An efficient method to determine the intended solution for a system of geometric constraints

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

August 29, 2005

Abstract

The number of solutions of a geometric constraint problem is generally exponential to the number of geometric elements in the problem. Finding a single intended solution, satisfying additional criteria, typically results in an NP-complete problem. A prototype-based selection scheme is presented here that avoids this problem. First, a resemblance relation between configurations is formally defined. This relation should be satisfied between the intended solution and a prototype configuration. The resemblance relation is in our approach satisfied by applying selection rules to subproblems in a bottom-up solving approach. The resulting solving algorithm is polynomial, because the selection rules are not used as search heuristic, but to unambiguously select a single solution such that no backtracking search is needed. For many applications, in particular CAD, this solution is both meaningful and intuitive.

Keywords:Geometric constraint solving; intended solution; feature modelling

1 Introduction

Geometric constraints are used in many applications to define parameterised geometric models. To generate instances of a model, the corresponding geometric constraint problem must be solved. In general, such a problem has a large number of solutions. Selecting a solution is an important issue in geometric constraint solving, known as the multiple-solution problem or the root identification problem. The selected solution should satisfy additional, often implicit, criteria, and is generally referred to as the intended solution.

Several selection criteria for the intended solution have been proposed by various authors. Bettig and Shah[1] provide an extensive overview and argue that declarative solution selectors are the most flexible and powerful approach. The user specifies additional constraints, or solution selectors, that narrow down the number of solutions until a single intended solution remains. Eleven basic selectors for various types of geometry have been identified, e.g. to specify that a point must be on a particular side of a curve.

A declarative solution selection scheme is also presented by Joan-Arinyo et al.[8] A genetic algorithm is used to select a solution that satisfies a number of additional constraints which determine on which side of a line a point should be. Drawbacks of this approach are that a large number of such additional constraints must be supplied, and that it is difficult to find optimal settings for the genetic algorithm.

The problem with the declarative approach in general, is that no efficient and complete algorithm for selecting the intended solution is known. The number of solutions for geometric constraint problems is generally exponential to the number of geometric elements. Finding all real solutions has been shown to be NP-complete[6]. Adding extra constraints or domain knowledge to select a solution is very likely to result in an NP-complete problem also[2].

The problem of finding all real solutions in an efficient way has been addressed using homotopic continuation techniques by Durand and Hoffmann[3]. First, the generic solution is represented as a system of equations. A representative set of solutions, called the start system, is then determined. Using numerical path-tracking techniques, a set of homotopy paths is computed, which describe how the start system is transformed when parameters are changed. Several homotopic continuation techniques have been effectively used to solve the octahedral problem involving points and planes. However, the number of real solutions is still exponential to the number of elements in the problem, and it is not clear how these techniques can be used to select a single intended solution.

For interactive applications, exponential time complexity for finding the intended solution is not acceptable. In particular, parametric and feature-based CAD models yield large systems of geometric constraints, which must be solved frequently. Also, such models often require that 3D problems can be solved, for which the number of solutions is even higher than for 2D problems.

The common alternative to the declarative approach is the heuristic approach. Several solving schemes[6, 2, 11] rely on heuristic rules based on the relative position and orientation of the geometry in a prototype, i.e. a sketch drawn by the user. By applying these heuristics, the intended solution may be found without searching through an exponential number of options. However, the heuristics do not always lead to a solution. A

complete algorithm, i.e. an algorithm that always finds a solution if there is one, still requires exponential search.

Essert-Villard et al.[4] present a formal framework for sketch-based heuristics, based on homotopy theory. They define a special S-homotopy relation for configurations that respect geometric constraints of a constraint system S. By selecting a root for each triangular subproblem using local criteria, a solution is obtained that is S-homotopic to the sketch. In this way, the tree of possible solutions is pruned. Still, several branches may remain, resulting in different solutions. Some additional selection process is thus needed.

Some solving schemes allow the user to explore the tree of solutions when the search heuristics do not result in a single solution, e.g. the work of Bouma et al.[2] and the FRONTIER system[10]. However, such a procedure can be cumbersome for large systems, or when systems are changed and need to be solved again.

A general framework for constructive geometric constraint solvers is presented by Joan-Arinyo et al.[9]. This framework includes an enumeration scheme for the solutions of a system of constraints, based on the maximum number of roots of the equations of the subproblems of the system. A solution instance is obtained by specifying parameter values and an index number to select one of the solutions. The previous approaches, and our own, all fit in this framework.

The differences between the various approaches boil down to how much control the user has in the solution selection process. We have taken a new view on the problem, formulating generic requirements for intended solutions.

In previous work on freeform feature modelling[12], we have identified several requirements for freeform feature classes and their instances. To define a feature class, a prototype feature is created first, which is then parameterised using geometric constraints. For users of the feature class, who generally have no knowledge of the constraints used for the parameterisation, discontinuities in the behaviour of the feature are unexpected and not desirable. By using the prototype in the solving process, feature instances are determined which respond to changes in parameters in a predictable way.

Similar requirements can also be applied to select the intended solution of geometric constraint problems in general. The intended solution should satisfy the following requirements: (1) the solution is a continuous function of the parameters of the problem; and (2) the solution uniquely resembles a given prototype. The prototype that is used, is simply a configuration of all the geometric variables in the problem. These requirements are similar to requirements proposed by Essert-Villard et al.[4], but our interpretation will always lead to a single unambiguous solution.

By allowing the user to explicitly declare solution selectors, it is likely that the selected solution does not satisfy the above-mentioned desirable

properties. We therefore advocate an approach where the selectors are determined by the decomposition algorithm and the prototype. To the user, the solution selection process is more or less transparent, but the given requirements ensure that there is an intuitive relation between the solution found by the solver and the prototype.

Although, in this approach, the user does have less modelling freedom than in the declarative approach, we believe this is not a major problem, and our approach has important advantages. In particular, it prevents the user from making discontinuous or invalid parameterisations. Also, we believe that users can quickly develop an intuitive feeling for the behaviour of the constraint systems defined with this approach, and thus learn to make good parameterisations.

In this paper, we will show that the intended solution, satisfying the requirements stated above, can be found using a simple bottom-up solving approach, such as presented by Bouma et al.[2], extended to 3D tetrahedral-decomposable problems. We present selection rules for triangular and tetrahedral subproblems, and a formal proof to show that these selection rules result in a uniquely-defined solution that satisfies the requirements for the intended solution. If no solution is found with these selection rules, then the intended solution does not exist, so there is no need to search for other solutions. The resulting algorithm is polynomial, and may be used for relatively simple, but large and practical geometric constraint problems.

