A constructive approach to calculate parameter ranges for systems of geometric constraints

Hilderick A. van der Meiden and Willem F. Bronsvoort
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
H.A.vanderMeiden/W.F.Bronsvoort@tudelft.nl

April 5, 2006

Abstract

Geometric constraints are at the heart of parametric and feature-based CAD systems. Changing values of geometric constraint parameters is one of the most common operations in such systems. However, because allowable parameter values are not known to the user beforehand, this is often a trial-and-error process. We present an approach for automatically determining the allowable range for parameters of geometric constraints. Considered are systems of distance and angle constraints on points in 3D that can be decomposed into triangular and tetrahedral subproblems, by which most practical situations in parametric and feature-based CAD systems can be represented. Our method uses the decomposition to find critical parameter values for which subproblems degenerate. By solving one problem instance for each interval between two subsequent critical values, the exact parameter range is determined for which a solution exists.

Keywords: Parametric and feature-based CAD, Geometric constraint solving, Parameter range computation

1 Introduction

In parametric and feature-based CAD systems, the shape of an object is represented by a parametric model and a 3D geometric model, e.g. a boundary representation. The parametric model imposes constraints on the entities of the geometric model, i.e. on its vertices, edges and faces. The user may quickly create variant objects by modifying parameter values of constraints corresponding to the dimensions of the object. The CAD system's constraint solver automatically modifies the geometric model to re-satisfy the constraints.

For some combinations of parameter values, however, it may not be possible to satisfy the system of constraints or to regenerate the geometric model. In general, the user does not know exactly which values are allowed and which are not [7]. In current CAD systems, the user is informed of any problem only after changing a parameter value. It would be very helpful if parameter ranges for which the CAD model is valid, could be calculated and presented to the user beforehand.

The relation between parameteric models and geometric models has been studied in the context of families of objects. A general framework for families of objects is presented by Shapiro and Vossler [15]. Basically, they associate with each model is a particular family of objects. For example, a parameter-space family represents the objects that can be obtained by modifying parameters, and cell-complex families represent the objects that can be obtained by operations on the corresponding cell-complexes, such as boundary representations. Related work such as [14] focuses on finding mappings between parameter-space families and boundary-representation families.

Our work relates to parameter-space families. Systems of geometric constraints underlie such families, and the problem of determining the parameter-space family for which the geometric constraints are satisfied, has not yet been addressed adequately. We present a method to automatically determine, given a well-constrained system of geometric constraints, the allowable range for a single parameter of the system, such that a solution exists for any value in that range.

Prior work concerning parameter range computation in geometric constraint systems or CAD is scarce.

Hoffman and Kim [7] consider a 2D problem with only horizontal and vertical line segments and distance constraints between them. They determine the range for the distance parameters such that the topology of the collection of line segments does not change. The considered constraint system is very simple, but they do make some useful observations concerning the problem in general. In particular, they observe that a system of geometric constraints in general has a large number of solutions, exponential with respect to the number of constraints in the problem, and that a different parameter range may be found for each solution. Also, they suggest that only one parameter at a time should be considered, because it will otherwise be very difficult to determine and to represent the ranges.

A solving approach for interval-based geometric constraints is presented by Joan-Arinyo and Mata [10]. This approach can also be used to find bounds for a parameter of a system of constraints such that a solution is feasible. Unlike the approach presented in [7], complex systems of geometric constraints in 2D and 3D can be solved. However, this approach cannot deal with parameter ranges that consist of several disjoint intervals, and, because it is based on sampling, it cannot determine the exact intervals.

Our approach is based on analysis of the construction plan found by a constructive geometric constraint solver, which enables us to determine exact parameter ranges. A range is computed for only one parameter, referred to as the variant parameter. We do not consider cases where several parameters are varied simultaneously.

The considered constraint systems are restricted to systems of distance and angle constraints on points in 3D that are well-constrained and decomposable into triangular and tetrahedral subproblems. These systems are representative for most practical problems in CAD. For example, such systems have been used for defining generic shapes of features and for attaching and positioning features

[16]. More in general, mappings from constraints on higher-dimensional entities to constraints on points are possible [17]. The presented method may also be useful for other applications than CAD in which geometric constraints are used.

Basically, the method involves finding those parameter values for which one or more subproblems degenerate, i.e. the critical parameter values. This is achieved by removing the constraint of which the parameter is investigated, adding other constraints that result in degenerate subproblems, and solving the new constraint systems with the geometric constraint solver. For each interval between two subsequent critical parameter values, the method determines whether the system can be solved, resulting in a single interval or a set of disjoint intervals for which the problem has a solution.

In Section 2 some background on geometric constraints and constraint solving is given. Section 3 presents the basic idea of our method for parameter range computation. The method requires that the degenerate solutions for subproblems are found, and an approach to find these is discussed in Section 4. An application of the method to a detailed example is given in Section 5. The last section discusses results and future work.

