system *wcs* into the constraint system. In the reduction process, *wcs* is considered to be a rigid body with six DOF. Then it is fixed and is the start point of the constraint evaluation phase, i.e. *wcs* has zero DOF in the constraint evaluation process. With an appropriate representation and some preprocessing, we may restrict ourselves to rigid bodies and pairwise constraints. For example, a point may be equally turned into a rigid body by adding three directions on the point and three *vector-plane-angle* constraints between those directions and the coordinate planes of *wcs*. Therefore, all the geometric elements considered in this paper are rigid bodies with six DOF each and all the constraints are pairwise constraints with one DOC each.

A geometric constraint system can be represented by a *constraint graph*  $G = (V, E)$ , in which each node is a geometric element and each edge is a constraint. Fig. 1 gives an example of nine polyhedrons and the corresponding constraint graph. A *structure graph* is a bipartite graph in which each node represents a constraint or a geometric element. The edges in the structure graph link each constraint node to two geometric element nodes constrained by that constraint.

Following Fudos and Hoffmann [6], we give the definitions of under-, well- and over-constraint problems. A graph with  $n$  nodes is structurally *over-constrained* if there is an induced sub-graph with  $m \leq n$  nodes and more than  $6m - 6$  edges. A graph is structurally *under-constrained* if it is not structurally over-constrained, and the number of edges is less than  $6n - 6$ . And a graph is structurally *well-constrained* if it is not structurally over-constrained, and the number of edges is equal to  $6n - 6$ . Note that a structurally well-constrained graph can be over-constrained in a geometric sense, for example, if there are three points  $p_1, p_2, p_3$  with three *distance* constraints, where  $distance(p_1, p_2) > distance(p_1, p_3) + distance(p_2, p_3)$ .

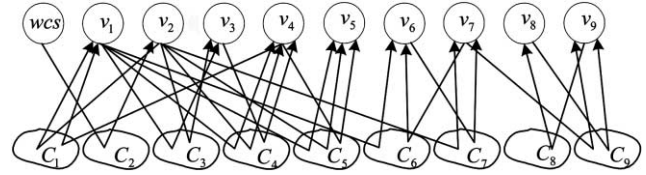


Fig. 2. The constraint partition result of Fig. 1(b).

Since *wcs* has zero DOF in the constraint evaluation phase, a well-constrained problem has a finite number of solutions if there is no numerical redundancies.

In order to illustrate the constraint propagation process, we specify one rigid body as the starting point. We define that rigid body as the *base* of the constraint system.

For each geometric element  $g$  in a well-constrained system, there must be some corresponding constraints such that the sum of DOCs of those constraints is equal to the DOF of  $g$ . Thus we have the following theorem.

**Theorem 1.** Let  $B = (V, C, E)$  be the structure graph of a well-constrained system, where  $V$  is the set of geometric element nodes,  $C$  is the set of constraint nodes and  $E$  is the set of edges. Then there exists a partition of  $C$  such that

1.  $\forall v_i \in V - \{base\}$  correspond to a constraint set  $C_i \subset C$  s.t.  $\forall c_j \in C_i, c_j$  and  $v_i$  are adjacent and

$$\sum_{c_j \in C_i} DOC(c_j) = DOF(v_i)$$

2.  $C = \bigcup_{1 \leq m \leq |V|} C_m$

$$\text{and } \forall C_m, C_n, C_m \cap C_n = \Phi.$$

**Proof.**  $\forall v_i \in V - \{base\}$ , replace  $v_i$  with  $DOF(v_i)$  nodes and copy the edges incident with  $v_i$  to each new node. Then each node of  $V$  stands for one variable and each node of  $C$  stands for one equation. Since the constraint system is well-constrained, we can find a complete matching according to the sparse matrix theory [4]. Thus  $\forall v_i \in V - \{base\}$ , the corresponding constraint set  $C_i$  is the set of matching constraints of the new nodes that have replaced  $v_i$ .  $\square$

From the proof, we can get the partition of  $C$  based on the maximal matching algorithm of bipartite graphs. The partition result of Fig. 1(b) is shown in Fig. 2, in which *wcs* is set to be the base. The arcs in Fig. 2 indicate the matching of geometric elements and constraint sets.

After constraint partition, we can convert the structure graph  $B = (V, C, E)$  into a directed graph via the following operations.  $\forall e_i = \langle v_p, c_q \rangle \in E$ , let  $v_r$  be the corresponding geometric element of constraint  $c_q$  in the partition result,

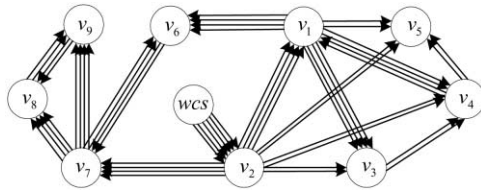


Fig. 3. The dependence graph of Fig. 1(b).

if  $r \neq p$ , add an directed edge  $\langle v_p, v_r \rangle$ . After deleting all the constraint nodes and the edges incident with them, we get a directed graph called *dependence graph*, which has at least two strong components (i.e. maximal strongly connected induced graph). Fig. 3 shows the dependence graph of Fig. 1(b).

**Theorem 2.** *The dependence graph of a well-constrained system becomes exactly the constraint graph after converting the directed edges to undirected ones.*

**Proof.** From the generating process of dependence graph we see the conclusion clearly.  $\square$

2.2. Relationships between graph rigidity and connectivity

In design applications, usually there is a set of geometric elements whose positions and orientations relative to each other are known according to the constraints between them. These geometric elements can form a rigid body with 6 DOF (3 translational DOF and 3 rotational DOF). In the constraint graph, the corresponding nodes and the edges between them constitute a well-constrained induced graph  $G_r$ .  $G_r$  is called a *cluster* and can be combined into a single node. The combination operation is called *reduction*, a reduction is denoted by  $\rightarrow^R$  and a sequence of reductions is denoted by  $\rightarrow^R$ . For a cluster  $G = (V, E)$ , there exists the relation

$$\sum_{v_i \in V} DOF(v_i) - \sum_{c_j \in E} DOC(c_j) = 6,$$

i.e.  $6|V| - |E| = 6$ .

