J. Symbolic Computation (2000) **30**, 493–519
doi:10.1006/jsco.2000.0392
Available online at http://www.idealibrary.com on IDEAL

A Systematic Framework for Solving Geometric Constraints Analytically

CASSIANO DURAND† AND CHRISTOPH M. HOFFMANN‡

Computer Science Department, Purdue University, West Lafayette, IN 47907-1398, U.S.A.

A systematic framework is presented for solving algebraic equations arising in geometric constraint solving. The framework has been used successfully to solve a family of spatial geometric constraint problems. The approach combines geometric reasoning, symbolic reduction, and homotopy continuation.

1. Introduction

A *geometric constraint solver* accepts instances of geometric constraint problems. A *geometric constraint problem instance* consists of a set of *geometric elements*, such as points, lines and planes, and *constraints* upon them, such as constraints of distance, angle, coincidence, and so on. The constraint solver then computes a suitable set of coordinates for each geometric element such that the constraints are satisfied, or else announces that no solution could be found.

Applications of geometric constraint solving abound in solid modeling, graphics, engineering, and many other fields (Durand, 1998). We are especially interested in applications in solid modeling, hence we concentrate on solvers that have to tackle nonlinear problems to satisfies the constraints. Incremental constraint satisfaction, an important subject in graphics and simulation, is not addressed in this paper.

A geometric constraint solver can operate in a single phase or in two phases. Single-phase solvers, also called instance solvers, directly translate the constraint problem instance into a representation suitable for solving the problem instance immediately—and then solve the instance. Two-phase solvers first preprocess the constraint system instance, making use of the structure of the constraints and using the constraints symbolically. A fundamental advantage of generic solvers is their ability to create *templates* to solve classes of constraint problems, e.g. Hoffmann and Vermeer (1994).

After a two-phase solver has preprocessed the problem, a second phase is required to determine actual coordinate values for the geometric elements subjected to the constraints. The work of this second phase differs from the work of single-phase solvers only in that the preprocessing has decomposed the constraint problem and possibly recognized characteristic patterns that are solvable by a repertoire of templates. The second phase has been described in Fudos (1995).

†E-mail: crbd@cs.purdue.edu
‡E-mail: cmh@cs.purdue.edu

0747–7171/00/110493 + 27 $35.00/0

We are interested in how to approach the second phase of two-phase solvers when the nonlinear systems that must be solved become daunting. We note that spatial constraint solving configurations with as few as six geometric elements may pose serious challenges to reliably finding one or more solutions.

The problems that arise for the second phase very naturally correspond to systems of simultaneous nonlinear equations. For reasons explained in detail in Durand (1998) and briefly noted in the next section, it is often insufficient to find only one solution of such systems: equation solvers that find only one solution may find one that is not acceptable to the application that formulated the constraint problem. For this reason we look for an algebraic approach that can, in principle and actuality, compute *all* solutions of the system. It is with this requirement in mind that we undertake to formulate a framework for solving nonlinear algebraic equations.

Our main goal is to provide a systematic solution framework for octahedral problems, which combines geometric reasoning, symbolic simplification and homotopy continuation. Previous solutions, e.g. Hoffmann and Vermeer (1994), have relied on reasoning about the geometry of the configuration. We do not consider degenerate cases. Unless otherwise stated, all the problems involve only nonzero distances and angles in the interval $(0, \pi)$.

Moreover, throughout the text, solving a constraint problem can be regarded as finding *all* the possible realizations which satisfy the given constraints.

Finally, note that all the running times reported were obtained on a Sun Sparc Station 20 with 128 MBytes of memory and operating system SunOS Release 5.5.1.

Section 2 presents a brief survey of constraint solving techniques. Section 3 introduces definitions, terminology and basic concepts which are used throughout the paper. It also defines the scope of this work. Section 4 reviews homotopy continuation methods for solving systems of algebraic equations. Section 5 introduces our solution framework and uses it for solving a family of basic constraint problems. Section 6 concludes this work.

2. Constraint Solving Techniques

2.1. ANALYTICAL SOLVERS

In analytical solvers, the constraints are represented by a system of nonlinear equations. Analytic solvers can be further classified as numerical and symbolic algebraic solvers.

NUMERICAL SOLVERS

Numerical solvers are instance solvers that use iterative methods to solve the system of equations representing the constraints.

An iterative technique in wide use is the Newton–Raphson method (Ortega and Rheinboldt, 1970; Stoer and Bulirsch, 1993). This method is distinguished by the ability to solve large problems, but it is very sensitive, requiring a sufficiently good initial guess. The difficulty predicting to which root the method will converge relies on the fact that the attraction basins[†] for the Newton–Raphson method are fractals (Peitgen and Ritcher, 1986). Therefore, if the sketch is used as the initial approximation, then it should nearly

[†]The set of points in the space of system variables such that an initial approximation chosen in this set evolves to a particular solution of the system.

satisfy the constraints to guarantee that the method would converge to the desired solution. In applications, this is seldom the case.

Based on the theory of nonlinear optimization, methods with global convergence properties were proposed (Dennis and Schnabel, 1983). These methods are referred as global, and converge to a solution from almost any initial guess. Convergence is achieved by defining an *energy* function that decreases as progress is made towards a solution, therefore assuring improvement at each iteration. However, this method can still occasionally fail by ending in a local minimum of the energy function. A combination of heuristics and a variation of this method were used in EMBED (Crippen and Havel, 1988), a practical, but complex, algorithm for solving molecular conformation problems.

The major drawback of the foregoing techniques is that they can converge to unwanted solutions. In that case, the method should be re-applied with different initial guesses until the desired solution is produced. However, there is no guidance on how the subsequent guesses might be made.

Homotopy continuation methods can be used to circumvent this problem (Allgower and Georg, 1990; Li, 1997; Rojas, 1999). Continuation methods are robust and versatile global methods capable of finding *all* solutions of a given system (Allgower and Georg, 1993). Although the theoretical foundations encompass many different areas of mathematics, the idea behind homotopy is rather intuitive: the solutions of a known "easy" system are deformed into the solutions of the wanted system. The method has been applied to problems in many areas, including robotics, kinematics of mechanisms, chemical equilibrium, geometric intersection (Morgan, 1987; Wampler *et al.*, 1990; Patrikalakis, 1992; Huber and Sturmfels, 1995; Huber, 1996; Verschelde, 1996, 1997b) and, more recently, to constraint solving (Lamure and Michelucci, 1995). Albeit powerful, homotopy requires a significant amount of computational work, usually limiting the set of solvable systems to those which are "small". Fortunately, algebraic tools can be used to reduce the size of the systems we are interested in, and broaden the applicability and relevance of continuation methods (Morgan, 1992).

SYMBOLIC ALGEBRAIC SOLVERS

Symbolic algebraic solvers use algebraic elimination methods to solve the system of equations representing the constraints coupled with (univariate) root finding.

The first approach is based on polynomial ideal theory, and generates special bases for the system, called *Gröbner bases* (Buchberger, 1965, 1985; Cox *et al.*, 1992). The original system is transformed into an equivalent triangular system (a Gröbner basis) which, therefore, can be easily solved by back-substitution and univariate root finding. The computation of a Gröbner basis is known as Buchberger's algorithm. Gröbner bases have been used extensively in algebra and geometry (Hoffmann, 1989; Cox *et al.*, 1992). In Kondo (1992), Gröbner bases are used in constraint solving.

The second approach is based on Ritt's construction of characteristic sets (also referred as triangular sets) (Ritt, 1932, 1950), a technique rediscovered and extended by Wu in the context of mechanical geometry theorem proving (Wu, 1986, 1994; Chou, 1988). This method decomposes the solution set of an algebraic system into set expressions involving the solutions of simpler systems. It is argued in Wang (1991) that the method can be used to solve a large number of systems found in the current literature. Wang (1998) generalized the notion of triangular sets to pairs of polynomials called simple systems, which were used to devise a method for solving polynomial systems. Lazard

(1991) and Kalkbrener (1993) also present methods for decomposing the solution set of polynomial systems into triangular sets. An extensive discussion about the different notions of triangular sets is presented in Lazard (1999).

The third approach uses resultants and is based on the theory of determinants. The main idea is to use the original system to generate a larger system where the terms of the original equations are regarded as distinct variables (Gelfand, 1994; Sturmfels, 1997). Sederberg uses this method in the context of curve and surface modeling (Sederberg, 1983). In Manocha (1993) and Manocha and Canny (1993), sparse resultants are used to compute the solutions of polynomial systems. Emiris and Mourrain (1996) and Emiris and Verschelde (1997) use a solver based on sparse resultants to solve problems arising in computational biology and chemistry.

Symbolic algebraic solvers can be regarded as instance solvers if the constraints values are used when manipulating the equations. The power of the approach is due to the fact that the constraints can be manipulated symbolically, producing parameterized solutions. Those solutions can be re-evaluated for different sets of constraint values.

Symbolic solvers are often very slow, usually requiring exponential running time. Moreover, symbolic computations are memory intensive. Therefore, some geometric restrictions are usually imposed in practice.

2.2. GRAPH-BASED SOLVERS

In graph-based solvers, the constraints are represented by a *constraint graph* which encodes the geometric and topological structure of the sketch. It is a two-phase approach: in the first phase, the constraint graph is analyzed and a decomposition and construction sequence is determined. In the second phase, the geometric elements are placed, i.e. their coordinates are computed, as the construction steps are carried out.

The solver described in Bouma (1995) uses this approach to solve problems in 2D. The construction sequence groups the vertices of the graph recursively into sets, called *clusters*. The clusters induce subgraphs whose underlying geometry can be solved algebraically. The algorithm recursively merges three clusters (forming a new augmented cluster), provided they are pairwise adjacent (when regarded as super-vertices of the graph). For a complete solution, all vertices must be grouped into a single cluster upon termination. Regardless of the fact that the clusters can be merged in many different ways, the solution is unique when applying simple rules for selecting from arising multiple roots (Fudos and Hoffmann, 1993). In Fudos and Hoffmann (1996), the authors describe how to construct conic blending arcs from constraints using the same approach. In Hoffmann and Joan-Arinyo (1997), a method that combines graph-based and numerical techniques is presented.