2 Geometric constraints and constraint solving

Geometric constraints are typically imposed on pairs of 2D or 3D geometric entities, such as points, lines, planes, spheres and cylinders, which are also called constraint variables in this context. These constraints are imposed on one or more variables, and reduce their relative freedom by one or more degrees, i.e. they reduce the number of translational and rotational freedoms of the objects involved. Geometric constraints with continuous parameters, such as a constraint on the distance between two points and a constraint on the angle between two lines, are also called dimension constraints. Examples of constraints with no parameters are parallelism and tangency constraints.

Systems of geometric constraints are typically solved using specialised geometric constraint solving algorithms, which use domain-specific knowledge to find solutions efficiently. Most of these solvers are so-called constructive geometric constraint solvers, which first determine a *decomposition-recombination plan* (DR-plan) for a constraint system. Such a DR-plan identifies smaller subproblems that can be solved independently, or in some particular order, the results of which can then be recombined to construct a solution for the complete system. Several DR-planning algorithms are compared and discussed in [8].

Constructive geometric constraint solvers operate in three stages. In the first stage, a DR-planner decomposes the problem into subproblems. These are also referred to as clusters, because the solutions of these subproblems are geometrically stable configurations, i.e. they have no internal degrees of freedom. In the second phase of the algorithm, subproblems are solved using fixed patterns or a general algebraic or numeric solver. In the third and final phase of the algorithm, the subproblem solutions are combined by applying rotation and translation transformations.

In general, geometric constraint systems yield a large number of solutions, exponential with respect to the number of constraints. Geometric constraint solvers therefore provide mechanisms for selecting a solution or for navigating the solution space. We are assuming that of all the solutions, the user is interested in only one of those solutions, referred to as the *intended solution*.

Finding the particular solution that meets the user's requirements, i.e. finding the intended solution, is a difficult subject for which many schemes have been proposed. For an extensive overview, see [1]. The major difficulty is that the number of solutions is generally exponential with respect to the number of constraints in the problem, and it has been shown that the general problem of finding a solution with particular properties is NP-hard [6]. Thus, it is necessary to find some compromise between the expressive power of the solution selector and the computational cost of finding the intended solution.

The most common way is to use heuristic rules to pick a single solution for each subproblem in the DR-plan such that the final solution closely matches a given prototype. For example, in some parametric CAD systems, the user first sketches geometry, and then adds dimensions. The sketch is used as a prototype, and the solver determines a solution that is similar to the prototype. This solution is considered to be the intended solution. Prototype-based heuristics have been used frequently, and the properties of the resulting solutions have been studied in detail [5, 17].

A different approach is presented in [10], where additional constraints are used to select a solution. The additional constraints declare on which side of a line a point should be, in 2D. A genetic algorithm is used to select the solution that satisfies the highest number of additional constraints.

Other geometric constraint solving approaches allow users to manually browse through a tree of solutions and to visually inspect the effect of choosing a particular branch [4, 12].

For our purpose, it is not relevant which mechanism is used to select subproblem solutions. However, we do assume that for every subproblem in the DR-plan, a single solution is selected and used throughout the process of calculating the parameter range. Stated differently, we assume that every subproblem in the DR-plan has a single intended solution which can be represented as a function of the parameters of the problem.

To simplify the notation and algorithms in this paper, the types of geometric entities that may be constrained are restricted to points in $\mathbb{R}^3$. Available constraints are the distance between a pair of points, and the angle between two intersecting line segments, defined on three points.

A geometric constraint problem is represented as a tuple $G = (V, C, \vec{x})$:

- $V = \{v_1, ...v_n\}$ is a set of point variables
- $C = \{c_1, ...c_m\}$ is a set of constraints, where $c_i \in C$ is either a distance constraint or an angle constraint
- $\vec{x} = (x_1, ..., x_m)$ is called the parameter vector, where $x_i \in \mathbb{R}$ is the value of the parameter of constraint $c_i \in C$; the space $X = \mathbb{R}^m$ represents all possible parameter vectors $\vec{x}$
- if constraint c_i is a distance constraint, then $c_i = (v_a, v_b)$, where v_a, $v_b \in V$, and the following predicate must be satisfied:

$$\|v_a - v_b\| = x_i$$

- if constraint c_i is an angle constraint, then $c_i = (v_a, v_b, v_c)$, where v_a, v_b, $v_c \in V$, and the following predicate must be satisfied:

$$\frac{(v_a - v_b)}{\|v_a - v_b\|} \cdot \frac{(v_c - v_b)}{\|v_c - v_b\|} = cos(x_i)$$

The constraint solver used in our implementation is a simple bottom-up solver, similar to the solvers presented in [4, 9], with a prototype-based solution selection scheme [17]. Constraint systems are solved as follows:

- Clusters of three points are formed by solving triangular subproblems. These subproblems involve three distances, two distances and one angle, or one distance and two angles, either given in constraints or derived from other clusters.

- Clusters of four points are formed by solving tetrahedral subproblems. These subproblems involve four points and six known distances between these points. The distances either are given in constraints or have been derived from other clusters.