In Section 2, the geometric constraint problem is defined and the requirements for the intended solution are formally stated. In Section 3, our constraint solving approach is presented. In Section 4, subproblems and their solutions are analysed. In Section 5, we show that by combining subproblem solutions, a uniquely-defined solution is obtained that satisfies the requirements for the intended solution. Finally, in Section 6, we present our conclusions.

2 Problem definition

The geometric constraint problem considered here involves distance constraints and angle constraints on points in 3-dimensional Euclidean space. The constraints are defined as:

Definition 1 *A distance constraint* $d(p_a, p_b, \delta)$ *on point variables* $p_a, p_b \in \mathbb{R}^3$ *with parameter* $\delta \in \mathbb{R}$*, is defined as:*

$$\|p_a - p_b\| = \delta \qquad \delta \geq 0$$

Definition 2 *An angle constraint* $a(p_a, p_b, p_c, \alpha)$ *on point variables* p_a, p_b, $p_c \in \mathbb{R}^3$ *with parameter* $\alpha \in \mathbb{R}$, *is defined as:*

$$\frac{p_a - p_b}{\|p_a - p_b\|} \cdot \frac{p_c - p_b}{\|p_c - p_b\|} = cos(\alpha)$$

The angle α is referred to as the unsigned angle; $a(p_a, p_b, p_c, \alpha)$ is equivalent to $a(p_a, p_b, p_c, -\alpha)$. Next, we define our geometric constraint problem:

Definition 3 *A geometric constraint problem* $G(x) = (V, C)$ *consists of* n *point variables* $V = \{v_1 \dots v_n\}$ *and* m *constraints* $C = \{c_1 \dots c_m\}$. *A constraint* $c_i \in C$ *is either a distance constraint or an angle constraint with parameter* $x_i \in \mathbb{R}$, $x = (x_1, ...x_m)$.

The parameters of the problem are represented by the parameter vector $x = (x_1, ...x_m) \in X$, where $X = \mathbb{R}^m$ is referred to as the *parameter space* of the problem. A solution for the problem is a *configuration* of points, i.e. a vector $y = (y_1, ..., y_n)$, $y_i \in \mathbb{R}^3$. The space $Y = \mathbb{R}^{n \times 3}$ is referred to as the *configuration space* of the problem.

Definition 4 *The general solution for a geometric constraint problem* $G(x)$ *is a set of solutions as a function of the parameters of the problem:*

$$S(x) = \{s; s \therefore G(x)\} \quad s \in Y$$

where $s \therefore G(x)$ *means that* s *satisfies the constraints in* $G(x)$.

In our approach, one particular solution in the general solution, i.e. the intended solution, is unambiguously determined using a prototype configuration $p \in Y$. The intended solution is represented by a function of the parameters of the problem and the prototype configuration:

Definition 5 *The intended solution for a geometric constraint problem* $G(x)$ *and a prototype* p *is a function* $s : X \times Y \rightarrow Y$ *such that (1)* $s(x, p) \in S(x)$ *and (2)* $s(x, p) \equiv_G p$, *where* $\equiv_G \subseteq Y \times Y$ *is called the resemblance relation for* G.

The resemblance relation is satisfied if the intended solution 'looks like' the prototype. The specific relation that we use for this is elaborated in the following sections, and will turn out to be an equivalence relation. The resemblance relation therefore partitions the configuration space into a number of equivalence classes. The intended solution is, by definition, the solution in the same equivalence class as the prototype. This is illustrated in Fig. 1.

A simple constraint solving algorithm is presented in this paper that implements the above ideas. We will show that the solutions determined by this algorithm have, in particular, the following properties:

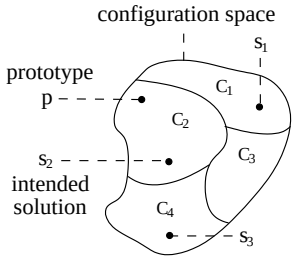


Figure 1: The configuration space is partitioned into a number of equivalence classes C_i. The intended solution (s_2) is the solution that is in the same equivalence class (C_2) as the prototype (p).

Property 1 *$x \rightarrow s(x,p)$ is a continuous mapping from X' to Y, for any $p \in Y$, where $X' = \{x^* : s(x^*,p)$ exists $\}$.*

Property 2 *Different solutions for the same parameter value do not resemble each other, i.e. they are in different equivalence classes. Formally: $\forall x \in X, \forall s_1, s_2 \in S(x) : s_1 \neq s_2 \Rightarrow s_1 \not\equiv_G s_2$.*

Property 3 *Every equivalence class is a connected set, i.e. for any two configurations $y, z \in Y$ such that $y \equiv_G z$, there exists a continuous function $f : [0,1] \rightarrow Y$ such that $f(0) = y$, $f(1) = z$ and $\forall \phi \in (0,1) : f(\phi) \equiv_G y$ (and $f(\phi) \equiv_G z$).*

These are in fact the desired properties introduced in Section 1, i.e. solutions with these properties are predictable from an end-user's viewpoint. Property 1 states that there is a continuous relation between the parameters of the constraints and the intended solution. Property 2 ensures that a solution can be selected unambiguously; given any combination of a prototype and a parameter vector, there is at most one solution. Property 3 states that all configurations in an equivalence class are connected; configurations that resemble each other are geometrically 'close' to each other. We believe this to be a reasonable interpretation of the intuitive meaning of resemblance.

Our approach is not complete in the sense that a solution for a geometric constraint problem is always found. For some combinations of the parameter and prototype, there may not be an intended solution, even though a real solution exists. Consider, for example, Fig. 2. Suppose we have defined a prototype such that for parameter $d = 4$ we find the solution shown on the left. For any parameter value $d \geq 3$, the solution is a continuous function of d. If, however, we instantiate the problem with parameter $d = 2$, there is no solution in the same equivalence class, i.e. there is no solution for $d = 2$ that can be reached continuously from the previous solutions. Thus, the configuration shown on the right is a real solution, but it is not the intended solution.

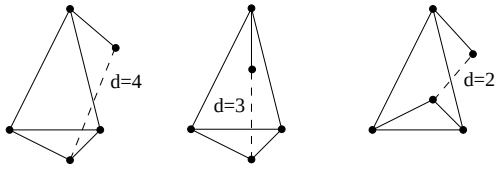


Figure 2: A constraint problem with a distance parameter d. The intended solution exists only for $d \geq 3$. For $d = 2$ there is a solution, but it is not the intended solution.