A cluster that does not contain sub-clusters is called a *basic cluster*. We divide the basic cluster patterns into two categories: pattern 1 is that there exists node  $g$  in the cluster such that the number of incident edges of  $g$  with *outer* nodes is not less than  $DOF(g)$ ; and the residuals are pattern 2. Fig. 4 illustrates patterns 1 and 2.

A constraint system may have many different reduction ways. The *order of a cluster* is the number of nodes in that cluster. The *order of a reduction result* is the maximal order of all the clusters in that reduction result, and the *order of a constraint graph* is the minimal order of *all* possible reduction results. The order of a reduction result can be used to evaluate whether the reduction method is satisfying.

There are certain relationships between the rigid induced graphs of the constraint graph and corresponding *strongly*

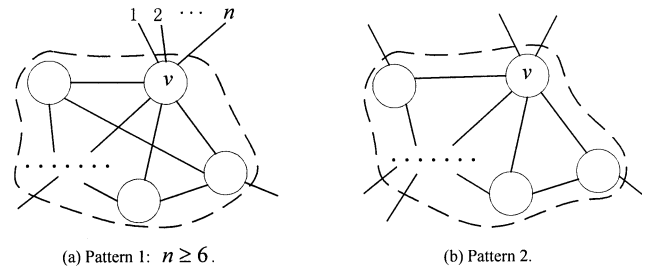


Fig. 4. Two basic cluster patterns.

*connected* induced graphs of the dependence graph. Theorems 3–6 provide a mathematical basis of the reduction method to well-constrained systems.

**Theorem 3.** *The geometric elements of a basic cluster of pattern 2 are strongly connected in the dependence graph if the base does not belong to that cluster.*

**Proof (by contradiction).** Let  $G_r = (V_r, E_r)$  be a basic cluster of pattern 2 and  $D_r = (V_r, A_r)$  be the corresponding induced graph of the dependence graph. Suppose  $\exists v_i, v_j \in V_r$ ,  $v_i$  and  $v_j$  are not strongly connected in  $D_r$ . Then  $D_r$  can be divided into  $D_{r1} = (V_{r1}, A_{r1})$  and  $D_{r2} = (V_{r2}, A_{r2})$  that are not strongly connected, as shown in Fig. 5. If  $|V_{r2}| = 1$ ,  $v_j$  corresponds to six outer constraints, i.e.  $n = 6$ , then  $G_r$  falls into basic cluster pattern 1. Therefore  $|V_{r2}| > 1$ . Because  $G_r$  does not contain sub-clusters, we know that  $6|V_{r2}| - 6 > |A_{r2}|$ .  $G_r$  is well-constrained, then each variable must have a corresponding constraint, i.e.  $6|V_{r2}| = |A_{r2}| + n$ . Since  $n \leq 6$ , the equation becomes  $6|V_{r2}| \leq |A_{r2}| + 6$ , it is contrary to  $6|V_{r2}| - 6 > |A_{r2}|$ .  $\square$

**Theorem 4.** *The geometric elements of basic cluster pattern 1 except for the node incident with  $n \geq 6$  outer edges are strongly connected in the dependence graph if the base does not belong to that cluster.*

**Proof.** The proof is similar to that of theorem 3.  $\square$

**Theorem 5.** *There exists a strong component  $D_r$  of the dependence graph  $D$  of a well-constrained system such that  $D_r$  corresponds to basic cluster pattern 1 or 2, and one outer node correspond to basic cluster pattern 1 after some steps of reduction.*

**Proof.** After reducing all the basic clusters of the strong

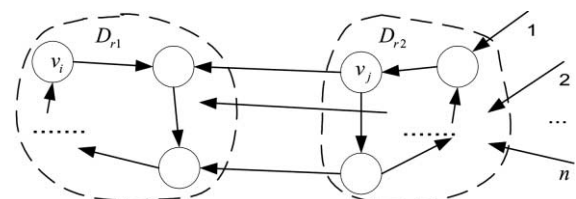


Fig. 5. Two parts of  $D_r$  that is not strongly connected.

components of  $D$  recursively,  $D$  becomes an acyclic digraph if each strong component is considered to be one node. If there exists  $D_r$  that corresponds to basic cluster pattern 1 or 2, the conclusion is true. Otherwise, two different cases can be distinguished according to the number of strong components of  $D$ . Case 1 (the number of strong components is 2): Let  $D_r$  and  $\{base\}$  be the two components. Because  $D$  is well-constrained and there are no sub-basic clusters in  $D_r$ ,  $D_r$  and  $\{base\}$  corresponds to basic cluster pattern 1. Case 2 (the number of strong components is more than 2): Assume there is no such  $D_r$  so as to  $D_r$  and an outer node correspond to basic cluster pattern 1. Because there is no sub-basic clusters in each strong component and  $D$  is well-constrained, we can see that  $D$  corresponds to basic cluster pattern 1. Then the induced graph  $G(D - \{base\})$  is strongly connected, this is contrary to the fact that  $D$  is acyclic.  $\square$

**Theorem 6.** *Let  $D$  be the dependence graph of a well-constrained system.  $D$  is a basic cluster if the number of strong components of  $D$  is 2 when arbitrary node  $n$  of  $D$  is regarded as the base.*

**Proof.** Assume there is induced graph  $D_r = (V_r, A_r)$  in  $D$  that corresponds to a cluster. Let  $n$  be a node of  $D_r$ . When  $n$  is regarded as the base, because  $6|V_r| - |A_r| = 6$  and every node of  $D$  except for  $n$  corresponds to six constraints, we know that  $\forall v \in V_r - \{n\}$ , the corresponding constraint set is a subset of  $A_r$ . Therefore, no outer arcs are directed towards  $D_r$ , this is contrary to the fact that  $D$  has only two strong components.  $\square$

### 2.3. Processing over-constrained cases

For over-constrained cases, we can use the constraint partition method to check whether a constraint is structurally redundant. After deleting all the redundant constraints, the over-constrained graph is translated to structurally under- or well-constrained graph.