- If two clusters share three points, then there are no relative degrees of freedom, and they can be merged by translating and rotating the clusters such that the shared points coincide.

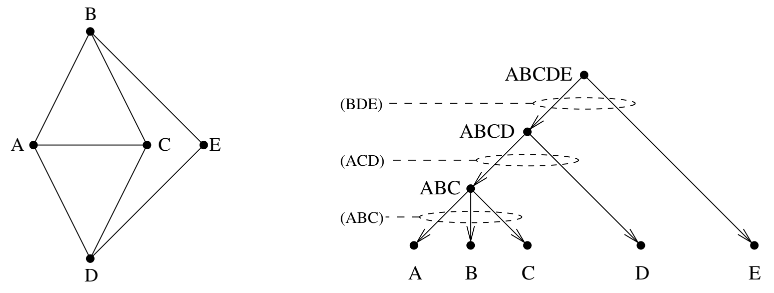
As already stated above, we are only interested in the intended solution, not in how this solution is selected. Therefore, we do not represent the selection criteria that lead to the solution. Instead, we assume that an intended solution has been defined for each subproblem in the DR-plan. The intended solutions for the subproblems and for the complete problem G are represented as functions of the parameters of the problem, as follows:

- The DR-plan is a directed acyclic graph of which the nodes represent clusters K_i. The arcs in the plan represent parent-child relations between the clusters. Clusters without children correspond to point variables $v_i \in V$. It is assumed that the system of constraints is well-constrained, and that there is a single root node in the DR-plan representing the solution for the whole system.

- For each cluster K_i in the DR-plan, there is an intended solution for the corresponding subproblem, represented by a function $s_i : X \rightarrow K_i$, such that if $s_i(\vec{x})$ exists for some $\vec{x} \in X$, then all constraints in the subgraph corresponding to cluster K_i are satisfied.

- The intended solution of G is a function $s_G : X \rightarrow Y$ such that if $s_G(\vec{x})$ exists for some $\vec{x} \in X$, then all constraints are satisfied. Here Y is the configuration space of the problem, equivalent to $\mathbb{R}^{3n}$.

A 2D example of a problem and its DR-plan are presented in Figure 1. A 3D example will be discussed in detail in Section 5.



(a) constraint graph (b) DR-plan

Figure 1: A 2D constraint problem and its DR-plan. Edges in the constraint graph represent distance constraints. Edges in the DR-plan represent merging of clusters. Shown in brackets are the subproblems to be solved in each merging step.

3 Parameter range computation

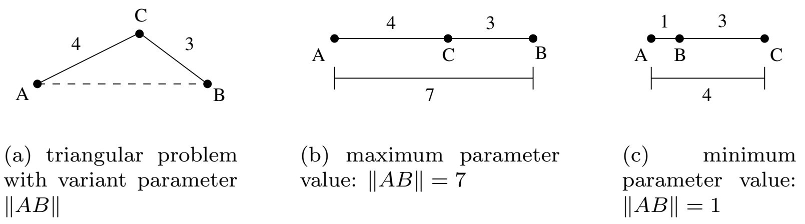
The goal of the parameter range computation is to find for a single parameter of a system of geometric constraints, such as described in the previous section, the range of parameter values for which the intended solution exists, with all other parameters kept constant. It is assumed that a geometric constraint solver is available that can determine a proper DR-plan, and find the intended solutions for the subproblems and the whole system.

More formally, the problem is the following: given a geometric constraint system $G = (V, C, \vec{x})$ with an intended solution $s_G(\vec{x})$, determine the range R_i such that if $x'_i \in R_i$, then $s_G(\vec{x'})$ exists. Here, $\vec{x'} = (x_1, \ldots, x'_i, \ldots, x_m)$ and R_i is a union of one or more disjoint intervals.

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.

The idea behind this approach is that after all critical parameter values have been determined, it is sufficient to test solvability by picking one value in each interval between two subsequent critical values, and solving the system with that parameter value, because the solvability does not change within an interval.

Because there is only one variant parameter, the system has degrees of freedom only in the subproblems that depend on this parameter. These subproblems are effectively underconstrained, 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 (see Section 4). 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 subproblem that depends on the variant parameter, either directly or indirectly.



(a) triangular problem with variant parameter $\|AB\|$

(b) maximum parameter value: $\|AB\| = 7$

(c) minimum parameter value: $\|AB\| = 1$

Figure 2: Degenerate cases for a triangular subproblem with a single variant parameter.

A subproblem depends directly on the variant parameter if it can be solved without using the solutions of other subproblems. If other subproblems must be solved first, then it depends indirectly on the parameter. Consider, for example, the constraint system in Figure 1(a) and the DR-plan for this system in Figure 1(b). From the DR-plan, we can infer that to solve this system, first the subproblem ABC is solved, then subproblem ACD. The results are then merged into cluster $ABCD$, from which $\|BD\|$ can be determined, and subsequently subproblem BDE can be solved. Thus only after solving the first two subproblems, can BDE be solved and merged with $ABCD$. If we consider the distance $\|AB\|$ to be the variant parameter, then subproblem ABC depends directly on this parameter, and subproblem BDE indirectly. For these two subproblems, we must determine the degenerate cases and the corresponding critical values of the parameter.