DCM (D-Cubed, 1994) is a commercial solver that also uses a graph-based approach. The constraint graph is partitioned into subgraphs that can be solved algebraically with respect to local coordinates. In the next phase, the subgraphs are placed with respect to each other by the application of rigid-body transformations to the underlying geometry of each subgraph (Owen, 1991).

The graph-based approach is fast and methodical. However, it is very sensitive to the types of geometric objects and constraints considered. Extensive modifications are required after adding new geometric types or new constraint types.

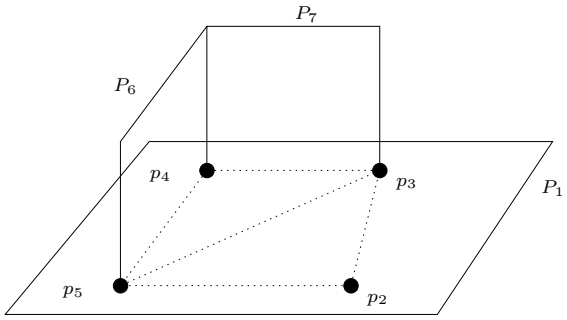

Figure 1. Sketch involving planes P_1, P_6, and P_7, and points p_2, p_3, p_4, and p_5.

2.3. RULE-BASED SOLVERS

In the rule-based approach, the constraints are represented as a set of rules and predicates. Rewrite rules are used to find a construction sequence that satisfies all constraints. Based on this procedure, the predicates representing the desired constraints are transformed into predicates defining the position of the geometric objects involved.

One of the first attempts to represent constraints as rules is described in Borning (1981), where the rules are classes in Smalltalk associated with methods that can be invoked to solve the constraints. Brüderlin (1987) calculates all solutions symbolically. Predicates are stored in a Prolog database with calls to the procedural language Modula-2 to evaluate the construction steps. Aldefeld (1988) presents a method based on geometric reasoning that uses a forward inference mechanism to solve problems in 2D involving points, tracks and line segments. Verroust *et al.* (1992) describes an approach capable of modeling dimensional, tangency and radius constraints. The sketch is represented by a set of mutually constrained distances (CD sets) and angles (CA sets) which are evaluated simultaneously. Joan-Arinyo and Soto (1997a,b) provide a correctness proof of a method based on an extension of the repertoire of the rules presented in Verroust *et al.* (1992).

Rule-based solvers are valued for the explicit and transparent representation of the geometric knowledge and separation of the knowledge from its processing. As a consequence, this approach is very flexible in the sense that new rules can be added incrementally without modification of the inference component. Nevertheless, it is a potentially slow method due to the exhaustive search and matching inherent in the inference mechanism.

3. Theoretical Background

3.1. PRIMITIVES AND CONSTRAINTS

A point or a plane in 3-space is referred to as a *primitive*. We denote points by $p, p_1, \ldots$ and planes by $P, P_1, \ldots$. By *sketch* we mean the (finite) set of primitives of a geometric constraint problem. We allow *constraints* of distance, angle, denoted *dist* and *ang*, respectively. We also allow the relations of incidence, perpendicularity, and parallelism, denoted in order by *on*, *perp*, and *para*.

The *constraint graph* captures the relationship between the primitives of a sketch. The graph vertices denote the primitives, and the graph edges denote the constraints and relations on them. Figure 1 shows a sketch involving planes P_1, P_6, and P_7, and points

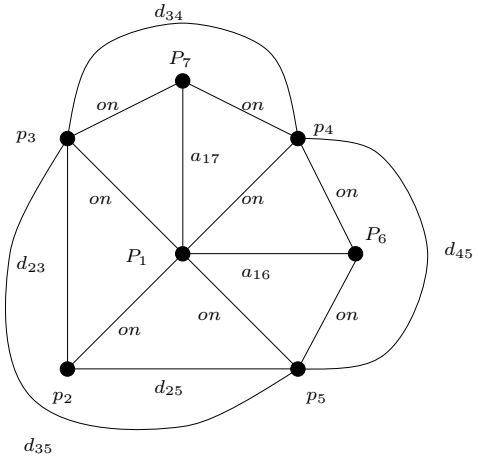

Figure 2. Constraint graph for the sketch of Figure 1.

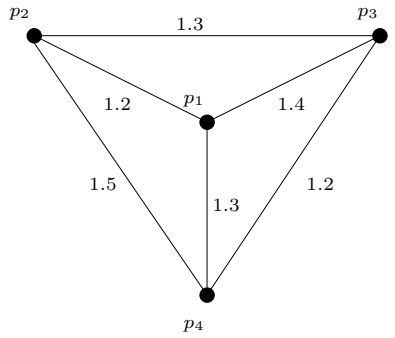

Figure 3. Constraint graph involving four points p_1, p_2, p_3, and p_4, constrained by distances.

Table 1. Two possible realizations of the constraint graph shown in Figure 3.

p_1	p_2	p_3	p_4
(0,0,0)	(1.2,0,0)	(0.7125,0,−1.20513)	(0.366667,1.03217,−0.700131)
(0,0,0)	(1.2,0,0)	(0.7125,0,−1.20513)	(0.366667,−1.03217,−0.700131)

p_2, p_3, p_4, and p_5. Figure 2 shows the corresponding constraint graph. Edges labeled d_{ij} or a_{ij} denote a distance or an angle constraint on the primitives i and j, respectively. The label *on* indicates that the adjacent primitives are incident.

For different sets of constraint values, we can compute one or more placements of the primitives that satisfy the given constraints. The placements are referred as *realizations* of the constraint graph. Figure 3 shows a graph involving four points p_1, p_2, p_3, and p_4, constrained by distances. The labels on the edges correspond to the distance values. Table 1 shows two possible realizations of Figure 3.

The problem of finding one or more realizations of a constraint graph is called a *geometric constraint problem* or simply a *constraint problem*.

If a constraint problem has infinitely many solutions, then it is *underconstrained*. If a problem has a finite number of solutions after deleting one or more constraints, then it is called *overconstrained*. If the solutions satisfy the deleted constraints, the overconstrained problem is said to be *consistent*, otherwise, it is *inconsistent* and has no solution. A problem with a finite number of solutions is called *well constrained* if it is not overconstrained.

3.2. BASIC CONFIGURATIONS

Deciding whether a problem is well constrained by inspection of the constraint graph is nontrivial. A survey of concepts and techniques can be found in Durand (1998).

In the plane, Laman's (1970) theorem provides a basis for such a test, but it is restricted to primitives with 3 degrees of freedom, such as points, planes, and circles with fixed radii. Another characterization is based on Henneberg n-sequences (Henneberg, 1911), which also leads to an algorithm for finding realizations for a restricted class of graphs, called 2-*simple* or *sequentially constructible*. In Bouma (1995), the idea is extended to a more general class of graphs. Initially, the constraint graph is partitioned into 2-simple subgraphs, called *clusters*, and their realizations computed locally. The realizations corresponding to three clusters can then be recursively merged, provided the clusters share a primitive with each other. Regardless of the effort, finding a fast algorithm to systematically produce a realization for any 2D abstract constraint system is, at the present time, an open problem that deserves further attention.

In three dimensions, the constraint solving is even more difficult. Even a test to check if a problem is well constrained is still unknown. Laman's and Henneberg's results, which provided the algorithmic foundation in two dimensions, cannot be fully extended to higher dimensions (Crippen and Havel, 1988).

Hoffmann *et al.* (1997a,b) propose an approach based on degree-of-freedom analysis where the constraint graph is augmented with a weight function that accounts for the number of degrees of freedom of a primitive and the number of degrees of freedom eliminated by a constraint. In Hoffmann and Vermeer (1994), the algorithm from Bouma (1995) is extended to three dimensions. Since the primitives considered there (points and planes) have 3 degrees of freedom, three pairwise constrained vertices are necessary to begin a cluster. Additional vertices can be added to the cluster provided they are incident to three nodes already in the cluster. This corresponds to a tetrahedral structure in the constraint graph. When no more vertices can be added, the cluster is deleted from the constraint graph and the process repeated. There may be unused edges in the constraint graphs since three pairwise constrained vertices are needed to start the cluster. These edges with the adjacent vertices form a *degenerate cluster*. A local realization is then computed for each cluster and the clusters are merged to produce the final realization. However, the necessary relationships between clusters required for merging are much more complicated than the ones found in the two-dimensional case. The Hoffmann and Vermeer (1994) paper identifies four configurations which define a well-constrained problem in general. They are shown in Figure 4(a)–(d).

The double tetrahedron and the decahedron can be decomposed into tetrahedra. The tetrahedron and the octahedron cannot be further decomposed and, for this reason, are called *basic configurations*. The corresponding problems are *basic problems*. They define

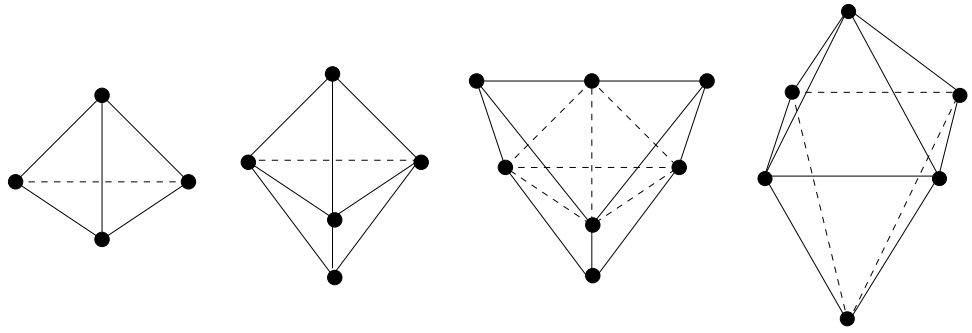

Figure 4. (a) Tetrahedron, (b) double tetrahedron, (c) decahedron and (d) octahedron.

intrinsically different construction steps and can be regarded as building blocks of more complex designs. In fact, by solving these two families of basic problems, one can, in principle, solve any problem which can be decomposed into tetrahedra and octahedra. This justifies our interest in finding efficient solution strategies for these problems.