**Theorem 7.** *Assume constraint system  $G = (V, E)$  becomes over-constrained by adding constraint  $c = \langle v_p, v_q \rangle$ , where  $v_p$  and  $v_q$  belong to  $V$ . Then there exists a constraint that has no corresponding geometric element when  $v_p$  is set to be the base.*

**Proof.** Because  $c$  over constrains  $G$ , we know that there is

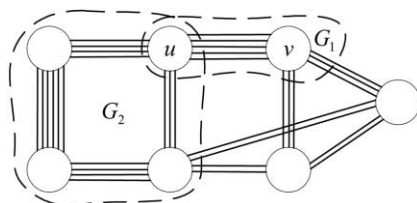


Fig. 6. An under-constrained system with two basic pseudo-clusters.

an induced graph  $G_i = (V_i, E_i)$  of  $G$  such that  $c \in G_i$  and  $6|V_i| - 6 < |E_i|$ . Since  $v_p \in G_i$  is set to be the base, there must be  $|E_i| - 6(|V_i| - 1)$  constraints that can not find corresponding geometric elements.  $\square$

**Algorithm 1.** The algorithm to deleting structurally redundant constraints.

*Input:*  $G = (V, E)$ , the constraint graph of a constraint system.

*Output:*  $G$ , the well- or under-constrained system.

```

procedure OVER_TO_UNDERORWELL( $G$ )
begin
1  generate graph  $G^* = (V^*, E^*)$  such that  $V^* = V$  and
    $E^* = \Phi$ ;
2  for  $c = \langle v_p, v_q \rangle \in E$  do
3  begin
4  add  $c$  to  $G^*$ ;
5  set  $v_p$  as the base of  $G^*$ ;
6  generate structure graph  $B^*$  of  $G^*$ ;
7  partite the constraint set of  $B^*$  using the maximal
   matching algorithm of bipartite graph;
8  if some constraint cannot find corresponding
   geometric element then
9  set  $c$  be structurally redundant and delete it from
    $G$  and  $G^*$ ;
10 end
11 return  $G$ ;
end
    
```

Obviously, Algorithm 1 identifies all the structurally redundant constraints. From graph theory [5] we know that a maximal matching of the bipartite graph can be found in  $O(|V||E|)$ . Since we have  $O(|E|) = O(|V|)$  for a design, Algorithm 1 takes  $O(|V|^3)$  in all.

### 2.4. Processing under-constrained cases

It is much more complex to deal with under-constrained cases. Let  $G_r$  be an induced graph of constraint graph  $G$ .  $G_r$  is called a *pseudo-cluster* if it becomes well-constrained after adding more than zero constraints that do not over constrain  $G$ . A pseudo-cluster that does not contain sub-pseudo clusters is called a *basic pseudo-cluster*. Note that a basic pseudo-cluster may contain sub-clusters, for example,  $G_2$  in Fig. 6. There are three rules to evaluate the additional constraints that convert an under-constrained graph into a well-constrained one.

- Rule 1. They do not over constrain the constraint graph.
- Rule 2. They do not change the structure of the constraint graph.
- Rule 3. The order of the constraint system does not increase.

It is explicit that Rule 1 must be satisfied for all additional

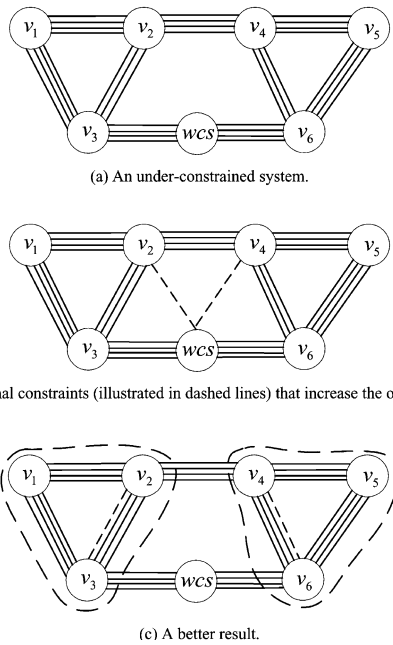


Fig. 7. An under-constrained system and the additional constraints.

constraints. In practical applications, usually the user ignores some constraints for simplicity and the related induced graphs correspond to basic pseudo-clusters. We expect that the additional constraints make the basic pseudo-clusters well-constrained. However, it is difficult to find an algorithm to satisfy this requirement. Obviously, the basic pseudo-clusters are determined by the topology and the numbers of edges between node pairs of the constraint graph. And if the additional constraints change the structure of the constraint graph, the basic pseudo-clusters with two nodes cannot become clusters. For example, if we do not add constraint between  $u$  and  $v$  in Fig. 6,  $G_1$  cannot become well-constrained. Therefore, we use Rule 2 to evaluate the additional constraints.

When decomposing the constraint system, we hope to get the best reduction result whose order is equal to that of the constraint system. And if the additional constraints increase the order of the constraint system, that of the reduction result will increase consequently. Thus Rule 3 is applied to judge whether the way of adding constraints is good. Actually, Rule 3 relates to Rule 2 because the order of a constraint system is decided by those of the basic clusters and pseudo-clusters of that system. As can be seen in below materials, our method that does not change the constraint graph structure increases the order of the constraint system by less than five times.

For example, we need two additional constraints to well constrain the under-constrained system illustrated in Fig. 7(a). If we place the two constraints as Fig. 7(b), the order of the reduction result is not minimal. A better result is shown in Fig. 7(c).

**Theorem 8.** Let  $S$  be the set of constraints that do not over

constrain the constraint system and translate basic pseudo-cluster  $G_p = (V_p, E_p)$  to well-constrained graph,  $x(u, v)$  be the number of edges between  $u$  and  $v$  that are two nodes in  $V_p$ . Then after replacing every sub-cluster of  $G_p$  with a single node, (1)  $|S| \leq 6 - \max_{u,v \in V_p} x(u, v)$  and (2) the constraints  $S$  of can be all added between arbitrary node  $u$  and  $v$ .

