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 (2): 402-408, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.402.408


# Research on Algorithm of Computing Parameter Interval in Geometric Models

Lijuan Sun and Xianguo Liu
College of Computer Science and Technology,
Harbin University of Science and Technology, Harbin 150080, People's Republic of China

**Abstract:** In the process of modeling, users often change parameter values using trial-and error method in some CAD system. This study aims to present a novel approach that determines automatically suitable parameter intervals for semantic feature modeling system. A new algorithm of computing parameter range was proposed after analyzing geometric constraint graphs in this method, critical parameter value was found by decomposition for variant parameter and each sub-problem instance was solved in each interval. The results reveal that the proposed method can efficiently process constraint problem.

**Key words:** CAD, geometric model, constraint solving, parameter interval, feature modeling

## INTRODUCTION

Most of current CAD systems are based on parameters and semantic features (Bronsvoort *et al*., 2006), geometric constraints and topological constraints are two important aspects in valid constraints (Bouma *et al*., 1995; Fudos and Hoffmann, 1997). Users can create efficiently kinds of models by modifying parameters in geometric constraints (Hoffmann and Peters, 1995; Hoffmann and Joan-Arinyo, 1998) and topology of model is established in this process (Van der Meiden and Bronsvoort, 2007; Nyirenda *et al*., 2006; Bidarra *et al*., 2004). Sometimes, users can also create new models using model exchange feature-based (Wei *et al*., 2008; Jin *et al*., 2008). However, it is a repeating activity that users change parameters of models and maybe it will result in invalid geometric model because of unsuitable parameters chosen.

There are few studies about computing parameter intervals in geometric models, Hoffman and Kim presented a kind of approach of computing parameter intervals (Hoffmann and Kim, 2001), but this method is limited in horizontal and vertical line segments and distance constraints between them in two dimension space. Few of researchers do this work by combining topological constraint and geometric constraint, so it is necessary to try to find a novel way.

A new approach is presented in this study, that suitable parameter interval is determined automatically for parameter system at the time of changing single parameter. Firstly, topological constraint graph need to be analyzed to find distance constraints and angle constraints, these constraints are added into corresponding geometric constraints. Secondly, construction graph is analyzed to discover one or more critical parameter values of degeneration sub-problems, then each specific value is tested in every interval, so precise parameter interval can be found. After determining parameter intervals, user can choose parameters in this interval to avoid producing unexpected situation.

## VALIDITY CONSTRAINT IN FEATURE MODELING

**Topological constraint system:** Topological constraint is established by attaching feature to feature in the process of creating models and system keeps a topological constraint graph which contains all of feature instances that preserve corresponding shape information, parameters and constraint relations and that is connected by attaching relations between features (Meiden and Bronsvoort, 2010). A topological constraint graph is a directed graph of all of solid relations in models, each edge represents a kind of attaching relation, features attached are called host features, on the other hand, the ones attaching main features are called guest feature and direction of a directed graph is from main features to guest features.

Parameters intervals are solved on the precise that system should not change model's topology. Therefore, this must refer to a question of topology degeneration.

**Definition 1:** Topology degeneration is that system topology changes when parameter value $x \le X_{min}$ or $x \ge X_{max}$.

**Corresponding Author:** Xianguo Liu, College of Computer Science and Technology,
Harbin University of Science and Technology, #52 Xuefu Road, Nangang District, Harbin,
Heilongjiang, People's Republic of China Tel: +86-137-960-95470

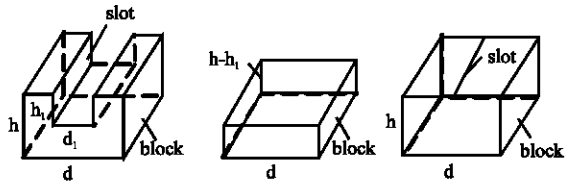


Fig. 1: Topology degeneration. (a) Instance model, (b) $d_1 = d$ and (c) $d_1 = 0$

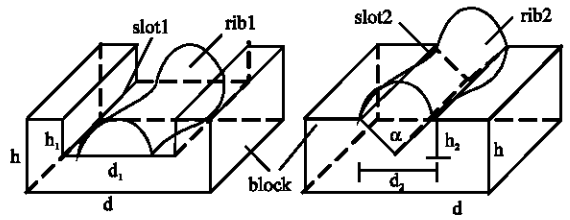


Fig. 2: Topological constraint graph. (a) Instance1 and (b) instance2


Fig. 3: Topological attaching graph in Fig. 2

Width of block model is d, height is h, width of slot attaching block is $d_1$, height is $h_1$, which is shown in Fig. 1a. Figure 1a will be Fig. 1b when $d_1$ is equal to d and that will be Fig. 1c when $d_1$ is equal to zero, so topology will be changed in these two cases. However, topology will not be changes when $d_1$ is more than zero and at the same time, less than d, so these two cases are called two kinds of topology degeneration. The other two cases of topology degeneration are that it will be change at the case of $h_1$ is equal to h and $h_1$ is equal to zero.

