http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (7): 1447-1451, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1447.1451

An Expanding Algorithm of Geometric Constraint Solving in Semantic Feature Modeling

[1,2]Xianguo Liu and [1]Lijuan Sun
[1]College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, People's Republic of China
[2]School of Software, Liaoning Technology University, Huludao 125105, People's Republic of China

Abstract: A new solving approach for constraint problem was proposed in this study, the constraint problem needed to solve was decomposed not into single sub-problems but into three types of sub-problems, namely, rigid subset, scalable subset and radial subset and each type of subset corresponds a cluster of constraint problem. Based on cluster rewriting rule approach, a small set of rewriting rules were applied in constraint system and an incremental algorithm was presented, the solving approach could get the generic solution when no available rewriting rule was available. By this approach, we can determine that constraint system is well-constrained, under-constrained or over-constrained. The results reveal that the proposed method can efficiently process constraint problem.

Key words: CAD, geometric model, constraint solving, incremental algorithm, semantic feature modeling

INTRODUCTION

Parametric design (Hoffmann and Kim, 2001; Sun and Liu, 2011) has become the core technology in CAD system in recent years and the geometric constraint solving (Hoffmann, 2005) is the most important part in semantic feature modeling (Bronsvoort *et al*., 2006). But so far, although geometric constraint has been brought into three-dimensional field, it can be better used in the field of two-dimension. It can solve some problems about parts assembling (Sun and Ding, 2010; Ding and Sun, 2009) but the existing solving methods (Xueliang *et al*., 2010; Wei *et al*., 2008) in three-dimensional geometry have many difficulties in the applications.

Now, cluster solving method (Bettig and Hoffmann, 2010) is the most mature approach, namely geometric constraint problems are decomposed into several well-constrainted sub-problems. Each single sub-problem is solved independently, then all sub-solutions are combined into a generic solution and the generic solution is the solution of the whole constraint system. Cluster solving method has two modes: rewriting rule method (Durand and Hoffmann, 2000; Hoffmann and Vermeer, 1995) and degree of freedom based rule method (Kramer, 1992; Gao *et al*., 2006).

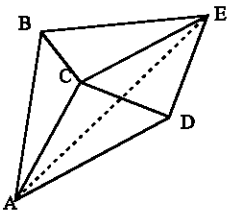
Rewriting rule method is that modeling system tries to find some subsets of constraint problem and these subsets must be well-constrained and they have accurate and efficient methods corresponding for solving, so the constraint system can be solved quickly. But this method is not perfect for three-dimensional problems because there is no rule which can decompose a constraint system into several well-constrained sub-problems. Therefore, this method can not be used to solve all of well-constrained problems and may result in under-constrained (Sun *et al*., 2010).

Degree of freedom based rule is that it determines well-constrained sub-problems by using degree of freedom analysis. For the specific sub-problems, solving algorithm may be unknown and algebraic method (Van-der-Meiden and Bronsvoort, 2005) is used to solve in this approach, it will increase greatly the cost of computation when constraint system is very large, so degree of freedom based rule is also imperfect.

Both of methods require constraint system can be decomposed into several well-constrained subsets, based on rewriting rule and degree of freedom based rule, a new solving approach is presented in this study. Firstly constraint system is decomposed into three types subset: rigid subset, scalable subset and radial subset. Then incremental algorithms are applied and at last the generic solution will be get. This approach can get solution for the constraint system that can not be decomposed into sub-problems solved independently.

A constraint problem instance is shown in Fig. 1. There are five points in three-dimension space where, ∠BAC, ∠DAC, ∠EAD, ∠BAE, ∠ABC, ∠ADC, ∠ADE, ∠EAB are constrained by angle constraint and a distance

Corresponding Author: Xianguo Liu, College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, People's Republic of China



<(A, B, C) <(C, A, D)
<(A, D, C) <(D, A, E)
<(A, D, E) <(E, A, B)
< (A, B, E) < (B, A, C)
Distance (B, E)

Fig. 1: Instance model and its constraint representation

constraint lies between point B and point F. It is easy to get hat the whole system is well-constrained but it is difficult to decompose the system into several well-constrained subsets. Therefore, the whole system must be solved in both of method aforementioned. Present approach can be used easily to decompose the system into several subset and solving process is effected very light when a few variables changes.