For a subproblem that depends directly on the variant parameter, it is relatively straightforward to determine the critical parameter values. Consider subproblem ABC, as shown in Figure 2(a). Such a triangle exists only if $\|AC\| + \|BC\| \geq \|AB\| \geq abs(\|AC\| - \|BC\|)$. The minimum and maximum value of $\|AB\|$ are its critical values. For these values, the triangle degenerates to a line segment (see Figure 2(b) and Figure 2(c)). In Section 4, methods are presented to determine the degenerate solutions for more complex subproblems.

For a subproblem that depends indirectly on the parameter, as is the case for subproblem BDE in the example in Figure 1, the relation between the critical parameter values and the degenerate solutions of the subproblem is more complex. To find these critical values, we first determine all degenerate solutions of the subproblem. Subproblem BDE is similar to the case of Figure 2, 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. The degree of freedom that is created by removing the first constraint is resolved by the newly added constraints, so the resulting system is again structurally well-constrained. For subproblem 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 subproblem. 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 $\|AB\|$ from the solution.

The backwards solving step is executed for each degenerate solution of each

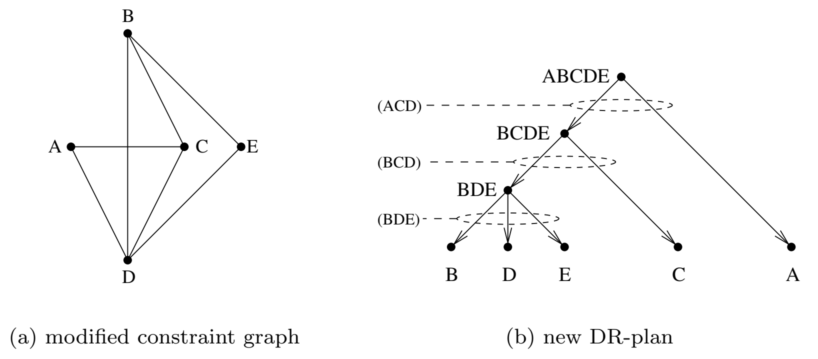


Figure 3: The problem of Figure 1 after removing the constraint on $\|AB\|$ and adding a constraint on $\|BD\|$.

subproblem that depends on the variant parameter. In each step, a new DR-plan is determined. Such a DR-plan will be similar to the original DR-plan, but subproblems that depend on the variant parameter will have disappeared, and new subproblems that depend on the currently considered degenerate solution will have appeared. A complication is that for the new subproblems, no intended solution has been defined. Thus, each backwards solving step may have several solutions, resulting in different critical parameter values.

Consider again the constraint system in Figure 1(a). By removing the constraint on $\|AB\|$, the variant parameter, and adding a constraint on $\|BD\|$ that corresponds to the distance in one of the degenerate solutions of BDE, we obtain the constraint system in Figure 3(a). The DR-plan for this system is shown in Figure 3(b). As can be seen, subproblem ABC has disappeared, subproblems ACD and BDE remain, and a new subproblem BCD has appeared. To solve this system, all three subproblems are solved independently, and then merged. However, subproblem BCD is new, and no intended solution has been defined for it. This subproblem 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 $\|AB\|$: two for the degenerate cases of subproblem ABC, two for the first degenerate case of BDE, and two for the second degenerate case of BDE.

Depending on the actual values of the other constraint parameters, the system of constraints in a backwards solving step may be overconstrained (even though it is structurally well-constrained) and therefore may have no solutions. This implies that there is no value for the variant parameter corresponding to the particular degenerate case of the particular subproblem of this step. So there is no corresponding critical value for this step either.

If the system is underconstrained (but structurally well-constrained), then it may still be possible to find a critical value for the variant parameter, but when solving the original system with this parameter value, this system will also be underconstrained. In such cases, both systems can be split into two well-constrained systems. The corresponding clusters are the same in both systems. The clusters cannot be merged rigidly, because two or more points in a cluster have the same position. In the backwards solving step, merging these clusters is not necessary for computing the critical value. However, for a valid solution of the original system, all clusters must be merged, so this critical value does not represent a valid solution, and should not be part of the parameter range. If the system has valid solutions for parameter values around this critical value, then the intervals adjacent to this critical value are half-open intervals.

For example, consider the system of Figure 4, where $\|AB\|$ is the variant parameter. Given that $\|BE\| = \|DE\|$, then for the degenerate case of triangle BDE we find $\|BD\| = 0$. By propagating this value, we can solve subsystem $ABCD$, and calculate the critical value for $\|AB\|$. When applying this critical value to the original system, we find that points B and D are coincident in cluster $ABCD$, and we cannot merge it with cluster BDE. So this critical value should not be part of the parameter range, and the intervals adjacent to it are half open intervals.