Deep first search algorithm is used to search a topological directed graph (Van der Meiden and Bronsvoort, 2006), each two features which have attaching relations are analyzed. Firstly, all kinds of degeneration cases of guest features are solved and then distance constraints and angle constraints which are found are added into corresponding geometric constraints.

Two model instances are shown in Fig. 2 and their topological attaching graphs are shown in Fig. 3. We can get that block feature is attached by slot feature and slot feature is attached by rib feature in Fig. 3, so slot feature is guest feature relative to block feature and rib feature is guest feature relative to slot feature. By analyzing these two guest features, we can get that slot1 feature in Fig. 2a has four types of topology degeneration, namely

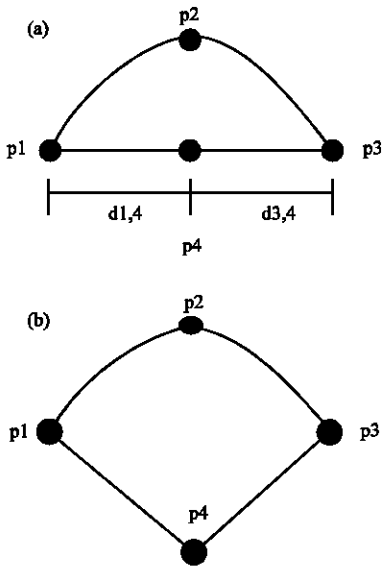


Fig. 4: FFDPs configuration prototype. (a) configuration prototype of rib1 and (b) configuration prototype of rib2

$h_1 = h$, $h_1 = 0$, $d_1 = d$ and $d_1 = 0$, so its topological constraints are $0<h_1<h$ and $0<d_1<d$, then this constraints will be applied in corresponding geometric constraints. Freeform feature rib1 in Fig. 2a was constructed as a configuration prototype shown in Fig. 4a by using FFDPs algorithm (Nyirenda *et al*., 2007). Geometric constraint between $d_{1,4}$ and $d_{3,4}$ originally is $d_{1,4} = d_{3,4}$ but after analyzing slot1 feature attached that its width is $d_1$ we can get two degenerations, namely $d_{1,4} = d_{3,4} = 0$ and $d_{1,4} = d_{3,4} = d_1/2$ , so its topological constraint is $0<d_{1,4} = d_{3,4} < d_1/2$ , at last, this constraint is added into original geometric constraint between $d_{1,4}$ and $d_{3,4}$.

As shown in Fig. 2b, slot2 feature has four degeneration situations, they are $h_2 = h$, $h_2 = 0$, $d_2 = d$ and $d_2 = 0$, angle α has also two degeneration situations, viz., $\alpha = 0$ and $\alpha = \pi$, so we can get some topological constraints from them, viz., $0<h_2<h$, $0<d_2<d$ and $0<\alpha<\pi$ and the three constraints are added into corresponding geometric constraints. Slot2 feature is attached by rib2 feature in Fig. 2b, angle of slot2 feature is α and configuration prototype of rib2 feature is shown in Fig. 4b, from the two Fig., we can get two topology degenerations, viz., $\beta = 0$ and $\beta = \alpha$, so its topological constraint is $0<\beta\leq\alpha$ and at last this constraint is added into original geometric constraint of angle β. After analyzing these topological constraint graphs, it will reduce lots of unnecessary compute in the process of geometric constraint solving and will avoid generating

invalid constraint graph and more importantly, it will help users maintain topological constraint relations of system. Geometric constraint system: some special geometric constraint solving algorithms are used in a geometric constraint system (Hoffmann *et al*., 2001), these algorithm can find effective solutions by using knowledge in a specific area. Most of geometric constraint solving system work by constructing constraint solving which creates a DR-plan graph that recognizes some sub-problems solved independently or combines sub-problems' solutions into some solution of whole system for system.