DEFINITION OF SUBSET

A cluster (Van-der-Meiden and Bronsvoort, 2006) basically represents a collection of distance and angle constraints on a set of points. We define three types of subset: rigid subset, scalable subset and radial subset. The type of a subset determines which distances and angles are constrained by it. Also, a set of configurations is associated with a subset, each of which determines an alternative set of values for the distances and angles constrained by the subset.

Subset distance constraints δ (m, n) are defined as follows:

$$\delta(m,n)=\sqrt{(m-n).(n-m)}$$

Subset angle constraints $\angle$(m, n, t) are defined as follows:

$$\angle(m,n,t)=\cos^{-1}\left(\frac{m-n}{\delta(m-n)}.\frac{t-n}{\delta(t,n)}\right)$$

where, m, n, tϵR^2 or R^3 are points in the cluster.

By this definition, distances and angles are unsigned; i.e., δ (m, n)$\geq$0 and 0$\leq\angle$(m, n, t)$\leq\pi$.

Definition 1: Rigid subset is a constraint set of all the points [a_1, a_2, …, a_n] such that the relation position of all points is constrained in two- or three-dimensional models, i.e. all distances and angles in point set are constrained. As shown in Fig. 2a, the rigid subset is well-constrained and has no internal degree of freedom but it can be translated or rotated. The notation for the subset on a set of points [a_1,a_2,…, a_n] is RIGD ([a_1,a_2, …, a_n]).

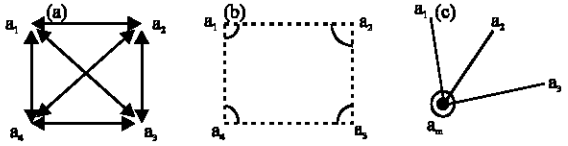


Fig. 2: Three types of clusters

Definition 2: Scalable subset is a constraint set of all the points [a_1, a_2, …, a_n] such that for all i, j, kϵ [1, n] and the angles $\angle(a_i, a_k, a_j)$ are constrained, shown in Fig. 2b. The scalable subset has one internal degree of freedom, it can be scaled uniformly. The notation for the subset on a set of points [a_1, a_2, …, a_n] is SCLB ([a_1, a_2, …, a_n]).

Definition 3: Radial subset is a constraint set of all the points [a_m, a_1, a_2, …, a_n] such that for all i, jϵ [1, n] and the angles $\angle(a_i, a_m, a_j)$ are constrained, shown in Fig. 2c. Point am is called the centre point and points [a_1, a_2,…, a_n] are called the radial points. The radial subset has n internal degrees of freedom. The notation for the subset on a set of points [a_m, a_1, a_2,…, a_n] is RADI ([a_m, a_1, a_2,…, a_n]).

REWRITING APPROACH

Re-writing process: To solve a constraint system, rewriting the system to a single rigid cluster by trying to apply a set of rewrite rules in this approach. A rewrite rule specifies a pattern, describing its input clusters and its output cluster in a generic way and it specifies a procedure to determine the configurations of the output cluster from the configurations of the input clusters. A rewrite rule can be applied if a set of clusters is found in the system that matches the input clusters in the pattern. The corresponding output cluster is added to the system and the configurations of the output cluster are determined by the procedure.

A pattern specifies a number of input clusters of a given type and a number of pattern variables. These pattern variables are matched by the solving algorithm to the point variables of the clusters in the system, such that the number of variables and the type of the cluster match. A pattern may also specify that an input cluster can match any cluster with a superset of the given variables. If a variable name occurs several times in the pattern, it must be matched with a single point variable that is constrained by several clusters in the constraint system.