If there are several subproblems in a DR-plan for which no intended solution has been defined, the constraint solver will generate every possible combination of the solutions of these subproblems. In a worst case scenario, the number of solutions generated in the parameter range computation is exponential with respect to the number of constraints in the problem. In this case, the decomposition yields a chain of dependencies involving all subproblems in the DR-plan, and when the system is modified for backwards solving, all subproblems in the chain are replaced with new subproblems. However, we argue that for typical CAD problems, the decomposition will yield relatively short chains of dependent subproblems. Modifying the system of constraints for backwards solving will then result in local modifications of the DR-plan, and only a few new subproblems will appear in each backwards solving step. So, an acceptable number of solutions will be generated for such typical problems.

A pseudo-code algorithm for the parameter range computation is presented in Algorithm 1.

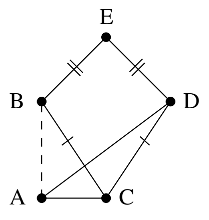


Figure 4: System with underconstrained degenerate cases.

Algorithm 1 Algorithm for parameter range computation

```
function CalcParameterRange (V,C,x,i)
  V: set of vertices
  C: set of constraints
  x: parameter vector
  i: index of the variant parameter
begin
  Y := empty set of critical values
  R := empty set of intervals
  D := DRPlan(V,C,x)
  S := set of subproblems in D dependent on x[i]
  remove c[i] from C
  for each s ∈ S:
    M := degenerate cases of s
    for each m ∈ M:
      U := constraints to fix m
      add U to C
      K := Solve(V,C,x)
      for each k ∈ K:
        calculate x[i] from k
        add x[i] to Y
        if k not underconstrained then add [x[i],x[i]] to R
      remove U from C
  add c[i] to C
  for each subsequent interval <y1,y2> in Y:
    x[i] := (y1+y2)/2
    K := Solve(V,C,x)
    if K is valid then add <y1,y2> to R
  return R
```

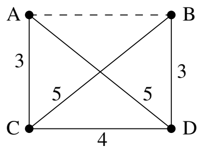
4 Degenerate subproblem solutions

Subproblems 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 subproblems, the degenerate cases must be determined.

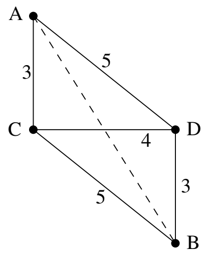
Generally, the term degenerate refers to limiting cases where a class of objects changes its nature to become another, usually simpler class [18]. For example, a triangle can degenerate to (three points on) a line. In general, the solutions of subproblems with one or more degrees of freedom form a continuous class. We use the term degenerate solution to refer to solutions that form the boundary of such a class.

As discussed in Section 2, the subproblems considered in this paper are triangular and tetrahedral subproblems with distance and angle constraints.

The simplest triangular subproblem, involving three distance constraints, has already been illustrated in Figure 2. The distance between A and B is unknown, resulting in one degree of freedom. As already stated in Section



(a) degenerate case corresponding to minimum value of $\|AB\|$



(b) degenerate case corresponding to maximum value for $\|AB\|$

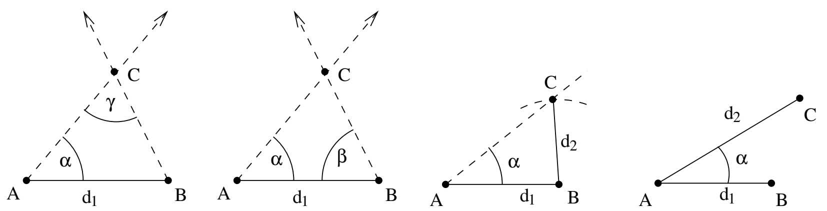
Figure 5: Degenerate cases of the tetrahedral problem.

3, for this subproblem there are two degenerate solutions, corresponding to a minimum and maximum value, $\|AB\| = 1$ and $\|AB\| = 7$ respectively. If $\|AB\|$ is increased above the maximum or decreased below the minimum, there is no solution for the problem in $\mathbb{R}^2$.

Note that the problem above is defined in $\mathbb{R}^2$, but the minimum and maximum values of $\|AB\|$ can be computed on the real line. Effectively, the space in which the problem is considered is reduced from $\mathbb{R}^2$ to $\mathbb{R}$. For 3D problems, a similar reduction from $\mathbb{R}^3$ to $\mathbb{R}^2$ can be used to find the degenerate cases.

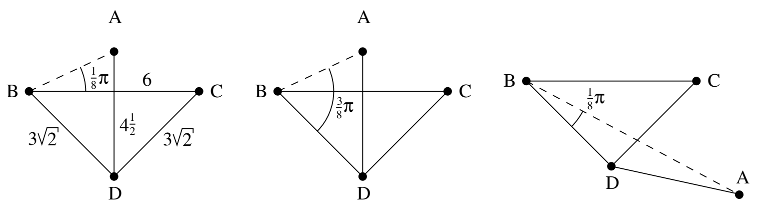
The basic 3D subproblem is the tetrahedral problem involving only distance constraints, as illustrated in Figure 5. The distance $\|AB\|$ is considered to be the variant parameter. With five known distances, the problem is underconstrained in 3D. However, in 2D the problem is well-constrained and has two solutions, corresponding to $\|AB\| = 4$ and $\|AB\| = 2\sqrt{13}$, shown in Figures 5(a) and 5(b) respectively. These solutions correspond to the degenerate cases of the 3D problem.