The algorithm built in our system has three steps, firstly, a problem is decomposed into some sub-problems which are also called cluster by a DR-plan graph, secondly, solving some sub-problems by determinate schemes and lastly, determining one or more whole solutions by combining sub-problems' solutions.

The core content of this paper and the mechanism of selecting sub-problems are not relevant, but it is supposed that a individual solution can be selected from each sub-problem in the DR-plan graph and the solution selected can be used in the process of computing parameter ranges.

## PARAMETER RANGE COMPUTATION

Purpose of computing parameter range is to find parameter value ranges for geometric constraint system and one or more solutions exist in this range, but the other parameters should be constant when the parameter is being computed.

Principle of parameter range computation: the basic approach is as follows: first find the critical values for the variant parameter, i.e., the parameter values for which the solvability of the system might change. Next, in each interval between two subsequent critical values, pick a value for the parameter and determine whether the system can be solved for this value. The parameter's range is the union of the intervals for which this is the case.

Because there is only one variant parameter, the system has degrees of freedom only in the sub-problems that depend on this parameter. These sub-problems are effectively under-constrained, because the parameter can vary, i.e., they have infinitely many solutions. Some of these are degenerate solutions, i.e., solutions that are on the boundary of the solution space. These degenerate solutions correspond to the critical values of the parameter. To determine all critical values, we must determine all parameter values corresponding to a degenerate solution of a sub-problem that depends on the variant parameter, either directly or indirectly.

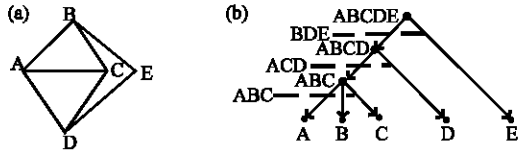


Fig. 5: A constraint problem and its DR-plan. (a) sub-problem solving instance graph and (b) DR-plan graph of Fig. a

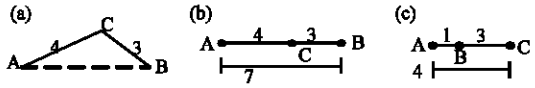


Fig. 6: Degenerate cases for a triangular sub-problem with a single variant parameter. (a) A triangular problem, (b) the first degeneration and (c) the second degeneration

A sub-problem depends directly on the variant parameter if it can be solved without using the solutions of other sub-problems. If other sub-problems must be solved first, then it depends indirectly on the parameter. Consider, for example, the constraint system in Fig. 5a and the DR-plan for this system in Fig. 5b. From the DR-plan, we can infer that to solve this system, first the sub-problem ABC is solved, then sub-problem ACD. The results are then merged into cluster ABCD, from which $\delta(B,D)$ can be determined and subsequently sub-problem BDE can be solved. Thus only after solving the first two sub-problems, can BDE be solved and merged with ABCD. If we consider the distance to be the variant parameter, then sub-problem ABC depends directly on this parameter and sub-problem BDE indirectly. For these two sub-problems, we must determine the degenerate cases and the corresponding critical values of the parameter.

For a sub-problem that depends directly on the variant parameter, it is relatively straightforward to determine the critical parameter values. Consider sub-problem ABC, as shown in Fig. 6a. Such a triangle exists only if: $\delta(A,C)+\delta(B,C)\geq\delta(A,B)\geq ABS(\delta(A,C)-\delta(B,C))$. The minimum and maximum value of $\delta(A,B)$ are its critical values. For these values, the triangle degenerates to a line segment (Fig. 6b, c).

For a sub-problem that depends indirectly on the parameter, as is the case for sub-problem BDE in the example in Fig. 1, the relation between the critical parameter values and the degenerate solutions of the sub-problem is more complex. To find these critical values, we first determine all degenerate solutions of the sub-problem. Sub-problem BDE is similar to the case of

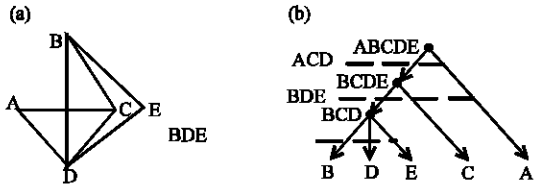


Fig. 7: The problem of Fig. 5 after adding a constraint on δ(b,D). (a) Modified constraint graph and (b) new DR-plan

Fig. 6, i.e., it has two degenerate solutions. Next, we modify the system of constraints by removing the constraint corresponding to the variant parameter and adding one or more constraints that correspond to the first degenerate solution. For sub-problem BDE, we add a distance constraint between points B and D that corresponds to the distance between those points in the first degenerate solution of the sub-problem. We then use the constraint solver to solve the modified system. From the result of this backwards solving step, we determine the first critical value of the variant parameter, simply by calculating the corresponding distance δ(A,B) from the solution.