To determine the configurations of the output cluster of a rewrite rule, the procedural part of the rule is applied for every combination of input cluster configurations. Suppose, for example, that two clusters are used as the input of a rewrite rule and that each cluster has two configurations associated with it; then four different configurations for the output cluster are computed.

The generic solution is initially empty. When the user adds a cluster to the problem (Algorithm 1), the cluster is added to the generic solution. The algorithm then searches for possible rewrite applications on that new cluster, i.e., rewrite rule applications where the cluster is used as input (Algorithm 3). When a cluster is removed (Algorithm 2), it is removed from the generic solution. The rewriting approach is implemented as follows:

Algorithm 1: Adding a subset

```
Function add subset (G,A,s)
G: Generic solution
A: Active set
s: Subset
begin
G. add (s)
A. add (s)
Search rewrites (G, A, s)
End
```

Algorithm 2: Removing a subset

```
Function remove subset (G, A, s)
G: Generic solution
A: Active set
s: Subset
begin
G. remove (s)
A. remove (s)
for each x in dependent subsets (G, s)
Remove subset (x)
for each y in deactivated subsets (A, s)
A. add (y)
Search rewrites (G, A, y)
End
```

Algorithm 3: Searching for rewrite rule applications

```
Function Search rewrites (G, A, s)
G: Generic solution
A: Active set
s: Subset
begin
subset := s + Overlapping subsets (A, s)
reference := Reference graph (subset)
for each rule in All rewrite rules
pattern := Pattern graph [rule]
matches = Subgraph isomorphisms (pattern, reference)
for each match in matches
rewrite := instantiate rule from match
if is Progressive (rewrite) then
G. add (rewrite)
A. add (rewrite.output)
for each i in rewrite. inputs
if I is Redundant (i) then
A. remove (i)
Search rewrites (rewrite. output)
end
```

Rewriting rules: If $a_1, a_2, a_3, \ldots, a_n \in R^3$.

Rule 1:

$$\text{RADI}(a_1, [a_3, a_2, \ldots]) \cup \text{RADI}(a_2, [a_1, a_3, \ldots]) \rightarrow \text{SCLB}(a_1, a_2, a_3])$$

This rule can be applied when two radial subset share three points. When a match is found, a new scalable subset is added to the system.

This rule can be applied to the problem in Fig. 1, as follows:

$$\text{RADI}(A, [B, C, D, E, F]) \cup \text{RADI}(B, [A, C]) \rightarrow \text{SCLB}([A, B, C])$$
$$\text{RADI}(A, [B, C, D, E, F]) \cup \text{RADI}(D, [C, A, E]) \rightarrow \text{SCLB}([A, C, D])$$
$$\text{RADI}(A, [B, C, D, E, F]) \cup \text{RADI}(D, [C, A, E]) \rightarrow \text{SCLB}([A, D, E])$$
$$\text{RADI}(A, [B, C, D, E, F]) \cup \text{RADI}(F, [A, E]) \rightarrow \text{SCLB}([A, E, F])$$

Rule 2:

$$\text{SCLB}(A = [a_1, a_2, \ldots]) \cup \text{SCLB}(B = [a_1, a_2, \ldots]) \rightarrow \text{SCLB}(AUB)$$

Two subsets can be combined into one new one when the shared points between two scalable subsets have the same coordinate. The rule can be applied repeatedly, in the example problem, as follows:

$$\text{SCLB}([A, B, C]) \cup \text{SCLB}([A, C, D]) \rightarrow \text{SCLB}([A, B, C, D])$$
$$\text{SCLB}([A, D, E]) \cup \text{SCLB}([A, E, F]) \rightarrow \text{SCLB}([A, D, E, F])$$
$$\text{SCLB}([A, B, C, D]) \cup \text{SCLB}([A, D, E, F]) \rightarrow \text{SCLB}([A, B, C, D, E, F])$$

Rule 3:

$$\text{SCLB}(A = [a_1, a_2, \ldots]) \cap \text{SCLB}([a_1, a_2, \ldots]) \rightarrow \text{RIGD}(A)$$

The rule can be applied to the example system, resulting in a cluster RIGD ([A, B, C, D, E, F]) which constrains all the variables of the problem.