The approach is, however, complete in the sense that it always finds the intended solution if it exists. In the following section, a simple constraint solving algorithm is presented that will find this intended solution in polynomial time. In the remaining sections of the paper, we will show that the algorithm has the properties given above.

3 Solving approach

The constraint solving method presented here is based on the constructive approach to constraint solving. In this approach, a constraint problem is decomposed into subproblems of which the solution can be determined relatively easily. The solutions to these subproblems are then combined to construct a solution for the original problem.

Subproblem solutions are represented by clusters of geometric elements of which the relative position and orientation are known. A cluster is a rigid body that can be translated and rotated, while remaining a solution. Clusters can be merged if they share a number of elements such that no degrees of freedom are left between these clusters. Merging a pair of clusters involves translating and rotating the clusters such that their shared elements coincide. By merging two clusters, the relative position and orientation of geometric elements in both clusters is fixed. This information may then be used to solve remaining subproblems. Subproblems must be solved and merged in such an order that information can be propagated from the known parameter values, through several subproblems, to a final solution. The geometric constraint problem has been completely solved when all geometric elements have been merged into a single cluster.

This approach has been used to solve 2D problems involving points and lines[2, 6], and 3D problems involving points and planes[7]. A correctness proof for a cluster rewriting algorithm has been presented by Fudos and Hoffmann[5].

The constraint solving approach presented here is similar, except that the only geometric primitives considered are points in 3D Euclidean space. The subproblems considered are mathematically the same as those in the work of Hoffmann and Vermeer[7], except for their formulation. The algorithm solves triangular and tetrahedral subproblems, in which three respectively

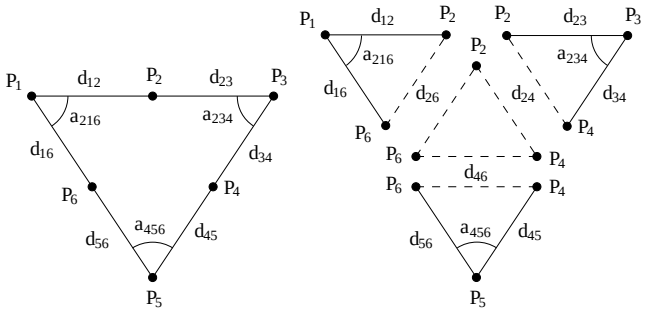


Figure 3: A 2D problem with distances and angles between points (left), and its decomposition (right)

four points are involved. All angle constraints are solved in the context of a triangular subproblem. Tetrahedral clusters are created when all six distances between the four points in the cluster are known. A pair of clusters can be merged if they share three point variables which are not on a line or in a single point. Only clusters of four points or more need to be merged. Triangular clusters are only used for propagating known distances, which are then used to determine tetrahedral clusters.

This solving approach is illustrated with a 2D example, in which triangular subproblems are solved and merged, shown in Fig. 3. The example problem involves six points, P_i, six distance constraints, d_{ij}, and three angle constraints, a_{ijk}. The problem is decomposed into four clusters shown on the right. Clusters $P_1P_2P_6$, $P_2P_3P_4$ and $P_4P_5P_6$ can be determined independently, using distances and angles specified in the problem. Cluster $P_2P_4P_6$ can be determined using distances derived from the first three clusters (d_{26}, d_{24} and d_{46}). The first three clusters each share two points with the fourth cluster, and they can be merged by rotating and translating them such that the shared points coincide.

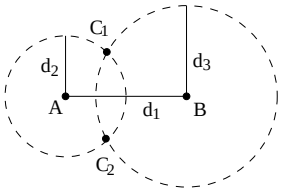
Each subproblem that is solved may have more than one solution, and a single solution must be selected in each case. To do this, for each type of subproblem a resemblance relation will be defined. The solution selection process compares every solution with the subconfiguration of the prototype that involves only the variables of the subproblem. The solution that resembles the subconfiguration is selected and used further in the solving process. The solver automatically determines the appropriate selection rules and subconfigurations of the prototype, so there is no need for user interaction during the solution selection process.

If a subproblem cannot be solved, or if solutions cannot be merged, the algorithm will not backtrack to try other subproblem solutions. Thus the algorithm does not find all solutions for the geometric problem, but it does find the intended solution if it exists, and, in addition, it runs in polynomial time.

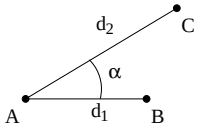
In the following section, we will define a resemblance relation for each type of subproblem, and we will show that these relations satisfy the properties introduced in Section 2.

4 Subproblem analysis

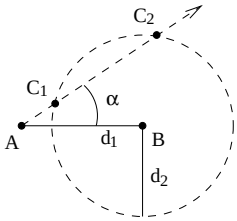
The basic subproblems that need to be solved are 'triangular' subproblems. A cluster of three points is determined using known distances and angles between these points. There are five well-constrained cases to consider, illustrated in Fig. 4. Each case is identified by a 3-letter code, where 'd' stands for a known distance, and 'a' stands for a known angle. The order of these letters indicates the configuration of the known distances and angles. Because three points are always in a plane, these problems are solved in 2D.



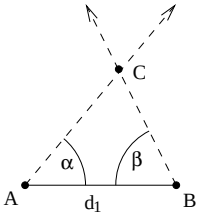
(a) Case 'ddd'



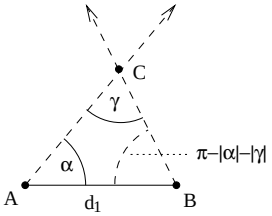
(b) Case 'dad'



(c) Case 'dda' (= 'add')



(d) Case 'ada'



(e) Case 'daa' (= 'aad')

Figure 4: Triangular subproblems

Case 'ddd': three distances are known, see Fig. 4(a).
constraints: $d(A,B) = d_1$, $d(A,C) = d_2$, $d(B,C) = d_3$
solution: $A = (0,0)$, $B = (d_1, 0)$, $C = (x, y)$
$x = \frac{(d_2)^2 - (d_3)^2 + (d_1)^2}{2d_1}$, $y = \pm\sqrt{(d_2)^2 - x^2}$