3.3. TETRAHEDRAL AND OCTAHEDRAL PROBLEMS

Tetrahedral problems involve four primitives which can be points and planes, constrained by distances and angles. There are four possible cases, which are shown in Figure 13 (Appendix B). The problems $Tetra_i$, $i = 1, \ldots, 4$ involve $i-1$ planes. Problems involving three planes are underconstrained. The *Tetra* family of problems can be solved directly by using many analytical methods (see Durand, 1998).

Octahedral problems involve six primitives among points and planes constrained by distances and angles. They are shown in Figures 14 and 15 (Appendix C). We consider six configurations which differ on the number of planes and points involved and on their topology. Problems with more than four planes are underconstrained and are therefore not discussed. Section 5 addresses the solution of octahedral problems.

3.4. THE ALGEBRAIC SYSTEM ASSOCIATED WITH A CONSTRAINT PROBLEM

With $||\ ||$, $\cdot$ and $\times$ we denote Euclidean norm, dot product and vector product, respectively. The point p_i is represented by its Cartesian coordinates

$$p_i : (x_i, y_i, z_i),$$

and the plane P_i, by the unit normal vector $n_i = (nx_i, ny_i, nz_i)$ and the signed distance from the origin d_i

$$P_i : (nx_i, ny_i, nz_i : d_i), \quad \text{q } ||n_i|| = 1.$$

Note that the plane P_i has the implicit equation

$$nx_i x + ny_i y + nz_i x + d_i = 0.$$

The condition $||n_i|| = 1$ is an *implicit constraint*. We give an algebraic representation of the constraints. The equations are presented in the vectorial and Cartesian format.

ANGLE BETWEEN TWO PLANES P_i AND P_j

$ang(P_i, P_j) = a_{ij}$	
Vector	$n_i \cdot n_j = \cos(a_{ij})$
Cartesian	$nx_i nx_j + ny_i ny_j + nz_i nz_j = \cos(a_{ij})$

The constraints *para* and *perp* are special cases of the *ang* constraint where the angles are 0° and 90°, respectively. Note that the definition of parallelism is different from the one presented in most geometry books, where the angle between the primitives can be either 0° or 180°. We choose *oriented parallelism* because it reduces the degree of the corresponding equation and, consequently, the number of solutions of the problem.

DISTANCE BETWEEN TWO POINTS p_i AND p_j

$dist(p_i, p_j) = d_{ij}$	
Vector	$\|\|p_i p_j\|\| = d_{ij}$
Cartesian	$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = d_{ij}^2$

DISTANCE FROM POINT p_i TO PLANE P_j

$dist(p_i, P_j) = d_{ij}$	
Vector	$n_j \cdot p_i + d_j = d_{ij}$
Cartesian	$x_i nx_j + y_i ny_j + z_i nz_j + d_j = d_{ij}$

The constraint *on* is a special case of the *dist* constraint where the distance is 0.

Given a constraint problem, we define the *associated algebraic system* as the polynomial system obtained by the union of the equations corresponding to the implicit and explicit constraints. Consider $Tetra_3$ in Figure 13(c), for instance. If the primitives are represented by

$$\begin{aligned} P_1 &: (x_0, x_1, x_2 : x_3) \\ P_2 &: (x_4, x_5, x_6 : x_7) \\ p_3 &: (x_8, x_9, x_{10}) \\ p_4 &: (x_{11}, x_{12}, x_{13}), \end{aligned}$$

then the algebraic system associated with $Tetra_3$ is

$$\begin{cases} x_0^2 + x_1^2 + x_2^2 - 1 = 0 \\ x_4^2 + x_5^2 + x_6^2 - 1 = 0 \\ x_0 x_4 + x_1 x_5 + x_2 x_6 - \cos(a_1) = 0 \\ x_8 x_0 + x_9 x_1 + x_{10} x_2 - x_3 - d_2 = 0 \\ x_{11} x_0 + x_{12} x_1 + x_{13} x_2 - x_3 - d_3 = 0 \\ x_8 x_4 + x_9 x_5 + x_{10} x_6 - x_7 - d_4 = 0 \\ x_{11} x_4 + x_{12} x_5 + x_{13} x_6 - x_7 - d_5 = 0 \\ (x_{11} - x_8)^2 + (x_{12} - x_9)^2 + (x_{13} - x_{10})^2 - d_6^2 = 0. \end{cases}$$

The first two equations correspond to the implicit constraints on P_1 and P_2. The other equations correspond to the distance and angle constraints.

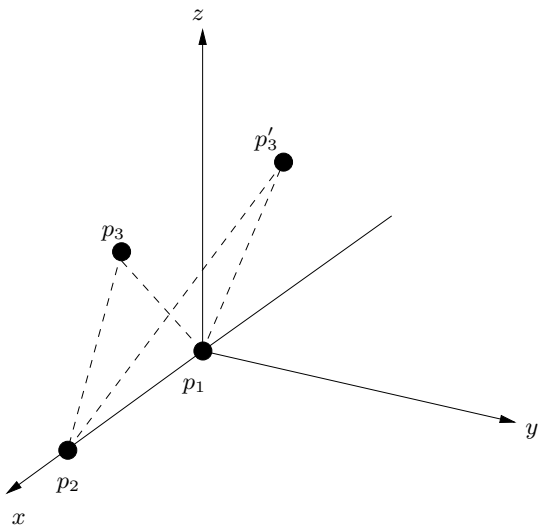

Figure 5. Placement of three points.

3.5. PLACEMENT RULES

The solutions of well-constrained problems are in general rigid realizations with 6 degrees of freedom (three translational and three rotational). Therefore some of the primitives must be placed with respect to a coordinate system to guarantee that the associated system can be solved.

Six degrees of freedom have to be eliminated. Since we are dealing only with points and planes, which have 3 degrees of freedom, we need to place three primitives constrained with respect to each other, i.e. forming a triangle on the constraint graph (Hoffmann and Vermeer, 1994). We use the following placement rules. Only distance and angles are considered. Moreover, in order to avoid degenerate cases, we assume that only nonzero distances and nontrivial angles ($\neq 0°, 180°$) occur.

PLACEMENT OF THREE POINTS (RULE *ppp*)

Let p_1, p_2 and p_3 be three points with distance constraints $d_{i,j}$. A generic placement can be obtained by the following rules; see also Figure 5:

(1) p_1 is placed at the origin.
(2) p_2 is placed on the positive side of the x-axis at distance d_{12} from p_1.
(3) p_3 is placed on the xz-plane according to the distances d_{13} and d_{23}.

In terms of generic coordinates, the primitives can be represented by

$$p_1 : (0, 0, 0)$$
$$p_2 : (x_0, 0, 0)$$
$$p_3 : (x_1, 0, x_2).$$

The placement rules and the constraints then determine the values of x_0, x_1 and x_2:

$$x_0 = d_{12} \qquad x_1 = \frac{1}{2}\frac{x_0^2 - d_{23}^2 + d_{13}^2}{x_0} \qquad x_2 = \sqrt{-x_1^2 + d_{13}^2}.$$

PLACEMENT OF TWO POINTS AND ONE PLANE (RULE ppP)

Let P_1 be a plane, p_2 and p_3 two points, and $d_{i,j}$ the distance constraints between them. A generic placement can be obtained by the following rules, illustrated in Figure 6:

- P_1 is placed as the xy-plane (with normal vector $(0, 0, 1)$).
- p_2 is placed on the positive side of the z-axis at distance d_{12} from P_1.
- p_3 in placed on the xz-plane according to the distances d_{13} and d_{23}.

In terms of generic coordinates, the primitives can be represented by

$$P_1 : (0, 0, 1 : 0)$$
$$p_2 : (0, 0, x_0)$$
$$p_3 : (x_1, 0, x_2).$$

Then the placement rules and the constraints determine the values of x_0, x_1 and x_2

$$x_0 = d_{12} \qquad x_1 = \sqrt{d_{23}^2 - d_{12}^2 - d_{13}^2 + 2d_{12}d_{13}} \qquad x_2 = d_{13}.$$

PLACEMENT OF ONE POINT AND TWO PLANES (RULE pPP)

Let P_1 and P_2 be two planes and p_3 a point, and assume the constraints $ang(P_1, p_2) = a_{12}$, $dist(P_1, p_3) = d_{13}$, and $dist(P_2, p_3) = d_{23}$. As shown in Figure 7, a generic placement can be obtained as follows:

- P_1 is placed as the xy-plane (with normal vector $(0, 0, 1)$).
- P_2 is placed in such a way that it satisfies the angle constraint a_{12}, and the intersection of P_1 and P_2 coincides with the y-axis.
- p_3 is placed on the xz-plane according to the distances d_{13} and d_{23}.

Therefore their coordinates can be represented generically by

$$P_1 : (0, 0, 1 : 0)$$
$$P_2 : (x_0, 0, x_1 : 0)$$
$$p_3 : (x_2, 0, x_3).$$

The values of x_0, x_1, x_2 and x_3 can be computed directly based on the constraints and placement rules

$$x_1 = cos(a_{12}) \qquad x_0 = \sqrt{1 - x_0} = \sin(a_{12}) \qquad x_3 = d_{13} \qquad x_2 = \frac{d_{23} - d_{13}\cos(a_{12})}{\sin(a_{12})}.$$

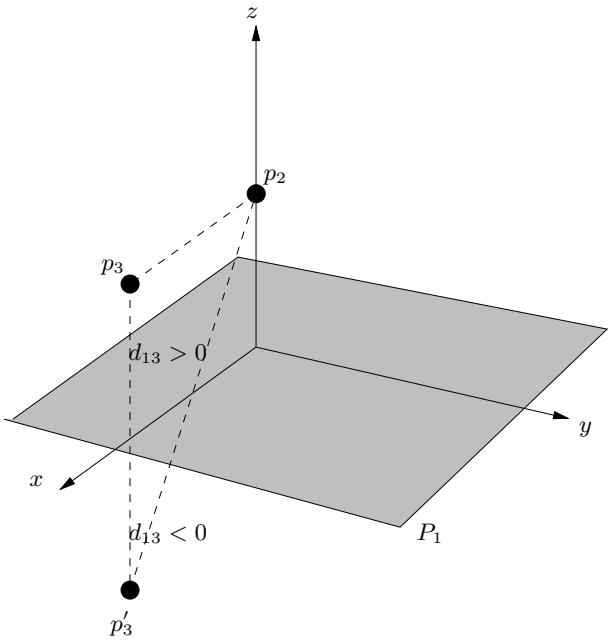

Figure 6. Placement of two points and one plane.