**Proof.** If  $|V_p| = 2$ , obviously the conclusions are correct. Otherwise, after replacing every cluster of  $G_p$  with a single node, let

$$x(u_0, v_0) = \max_{u,v \in V_p} x(u, v)$$

and  $G_0 = (V_0, E_0)$  be an arbitrary induced graph of  $G_p$  such that  $u_0, v_0 \in V_0$ . We know that  $6|V_0| - 6 > |E_0|$ . Since  $G_p$  is a basic pseudo-cluster and does not contain sub-pseudo-clusters, the relationship  $6|V_0| - 6 > |E_0|$  must still be satisfied after we add one edge between  $u_0$  and  $v_0$ , and the additional constraint do not over constrain  $G_p$ . We repeat the process of adding edges between  $u_0$  and  $v_0$ , and  $u_0$  and  $v_0$  become a well-constrained induced graph after adding  $6 - x(u_0, v_0)$  edges. Therefore,  $G_p$  must become well-constrained after adding less than  $6 - x(u_0, v_0)$  edges, i.e.

$$|S| < 6 - \max_{u,v \in V_p} x(u, v).$$

And from the process of adding edges we can see that the constraints of  $S$  can be all added between arbitrary node  $u$  and  $v$ .  $\square$

**Theorem 9.** Let  $u$  and  $v$  be two arbitrary nodes of an under-constrained graph and  $x(u, v)$  be the number of edges between  $u$  and  $v$ . Then after adding less than or equal to  $6 - x(u, v)$  edges between  $u$  and  $v$ , there exists an induced graph  $G_r$  such that (1)  $u, v \in G_r$  and (2)  $G_r$  is a cluster and the whole constraint system is not over-constrained.  $G_r$  is noted as  $well(u, v)$ .

**Proof.** The proof is similar to that of Theorem 8.  $\square$

Obviously, if we set node  $u$  as the base,  $well(u, v)$  is the combination of the strong component containing  $v$  and all the prior strong components of the dependence graph. And  $well(u, v)$  need not to be a basic pseudo-cluster after deleting the additional constraints, as illustrated in Fig. 8. Then we

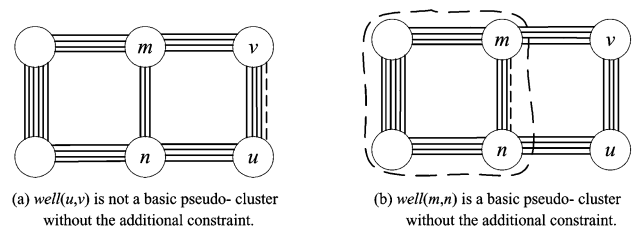


Fig. 8. Adjusting the additional constraints.

must adjust the additional constraints to get a more preferable result.

Assume  $G_p = (V_p, E_p)$  become well-constrained after adding some constraints and  $E_r = \{ \langle u_r, v_r \rangle \mid u_r, v_r \in V_p \} \subseteq E_p$  be the edge set such that we can add more than zero constraints between  $u_r$  and  $v_r$ . From the definition of basic pseudo-cluster, we know that  $G_p$  is a basic pseudo-cluster if  $well(u_r, v_r)$  is  $G_p$  for arbitrary edge  $\langle u_r, v_r \rangle \in E_r$ .

It is clear that all nodes of a basic pseudo-cluster are connected, i.e. a basic pseudo-cluster contains at least one existing edge. Therefore, we can add constraints between the nodes that are adjacent to well constrain the basic pseudo-cluster. In this way, the topology of the constraint graph remains the same and the structure of the graph does not change.

From above discussion, we can give an algorithm to convert under-constrained cases to well-constrained cases.

**Algorithm 2.** The algorithm to convert under-constrained cases to well-constrained cases.

*Input:*  $G = (V, E)$ , the constraint graph of an under-constrained system.

*Output:*  $G$ , the well-constrained system.

```

procedure UNDER_TO_WELL( $G$ )
begin
1  if there are more than one connected components in  $G$ 
   then
2    for all connected component  $G_r$  that do not contain
      $wcs$  do
3      add six edges between  $wcs$  and arbitrary  $v_0 \in G_r$ ;
4       $E' \leftarrow E$ ;
5      for  $\langle u, v \rangle \in E'$  and the priority of  $\langle u, v \rangle$  is
     minimal do
6        begin
7          delete all the edges between  $u$  and  $v$  from  $E'$ ;
8          add  $x < 6$  edges that is not structurally redundant to
            $E$  between  $u$  and  $v$ . Then a cluster containing  $u$ 
           and  $v$  is generated;
9          if  $x = 0$  then
10             continue;
11           $u_0 \leftarrow u, v_0 \leftarrow v$ ;
12          set  $u$  as the base of  $G$ ;
13          generate structure graph  $B$  and dependence graph  $D$ 
           of  $G$ ;
14           $G' \leftarrow well(u, v)$ ;
15          delete all the additional edges from  $G'$  and  $G$ ;
16          if  $G' \neq G$  then
17             begin
18               delete all edges that appear in  $G'$  from  $E'$ ;
19               UNDER_TO_WELL( $G'$ );
20             end
21          end
22          if  $G$  is not well-constrained then

```

```

23      add constraints between  $u_0$  and  $v_0$  to well
        constrain  $G$ ;
24  return  $G$ ;
end

```

In line 5 of Algorithm 2, if we have added edges between  $u$  and  $v$ , then the priority of  $\langle u, v \rangle$  is maximal. Otherwise, the priority of  $\langle u, v \rangle$  is minimal means that the number of edges between  $u$  and  $v$  is minimal. In Algorithm 2, the process of adding constraints between  $u$  and  $v$  costs  $O(6|E|^2)$  in the worst case. If there are  $x$  edges in  $G'$  in line 19, the loop of line 5–21 will be repeated for  $O(|E|) - x$  times. Since  $O(|E|) = O(|V|)$ , we can see that Algorithm 2 requires  $O(|V|^3)$  in all.