Case 'dad': two distances and the enclosed angle are known, see Fig. 4(b).
constraints: $d(A,B) = d_1$, $a(B,A,C) = \alpha$, $d(A,C) = d_2$
solution: $A = (0,0)$, $B = (d_1,0)$, $C = (x,y)$
$x = d_2 \cdot cos(\alpha)$, $y = \pm d_2 \cdot sin(\alpha)$

Case 'dda' (= 'add'): two distances and one adjacent angle, see Fig. 4(c).
constraints: $d(A,B) = d_1$, $d(B,C) = d_2$, $a(B,A,C) = \alpha$
solution: $A = (0,0)$, $B = (d_1,0)$, $C = (x,y)$
$x = t \cdot cos(\alpha)$, $y = \pm t \cdot sin(\alpha)$
$t = q \pm \sqrt{4q^2 - 4r}, \quad t \geq 0$
$q = d_1 \cdot cos(\alpha), \quad r = (d_1)^2 - (d_2)^2$

Case 'ada': one distance and two adjacent angles are known, see Fig. 4(d).
constraints: $a(C,A,B) = \alpha$, $d(A,B) = d_1$, $a(A,B,C) = \beta$
solution: $A = (0,0)$, $B = (d_1,0)$, $C = (x,y)$
$x = t \cdot cos(\alpha)$, $y = \pm t \cdot sin(\alpha)$
$t = \frac{d_1 \cdot cos(\beta)}{sin(\alpha) \cdot cos(\beta) + sin(\beta) \cdot cos(\alpha)}$

Case 'daa' (= 'aad'): one distance, one adjacent angle and the opposite angle are known, see Fig. 4(e).
constraints: $a(C,A,B) = \alpha$, $a(B,C,A) = \gamma$, $d(A,B) = d_1$
solution: $A = (0,0)$, $B = (d_1,0)$, $C = (x,y)$
$x = t \cdot cos(\alpha)$, $y = \pm t \cdot sin(\alpha)$
$t = \frac{d_1 \cdot cos(\beta)}{sin(\alpha) \cdot cos(\beta) + sin(\beta) \cdot cos(\alpha)}, \quad \beta = \pi - |\alpha| - |\gamma|$

For each subproblem, the intended solution is represented by a function $s(x,p)$, where x is the parameter vector containing the parameters of the constraints in the subproblem, and p is the subconfiguration of the prototype for the point variables involved in the subproblem. The prototype is used as a solution selector in this function, i.e. depending on p, a particular solution is returned. The selected solution should have a resemblance relation with the prototype (see Definition 5), so that it is the intended solution.

For each subproblem case, a specific resemblance relation will be defined in this section. For the 'ddd', 'dad', 'ada' and 'daa' cases, the same resemblance relation is used, denoted by $\equiv^*$. For the 'dda' case, the resemblance relation is denoted by $\equiv^{dda}$, and for the tetrahedral subproblem by $\equiv^{tet}$. The implementation of the different functions $s(x,p)$ is trivial given these resemblance relations. For our analysis it is sufficient to note that, depending on the specific subproblem case, $s(x,p) \equiv^* p$, $s(x,p) \equiv^{dda} p$ or $s(x,p) \equiv^{tet} p$.

The intended solution and resemblance relation for each subproblem must satisfy the properties defined in Section 2. In the following, we will define the resemblance relations and prove the required properties in each case.

First note that for each of the triangle cases described above, there are mirror-symmetrical solutions, i.e. solutions for $y > 0$ and $y < 0$.

Lemma 1 *Mirror-symmetrical solutions in 2D are congruent in 3D.*

Proof: The plane in which these solutions are constructed is arbitrary. In 3D, the solutions may be rotated 180° about the symmetry axis, so they are congruent.

Lemma 2 *For the 'ddd', 'dad', 'ada' and 'daa' cases, $x \rightarrow s(x, p)$ is a continuous mapping, satisfying Property 1.*

Proof: For each case, there are only two solutions, which are congruent in 3D (Lemma 1). In the solution formulas given above, the sign of y may be chosen arbitrarily. In each case, the solution is a continuous function of the constraint parameters, and thus satisfies Property 1.

Due to Lemma 1, for all the above cases except the 'dda' case, there is only one solution. The definition of the resemblance relation for these cases is therefore trivial:

Definition 6 *For the 'ddd', 'dad', 'ada' and 'daa' cases, the resemblance relation $\equiv^* \subseteq Y \times Y$ is defined by:*

$$y_1 \equiv^* y_2 \iff y_1, y_2 \in Y$$

This relation is obviously reflexive ($a \equiv^* a$), symmetrical ($a \equiv^* b \iff b \equiv^* a$) and transitive ($a \equiv^* b$ and $b \equiv^* c \Rightarrow a \equiv^* c$). Thus it is an equivalence relation.

Lemma 3 *The resemblance relation $\equiv^*$ satisfies Properties 2 and 3.*

Proof: For any given parameter value, there is at most one solution, thus Property 2 is satisfied. The $\equiv^*$ relation defines only one equivalence class, equal to $Y = R^{3n}$, which is connected. Thus Property 3 is also satisfied.

In the 'dda' case, there may be up to four solutions in 2D. One pair of solutions may be discarded due to symmetry (Lemma 1). From the remaining pair, one solution is selected using the prototype configuration. Given a 'dda' problem on a triangle ABC, as shown in Fig. 4(c), and a prototype $p = (A_p, B_p, C_p)$, then the solution $s = (A_s, B_s, C_s)$ is selected such that angle $\angle B_p C_p A_p$ and angle $\angle B_s C_s A_s$ are either both acute or both non-acute. The *acuteness* Γ of an angle $\angle Q$ is defined as:

$$\Gamma(\angle Q) = \begin{cases} \textit{acute} & \iff & 0 \leq \angle Q < \frac{\pi}{2} \\ \textit{non-acute} & \iff & \frac{\pi}{2} \leq \angle Q \leq \pi \end{cases}$$

Definition 7 *For the 'dda' case, the resemblance relation* $\equiv^{dda} \subseteq Y \times Y$ *is defined by:*

$$ABC \equiv^{dda} A'B'C' \iff \Gamma(\angle BCA) = \Gamma(\angle B'C'A')$$

Because equality is an equivalence relation, $\equiv^{dda}$ is also an equivalence relation.