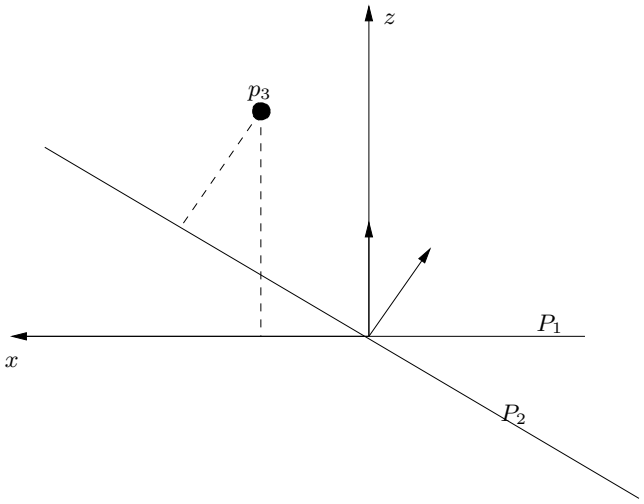

Figure 7. Placement of two planes and one point. Only the projection on the plane xy is shown.

PLACEMENT OF THREE PLANES (RULE PPP)

Let P_1, P_2 and P_3 be three planes, and let a_{ij} denote the angle constraints between them. A generic placement can be obtained by the following rules:

- P_1 is placed as the xy-plane (with normal vector $(0, 0, 1)$).

- P_2 is placed in such a way that it satisfies the angle constraint a_{12}, and the intersection of P_1 and P_2 coincides with the y-axis.
- P_3 is placed in such a way that it contains the origin and satisfies the angle constraints a_{13} and a_{23}.

Therefore their coordinates can be represented generically by

$$P_1 : (0, 0, 1 : 0)$$
$$P_2 : (x_0, 0, x_1 : 0)$$
$$P_3 : (x_2, x_3, x_4 : 0).$$

The placement rules and constraints completely determine the values of x_0, x_1, x_2, x_3 and x_4. In this case x_3 can assume two distinct values.

$$x_1 = cos(a_{12}) \qquad x_0 = \sqrt{1 - x_0} = \sin(a_{12}) \qquad x_4 = cos(a_{13})$$
$$x_2 = \frac{\cos(a_{23}) - \cos(a_{13})\cos(a_{12})}{\sin(a_{12})} \qquad x_3 = \pm\sqrt{1 - x_2^2 - x_4^2}.$$

4. Homotopy Continuation Methods

4.1. OVERVIEW

For more than a century, homotopy has played an important role in many areas of modern mathematics, and its use as a tool to solve systems of linear equations can be traced back at least to Lahaye (1934).

Let $F(x) = 0$, $x = (x_1, x_2, \ldots, x_n)$, $F = (f_1, f_2, \ldots, f_n)$, be a system with finitely many solutions in C^n. The *homotopy equation* is defined by

$$H(x, \lambda) = (1 - \lambda)G(x) + \lambda F(x), \tag{1}$$

where $\lambda \in [0, 1)$. $F(x)$ is called the *target system* and $G(x)$ the *start system*.

The system (1) is underdetermined and implicitly defines a curve in $C^n \times [0, 1)$, the *homotopy path*. The term *homotopy continuation* refers to a set of techniques for numerically approximating the homotopy path. The solutions of $H(x, 0) = G(x) = 0$ are the start points, and, as λ approaches 1, the start points are deformed into the solutions of the target system.

Most homotopy continuation methods use a *predictor–corrector* scheme, similar to the one depicted in Figure 8. Suppose x^* is on the homotopy path for $\lambda = \lambda_0$. The predictor function computes x', which approximates $H(x, \lambda_0 + \delta) = 0$, and the corrector uses x' to compute the point on the homotopy path for $\lambda_0 + \delta$. For a review of path-following techniques, see Allgower and Georg (1990, 1997).

The choice of the start system, and the start points to follow, is crucial for designing an efficient homotopy, because the number of paths to be followed corresponds to the number of start points selected. The problem arises because some start points may produce divergent paths, corresponding to solutions at infinity of $F(x) = 0$ (Morgan, 1987). Solutions at infinity are difficult to detect, expensive to compute, and usually have no practical interpretation.

The topology of the homotopy paths may also impose extra difficulties: paths may cross, have singularities, or become arbitrarily close, causing many numerical problems.

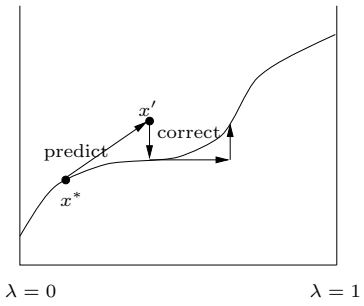

Figure 8. Predictor–corrector scheme. x^* is a point on the homotopy path and x' is the predicted point.

Fortunately, an adequate selection of the start system can usually minimize such situations in practice. In our situation, start system selection was adjusted after solving a configuration for the first time, using the approach of cheater homotopy described later.

4.2. TYPES OF HOMOTOPY CONTINUATION METHODS

Homotopy continuation methods are used to compute all solutions of polynomial systems. Even though the underlying idea is the same, they can rely on different theoretical principles, which define the strategy used for computing the start system and corresponding start points, and the space in which the computations are going to be performed.

Projective homotopies are based on *Bezout's* theorem (Morgan, 1987), which states that the number of isolated solutions of $F(x) = 0$ is bounded above by its total degree. Morgan (1986a) describes how to build a generic start system, whose number of solutions is equal to the total degree of the target system. In Morgan (1986b), he also introduces a projective transformation, which avoids path crossing and solutions at infinity. The resulting homotopy is known as *standard homotopy.*

Bezout's theorem uses only the degree of the polynomials and often overcounts the actual number of isolated solutions. Consequently, standard homotopy usually finds a large number of homotopy paths that lead to solutions at infinity.

Polyhedral homotopies take into account the sparse structure of the system, based on the monomials that appear in each equation. These homotopies rely on Bernstein's theorem, which states that the number of isolated solutions of a system in $(C^*)^n$ is bounded above by the *mixed volume* of the *Newton polytopes* (Verschelde, 1996). The theorem forms the basis of *sparse elimination theory* with methods, also known as *polyhedral* methods, that use a geometric approach to exploit the structure of the equations.

Bernstein's bound is at most as high as Bezout's bound, but is significantly smaller for systems we have encountered in our applications. The bound is also known as the BKK bound, because it relies on work by Bernstein, Khovanskii and Kushnirenko (see Kushnirenko, 1975; Khovanskii, 1977, 1978; Dyer *et al.*, 1998). Mixed volumes can be computed by several methods; in particular, those of Emiris and Canny (1995), Huber and Sturmfels (1995), Verschelde *et al.* (1996) and Dyer *et al.* (1998). For additional theoretical background and standard tools refer to Bonnesen and Fenchel (1987), Betke (1992), Schneider (1993) and Dyer *et al.* (1998).

Sparse elimination also provides the basis for solving systems of equations by continuation (Huber and Sturmfels, 1995; Verschelde *et al.*, 1996; Huber and Sturmfels, 1997).

The central computation in this method is finding a *mixed subdivision* of the supports associated with the polynomials of the system, which defines a monomial basis of the coordinate ring and permits the computation of the number of solutions and numeric approximation of the solution vectors. This method is called *polyhedral homotopy continuation.*

Unlike projective homotopies, it is not necessary to homogenize any of the systems involved, because the continuation is performed in affine space, not in projective space.

Note that polyhedral homotopies have to follow a number of paths equal to the BKK bound of the target system, and do not take into account any relationship between the coefficients, which happens, for instance, when the coefficients are given by parameters. Therefore, the BKK bound can still overcount the number of affine solutions of the system. For more details refer to Verschelde (1996).

In many practical applications we need to solve different instances of a system. That is particularly true when the coefficients of the target system depend on certain parameters. For instance, the coefficients of systems associated with geometric constraint problems depend on the constraints defined between the primitives.

Morgan and Sommese (1989) show that such parametric structure can be exploited, by performing the continuation in parameter space, instead of coefficient space. Therefore, fewer paths need to be tracked, and the total numerical cost is substantially reduced. The method is called in the literature *parameter-based homotopy.*

The same idea is the basis of the so-called *cheater's homotopies* (Li *et al.*, 1989; Verschelde, 1998), which are introduced to solve repeatedly a polynomial system with parametric structure. The procedure assumes that one has solved the polynomial system once—the cheating part—for a generic set of complex parameter values. Afterwards, we can use that system and *only* its nonsingular affine solutions as the start system and start points in a homotopy to solve any other system with the same parameter structure. Since only the nonsingular affine solutions are used as start points, much fewer paths have to be tracked, when compared with standard homotopy, for instance. See Morgan and Sommese (1989) for further detail.

4.3. IMPLEMENTATIONS USED

We use two software packages that implement different flavors of homotopy: *Continuum* (Durand and Hoffmann, 1998), which uses the projective approach, and *PHC* (Verschelde, 1997a,b), which implements polyhedral homotopy.