**Theorem 10.** Algorithm 2 converts an under-constrained graph into a well-constrained one.

**Proof.** In Algorithm 2, line 1–3 make the constraint graph connected. For arbitrary edge  $\langle u, v \rangle$ ,  $well(u, v)$  is well-constrained. Since  $well(u, v)$  is connected with outer nodes, it is an induced graph of other well-constrained graph. Algorithm 2 processes all the edges, thus it converts an under-constrained graph into a well-constrained one.  $\square$

After adding some constraints to a basic pseudo-cluster, it is possible that we can not add constraints to some other basic pseudo-clusters. As shown in Fig. 9, when we add some constraints to  $G_p$ ,  $G_1$  and  $G_2$  disappear.

Let  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  be two basic pseudo-clusters and  $|V_1| > |V_2|$ ,  $x(u, v)$  be the number of edges between node  $u$  and  $v$ . Generally speaking, we have the relationship

$$\min_{u, v \in V_1, x(u, v) > 0} x(u, v) < \min_{m, n \in V_2, x(m, n) > 0} x(m, n).$$

The loop of line 5–21 in Algorithm 2 searches the basic pseudo-clusters from the minimal  $x(u, v)$ , this strategy will keep the basic pseudo-clusters with more nodes in general.

**Lemma 1.** Let  $G_p$  and  $G_1, G_2, \dots, G_n$  be basic pseudo-clusters. Assume  $G_1, G_2, \dots, G_n$  disappear if  $G_p$  becomes a basic cluster after adding some constraints, and  $G_p$  disappear if  $G_1, G_2, \dots, G_n$  become basic clusters. Then  $n \leq 5$ .

**Proof.** Let  $s(G)$  be the number of edges that well

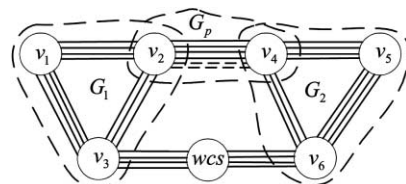


Fig. 9.  $G_1$  and  $G_2$  disappear when we add some constraints to  $G_p$ .

constrain basic pseudo-cluster  $G$ . From Theorem 8 we know that  $1 \leq s(G) \leq 5$ , then  $G_p$  ‘occupies’ the constraints of no more than five other basic pseudo-clusters. Thus  $n \leq 5$ . $\square$

**Lemma 2.** *The node number of the intersection of two basic pseudo-clusters is not more than one.*

**Proof.** Let  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  be two basic pseudo-clusters,  $G_s = (V_s, E_s)$  be the intersection, and  $G_{\text{total}} = (V_{\text{total}}, E_{\text{total}})$  be the union of  $G_1$  and  $G_2$ . Obviously  $G_s \neq G_1$  and  $G_s \neq G_2$ . Let assume  $6|V_1| - 6 = |E_1| + p$ , then after adding  $p$  constraints to  $G_1$ , we know that  $|E_{\text{total}}| = |E_1| + p + |E_2| - |E_s| = 6|V_1| - 6 + |E_2| - |E_s| \leq 6|V_{\text{total}}| - 6$ . Since  $|V_{\text{total}}| = |V_1| + |V_2| - |V_s|$ , we get  $6|V_s| - |E_s| \leq 6|V_2| - |E_2|$ . If  $|E_s| \neq 0$ ,  $G_s$  becomes a cluster after adding less than  $6|V_2| - |E_2| - 6$  edges. This is contrary to the fact that  $G_2$  is a basic pseudo-cluster. $\square$

**Theorem 11.** *Assume Algorithm 2 convert under-constrained graph  $G$  to well-constrained graph  $G'$ . Let  $p$  be the order of  $G$  and  $q$  be that of  $G'$ . Then  $q \leq 5p$ .*

**Proof.** Immediately from Lemma 1 and 2. $\square$

### 2.5. Reduction algorithms

After processing over- and under-constrained cases, the whole constraint system is converted into a well-constrained graph. From Theorem 5 we know that we can identify the clusters of a well-constrained graph via searching the strong components of the dependence graph. By applying the searching process *recursively*, the whole constraint system can be translated into a tree structure such that the none-leaf nodes are basic cluster nodes and the leaf nodes are geometric elements. Algorithm 3 gives the process to well-constrained system reduction and Algorithm 4 gives the top-level description to constraint system reduction.

**Algorithm 3.** The algorithm to well-constrained system reduction.

*Input:*  $G = (V, E)$ , the constraint graph of a well-constrained system.

*Output:*  $n$ , the root vertex after reduction.

```

procedure WELL_GRAPH_REDUCTION( $G$ )
begin
1   for  $v \in V$  do
2   begin
3     set  $v$  as the base;
4     generate structure graph  $B$ ;
5     generate dependence graph  $D$  using the maximal
      matching algorithm of bipartite graph;
6     search all the strong components  $D_1, D_2, \dots, D_m$ ;
7     if  $m \neq 2$  then
8     begin

```

```

9       search all well-constrained induced graphs
       $G_1, G_2, \dots, G_r$ , which are irrelevant to each
      other;
10      for  $i \leftarrow 1$  until  $r$  do
11        WELL_GRAPH_REDUCTION( $G_i$ );
12      end
13    end
14    replace  $G$  with a single cluster node  $n$ ;
15  return  $n$ ;
end

```

Since  $|E| = 6|V| - 6$ , line 5 costs  $O(|V|^2)$  and line 4, 6 and 9 cost each  $O(|V|)$ . If there are  $x$  nodes in  $G_i$  in line 11, the loop of line 1–13 will be repeated for  $|V| - x$  times. Therefore, Algorithm 3 requires  $O(|V|^3)$  in all.

A well-constrained system may have many different reduction ways. Using the proposed method, we can obtain the minimum order-reduction result of a well-constrained system.