Lemma 4 *For the 'dda' case,* $x \to s(x,p)$ *is a continuous mapping, satisfying Property 1.*

Proof: The sign of the term $t = q \pm \sqrt{4q^2 - 4r}$ in the general 'dda' solution, presented at the beginning of this section, is determined only by the prototype p. Both solutions are continuous and real for $4q^2 > 4r$. By substitutions for q and r, we obtain:

$$cos^2(\alpha) > 1 - (\frac{d_2}{d_1})^2$$

By substituting $cos^2(\alpha) + sin^2(\alpha) = 1$, we obtain:

$$sin^2(\alpha) < (\frac{d_2}{d_1})^2$$

The parameter domain corresponds to:

$$(|\alpha| < sin^{-1}(\frac{d_2}{d_1}) \cap d_1 \geq d_2) \cup (d_1 < d_2)$$

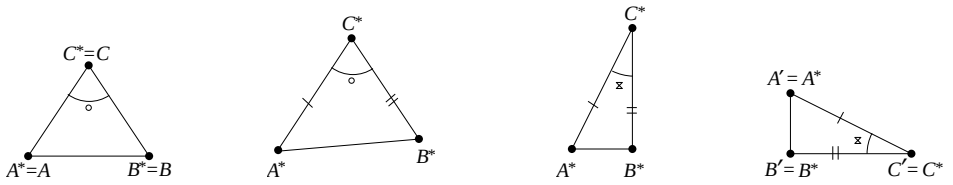
For any parameter vector in this domain, Property 1 is satisfied.

Lemma 5 *The resemblance relation* $\equiv^{dda}$ *satisfies Property 2.*

Proof: There are two distinct solutions of the 'dda' case, for the same parameter values, corresponding to $t = q - \sqrt{4q^2 - 4r}$ and $t = q + \sqrt{4q^2 - 4r}$. Given that ABC is a solution corresponding to $0 \leq t \leq q$, then $\|AC\| = t \leq q = d_1 cos(\alpha)$ and $\angle BCA \geq \frac{\pi}{2}$. Note that if $t = q$, then $\|AC\| = t = q = d_1 cos(\alpha)$, and thus $\angle BCA = \frac{\pi}{2}$. In this case there is only one solution, and no selection is needed.

The other solution, corresponding to $t > q$ is a triangle $A'B'C'$, where $\angle B'C'A' < \frac{\pi}{2}$. Angle $\angle B'C'A'$ is always acute, whereas $\angle BCA$ is always non-acute, i.e. $ABC \not\equiv^{dda} A'B'C'$. Thus, Property 2 is satisfied.

Lemma 6 *The resemblance relation* $\equiv^{dda}$ *satisfies Property 3.*



(a) $f(0)$ (b) $f(\frac{1}{3})$ (c) $f(\frac{2}{3})$ (d) $f(1)$

Figure 5: A continuous acuteness-preserving transformation

Proof: Given a configuration ABC (Fig. 5a) such that $\angle BCA$ is acute and a configuration $A'B'C'$ (Fig. 5d) such that $\angle B'C'A'$ is also acute. Then there is a continuous function f, such that $f(0) = ABC$, $f(1) = A'B'C'$, and $\forall\phi \in (0,1)$: $f(\phi) = A^*B^*C^*$ such that $\angle B^*C^*A^*$ is acute.

This function may be constructed as follows. From $\phi = 0$ to $\phi = \frac{1}{3}$, the edges A^*C^* and B^*C^* are scaled continuously from $\|AC\|$ to $\|A'C'\|$ and from $\|BC\|$ to $\|B'C'\|$. The angle $\angle B^*C^*A^*$ remains constant (Fig. 5b). Then, angle $\angle B^*C^*A^*$ is scaled continuously from $\angle BCA$ at $\phi = \frac{1}{3}$ to $\angle B'C'A'$ at $\phi = \frac{2}{3}$. The edges A^*C^* and B^*C^* remain of constant length, but the edge A^*B^* is now scaled (Fig. 5c). $\angle B^*C^*A^*$ remains acute during this transformation, because $\angle BCA < \frac{\pi}{2}$ and $\angle B'C'A' < \frac{\pi}{2}$. $A^*B^*C^*$ is now congruent with $A'B'C'$. From $\frac{2}{3} < \phi < 1$, the triangle $A^*B^*C^*$ undergoes a rigid motion such that at $\phi = 1$ it is equal to $A'B'C'$ (Fig. 5d). $\angle B^*C^*A^*$ remains constant during this motion. Thus for any $\phi \in [0,1]$: $\angle B^*C^*A^*$ is acute. The equivalence class corresponding to an acute angle is thus connected, satisfying Property 3. For $\angle BCA$ and $\angle B'C'A'$ being non-acute, a similar proof can be constructed.

The basic 3D subproblem is the tetrahedral subproblem. This problem involves four points. Only the case where six distances between these points are known needs to be considered. Angle constraints are solved in triangular subproblems.

A tetrahedron $ABCD$ is constructed using six known distances: d_{AB}, d_{AC}, d_{AD}, d_{BC}, d_{BD} and d_{CD}. First, a triangle ABC is constructed. Point D is determined by the intersection of three spheres, centred in A, B and C, with radii d_{AD}, d_{BD} and d_{CD} respectively.

There may be zero, one or two solutions for the intersection. If there are two solutions, then these are symmetrical, mirrored in the plane through ABC (see Fig. 6). To distinguish these solutions, we determine the *handedness* Θ of the corresponding tetrahedra. For a tetrahedron $PQRS$ this is

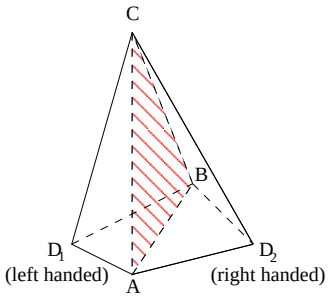


Figure 6: Lefthanded and righthanded solutions $ABCD_1$ and $ABCD_2$, given d_{AB}, d_{AC}, d_{AD}, d_{BC}, d_{BD} and d_{CD}.

defined as:

$$\Theta(PQRS) = \begin{cases} right & \iff & (\overrightarrow{PQ} \times \overrightarrow{PR}) \cdot \overrightarrow{PS} \geq 0 \\ left & \iff & (\overrightarrow{PQ} \times \overrightarrow{PR}) \cdot \overrightarrow{PS} < 0 \end{cases}$$

Definition 8 *For the tetrahedral subproblem, the resemblance relation* $\equiv^{tet} \subseteq Y \times Y$ *is defined by:*

$$ABCD \equiv^{tet} A'B'C'D' \iff \Theta(ABCD) = \Theta(A'B'C'D')$$

Because equality is an equivalence relation, $\equiv^{tet}$ is also an equivalence relation.