5. Solving the Octahedron

Some octahedral problems have been studied in different contexts, from kinematics (Nanua *et al.*, 1990) to computational chemistry (Emiris and Mourrain, 1996). For geometric constraint solving, the interest in the octahedron comes from the fact that it is the smallest nontrivial configuration which cannot be decomposed into tetrahedra.

Due to its topological symmetry, any of the eight triangular faces of the octahedron can be selected to be placed, and this gives some flexibility when choosing a placement order that leads to the simplest associated system. Usually, we found that placing the faces with planes produces an associated system which is easier to simplify. Therefore, we use rule *ppp* in $Octa_1$, rule *ppP* in $Octa_2$ and $Octa_4$, rule *pPP* in $Octa_3$ and $Octa_6$, and rule *PPP* in $Octa_5$, $Octa_7$, and $Octa_8$.

We define a four-step framework for solving octahedral problems:

(1) Equation formulation: in this phase, we compute the system associated with the problem using the placement rules and representation of primitives and constraints introduced in Section 3.
(2) Algebraic simplification: in this phase, we simplify the associated system by applying the following sequence of predefined steps.
   (a) Gaussian elimination. The resulting system should have as few squared variables as possible.
   (b) Eliminate univariate equations, since the variables involved can be determined directly.
   (c) Parameterize the variables appearing in all bilinear equations and replace them with their corresponding parametric expressions.
   (d) Parameterize the variables appearing in all bivariate quadratic equations (using *sin* and *cos*) and replace them with their corresponding parametric expressions.
   (e) Use the standard trigonometric substitution $\cos(\alpha_i) = \frac{1-y_i^2}{1+y_i^2}$, $\sin(\alpha_i) = \frac{2y_i}{1+y_i^2}$, where $y_i = \tan\left(\frac{\alpha_i}{2}\right)$.

   The resulting system is called the *core system* which is used as a pattern to solve all problems with the same structure.
(3) Homotopy continuation: in this phase, we use homotopy continuation to compute all the solutions of the core system.
(4) Realization: in this phase we compute the realizations using the solutions of the core system.

In what follows, we apply the the framework to solve $Octa_1$. The solutions of the other octahedral problems follow the same steps.

Initially, we position three points according to rule *ppp* defined in Section 3.5. The primitives can be represented in terms of coordinates by:

$$p_1 : (0,0,0), \qquad p_2 : (x_0,0,0), \qquad p_3 : (x_1,0,x_2),$$
$$p_4 : (x_3,x_4,x_5), \qquad p_5 : (x_6,x_7,x_8), \qquad p_6 : (x_9,x_{10},x_{11}),$$

where $x_0,\ldots,x_{11}$ are the unknowns of our problem. The associated system obtained using this coordinatization is

$$\{f_i\}_{i=1}^{12} = \begin{cases} x_0^2 - d_1^2 = 0 \\ x_1^2 + x_2^2 - d_2^2 = 0 \\ x_3^2 + x_4^2 + x_5^2 - d_3^2 = 0 \\ x_6^2 + x_7^2 + x_8^2 - d_4^2 = 0 \\ (x_1 - x_0)^2 + x_2^2 - d_5^2 = 0 \\ (x_3 - x_1)^2 + x_4^2 + (x_5 - x_2)^2 - d_6^2 = 0 \\ (x_6 - x_3)^2 + (x_7 - x_4)^2 + (x_8 - x_5)^2 - d_7^2 = 0 \\ (x_0 - x_6)^2 + x_7^2 + x_8^2 - d_8^2 = 0 \\ (x_0 - x_9)^2 + x_{10}^2 + x_{11}^2 - d_9^2 = 0 \\ (x_1 - x_9)^2 + x_{10}^2 + (x_2 - x_{11})^2 - d_{10}^2 = 0 \\ (x_3 - x_9)^2 + (x_4 - x_{10})^2 + (x_5 - x_{11})^2 - d_{11}^2 = 0 \\ (x_6 - x_9)^2 + (x_7 - x_{10})^2 + (x_8 - x_{11})^2 - d_{12}^2 = 0. \end{cases} \tag{2}$$

System (2) has 12 equations in 12 variables. Furthermore, despite its sparseness, its total

degree and BKK bound equal 2^{12}. Therefore 4096 homotopy paths must be tracked to solve the system directly. Considering that each path is computed in one second, more than one hour would be required to solve the problem.

We apply Gaussian elimination to system (2) (Step 2(a)). The following steps are performed sequentially:

$$\begin{aligned} f_5 &:= f_5 - f_1 - f_2 \\ f_6 &:= f_6 - f_2 - f_3 \\ f_7 &:= f_7 - f_3 - f_4 \\ f_8 &:= f_8 - f_1 - f_4 \\ f_9 &:= f_9 - f_1 \\ f_{10} &:= f_{10} - f_2 - f_9 \\ f_{11} &:= f_{11} - f_3 - f_9 \\ f_{12} &:= f_{12} - f_4 - f_9. \end{aligned}$$

The resulting equations f_1, f_2, f_5, and f_8 can be eliminated (Step 2(b)) since the values of x_0, x_1, x_2, and x_6 are completely determined. We use rule *ppp* to decide the sign of x_0 and x_2. The total degree of the resulting system is 64.

We parameterize the variables appearing in bilinear equations (Step 2(c)). For instance, we can derive parametric expressions for x_5 (in terms of x_3) and x_{11} (in terms of x_9) from equations f_6 and f_{10}, respectively. The resulting system has only six quadratic equations, namely, f_3, f_4, f_7, f_9, f_{11}, and f_{12}, in the variables x_3, x_4, x_7, x_8, x_9, and x_{10}. Note that this step does not reduce the degree of the system any further.

Equations f_3, f_4, and f_9 are biquadratic, involving the pairs of variables (x_3, x_4), (x_7, x_8), and (x_9, x_{10}), respectively. Each pair can be parameterized in terms of sines and cosines of an angle θ_i, $i = 1, 2, 3$, $0 \leq \theta_i \leq 2\pi$ (Step 2(d)). Finally, we perform the standard trigonometric substitution (Step 2(e)). This step does not reduce the total degree of the system, but simplifies the structure of the system. The resulting core system

$$\begin{cases} (\alpha_1\, y_2^2 + \alpha_2)\, y_1^2 + \alpha_3\, y_2\, y_1 + \alpha_4\, y_2^2 + \alpha_5 = 0 \\ (\beta_1\, y_3^2 + \beta_2)\, y_1^2 + \beta_3\, y_3\, y_1 + \beta_4\, y_3^2 + \beta_5 = 0 \\ (\gamma_1\, y_3^2 + \gamma_2)\, y_2^2 + \gamma_3\, y_3\, y_2 + \gamma_4\, y_3^2 + \gamma_5 = 0, \end{cases} \tag{3}$$

has only three equations of degree 4 in y_1, y_2, and y_3. The coefficients α_i, β_i, and γ_i, $i = 1, \ldots, 5$ depend exclusively on the distance constraints and can be recomputed for different instances of the problem. Furthermore, given a solution of the core system, a solution of the original system, and, consequently, a realization of the problem can be easily computed (Durand, 1998). The core systems of problems $Octa_2, \ldots, Octa_8$ are obtained by following the same steps. Their structures are shown in Appendix A.

The total degree of the system (3) is 64 and its BKK bound is 16. Therefore, standard homotopy requires 64 paths to be tracked, and polyhedral homotopy, only 16. Moreover, we solved generic instances of system (3) using Continuum and found that 48 out of the 64 paths lead to solutions at infinity. Consequently, we can use cheater's homotopy to our advantage, by following only the paths leading to the remaining 16 affine solutions.

Selecting the core system for a specific constraint problem is not a deterministic procedure. The applicability of some symbolic reduction and simplification techniques depends strongly on the structure of the system, which, in its turn, relies on the algebraic representation selected for primitives and constraints involved in the problem. The framework

Table 2. Summary of the results of the application of homotopy continuation on a generic instance of $Octa_1$.

		$Octa_1$
Continuum Standard homotopy	# paths	64
	time (in seconds)	27
Continuum Cheater's homotopy	# paths	16
	time (in seconds)	2
PHC	# paths	16
	time (in seconds)	2
Solutions	Real	8
	Complex	8
	Geometric	8

introduced here provides a systematic tool to find the core systems and solve octahedral problems, in a way consistent with results previously reported in the literature (Nanua *et al.*, 1990; Hoffmann and Vermeer, 1995; Emiris and Mourrain, 1996).

As pointed out by one of the referees, computing the BKK bound can be as hard as solving the original system. We emphasize that computing the BKK bound for the systems is not part of the solution process. It is a valuable tool for selecting the core system. Once such a system is chosen, it can be used in a numeric context to solve various instances of the same problem.

Since α_i, β_i, and γ_i, $i = 1, \ldots, 5$ are functions of the distance constraints, we can determine a generic set of coefficients for system (3) by selecting random distance values. The resulting system and its solutions are then used in a continuation to solve any other $Octa_1$ problem (cheater's homotopy).

Table 2 summarizes the application of homotopy continuation to a generic instance of $Octa_1$ (Step 3). Continuum (using cheater's homotopy) and PHC can solve the problem in two seconds. System (3) has eight real and eight complex solutions. The number of *Geometric solutions* corresponds to the number of realizations. In this example, it equals the number of real solutions. Nevertheless, we point out, that the number of realizations may be different from the number of real solutions in some problems (Durand, 1998).

Figures 9–12 show four realizations of an instance of $Octa_1$ where:

$d_1 = 1.00796$	$d_2 = 1.15857$	$d_3 = 1.19071$	$d_4 = 1.18592$
$d_5 = 1.12482$	$d_6 = 1.16643$	$d_7 = 1.17417$	$d_8 = 1.17389$
$d_9 = 1.18117$	$d_{10} = 1.06129$	$d_{11} = 1.07569$	$d_{12} = 1.11983$

The other four realizations can be obtained from these by reflecting the solutions with respect to the xz-plane.