The solving constraint problem process is indicated by a directed acyclic graph of subset and rewrite rules. The solving process of Fig. 1 is shown in Fig. 3. In Fig. 3 arrows indicate dependency relations between subset created by rewrite rules. In Fig. 3, the subset with no incoming arrows are problem subsets and the subset with no outgoing arrows is the solution subsets.

From its process, we can know whether a problem is well-constrained, under-constrained and over-constrained:

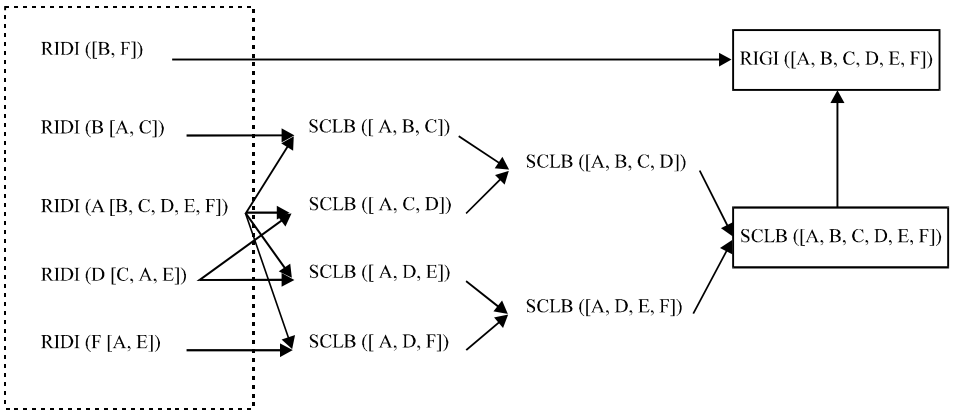


Fig. 3: Generic solution fo the problem in Fig. 1

- A problem is under-constrained if its generic solution has more than one solution subset or a single non-rigid solution subset
- A problem is over-constrained if any distance constraint or angle constraint has more than one source subset
- A problem is well-constrained if it is not under-constrained and not over-constrained. Note that these conditions are not mutually exclusive

REWRITING RULE APPLICATION

Specific steps of subset rewriting approach are explained in a instance, as shown in Fig. 1, based on the approach introduced in this study, geometric constraint system can be indicated as follows:

RADI (A, [B, C, D, E, F])
RADI (B, [A, C])
RADI (D, [C, A, E])
RADI (F, [A, E])
RIGD ([B, F])

Re-writing constraint system using rule 1 as follows:

RADI (A [B, C, D, E]) ∪ RADI (B, [A, C]) → SCLB ([A, B, C])
RADI (A [B, C, D, E]) ∪ RADI (B, [C, A, E]) → SCLB ([A, C, D])
RADI (A [B, C, D, E]) ∪ RADI (B, [A, E]) → SCLB ([A, D, E])
RADI (A [B, C, D, E]) ∪ RADI (B, [A, E]) → SCLB ([A, E, B])
RIGD ([B, F])

Re-writing constraint system using rule 2 as follows:

SCLB ([A, B, C]) ∪ SCLB ([A, C, D]) → SCLB ([A, B, C, D])
SCLB ([A, D, E]) ∪ SCLB ([A, E, B]) → SCLB ([A, D, E, B])
RIGD ([B, F])

Rule 2 was applied again:

SCLB ([A, B, C, D]) ∪ SCLB ([A, D, E, B]) → SCLB ([A, B, C, D, E])
RIGD ([B, F])

Re-writing constraint system using rule 3 as follows:

SCLB ([A, B, C, D, E]) ∪ RIGD ([B, F]) → RIGD ([A, B, C, D, E, F])

So, we can get generic solution of geometric constraint problems shown in Fig. 1 by using subset rewriting approach.