Lemma 7 *For tetrahedral subproblems,* $x \to s(x,p)$ *is a continuous mapping, satisfying Property 1.*

Proof: For any combination of the parameters, there are two symmetrical solutions, there are no solutions, or there is one degenerate solution. In the latter two cases, no solution selection is needed. If there are two solutions, then one solution is lefthanded and the other is righthanded, due to symmetry. The solution of which the handedness is equal to the handedness of the prototype, is the intended solution. Thus, the parameter domain for which the intended solution is continuous, is equal to the parameter domain for which a solution exists.

This domain can be characterised as follows. For each triangle in the tetrahedron, a solution must exist, which is expressed by the triangle inequality. Each parameter corresponds to an edge in two triangles, thus for each parameter two inequalities can be formulated:

$$d_{pq} < d_{pr} + d_{qr}$$

$$d_{pq} < d_{ps} + d_{qs}$$

where p, q, r and s should be replaced by any combination of A, B, C and D, resulting in a total of twelve inequalities. Each of these inequalities can be described geometrically as a half-space $X_i \in X$, $i = 1 \ldots 12$, where X is the parameter space of the problem. The domain of the parameters for which a solution exists is $X' = \bigcap X_i$. It can easily be shown, using elementary calculus, that this is a continuous domain. Thus, for any parameter vector $x \in X'$, the solution is a continuous mapping, satisfying Property 1.

Lemma 8 *The resemblance relation $\equiv^{tet}$ satisfies Property 2.*

Proof: The two solutions for a tetrahedron $ABCD$ are mirror-symmetrical. If the triangle ABC is first constructed in the plane $z = 0$, then the solutions D_1 and D_2 for point D are mirror images on different sides of this plane. Suppose $ABCD_1$ is righthanded, then $(\overrightarrow{AB} \times \overrightarrow{AC}) \cdot \overrightarrow{AD_1} > 0$, and $(\overrightarrow{AB} \times \overrightarrow{AC}) \cdot \overrightarrow{AD_2} < 0$, and thus $ABCD_2$ is lefthanded. If $ABCD_1$ is lefthanded, then vice versa. In both cases $\Theta(ABCD_1) \neq \Theta(ABCD_2)$ and $ABCD_1 \not\equiv^{tet} ABCD_2$, thus Property 2 is satisfied.

Lemma 9 *The resemblance relation $\equiv^{tet}$ satisfies Property 3.*

Proof: Each equivalence class in $\equiv^{tet}$ should be a connected set, i.e. all righthanded tetrahedra should be connected and all lefthanded tetrahedra should be connected. Given two righthanded configurations $ABCD$ and $A'B'C'D'$. Then there is a function f such that $f(0) = ABCD$, $f(1) = A'B'C'D'$, and $\forall \phi \in (0, 1)$: $f(\phi) = A^*B^*C^*D^*$, such that $A^*B^*C^*D^*$ is righthanded. Such a function may be constructed as follows. At $\phi = 0$, the edges of $A^*B^*C^*D^*$ are equal to the corresponding edges in $ABCD$. The edges are then scaled continuously such that they have the same length as the corresponding edges in $A'B'C'D'$ at $\phi = \frac{1}{2}$. To scale each edge, a scaling/shearing transformation is applied to the tetrahedron, which never scales an edge negatively, and thus the handedness of the tetrahedron does not change. From $\phi = \frac{1}{2}$ to $\phi = 1$, a rigid rotation/translation transforms $A^*B^*C^*D^*$ such that it exactly matches $A'B'C'D'$. This also does not change the handedness of the tetrahedron. Thus, we have shown that all righthanded tetrahedra are connected. For lefthanded tetrahedra, the same function may be used to proof the lemma.

5 Construction analysis

The intended solution for a problem $G = (V, C)$ is obtained by solving a sequence of subproblems $\mathcal{H} = \{H_1, \ldots H_k\}$. For each $H_i \in \mathcal{H}$ we have $H_i = (V_i, C_i)$, where $V_i \subset V$, and the constraints in C_i either are in C, or result from previously solved subproblems. The parameter vector of a subproblem H_i is denoted x_i and consists of the parameters of the constraints

C_i. The prototype p_i is a subconfiguration of the prototype p, corresponding to the variables in V_i. The intended solution for each subproblem H_i is represented by a vector s_i.

Subproblems are solved in a specific order, such that for $i < j$, H_i is independent of H_j. If H_j depends on H_i, then they share two point variables, of which the relative position is determined by H_i, and the distance between these points is needed to solve H_j.

Lemma 10 *Given $H_i = (C_i, V_i)$ and $H_j = (C_j, V_j)$ such that $V_i \cap V_j = \{v_a, v_b\}$. Also given is that in V_i we find $|v_a - v_b| = d_{ab}$ and in C_j there is a constraint $d(v_a, v_b, d_{ab})$. If $x_i \rightarrow s_i$ satisfies Property 1 and $x_j \rightarrow s_j$ satisfies Property 1, then $x_i \rightarrow s_j$ satisfies Property 1.*

Proof: If $x_i \rightarrow s_i$ is continuous, then $x_i \rightarrow d_{ab}$ is also continuous. Since $d_{ab} \in x_j$ and $x_j \rightarrow s_j$ is continuous, $x_i \rightarrow s_j$ is a composition of continuous functions, and thus also a continuous function.

Given a subproblem $H_i = (V_i, C_i)$ and a solution s_i, where $V_i = \{v_{i1}, \ldots, v_{ik}\}$ and $s_i = (s_{i1}, \ldots, s_{ik})$, then a cluster $K_i = \{(v_{il}, s_{il}); l = 1 \ldots k\}$ can be constructed. The cluster represents a collection of (variable, value) pairs, corresponding to assignments that satisfy the constraints in the problem.