Consider again the constraint system in Fig. 5a. By removing the constraint on δ(A,B), the variant parameter and adding a constraint on δ(B,D) that corresponds to the distance in one of the degenerate solutions of BDE, we obtain the constraint system in Fig. 7a. The DR-plan for this system is shown in Fig. 7b. As can be seen, sub-problem ABC has disappeared, sub-problems

ACD and BDE remain and a new sub-problem BCD has appeared. To solve this system, all three sub-problems are solved independently and then merged. However, sub-problem BCD is new and no intended solution has been defined for it. This sub-problem has two solutions: one where the relative orientation of the points is clockwise and one where it is counter-clockwise.

All in all, we search for six critical values for δ(A,B): two for the degenerate cases of sub-problem ABC, two for the first degenerate case of BDE and two for the second degenerate case of BDE. There is no value for the variant parameter corresponding to the particular degenerate case of the particular sub-problem of this step. So, there is no corresponding critical value for this step either.

**Sub-problem degeneration solutions:** Sub-problems in the DR-plan that depend on the variant parameter have one or more degrees of freedom when the constraint corresponding to the parameter is removed. For these sub-problems, the degenerate cases must be determined.

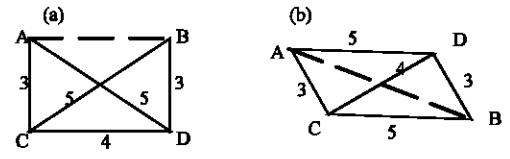


Fig. 8: Degenerate cases of the tetrahedral problem. (a) First degeneration and (b) the second degeneration

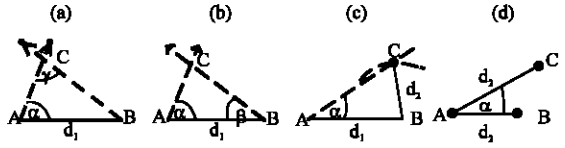


Fig. 9: Four cases of triangle sub-problem

The basic 3D sub-problem is the tetrahedral problem involving only distance constraints, an example tetrahedral problem is shown in Fig. 8. Suppose that the distance δ(A,B) is the variant parameter. With five known distances, the problem is under-constrained in 3D. However, in 2D the problem is well-constrained and has two solutions, corresponding to δ(A,B) and $\delta(A,B) = 2\sqrt{3}$, shown in Fig. 8a and b, respectively. These solutions correspond to the co-planar degenerate case of the tetrahedral problem.

Angle constraints need to be considered only in 2D sub-problems. There are four cases of triangular sub-problems with angle constraints. These are labeled aad, ada, add and dad, where a's and d's represent angles and distances, respectively and the order indicates the configuration of these angles and distances, as shown in Fig. 9. If an angle is the variant parameter, the triangular problems in which the parameter is directly involved may again degenerate. all angle parameters are in the range [0,∏]. When solving degenerate cases of tetrahedral sub-problems with angle constraints, the above triangular sub-problems also occur. Consider the problem in Fig. 10a. Two degenerate solutions are found, again by solving the system in 2D.

A pseudo-code algorithm for parameter range computation is presented in Algorithm 1. The algorithm takes two input arguments, a geometric constraint problem (argument GCP) and a variant parameter (argument VP). The variant parameter is mapped to a geometric constraint when a value is set for it (method SetParameter). If the value is cleared (method FreeParameter), then the geometric constraint is removed. Geometric constraints can be added to (method AddConstraint) and removed from the GCP (method RemoveConstraint). We assume that the generic solution

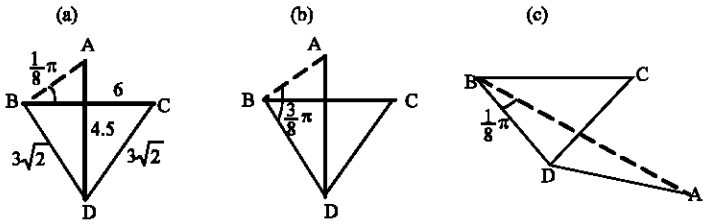


Fig. 10: A tetrahedral sub-problem with an angle constraint. (a) Tetrahedral constraint graph, (b) the first degeneration and (c) the second degeneration

(function GenericSolution) is automatically updated by the GCP, resulting in an efficient algorithm, but this is not strictly required.