**Lemma 3.** *The node number of the intersection of two basic clusters is not more than one.*

**Proof.** Let  $R_1 = (V_1, E_1)$  and  $R_2 = (V_2, E_2)$  be two basic clusters,  $S = (V_s, E_s)$  be the intersection, and  $S_{\text{total}} = (V_{\text{total}}, E_{\text{total}})$  be the union of  $R_1$  and  $R_2$ . Then we have  $|E_{\text{total}}| = |E_1| + |E_2| - |E_s| = 6|V_1| - 6 + 6|V_2| - 6 - |E_s| = 6|V_{\text{total}}| - 12 + 6|V_s| - |E_s|$ . Since  $|E_{\text{total}}| \leq 6|V_{\text{total}}| - 6$ , we get  $6|V_s| \leq |E_s| + 6$ . If  $|E_s| \neq 0$ , since  $R_1$  and  $R_2$  are basic clusters, we get  $|E_s| < 6|V_s| - 6$ , it is contrary to  $6|V_s| \leq |E_s| + 6$ . Thus  $S$  has one or zero node. $\square$

**Theorem 12.** *Algorithm 3 gets the minimum order-reduction result of a well-constrained system.*

**Proof.** Let  $p$  be the maximal order of basic clusters (may after some reduction steps) of a well-constrained system  $G$ . Obviously the minimum order of the all possible reduction results is  $p$ . In Algorithm 3, every time a basic cluster is reduced. Three different cases can be distinguished based on the relationship of current reducing cluster  $G_1$  and arbitrary cluster  $G_2$ . Case 1:  $G_1$  and  $G_2$  are not intersected. The two reduction steps are independent, so the orders of  $G_1$  and  $G_2$  are not more than  $p$ . Case 2:  $G_1$  and  $G_2$  are intersected. From Lemma 3 we know that one cluster can be a node of the other, thus the orders are not more than  $p$ . Case 3:  $G_2$  becomes basic cluster after reducing  $G_1$ . Obviously the orders of  $G_1$  and  $G_2$  are not more than  $p$ . From the previous we know the order of the reduction result generated by Algorithm 3 is  $p$ . $\square$

**Algorithm 4.** The top-level algorithm to constraint system reduction.

*Input:*  $G = (V, E)$ , the constraint graph of a constraint system.

*Output:*  $n$ , the root vertex after reduction.



```

procedure CONSTRAINT_GRAPH_REDUCTION
(G)
begin
  process the geometric elements and constraints such
  that each nodes of G has six DOF and each
  constraint has one DOC;
  G ←OVER_TO_UNDERORWELL(G);
  G ←UNDER_TO_WELL(G);
  n← WELL_GRAPH_REDUCTION(G);
return n;
end
  
```

It is clear that Algorithm 4 takes  $O(|V|^3)$  in all. And the correctness proof of the proposed reduction method can be found in the appendix.

### 3. Constraint evaluation process

After the reduction process, the constraints entailed must be solved. In order to solve a single cluster efficiently, we apply analytic solution if the combination pattern of constraints is supported. Otherwise, we use the Newton-Raphson method to solve the equations.

Once a cluster being solved, the result should be propagated to the whole constraint system. In the design process, many modifications of designs are generated by modifying one parameter of a constraint or a geometric element. In such cases, We can find the cluster containing the modified parameter and solve it, then solve the whole constraint system by solving parent of current cluster in series till the root cluster. When several parameters are modified simultaneously, we may solve the whole constraint system by solving every branch of the cluster tree. And the solving process of one branch is irrelevant to those of other branches.

### 4. Implementation

The proposed method has been implemented as an assembly constraint solver in a feature-based parametric CAD system of GEMS 5.0. In the reduction process, all the basic clusters are reduced recursively. And in the constraint evaluation process, the clusters are solved using analytic or iterative methods. The reduction process of Fig. 1(b) is given in Fig. 10.

Fig. 11 gives an example of a bicycle assembly, which contains 27 parts and 62 assembly constraints. Each part has six DOF, and after preprocessing, the assembly constraints are translated into 157 valid geometric constraints with one DOC each. Then we get the well-constrained system after deleting seven structurally redundant constraints and adding 12 constraints. And the assembly is constructed by solving all the constraints entailed using the Newton-Raphson method.