A pair of clusters K_i and K_j can be merged if $V_i \cap V_j = \{v_a, v_b, v_c\}$, and the configuration of these points in both clusters is the same. If the three points are on a line or in a single point, then the problem is underconstrained. Otherwise, a new cluster $K_m = K_i \otimes K_j$ is obtained. Here $K_m = \{(v, s^*); v \in V_i \cup V_j, s^* \in s_i \cup T(s_j)\}$. For each pair $(v, s) \in K_i$, there is a corresponding pair $(v, s) \in K_m$. For each pair $(v, s) \in K_j$ and $(v, s) \notin K_i$, there is a pair $(v, T(s)) \in K_m$. The transformation T is a rotation/translation such that the configuration of v_a, v_b and v_c in K_j is mapped onto the configuration of these variables in K_i. Thus, if (v_a, s_{ia}), (v_b, s_{ib}), $(v_c, s_{ic}) \in K_i$ and (v_a, s_{ja}), (v_b, s_{jb}), $(v_c, s_{jc}) \in K_j$, then $T(s_{ja}) = s_{ia}$, $T(s_{jb}) = s_{ib}$ and $T(s_{jc}) = s_{ic}$.

Lemma 11 *Let $K_m = K_i \otimes K_j$. If $x_i \rightarrow s_i$ and $x_j \rightarrow s_j$ are continuous, then $x_i \rightarrow s_m$ and $x_j \rightarrow s_m$ are also continuous.*

Proof: For each pair $(v, s) \in K_i$, there is a corresponding pair $(v, s) \in K_m$. Since $x_i \rightarrow s_i$ is continuous, $x_i \rightarrow s_m$ is also continuous. For each pair $(v, s) \in K_j$ and $(v, s) \notin K_i$, there is a pair $(v, T(s)) \in K_m$. Since $x_j \rightarrow s_j$ is continuous, and T is a continuous transformation, $x_j \rightarrow s_m$ is also continuous.

Theorem 1 *The intended solution $s(x, p)$ satisfies Property 1.*

Proof: $s(x, p)$ is obtained by solving a sequence of subproblems, by propagating distances, and by merging clusters. From Lemmas 2, 4 and 7, we

infer that all subproblem solutions satisfy Property 1. Using Lemma 10 and Lemma 11, we infer that the final solution also satisfies Property 1.

Definition 5 states that $s(x,p) \equiv_G p$, where $s(x,p)$ is the intended solution for a problem G. We can now define the resemblance relation $\equiv_G$ in terms of the resemblance relations $\equiv_i$ of the subproblems H_i that are solved to solve G.

Definition 9 *Given a problem G, and a set of subproblems $\mathcal{H} = \{H_1, \ldots, H_k\}$, then $\equiv_G$ is defined as:*

$$y \equiv_G z \iff \forall i \in 1 \ldots k : y_i \equiv_i z_i$$

Here, y_i and z_i are the subconfigurations of y and z, for the point variables V_i of each subproblem $H_i \in \mathcal{H}$, and each relation $\equiv_i$ is appropriately chosen as $\equiv^*$, $\equiv^{dda}$ or $\equiv^{tet}$, defined in Definitions 6, 7 and 8 respectively.

Theorem 2 *The resemblance relation $\equiv_G$ is uniquely defined by G.*

Proof: First, consider problems with no angle constraints. A problem G^n consisting of n points and $k = 3n-6$ distance constraints is well-constrained and can be solved by decomposing it into $t = n-3$ tetrahedral subproblems (to see this, consider that for four points, six distances are needed to solve one tetrahedron, and that for each extra point, three more constraints and one more tetrahedron are needed).

We construct the proof by induction. First, if $n = 4$, then $k = 6$ and $t = 1$, and the theorem obviously holds: G^4 is the tetrahedral subproblem. Next, assume that for any $4 \leq m < n$ the set $\mathcal{H}^m$ of subproblems used for solving G^m is uniquely determined by G^m.

Suppose the G^n problem contains s independent tetrahedral subproblems $G^4_i \subset G^n$, where $1 \leq i \leq s$ and $s \leq t$. Two subproblems are independent if they can be solved in any order. The independent subproblems are completely determined by the system of constraints in G^n. Suppose also that the solutions of the G^4 subproblems can be merged into c rigid clusters $C_1 \ldots C_c$. Figure 7 shows the relations between subproblems and clusters used in this proof. Because the problem is well constrained, $s \geq 1$ and $c \geq 1$.

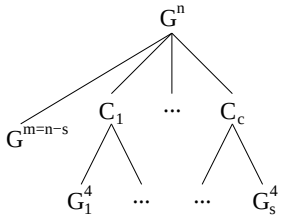


Figure 7: Relations between subproblems and clusters used in proof of Theorem 2.

If $s = t$, then all tetrahedral solutions can be merged into one cluster, and the problem is solved. If $s < t$, then $c > 1$, and we construct a new problem that shares three points with each cluster, so all clusters can be merged into a single cluster after this new problem has been solved. This problem is constructed as follows. Every constraint that is not in any of the G_i^4 subproblems, and the points those constraint are imposed on, are included in the new problem. Any distance, between a pair of points in the new problem that is determined by one of the clusters, is added as a constraint to the new problem. The new problem is thus completely determined by G^n. Each G_i^4 subproblem fixes one point relative to the new problem, because it shares three points with it, or because it shares three points with a cluster that shares three points with the new problem. The new subproblem thus contains $m = n - s$ points, and because $s > 0$, $m < n$.

Because the G^n problem is well-constrained, the new problem is a well-constrained G^m problem, with $m < n$, for which we have assumed that $\mathcal{H}^m$ is uniquely determined by G^m. As already stated above, the independent G^4 subproblems are also uniquely determined by G^n, thus $\mathcal{H}^n = \mathcal{H}^m + \{G_i^4; 1 \leq i \leq s\}$ is uniquely determined by G^n.

Now consider angle constraints; these are solved in triangular subproblems. The distances determined by these subproblems are then used to solve tetrahedral subproblems. The configuration of angle constraints in the problem G determines which triangular patterns (i.e. dda, dad, daa or ada) are used. A triangular subproblem can be solved directly using the constraints in G, or a triangular subproblem can be solved using distances determined by merging clusters. In both cases, the configuration of the angles in the problem determines the pattern used.

We therefore conclude that $\mathcal{H}$ is uniquely determined by the constraints in G. From Definition 9 it is clear that the resemblance relation $\equiv_G$ depends only on the set of subproblems in $\mathcal{H}$, and not on the order in which subproblems are solved, so $\equiv_G$ is uniquely determined by G.