Algorithm 1: Geometric parameter range computation

```
function GeometricParameterRange (GCP,VP)
GCP: geometric constraint problem
VP: variant parameter
begin
    assert WellConstrained(GCP)
    generic := GenericSolution(GCP)
    dependent := DependentSubproblems(generic,VP)
    FreeParameter(VP, GCP)
    Y := empty set of critical values
    R := empty set of intervals
    for each subproblem in dependent
        cases := DegenerateCases(subproblem)
        for each case in cases
            AddConstraints(case, GCP)
    newgeneric := GenericSolution(GCP)
            cluster := find Rigid in
        newgeneric containing VP and case
for each configuration in cluster
y := calculate VP from configuration
add y to Y
add [y,y] to R
RemoveConstraints(case, GCP)
for each subsequent interval (y1,y2) in Y
    x := (y1+y2)/2
    SetParameter(VP=x,GCP)
    if WellConstrained(GCP) then add (y1,y2) to R
    return R
end
```

The algorithm returns a set of intervals (R) that represents the parameter range. The set may contain open intervals (e.g., (y1, y2)), half-open intervals (e.g., [y1, y2)) and closed intervals (e.g., [y1, y2]). When intervals are added to the set, overlapping intervals are automatically combined, e.g., (0, 1) + [1, 2] → (0, 2].

To start, the algorithm asserts that the GCP is well-constrained, otherwise the result found by the algorithm is not a correct parameter range. It then determines the generic solution of the problem and from the generic solution it determines which sub-problems are dependent on the variant parameter.

The algorithm then modifies the system of constraints by removing the constraint corresponding to the variant parameter. Now the GCP is under-constrained. For each degenerate case of each sub-problem that is dependent on the variant parameter, we add constraints to the GCP that correspond to the degenerate case and we determine the generic solution of the modified system (preferably by an incremental solving method). The sub-problems that depend on the variant parameter will have disappeared in the new generic solution and new sub-problems that depend on the newly added constraints will have appeared.

## EXAMPLE CONSTRAINT PROBLEM

The example 3D constraint problem considered in this section is shown in Fig. 11a and its generic solution is shown in Fig. 11b. The distance between point B and point C is chosen to be the variant parameter, i.e. the parameter for which the range is to be computed. From the generic solution, we infer that the following sub-problems are dependent on $\delta$(B,C): ABC, BCD, BCF, ABCD, BCDF and ABCDF, ACEF and ABCDEF. For these sub-problems, we must find the degenerate solutions and the corresponding critical values of the variant parameter.

Sub-problem ABC is the simplest type of triangular sub-problem and degenerates for $\delta$(B,C) = 1 and $\delta$(B,C) = 7. Sub-problem BCF is similar and degenerates for $\delta$(B,C) = 1 and $\delta$(B,C) = 9. However, for these values, the complete system is over-constrained, thus no critical values are recorded.

Sub-problem BCD degenerates for $\delta$(B,C) = 0. Again, for this value, the complete system is over-constrained, thus no critical values are recorded. Sub-problem ABCD is shown in Fig. 12. This sub-problem degenerates for $\delta$(B,C) ≈ 1.01 and $\delta$(B,C) ≈ 6.97. Once again, for these values, the complete system is over-constrained and no critical values are recorded. Sub-problem BCDF is similar to ABCD and no critical values are found for this sub-problem either. These clusters are merged into sub-problem ABCDF, for which also no critical values are found.