Angle constraints need to be considered only in 2D subproblems. There are four cases of triangular subproblems with angle constraints. These are labelled '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 Figure 6.



(a) 'aad' case (b) 'ada' case (c) 'add' case (d) 'dad' case

Figure 6: Triangle cases for solving angle constraints



(a) problem (b) first solution (c) second solution

Figure 7: Degenerate solutions of a tetrahedral subproblem with an angle constraint.

If an angle is the variant parameter, the triangular problems in which the parameter is directly involved may again degenerate. Critical values for angles in these subproblems are given in Table 1. Note that all angle parameters are in the range $[0, \pi]$.

When solving degenerate cases of tetrahedral subproblems with angle constraints, the above triangular subproblems also occur. Consider the problem in Figure 7(a). Two degenerate solutions are found, again by solving the system in 2D. Note that if we adhere strictly to the triangular patterns in Figure 6, then the constraint solver must be able to propagate angles between triangular subproblems. In this example, the solver determines that $\angle CBD = \frac{1}{4}\pi$, and $\angle ABD = \angle CBD + \angle ABC = \frac{3}{8}\pi$ (Figure 7(b)), or $\angle ABD = \angle CBD - \angle ABC = \frac{1}{8}\pi$ (Figure 7(c)), depending on which side of line BC point A is located. Angle propagation can be implemented in constructive solvers, as described in [13], and is supported by many other constraint solving approaches too.

As described in Section 3, during the backwards solving phase, the constraint corresponding to the variant parameter is removed from the system of constraints, resulting in an underconstrained system. To make it well-constrained again, constraints are added to the system that represent the degenerate solutions of subproblems.

If the subproblem under consideration has a single degree of freedom, as we have assumed until now, then it has a finite number of degenerate solutions; in the cases considered here, there are always two solutions. We can first determine these degenerate solutions, and then, for each solution, add a single constraint to the underconstrained system. This constraint may be any distance or angle constraint that is not in the original system, and the distance or angle value can

Table 1: Critical angle values.

case	variable	min	max	
aad	α	0	$\pi - \gamma$	
	γ	0	$\pi - \alpha$	
ada	α	0	$\pi - \beta$	
	β	0	$\pi - \alpha$	
add	α	0	$sin^{-1}(\frac{d_2}{d_1})$	$d_1 \geq d_2$
			π	$d_1 < d_2$
dad	α	0	π	

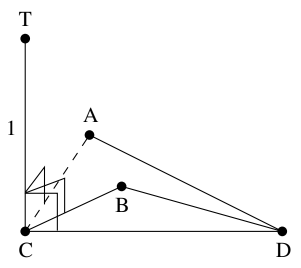


Figure 8: Adding an extra point T and constraints to $ABCD$ to represent degenerate solutions.

be calculated from the degenerate solution.

The above approach fails if a subproblem has more than one degree of freedom, because it will then have an infinite number of degenerate solutions, so we cannot generate them one by one. This situation may occur if a subproblem depends indirectly on the variant parameter via two or more independent subproblems. In other words, the variant parameter induces more than one degree of freedom in the subproblem, via several paths in the DR-plan.

In general, to deal with this situation, we must somehow represent the whole range of degenerate solutions. We can do this by adding constraints to the subproblem such that it is still underconstrained, but all the solutions are degenerate. Combined with all the other constraints, the system should be structurally well-constrained, and we can solve for the critical value of the variant parameter.

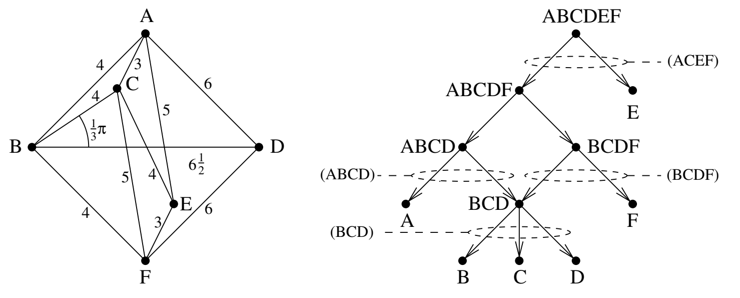
For example, to find the degenerate solutions of a tetrahedral subproblem with more than one degree of freedom, constraints are added that force all four points into a plane. With only distance and angle constraints, this can be realized by adding a new point variable, and constraints as shown in Figure 8. In this example, a tetrahedral subproblem $ABCD$ with only four known distances is considered, i.e. with two degrees of freedom in $\mathbb{R}^3$. By adding a point T and angle constraints between T and the other points, the latter points are forced into a plane and only one degree of freedom remains in this subproblem. In the backwards solving step we will find a finite number of solutions, and thus a finite number of critical values, because the system is well-constrained. We know that the solutions represent degenerate cases of $ABCD$, because of the added constraints.