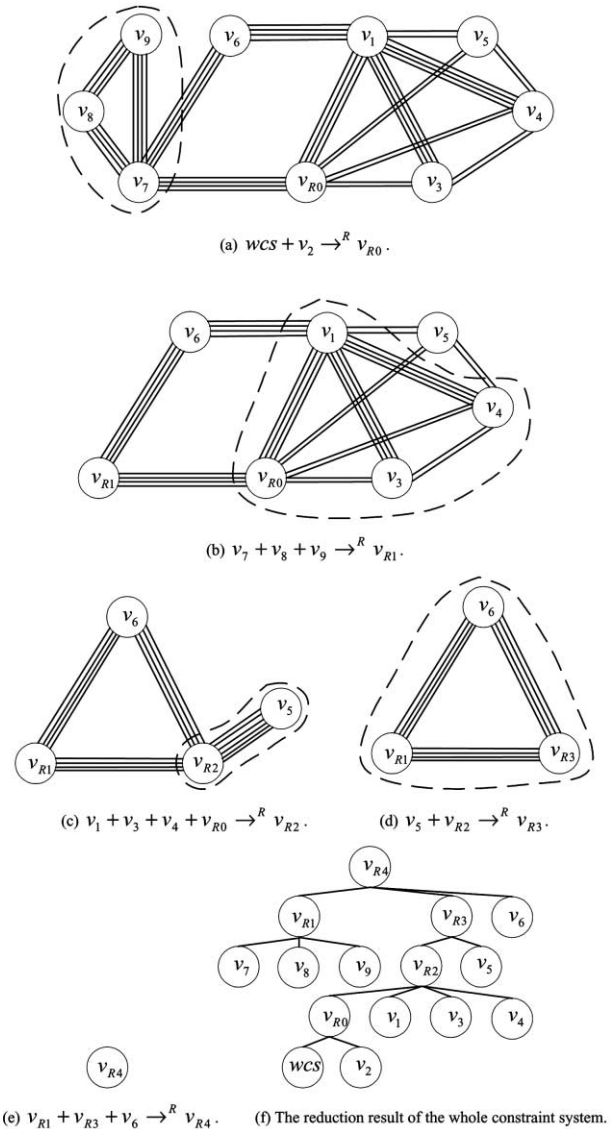


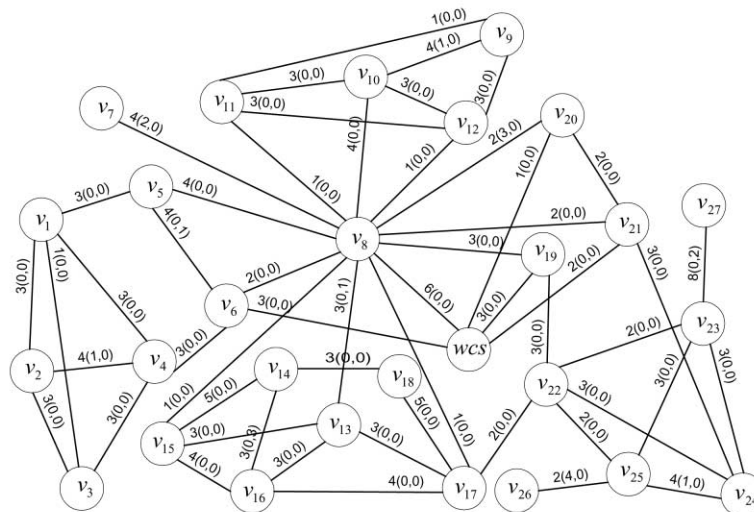
Fig. 10. The reduction process of Fig. 1(b).

### 5. Conclusion

We have presented a new DOF-based graph constructive method to geometric constraint systems solving based on the dependence analysis of the geometric elements. The basic idea is that the geometric elements whose solutions depend on each other must be solved together. In our approach, we make a constraint graph well-constrained by first deleting all structurally redundant constraints, and then adding constraints by three rules to the under-constrained system. And we prove that the order of a constraint system after processing under-constrained cases is not more than that of the original system multiplied by 5. After that, we apply a recursive searching process to identify all the clusters, which can get the minimum order-reduction result of a well-constrained system. We also briefly describe the constraint evaluation



(a) The parts that correspond to a bicycle.



(b) The constraint graph of the bicycle, where the first number on an edge stands for the initial DOC and the other two numbers in the parenthesis stand for the additional and redundant DOC of that edge, respectively.

Fig. 11. A bicycle assembly.

phase. Finally we give the implementation results of our approach.

The proposed method is capable of efficiently handling well-, over- and under-constrained systems and can process *any* type of constraint systems. Algorithms, such as those described in Owen [17], Bouma et al. [3] and Lee and Kim [13], are more efficient than the algorithms proposed in this paper in 2-D plane, but their domain is restricted. In addition, they cannot handle 3-D constraint problems efficiently.

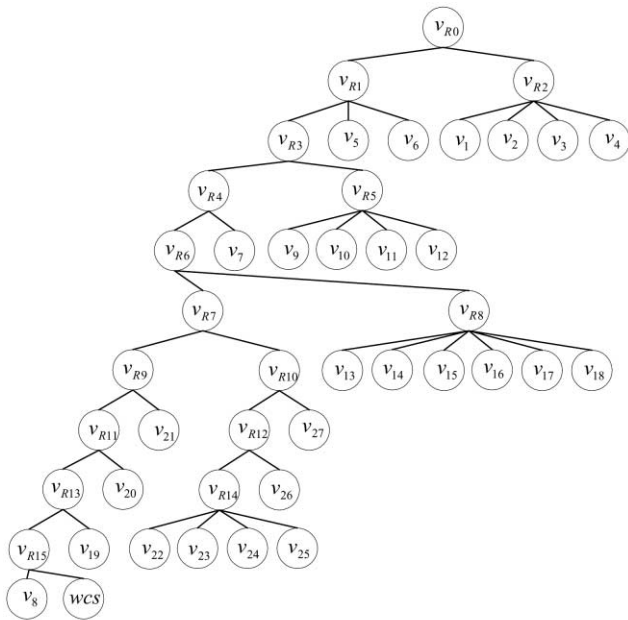
Some of the future works are listed below.

- Although we have presented algorithms to reduce the constraint systems, the efficiency of the algorithms may be improved. More efficient ways should be found to process over- and under-constrained cases and accelerate the reduction process.

- The stability of numerical solving should be improved and the optimization methods are considered. According to Ge et al. [8], the BFGS method is stable and effective. And it may be considered to adopt a heuristic way to find the intended solutions.
- The proposed method is a generic one and disposes of the structural constraint types. Future work is needed to deal with the numerical redundancy problem.

### Acknowledgements

This research was supported in part by the National Natural Science Foundation of China (Project No. 69902004) and NKBRFS (Project No. 1998030600).



(c) The reduction result of the constraint graph.



(d) Assemble all the parts using the proposed constraint evaluation method.

Fig. 11. (continued)

### Appendix. The correctness proof of the reduction method

Following Fudos and Hoffmann [6] and Lee and Kim [13], we present the correctness proof of the proposed method by proving that the reduction algorithm is *canonical* for well-constrained problems. For the definitions of canonical and confluent, see [6] or [13].