This method indicates that only one sub set of rewriting rules is changed when any distance or angle constraint in models is being modified and the change can only affect the sub set which it located, not affect as traditional approach that all constraint sub-system was modified.

CONCLUSION

Two types of representation of geometric constraint system are proposed in this study, namely, scalable subset and radial subset. This approach expends the original rewriting method and using a simple cluster rewriting approach, larger class of constraint problems can be solved than with only rigid subset. An incremental algorithm is presented for this approach. The advantages of this method are that large problems can be solved efficiently and that an incremental solving algorithm is easy to develop and implement.

ACKNOWLEDGMENTS

The authors are very grateful to Editor and Reviewers for their comments and constructive suggestions which

help to enrich the content and improve the presentation of this study. The study was supported by the National Natural Science Foundation of China under Grant No. 60173055.

REFERENCES

Bettig, B. and C.M. Hoffmann, 2010. Geometric Constraint Solving in CAD. http://www.cs.purdue.edu/homes/cmh/distribution/PapersChron/ConstraintSurvey2010.pdf.

Bronsvoort, W.F., R. Bidarra and P.J. Nyirenda, 2006. Developments in feature modeling. Comput. Aided Design Appl., 3: 655-664.

Ding, B. and L. Sun, 2009. Ontology-based model for software resources interoperability. Inform. Technol. J., 8: 871-878.

Durand, C. and C.M. Hoffmann, 2000. A systematic framework for solving geometric constraints analytically. J. Symbolic Comput., 30: 493-519.

Gao, X.S., Q. Lin and G.F. Zhang, 2006. A C-tree decomposition algorithm for 2D and 3D geometric constraint solving. Computer-Aided Design, 38: 1-13.

Hoffmann, C.M. and K.J. Kim, 2001. Towards valid parametric CAD models. Comput. Aided Design, 33: 81-90.

Hoffmann, C.M. and P.J. Vermeer, 1995. Geometric Constraint Solving in R^2 and R^3. In: Computing in Euclidean Geometry, Du, D.Z. and F. Huang (Eds.). 2nd Edn., World Scientific Publishing, Singapore, pp: 266-298.

Hoffmann, C.M., 2005. Constraint-based CAD. J. Comput. Inform. Sci. Eng., 5: 182-197.

Kramer, G.A., 1992. Solving Geometric Constraint Systems: A Case Study in Kinematics. MIT Press, Cambridge, New York.

Sun, L. and B. Ding, 2010. Ontology-based semantic interoperability among heterogeneous CAD systems. Inform. Technol. J., 9: 1635-1640.

Sun, L.J., Y.H. Jin and D.S. Sun, 2010. Research on cube-based boolean operation in cellular semantic feature modeling system. Inform. Technol. J., 9: 956-961.

Sun, L. and X. Liu, 2011. Research on algorithm of computing parameter interval in geometric models. Inform. Technol. J., 10: 402-408.

Van der Meiden, H.A. and W.F. Bronsvoort, 2005. An efficient method to determine the intended solution for a system of geometric constraints. Int. J. Comput. Geometry Appl., 15: 279-298.

Van der Meiden, H.A. and W.F. Bronsvoort, 2006. A constructive approach to calculate parameter ranges for systems of geometric constraints. Comput. Aided Design, 38: 275-283.

Wei, S., M. Tieqiang and L. Tao, 2008. Constraint conversion method in feature-based heterogeneous CAD model exchange. Inform. Technol. J., 7: 783-789.

Xueliang, H., C. Liping, W. Boxing and H. Yunbao, 2010. 3D geometric constraint solving for integrated variational design. J. Comput. Aided Design Comput. Graphics, 22: 30-36.