5 Example constraint problem

The example 3D constraint problem considered in this section is presented in Figure 9(a). To solve this problem, the DR-plan given in Figure 9(b) is used.

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 DR-plan, we infer that the following subproblems are dependent on $\|BC\|$: BCD, $ABCD$, $BCDF$ and $ACEF$. For these subproblems, we must find the degenerate solutions and the corresponding critical values of the variant parameter.

Subproblem BCD, which depends directly on $\|BC\|$, has a solution for any value of $\|BC\| \geq 0$. Note that $\|BC\|$ may not be negative, by the definition of



(a) Constraint system (b) DR-plan

Figure 9: 3D example problem

the distance constraint given in Section 2, thus we consider 0 to be a critical value.

Subproblem $ABCD$, shown in Figure 10, is similar to the case presented in Figure 7(a), i.e. it differs only in the names of the variables and the values of the parameters. In this case, we find two degenerate solutions, corresponding to $\|BC\| \approx 1.00$ and $\|BC\| \approx 6.97$. This subproblem also depends directly on the parameter, thus these values are its critical parameter values.

Subproblem $BCDF$ is identical to $ABCD$, therefore no new critical values are found.

Subproblem $ACEF$ is equivalent to the case presented in Figure 5. It has two degenerate solutions, corresponding to $\|AF\| = 4$ and $\|AF\| = 2\sqrt{13}$. This subproblem depends indirectly on the variant parameter, therefore we modify the system of constraints by removing the constraint on $\|BC\|$, and adding a constraint $\|AF\| = 4$ or $\|AF\| = 2\sqrt{13}$. The modified constraint system is shown in Figure 11(a), and the new DR-plan for this problem is shown in Figure 11(b). This DR-plan is used in the backwards solving steps to find the critical parameter values for both degenerate cases of the subproblem. In the DR-plan, there are two new subproblems, $ABDF$ and $ABCF$, for which no intended solution has been defined. Each has two solutions, which, when combined, result in four solutions for each degenerate case of $ACEF$. So, in total there are eight solutions for this subproblem, but it turns out that, due to symmetry, there are only four different critical values for $\|BC\|$, approximately: 1.49, 1.90, 2.50 and 6.76.

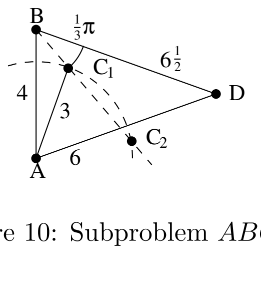
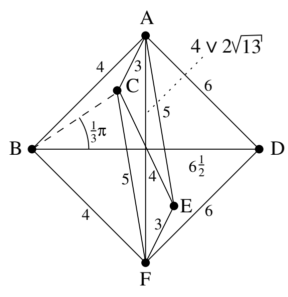
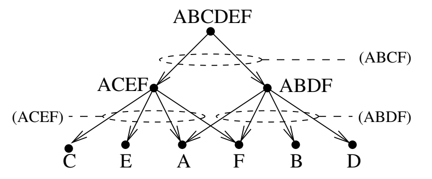


Figure 10: Subproblem $ABCD$.



(a) Constraint system



(b) DR-plan

Figure 11: Modified constraint system corresponding to the degenerate solutions of subproblem $ACEF$.

Now that we have determined all critical values for $\|BC\|$, 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 $\|BC\| \in [1.49, 1.90] \cup [2.50, 6.76]$.

6 Conclusions

An approach has been presented to calculate the range of values for a single parameter of a geometric constraint system, such that a solution exists for any value in that range. Systems with distance and angle constraints on points in 3D that are well-constrained and decomposable into triangular and tetrahedral subproblems have been considered. The approach uses the decomposition of the constraint system to find the critical parameter values for which subproblems degenerate, and computes the range from these values.

The worst-case complexity of the method is exponential. However, we believe that such worst cases will not occur in practice, because parameters will typically effect the constraint system only locally. Some improvements on the performance of the algorithm may be possible. For example, it may not be necessary to compute all geometric solutions in the backwards solving phase, because not all critical values contribute to the actual intervals of the parameter range. For complex models, it would therefore be useful to research optimization strategies.

For models that can not be represented with the constraint systems considered here, it would be interesting to extend the approach to more general constraint systems, with other geometric entities, other constraints, and other decompositions. We believe that the basic approach, i.e. finding critical values by introducing degenerate cases of subproblems in the constraint system, can be applied to any decomposable system, but more general methods are needed to determine degenerate cases for a wider range of subproblems.