**Lemma A1.** *If  $G_I \xrightarrow{*R} G_M$ , the induced graph that corresponds to a cluster in  $G_M$  is structurally well-constrained.*

**Proof.** We prove the conclusion by induction on the length of the reduction sequence that drives  $G_M$  from  $G_I$ . The induction basis is  $G_M = G_I$  and is trivial. For the induction step, consider the last reduction step that merges the cluster  $R_1, R_2, \dots, R_k$  into a new cluster  $R$ . Let  $G_i$  be the sub-graph of  $G_I$  corresponding to  $R_i (1 \leq i \leq k)$ . Let  $n_i$  and  $e_i$  be the number of nodes and edges in the sub-graph  $G_i$ , respectively. Because the node sets of  $G_i$  are disjoint, we

know that  $R$  has

$$n = \sum_{1 \leq i \leq k} n_i$$

nodes and

$$e = \sum_{1 \leq i \leq k} e_i + 6k - 6$$

edges. Since  $e_i = 6n_i - 6 (1 \leq i \leq k)$ , we get  $e = 6n - 6$ , thus the sub-graph is structurally well-constrained. •

**Theorem A1.** *A reduction is locally confluent.*

**Proof.** Assume that  $G_I \xrightarrow{*R} G_M$  and there are two different reductions  $G_M \xrightarrow{R_1} G_{M1}$  and  $G_M \xrightarrow{R_2} G_{M2}$ . Then we have to show that  $G_{M2} \xrightarrow{R_1} G_{M3}$  and  $G_{M1} \xrightarrow{R_2} G_{M3}$ . From Lemma 3 we know that the intersection of two basic clusters can only be one node, thus after one reduction step, the new cluster node becomes one node of the other cluster, then  $\xrightarrow{R_1}$  and  $\xrightarrow{R_2}$  commute. •

**Theorem A2.** *The proposed algorithm terminates.*

**Proof.** In line 14 of Algorithm 3, each cluster reduction replaces at least two nodes in the constraint graph by a single node. Therefore, the number of nodes in the graph decreases monotonically and the process must terminate when there is a single cluster node in the graph. •

**Theorem A3.** *The proposed reduction process is confluent.*

**Proof.** Immediately from Theorem A1 and A2. •

**Theorem A4.** *The proposed reduction algorithm is canonical for well-constrained problems.*

**Proof.** Let assume two single node graphs be the results of reduction process for graph  $G$ . Because the reduction process is confluent and the two graphs are reduction results of  $G$ , they may conflate to a common reduction  $G'$ . And they cannot be further reduced so that they must be identical since they are single node graphs. Therefore, all single node graphs obtained at the end of the reduction process are identical, and the process is canonical. •

### References

- [1] Alfeld B. Variation of geometries based on a geometric-reasoning method. *Computer Aided Design* 1988;20(3):117–26.
- [2] Borning A. The programming language aspect of ThingLab. *ACM Trans. On Programming Language and Systems* 1981;3(4):353–87.
- [3] Bouma W, Fudos I, Hoffmann CM, Cai J, Paige R. Geometric constraint solver. *Computer Aided Design* 1995;27(6):487–501.
- [4] Duff IS, Erisman AM, Reid JK. *Direct methods for sparse matrices*. Oxford Scientific Publications, 1986.
- [5] Foulds LR. *Graph theory application*. Springer-Verlag, 1992.
- [6] Fudos I, Hoffmann CM. *Correctness proof of a geometric constraint*

- solver. *International Journal of Computational Geometry & Applications* 1996;6(4):405–20.
- [7] Gao XS, Chou SC. Solving geometric constraint systems. II. A symbolic approach and decision of rc-constructibility. *Computer Aided Design* 1998;30(2):115–22.
- [8] Ge JX, Chou SC, Gao XS. Geometric constraint satisfaction using optimization methods. *Computer Aided Design* 1999;31(14):867–79.
- [9] Heydon A, Nelson G. The Juno-2 constraint-based drawing editor, SRC Research Report 131a, 1994.
- [10] Hoffmann CM, Lomonosov A, Sitharam M. Finding solvable sub-sets of constraint graphs. In: Smolka G, editor. Springer LNCS 1330, 1997. p. 463–77.
- [11] Hoffmann CM, Vermeer PJ. Geometric constraint solving in R2 and R3. In *Computing in euclidean geometry*, 2nd ed. World Scientific Publishing, 1994.
- [12] Latham RS, Middleditch AE. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design* 1996;28(11):917–28.
- [13] Lee JY, Kim K. A 2-D geometric constraint solver using DOF-based graph reduction. *Computer Aided Design* 1998;30(11):883–96.
- [14] Light RA, Gossard DC. Modification of geometric models through variational geometry. *Computer Aided Design* 1982;14(4):209–14.
- [15] Kondo K. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer Aided Design* 1992;24(3):141–7.
- [16] Kramer GA. A geometric constraint engine. *Artificial Intelligence* 1992;58:327–60.
- [17] Owen JC. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications*, ACM Press, 1991. p. 379–407.
- [18] Roller D. A system for interactive variation design. In: Wozny J, Turner, Preiss K, editors. *Geometric modelling for product engineering*, North Holland, 1990. p. 207–19.
- [19] Sansone G. *Orthogonal functions*. Revised English Edition. Dover, 1991.
- [20] Serrano D, Gossard DC. Combining mathematical models with geometric models in CAE systems. In *Proceedings of the ASME Computer in Engineering Conference*, Chicago, IL, 1986. p. 277–84.
- [21] Sunde, G., A CAD system with declarative specification of shape, Eurographics workshop on intelligent CAD systems, Noorwijkerhout, the Netherlands, 1987, pp. 90–104.
- [22] Sutherland I. Sketchpad, a man-machine graphical communication system. In *Proceedings of the Spring Joint Comp. Conference*, North-Holland, 1963. p. 329–45.
- [23] Verroust A, Schonec F, Roller D. Rule oriented method for parameterized computer-aided design. *Computer Aided Design* 1992;24(10):531–40.



**Yan-Tao Li** is a PhD student in the Department of Computer Science and Technology, Tsinghua University, Beijing, P. R. China. He received a BSc degree from the Department of Precision Instruments and Mechanology, Tsinghua University in 1997. His current research interests include geometric constraint solving, feature-based design, and image based modelling.



**Shi-Min Hu** is an associate professor in the Department of Computer Science and Technology, Tsinghua University. He received a BSc in Mathematics from Jilin University in 1990, a MSc and a PhD degree in Computational Geometry and Computer Graphics both from Zhejiang University in 1993 and 1996. His research interests are in computer aided geometric design, computer graphics, and content-based image indexing.



**Jia-Guang Sun** is a professor in the Department of Computer Science and Technology, Tsinghua University. He is also director of the National CAD Engineering Center at Tsinghua University, and Academician of Chinese Academy of Engineering. His research interests are in computer aided geometric design, computer graphics, and product data management.