6. Discussion

6.1. application considerations

In our view, instance solvers that find only one solution, such as Newton–Raphson-based solvers, are not very well suited to geometric constraint solving. We believe that it is necessary, from time to time, to explore other solutions of an equation system, since the

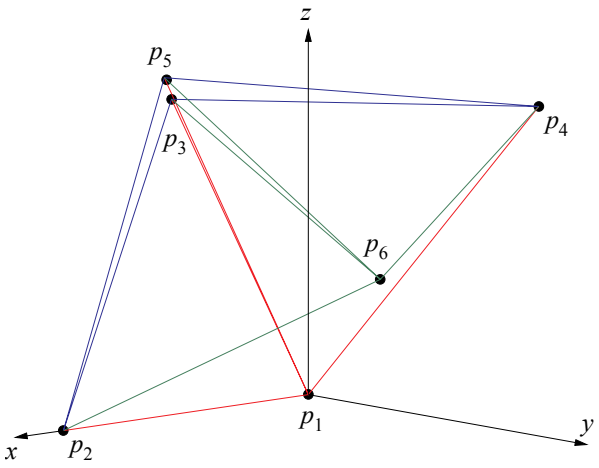

Figure 9. Realization #1 of a typical instance of $Octa_1$.

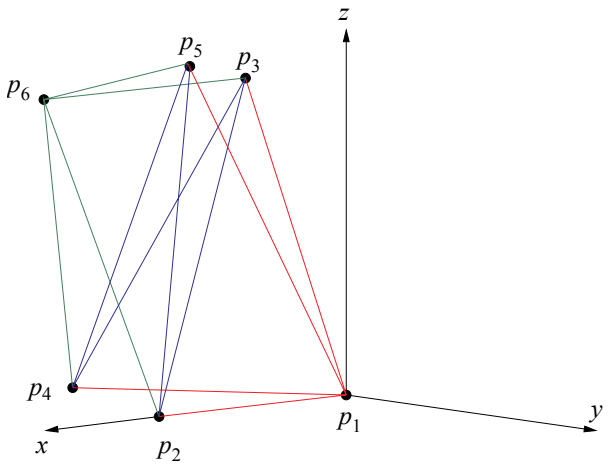

Figure 10. Realization #2 of a typical instance of $Octa_1$.

process of identifying a solution that realizes the application intent is not well understood and can have high computational complexity.

It has been argued that users of constraint solvers would probably present the input problem in a shape that is already close to the intended solution, and that this would lead with high probability to good starting values for iterative instance solvers. This argument is plausible in applications where the use of the solver is only for one-time problems. However, it is often the case that the input problem is understood as a generic design, and that different instances, or variants, are sought from different dimensional constraint values. In such a situation starting values for one instance, to a Newton iteration, are not necessarily good starting values for a different instance. However, as we pointed out,

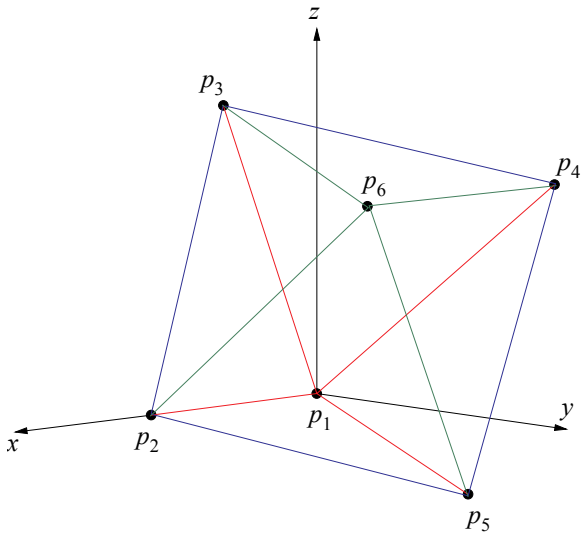

Figure 11. Realization #3 of a typical instance of $Octa_1$.

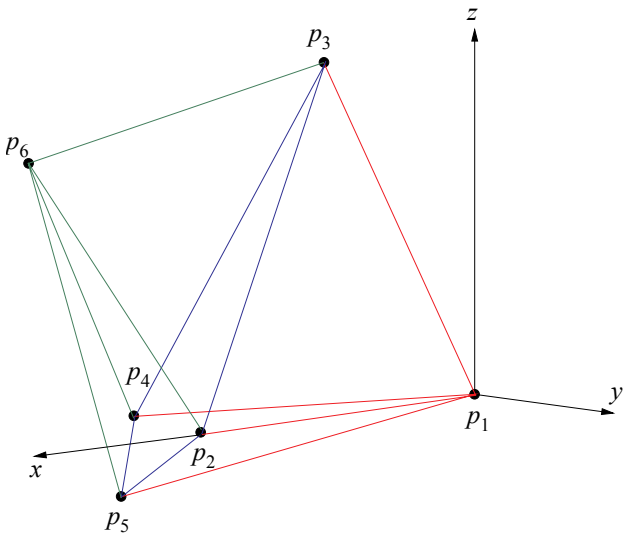

Figure 12. Realization #4 of a typical instance of $Octa_1$.

cheater's homotopy is ideally suited to that situation, because it leverages the knowledge of which paths lead to affine solutions. Other paths need not be re-evaluated. Thus, the techniques of this paper apply especially well to variational constraint problems in which different instances are constructed from the same input problem using various values for distance and angle constraints.

6.2. NONLINEAR EQUATIONS

General solutions to large-scale nonlinear equation systems are too complex, hence are not an attractive alternative. This has motivated us to approach the problem by decomposing it into patterns and devising solution templates. Restricting those patterns to small simultaneous problems involving only planes and points, the case we considered here, allows the systematic approach we have presented. This result has been foreshadowed in earlier work that approached the problem analysis with a pragmatic mixture of geometric reasoning and classical algebraic tools such as resultants.

Having a successful systematic analytical technique is encouraging, because a geometric reasoning approach must use specific individual properties of the problem, and is therefore hard to transfer to other problems with different combinations of geometric elements and different patterns of constraints between them. What is needed is a systematic approach that establishes a good methodology. This has been the objective of our work.

Instead of using elimination and reducing the numerical part to root finding, we opted to explore homotopy continuation. Our motivation is that the variable elimination computations needed to reduce the system to triangular form can become prohibitive. For example, a straightforward attack on octahedral problems without first reducing to the core system, using Gröbner bases, is at the limits of what can be computed with the current technology. Hence it does not lend itself to interactive spatial constraint solving. Clearly, future research is needed to expand the scope of problems amenable to systematic solution. This research could progress along the following lines.

BKK bounds work well for generic systems. However, as evident from the core system, the equations we eventually obtain using a systematic sequence of transformations have structure that could be exploited. In past research of this and related problems geometric reasoning was employed. It should be possible to focus exclusively on the algebraic structure instead, thereby unlocking a greater generality of solution techniques and making progress on some of the more challenging configurations with a richer set of geometric elements. Progress in this direction could help close the current gap of understanding the relationship between structure in the algebraic sense and geometric structure.

References

Aldefeld, B. (1988). Variation of geometries based on a geometric-reasoning method. *Comput. Aided Des.*, **20**, 117–126.

Allgower, E. L., Georg, K. (1990). *Numerical Continuation Methods. An Introduction.* New York, Springer-Verlag.

Allgower, E. L., Georg, K. (1993). Continuation and path following. *Acta Numerica*, **2**, 1–64.

Allgower, E. L., Georg, K. (1997). Numerical path following. In Ciarlet, P. G., Lions, J. L. eds, *Techniques of Scientific Computing (Part 2),* volume 5 of *Handbook of Numerical Analysis*, pp. 3–203. Amsterdam, North-Holland.

Betke, U. (1992). Mixed volumes of polytopes. *Arch. Math.*, **58**, 388–391.

Bonnesen, T., Fenchel, W. (1987). *Theory of Convex Bodies.* BCS Associates.

Borning, A. H. (1981). The programming language aspects of ThingLab, a constraint oriented simulation laboratory. *ACM TOPLAS*, **3**, 353–387.