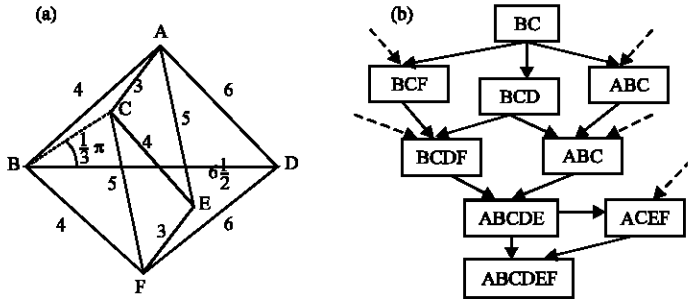


Fig. 11: 3D example problem. (a) constraint problem with variant parameter δ (B,C) and (b) part of generic solution dependant on δ (B,C)

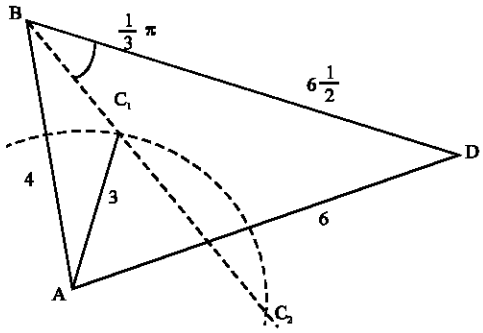


Fig. 12: Sub-problem ABCD

For sub-problem ABCDEF, again no critical values are found. No other sub-problems need to be tested for degenerate cases. Now that we have determined all critical values for δ(B,C), we test the solvability of the system by picking a parameter value in each interval between two subsequent critical values and solving the system. We find that the intended solution for this problem exists for δ(B,C) ∈ [c1, c2] ∈ [c3, c4].

To determine this parameter range, the problem was solved for a relatively small number of degenerate cases and intervals. In comparison, a naive sampling approach, i.e., an algorithm that simply tries to solve the system for different values, would need to solve the problem a very large number of times to achieve any reasonable accuracy. Also, a sampling approach may not find all intervals in the parameter range, since the minimum and maximum parameter values and the minimum sampling resolution are not known.

## CONCLUSIONS

A method of computing parameter range in topological and geometric constraint system was proposed in this paper, constraint system could find one or more solutions in this interval. Topological constraint and geometric constraint were combined in this approach, critical values of generation sub-problems were found by decomposing a whole problem and computing these critical values in polynomial time. It was available to compute parameter range in some CAD system based on parameter and semantic feature and it had the same effect in some other areas about topological and geometric constraint.

## 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

Bidarra, R., A. van Bunnik and W.F. Bronsvoort, 2004. Direct manipulation of feature models in web-based collaborative design. Proceedings of the ASME Design Engineering Technical Conferences, September 2004, The Netherlands, pp: 27-33.

Bouma, W., I. Fudos, C. Hoffmann and J. Cai and R. Paige, 1995. A geometric constraint solver. Comput. Aided Design, 27: 487-501.

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

Fudos, I. and C.M. Hoffmann, 1997. A graph-constructive approach to solving systems of geometric constraints. ACMT Graphic, 6: 179-216.

Hoffmann, C.M. and J. Peters, 1995. Geometric Constraints for CAGD. In: Mathematical Methods for Curves and Surfaces, Daehlen, M., T. Lyche and L. Schumaker (Eds.). Vanderbilt University Press, United States, pp: 237-254.

Hoffman, C.M. and R. Joan-Arinyo, 1998. CAD and the product master model. Comput. Aided Design, 30: 905-918.

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

Hoffmann, C.M., A. Lomonosov and M. Sitharam, 2001. Decomposition plans for geometric constraint systems, Part I: Performance measures for CAD. J. Symbolic Comput., 31: 376-408.

Jin, P., X. Zhang, S. Wan and L. Yue, 2008. Constraint rectangle: A novel approach to modeling continuously moving objects. Inform. Technol. J., 7: 607-614.

Meiden, H.A.V.D. and W.F. Bronsvoort, 2010. Tracking topological changes in feature models. Comput. Aided Geometric Des., 27: 281-293.

Nyirenda, P.J., M. Mulbagal and W.F. Bronsvoort, 2006. Definition of freeform surface feature classes. Comput. Aided Design Appl., 3: 665-674.

Nyirenda, P.J., R. Bidarra and W.F. Bronsvoort, 2007. A semantic blend feature definition. Comput. Aided Design Appl., 4: 795-806.

Van der Meiden, H.A. and W.F. Bronsvoort, 2006. Solving topological constraints for declarative families of objects. Proceedings of the ACM Symposium on Solid and Physical Modeling, June 6-8, Cardiff, Wales, United Kingdom, pp: 63-71.

Van der Meiden, H.A. and W.F. Bronsvoort, 2007. Solving topological constraints for declarative families of objects. Comput. Aided Design, 39: 652-662.

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