Lemma 12 *The resemblance relation $\equiv_G$ is an equivalence relation.*

Proof: For two configurations y and z, if $y = z$, then for each subproblem H_i, $y_i = z_i$. From Definitions 6, 7 and 8, we infer that each $\equiv_i$ is a resemblance relation, and therefore $y_i \equiv_i z_i$. From Definition 9 follows $y \equiv_G z$, thus $\equiv_G$ is reflexive. Also, for each subproblem, if $y_i \equiv_i z_i$ then $z_i \equiv_i y_i$, therefore, if $y \equiv_G z$ then $z \equiv_G y$. Thus, $\equiv_G$ is symmetrical. Given three configurations y, z and r such that $y \equiv_G z$ and $z \equiv_G r$, then, for every subproblem, $y_i \equiv_i z_i$ and $z_i \equiv_i r_i$. Because each $\equiv_i$ is transitive, we also have $y_i \equiv_i r_i$. From Definition 9 follows $y \equiv_G r$, thus the relation $\equiv_G$ is transitive. The relation $\equiv_G$ is thus reflexive, symmetrical and transitive, i.e. an equivalence relation.

Theorem 3 *The resemblance relation $\equiv_G$ satisfies Property 2.*

Proof: Given two solutions $s \therefore G$ and $t \therefore G$, for the same parameter vector, and a decomposition $\mathcal{H}$ of G. The first solution, s, is constructed by merging solutions $s_i \therefore H_i$, and t is constructed by merging solutions $t_i \therefore H_i$. If $s \neq t$ then there must be a subproblem H_i for which $s_i \neq t_i$, because merging clusters involves a rotation/translation transformation, which preserves the relative positions of the points in each cluster. From Lemma 3, 5 or 8, we obtain $s_i \not\equiv_i t_i$. From Definition 9, we obtain $s \not\equiv_G t$, and thus $\equiv_G$ satisfies Property 2.

Theorem 4 *The resemblance relation $\equiv_G$ satisfies Property 3.*

Proof: Given two configurations $y, z \in Y$, $y \equiv_G z$, then by Definition 9, for every subproblem $H_i \in \mathcal{H}$, $y_i \equiv_i z_i$. According to Lemmas 3, 6 and 9, all $\equiv_i$ satisfy Property 3, i.e. the equivalence classes in these relations are connected, and thus each pair y_i, z_i is connected. Then y and z are also connected. Thus the relation $\equiv_G$ satisfies Property 3.

The intended solution thus satisfies all three properties given in Section 2.

6 Conclusions

We have presented a mechanism to select a single solution for a system of constraints on points using a prototype configuration. The use of a prototype has been given a theoretical basis by introducing a formal resemblance relation. This allows us to select a solution in a meaningful way, i.e. the selected solution is really the intended one. These concepts have here been elaborated for a simple bottom-up solving scheme. The solver finds the intended solution in polynomial time, making it suitable for large systems of constraints.

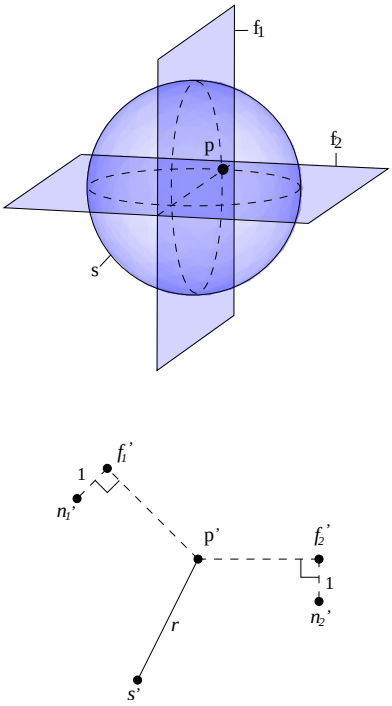
It may be difficult to extend our approach to general geometric constraint systems, in particular where larger subproblems, e.g. octahedral subproblems[3], are involved. However, we aim at systems of constraints common in CAD and other applications, which, in general, do not contain such subproblems.

A relatively simple extension that may be added is propagation of angles[11]. The solution selection process may also be extended to deal with negative distances and angles, allowing different solutions to be selected based on the sign of the input parameters[2].

Our resemblance relation is defined in terms of 3D points, whereas typical CAD problems contain other primitives such as lines, planes, blocks, cylinders and spheres. A solution for this is to map such problems to systems of constraints on points. Such mappings are plausible because the primitives can be characterised by a small number of points, and any point on the surface of a primitive can be expressed in terms of distances and angles relative to the characteristic points. For example, a plane may be characterised by

two points, one representing a point in the plane, and one at a fixed distance from the first, in the direction of the plane normal. Figure 8 shows how a problem with one point, two planes and one fixed-radius sphere, can be mapped to constraints on points. In addition to constraint systems on the geometric primitives mentioned above, also constraint systems on features may be mapped to points, which has been shown in practice for freeform features[12].

By mapping geometric problems in this way to constraints on points, we can use the solution selection rules presented here in problem domains with more complex primitives too, though it remains to be seen whether new selection rules might be needed in specific cases. Our approach can therefore be used to define and determine meaningful solutions for geometric constraint problems in CAD systems, in particular feature modelling systems, and other applications.



point p
plane f_1
plane f_2
sphere s, radius r
p on s
p in f_1
p in f_2

points p', f_1', n_1', f_2', n_2', s'
$p = p'$
f_1 through f_1', normal to $n_1' - f_1'$
f_2 through f_2', normal to $n_2' - f_2'$
s centre s' radius r
$d(p', s') = r$
$d(f_1', n_1') = 1$
$d(f_2', n_2') = 1$
$\alpha(n_1', f_1', p') = 90°$
$\alpha(n_2', f_2', p') = 90°$

Figure 8: Mapping constraints on plane and sphere primitives to constraints on points

Acknowledgements

We would like to thank the reviewers for their insightful comments; one of them in particular had a significant influence on this paper.
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. *Journal of Mechanical Design*, 125(3):443–451, 2003.

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

[3] C. Durand and C. M. Hoffmann. A systematic framework for solving geometric constraints analytically. *Journal of Symbolic Computation*, 30(5):493–519, 2000.

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

[5] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, 6(4):405–420, 1996.

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

[7] 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.

[8] R. Joan-Arinyo, M. V. Luzón, and A. Soto. Genetic algorithms for root multiselection in constructive geometric constraint solving. *Computers & Graphics*, 27(1):51–60, 2003.

[9] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana-Pastó. Declarative characterization of a general architecture for constructive geometric constraint solvers. In *3iA 2002 International Conference on Computer Graphics and Artificial Intelligence, May 14 - 15, 2002, Limoges, France*, pages 63–76, 2002.

[10] 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.

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

[12] 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.