Bouma, W., Fudos, I., Hoffmann, C. M., Cai, J., Paige, R. (1995). A geometric constraint solver. *Comput. Aided Des.*, **27**, 485–501.
Brüderlin, B. D. (1987). Rule-Based Geometric Modeling. Ph.D. Thesis, Swiss Federal Institute of Technology, ETH No. 8382.
Buchberger, B. (1965). An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (in German). Ph.D. Thesis, Institute of Mathematics, University of Innsbruck, Austria.
Buchberger, B. (1985). Gröbner bases: An algorithmic method in polynomial ideal theory. In Bose, N. K. ed., *Recent Trends in Multidimensional Systems Theory*, pp. 184–232. Dordrecht, D. Reidel.
Chou, S. C. (1988). An introduction to Wu's method for mechanical theorem proving in geometry. *J. Autom. Reasoning*, **4**, 237–267.
Cox, D., Little, J., O'Shea, D. (1992). *Ideals, Varieties and Algorithms*. New York, Springer-Verlag.
Crippen, G. M., Havel, T. F. (1988). *Distance Geometry and Molecular Conformation*. Somerset, England, Research Studies Press.
D-Cubed. (1994). *The Dimensional Constraint Manager*, Version 2.7. Cambridge, England.
Dennis, J. E., Schnabel, R. B. (1983). *Numerical Methods of Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ, Prentice-Hall.
Durand, C. (1998). Symbolic and Numerical Techniques for Constraint Solving. Ph.D. Thesis, Department of Computer Sciences, Purdue University, West Lafayette, IN, U.S.A.
Durand, C., Hoffmann, C. M. (1998). Continuum: A homotopy continuation solver for systems of algebraic equations. Technical Report TR 98-028, Department of Computer Sciences, Purdue University, West Lafayette, IN, U.S.A.
Dyer, M., Gritzmann, P., Hufnagel, A. (1998). On the complexity of computing mixed volumes. *SIAM J. Comput.*, **17**, 356–400.
Emiris, I. Z., Canny, J. F. (1995). Efficient incremental algorithm for the sparse resultant and the mixed volume. *J. Symb. Comput.*, **20**, 117–149.
Emiris, I. Z., Mourrain, B. (1996). Polynomial system solving and the case of the six-atom molecule. Technical Report 3075, INRIA.
Emiris, I. Z., Verschelde, J. (1997). How to count efficiently all affine roots of a polynomial system. Technical Report 3212, INRIA.
Fudos, I. (1995). Constraint Solving for Computer Aided Design. Ph.D. Thesis, Department of Computer Science, Purdue University, West Lafayette, IN, U.S.A.
Fudos, I., Hoffmann, C. M. (1993). Correctness of a geometric constraint solver. Technical Report CSD 93-076, Department of Computer Sciences, Purdue University, West Lafayette, IN, U.S.A..
Fudos, I., Hoffmann, C. M. (1996). Constraint-based parametric conics for CAD. *Comput. Aided Des.*, **28**, 91–100.
Gelfand, I. M. (1994). *Discriminants, Resultants, and Multidimensional Determinants*. Basel, Birkhauser.
Henneberg, C. M. (1968). Die Graphische Statik der Starren Systeme. Liepzig, 1911. Johnson reprint.
Hoffmann, C. M. (1989). *Solid and Geometric Modeling*. California, U.S.A., Morgan Kaufmann.
Hoffmann, C. M., Joan-Arinyo, R. (1997). Symbolic constraints in constructive geometry. *J. Symb. Comput.*, **23**, 287–300.
Hoffmann, C. M., Lomonosov, A., Sitharam, M. (1997a). Finding dense subgraphs of constraint graphs. In Smolka, G. ed., *Constraint Programming '97*, LNCS **1330**, pp. 463–478. New York, Springer-Verlag.
Hoffmann, C. M., Lomonosov, A., Sitharam, M. (1997b). Finding solvable subsets of constraint graphs. In Smolka, G. ed., LNCS **1330**, pp. 463–477. New York, Springer.
Hoffmann, C. M., Vermeer, P. J. (1994). Geometric constraint solving in R^2 and R^3. In *Computing in Euclidean Geometry*, 2^{nd} edn. Singapore, World Scientific Publishing.
Hoffmann, C. M., Vermeer, P. J. (1995). A spatial constraint problem. In *Proceedings of the Computational Kinematics Workshop, Nice, France, September 1995*. New York, ACM.
Huber, B. (1996). Solving Sparse Polynomial Systems. Ph.D. Thesis, Cornell University.
Huber, B., Sturmfels, B. (October 1995). A polyhedral method for solving sparse polynomial systems. *Math. Comput.*, **64**, 1541–1555.
Huber, B., Sturmfels, B. (1997). Bernstein's theorem in affine space. *Discrete Comput. Geom.*, **17**, 137–141.
Joan-Arinyo, R., Soto, A. (1997a). A correct rule-based geometric constraint solver. *Comput. Graph.*, **21**, 599–609.
Joan-Arinyo, R., Soto, A. (1997b). A ruler-and-compass geometric constraint solver. In Pratt, M. J., Sriram, R. D., Wozny, M. J. eds, *Product Modeling for Computer Integrated Design and Manufacture*, pp. 384–393. London, Chapman and Hall.
Kalkbrener, M. (1993). A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comput.*, **15**, 143–167.
Khovanskii, A. G. (1977). Newton polyhedra and toroidal varieties. *Funct. Anal. Appl.*, **11**, 289–296.

Khovanskii, A. G. (1978). Newton polyhedra and the genus of complete intersections. *Funktsional'nyi Analiz i Ego Prilozheniya*, **12**, 51–61.
Kondo, K. (1992). Algebraic method for manipulation of dimensional relationships in geometric models. *Comput. Aided Des.*, **24**, 141–147.
Kushnirenko, A. G. (1975). A Newton polytope and the number of solutions of a system of k equations in k unknowns. *Uspekhi Matematicheskikh Nauk.*, **30**, 266–267.
Lahaye, E. (1934). Une méthode de resolution d'une categorie d'equations transcendantes. *Comptes Rendus Séances Acad. Sci.*, **198**, 1840–1842.
Laman, G. (1970). On graphs and the rigidity of plane skeletal structures. *J. Eng. Math.*, **4**, 331–340.
Lamure, H., Michelucci, D. (1995). Solving geometric constraints by homotopy. In *Third Symposium on Solid Modeling and its Applications, Salt Lake City, UT*, pp. 263–269. New York, ACM.
Lazard, D. (1991). A new method for solving algebraic systems of positive dimension. *Discrete Appl. Math.*, **33**, 147–160.
Lazard, D. (1999). On theories of triangular sets. *J. Symb. Comput.*, **28**, 105–124.
Li, T. Y. (1997). Numerical solutions of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica*, **6**, 399–436.
Li, T. Y., Sauer, T., Yorke, J. A. (1989). The Cheater's homotopy: an efficient procedure for solving system of polynomial equations. *SIAM J. Numer. Anal.*, **26**, 1241–1251.
Manocha, D. (1993). Efficient algorithms for multipolynomial resultant. *Comput. J.*, **36**, 485–496. Special issue on Quantifier Elimination.
Manocha, D., Canny, J. (1993). Multipolynomial resultant algorithms. *J. Symb. Comput.*, **15**, 99–122.
Morgan, A. P. (1986a). A homotopy for solving polynomial systems. *Appl. Math. Comput.*, **18**, 87–92.
Morgan, A. P. (1986b). A transformation to avoid solutions at infinity for polynomial systems. *Appl. Math. Comput.*, **18**, 77–86.
Morgan, A. (1987). *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. New Jersey, Prentice-Hall.
Morgan, A. P. (1992). Polynomial continuation and its relationship to the symbolic reduction of polynomial systems. In Donald, B. R., Kapur, D., Mundy, J. L. eds, *Symbolic and Numerical Computation for Artificial Inteligence*, San Diego, Academic Press.
Morgan, A. P., Sommese, A. J. (1989). Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, **29**, 123–160.
Nanua, P., Waldron, K. J., Murthy, V. (1990). Direct kinematic solution of a Stewart platform. *IEEE Trans. Robot. Autom.*, **6**, 438–443.
Ortega, J., Rheinboldt, W. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. San Diego, Academic Press.
Owen, J. (1991). Algebraic solution for geometry from dimensional constraints. In *ACM Symposium on the Foundations of Solid Modeling, Austin, TX*, pp. 397–407. New York, ACM.
Patrikalakis, N. M. (1992). Surface-to-surface intersections. *IEEE Comput. Graph. Appl.*, **13**, 89–95.
Peitgen, H. O., Ritcher, P. H. (1986). *The Beauty of Fractals, Images of Complex Dynamical Systems*. New York, Springer-Verlag.
Ritt, J. F. (1932). *Differential Equations from the Algebraic Standpoint*. Providence, R.I., U.S.A., American Mathematical Society.
Ritt, J. F. (1950). *Differential Algebra*. Providence, R.I., U.S.A., American Mathematical Society.
Rojas, J. M. (1999). Solving degenerate sparse polynomial systems faster. *J. Symb. Comput.*, **28**, 155–186.
Schneider, R. (1993). *Convex Bodies*. Cambridge, Cambridge University Press.
Sederberg, T. W. (August 1983). Implicit and Parametric Curves and Surfaces. Ph.D. Thesis, Mechanical Engineering, Purdue University, West Lafayette, IN, U.S.A.
Stoer, J., Bulirsch, R. (1993). *Introduction to Numerical Analysis*, 2nd edn. New York, Springer-Verlag.
Sturmfels, B. (1997). Introduction to Resultants. *Lecture Notes at the AMS Short Course on Applications of Computational Algebraic Geometry, San Diego, January 1997*.
Verroust, A., Schonek, F., Roller, D. (1992). Rule-oriented method for parametrized computer-aided design. *Comput. Aided Des.*, **24**, 531–540.
Verschelde, J. (1996). Homotopy Continuation Methods for Solving Polynomial Systems. Ph.D. Thesis, Katholieke Universiteit Leuven.
Verschelde, J. (1997). PHCPACK. Available at `http://www.math.msu.edu./~jan/`.
Verschelde, J. (1997). PHCPACK: a general-purpose solver for polynomial systems by homotopy continuation. Technical Report TW265, Department of Computer Science, Katholieke Universiteit Leuven.
Verschelde, J. (1998). Numerical evidence for a conjecture in real algebraic geometry. Available at `http://www.math.msu.edu/~jan/`.
Verschelde, J., Gatermann, K., Cools, R. (1996). Mixed-volume computation by dynamic lifting applied to polynomial system solving. *Discrete Comput. Geom.*, **16**, 69–112.
Wampler, C. W., Morgan, A. P., Sommese, A. J. (1990). Numerical continuation methods for solving systems arising in kinematics. *J. Mech. Des.*, **112**, 59–68.
Wang, D. (1991). *On Wu's Method for Solving Systems of Algebraic Equations*, RISC-Linz Series 91-52.0. Austria, Johannes Kepler University.
Wang, D. (1998). Decomposing polynomial systems into simple systems. *J. Symb. Comput.*, **25**, 295–314.
Wu, W. (1986). Basic principles of mechanical theorem proving in elementary geometries. *J. Autom. Reasoning*, **2**, 221–252.
Wu, W. (1994). *Mechanical Theorem Proving in Geometries: Basic Principles*. New York, Springer-Verlag.

Appendix A. Structure of the Core System of the Remaining Octahedral Problems

In the following systems, the coefficients α_i, β_i, and γ_i depend solely on the constraint values and are different in each case.