A topic of our future research will be to also consider the geometric realization of CAD models, instead of only the parametric definition as has been done here. In particular, we would like to determine parameter ranges for semantic feature models [2], which, in addition to geometric constraints, contain topological constraints. Parameter range computation in this context may be based on the method presented here, with additional critical values corresponding to topological changes in the underlying cellular model.

For some applications, it would be useful to consider several variant parameters simultaneously. For example, the range of a pair of parameters could be presented to a user as a 2D graphic. Considering even more variant parameters at a time is useful for automatic model adjustment [11] and model optimisation algorithms. Our method cannot be easily generalised to solve problems with several simultaneously varying parameters, but it may be useful to help reduce the search space of such problems.

Methods for calculating ranges of parameters can be of great help in parametric and feature-based CAD systems, but also in other applications in which geometric constraints are used. Changing a parameter is a common operation in such applications, and knowing the allowable range of a parameter beforehand can prevent this from becoming a trial-and-error process.

Acknowledgements

H.A. van der Meiden's work is supported by the Netherlands Organisation for Scientific Research (NWO).

References

[1] B. Bettig and J. Shah. Solution selectors: a user-oriented answer to the multiple solution problem in constraint solving. *J Mech Design*, 125(3):443–451, 2003.

[2] R. Bidarra and W. F. Bronsvoort. Semantic feature modelling. *Comput Aided Des*, 32(3):201–225, 2000.

[3] R. Bidarra, K. J. de Kraker, and W. F. Bronsvoort. Representation and management of feature information in a cellular model. *Comput Aided Des*, 30(4):301–313, 1998.

[4] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Comput Aided Des*, 27(6):487–501, 1995.

[5] C. Essert-Villard, P. Schreck, and J.-F. Dufourd. Sketch-based pruning of a solution space within a formal geometric constraint solver. *Artif Intell*, 124(1):139–159, 2000.

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

[7] C. M. Hoffmann and K.-J. Kim. Towards valid parametric CAD models. *Comput Aided Des*, 33(1):81–90, 2001.

[8] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for geometric constraint systems, Part I: performance measures for CAD. *J Symb Comput*, 31(4):376–408, 2001.

[9] C. M. Hoffmann and P. J. Vermeer. Geometric constraint solving in $\mathbb{R}^2$ and $\mathbb{R}^3$. In D. Du and F. Hwang, editors, *Computing in Euclidean Geometry, Second Edition*, pages 266–298, Singapore, 1995. World Scientific Publishing.

[10] R. Joan-Arinyo and N. Mata. Applying constructive geometric constraint solvers to geometric problems with interval parameters. *Nonlinear Anal-Theor*, 47(1):213–224, 2001.

[11] A. Noort and W. F. Bronsvoort. Enforcing model validity by automatic adjustment. *J Inf Sci Eng*, 1(4):311–319, 2001.

[12] J. Oung, M. Sitharam, B. Moro, and A. Arbree. FRONTIER: fully enabling geometric constraints for feature-based modeling and assembly. In D. C. Anderson and K. Lee, editors, *Proceedings of Solid Modeling '01, Sixth ACM Symposium on Solid Modeling and Applications, June 4-8, Ann Arbor, USA*, pages 307–308, New York, 2001. ACM Press.

[13] D. Podgorelec. A new constructive approach to constraint-based geometric design. *Comput Aided Des*, 34(11):769–785, 2002.

[14] S. Raghothama and V. Shapiro. Consistent updates in dual representation systems. *Comput Aided Des*, 32(8-9):463–477, 2000.

[15] V. Shapiro and D. L. Vossler. What is a parametric family of solids? In C. M. Hoffmann and J. R. Rossignac, editors, *Proceedings of the Third ACM/IEEE Symposium on Solid Modelling and Applications, May 17-19, Salt Lake City, USA*, pages 43–54, New York, 1995. ACM Press.

[16] E. van den Berg, H. A. van der Meiden, and W. F. Bronsvoort. Specification of freeform features. In G. Elber and V. Shapiro, editors, *Proceedings of the Eight ACM Symposium on Solid Modeling and Applications, June 16-20, 2003, Seattle, Washington, USA*, pages 56–64, New York, 2003. ACM Press.

[17] H. A. van der Meiden and W. F. Bronsvoort. An efficient method to determine the intended solution for a system of geometric constraints. *Int J Comput Geom*, 15(3):279–298, 2005.

[18] E. W. Weisstein. Degenerate. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Degenerate.html, 2004.

Vitae

Hilderick A. van der Meiden is PhD student at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He graduated in computer science at the same university in 2004. His research interests include feature modeling, constraint solving, and semantics of families of objects.

Willem F. Bronsvoort is associate professor CAD/CAM at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He received his MSc degree in computer science from the University of Groningen in 1978, and his PhD degree from Delft University of Technology in 1990. His main research area is feature modeling, in particular semantic feature modeling, multiple-view feature modeling, freeform feature modeling, and mesh generation from feature models. He has published numerous papers in international journals, books and conference proceedings, is on the editorial board of several journals, and has served as co-chair and member of many program committees of conferences.