$Octa_2$

$$\begin{cases} (\alpha_1\, y_2^2 + \alpha_2)\, y_1^2 + \alpha_3\, y_2\, y_1 + \alpha_4\, y_2^2 + \alpha_5 = 0 \\ (\beta_1\, y_3^2 + \beta_2)\, y_1^2 + \beta_3\, y_3\, y_1 + \beta_4\, y_3^2 + \beta_5 = 0 \\ (\gamma_1\, y_3^2 + \gamma_2)\, y_2^2 + \gamma_3\, y_3\, y_2 + \gamma_4\, y_3^2 + \gamma_5 = 0 \end{cases} \tag{A1}$$

$Octa_3$

$$\begin{cases} (y_2^2 + \alpha_1)\, y_1^2 + \alpha_2\, y_2\, y_1 + y_2^2 + \alpha_3 = 0 \\ (\beta_1\, y_3^2 + \beta_2)\, y_1^2 + \beta_3\, y_3\, y_1 + \beta_4\, y_3^2 + \beta_5 = 0 \\ (y_3^2 + \gamma_1)\, y_2^2 + \gamma_2\, y_3\, y_2 + y_3^2 + \gamma_3 = 0 \end{cases} \tag{A2}$$

$Octa_4$

$$\begin{cases} (\alpha_1\, y_2^2 + \alpha_2)\, y_1^2 + \alpha_3\, y_2\, y_1 + \alpha_4\, y_2^2 + \alpha_5 = 0 \\ (\beta_1\, y_3^2 + \beta_2)\, y_1^2 + \beta_3\, y_3\, y_1 + \beta_4\, y_3^2 + \beta_5 = 0 \\ (\gamma_1\, y_3^2 + \gamma_2)\, y_2^2 + \gamma_3\, y_3\, y_2 + \gamma_4\, y_3^2 + \gamma_5 = 0 \end{cases} \tag{A3}$$

$Octa_5$

$$\begin{cases} \alpha_1 + \alpha_2\, y_1 + \alpha_3\, y_2 + \alpha_4\, y_1^2 + \alpha_5\, y_2^2 + \alpha_6\, y_2\, y_1 = 0 \\ \beta_1 + \beta_2\, y_1 + \beta_3\, y_3 + \beta_4\, y_1^2 + \beta_5\, y_3^2 + \beta_6\, y_3\, y_1 = 0 \\ \gamma_1 + \gamma_2\, y_2 + \gamma_3\, y_3 + \gamma_4\, y_2^2 + \gamma_5\, y_3^2 + \gamma_6\, y_3\, y_2 = 0 \end{cases} \tag{A4}$$

$Octa_6$

$$\begin{cases} (y_2^2 + \alpha_1)\, y_1^2 + \alpha_2\, y_2\, y_1 + y_2^2 + \alpha_3 = 0 \\ (\beta_1\, y_3^2 + \beta_2)\, y_1^2 + \beta_3\, y_3\, y_1 + \beta_4\, y_3^2 + \beta_5 = 0 \\ \gamma_1\, y_2^2 + \gamma_2\, y_3\, y_2 + \gamma_3 = 0 \end{cases} \tag{A5}$$

$Octa_7$

$$\begin{cases} (\alpha_1 + \alpha_2 y_3 + \alpha_3 y_1) y_2 + \alpha_4 y_1 + \alpha_5 = 0 \\ \beta_1 y_2^2 + \beta_2 y_2 + \beta_3 = 0 \\ \gamma_1 y_3^2 + \gamma_2 y_3 + \gamma_3 = 0 \end{cases} \tag{A6}$$

$Octa_8$

$$\begin{cases} (\alpha_1 y_2^2 + \alpha_2) y_1^2 + \alpha_3 y_1 y_2 + \alpha_4 y_2^2 + \alpha_5 = 0 \\ \beta_1 y_1^2 + \beta_2 y_1 y_3 + \beta_3 = 0 \\ \gamma_1 y_2^2 + \gamma_2 y_2 y_3 + \gamma_3 = 0. \end{cases} \tag{A7}$$

Appendix B. Tetrahedral Problems

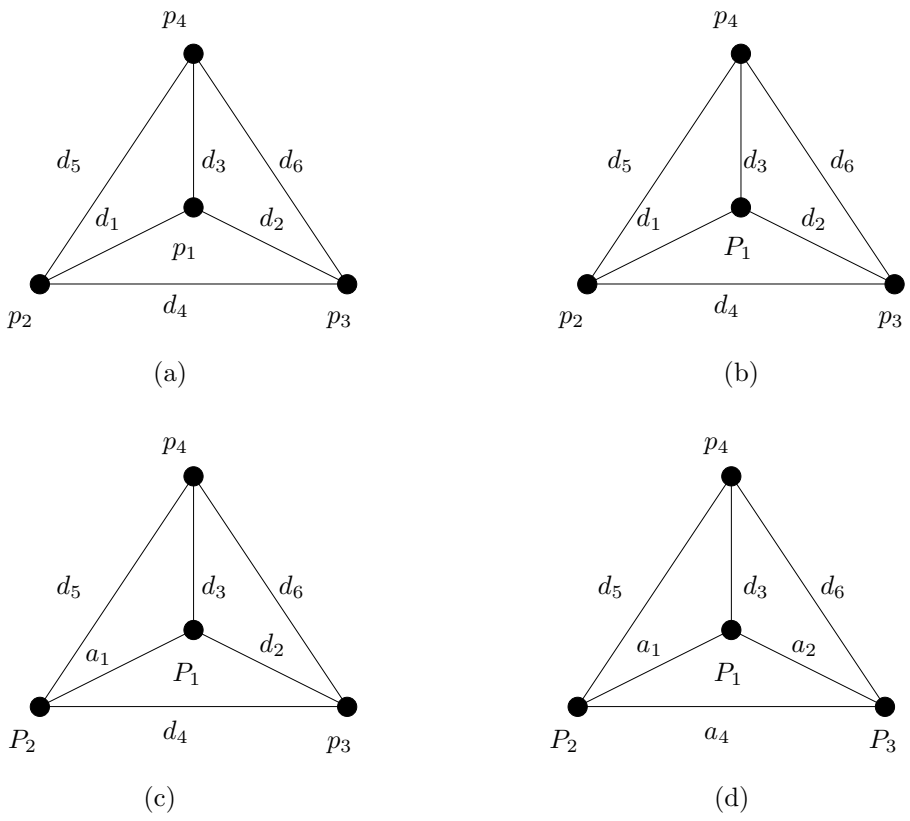

Figure 13. (a) $Tetra_1$, (b) $Tetra_2$, (c) $Tetra_3$ and (d) $Tetra_4$.

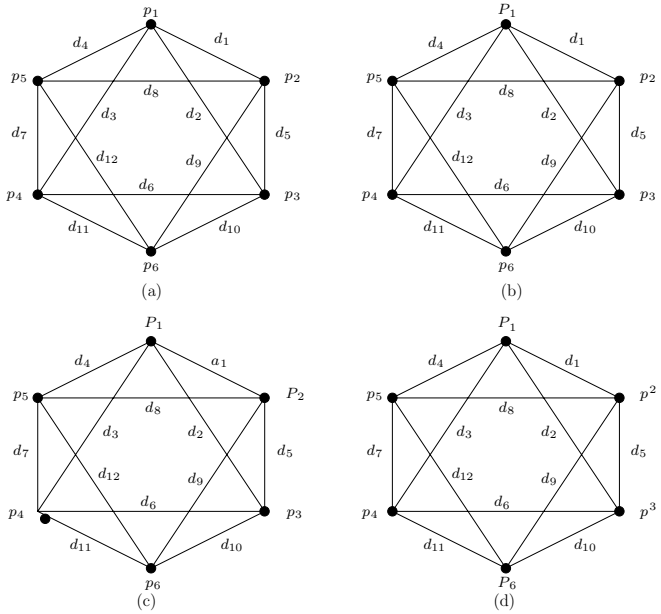

Figure 14. (a) $Octa_1$, (b) $Octa_2$, (c) $Octa_3$, (d) $Octa_4$.

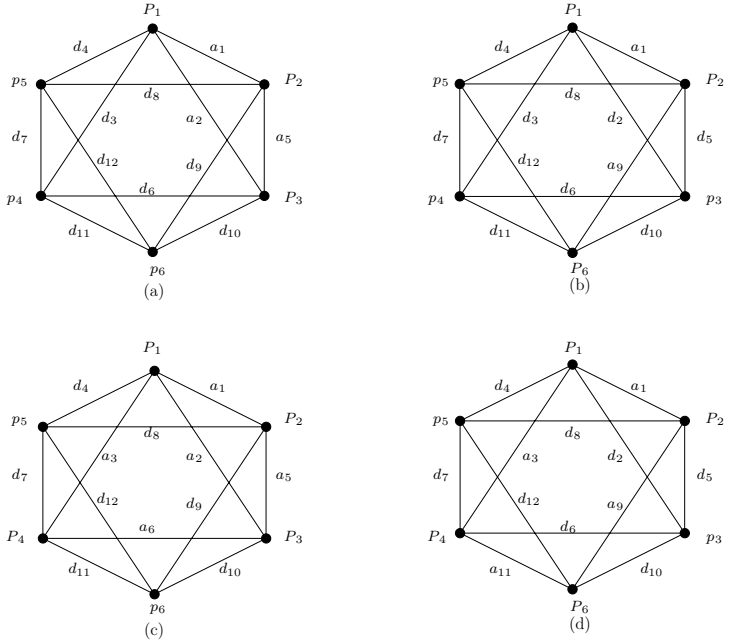

Figure 15. (a) $Octa_5$, (b) $Octa_6$, (c) $Octa_7$, (d) $Octa_8$.

Appendix C. Octahedral Problems

- $Octa_1$: six points.
- $Octa_2$: five points and one plane.

- $Octa_3$: four points and two planes that are adjacent in the constraint graph.
- $Octa_4$: four points and two planes that are not adjacent in the constraint graph.
- $Octa_5$: three points and three planes that form a triangle in the constraint graph.
- $Octa_6$: three points and three planes that form a path in the constraint graph.
- $Octa_7$: four planes and two points that are adjacent in the constraint graph.
- $Octa_8$: four planes and two points that are not adjacent in the constraint graph.

Originally Received 27 July 1999
Accepted 26 April 2000