A Systematic Framework for Solving Geometric Constraints Analytically

Cassiano Durand Christoph M. Hoffmann

Computer Science Department
Purdue University
West Lafayette, IN 47907-1398, USA
{crbd,cmh}@cs.purdue.edu

June 19, 2000

Abstract

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; [Dur98]. We are especially interested in applications in solid modeling, hence we concentrate on solvers that have to tackle nonlinear problems to satisfy 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., [HV94].

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, e.g., [Fud95].

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 in 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 [Dur98] 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., [HV94], 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 [SB93, OR70]. 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[1] for the Newton-Raphson method are fractals [PR86]. Therefore, if the sketch is used as the initial approximation, then it should nearly 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 [DS83]. 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 [CH88], 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 how the subsequent guesses might be made.

Homotopy continuation methods can be used to circumvent this problem [AG90, Li97, Roj99]. Continuation methods are robust and versatile global methods capable of finding *all* solutions of a given system [AG93]. 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 [Mor87, WMS90, Pat92, Ver96, Ver97b, HS95, Hub96] and, more recently, to constraint solving [LM95]. Albeit powerful, homotopy requires a significant amount of computational work, usually limiting the set of solvable systems to those which are "small". Fortunately,

[1] 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

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 [Mor92].

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* [Buc65, Buc85, CLO92]. 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 [CLO92, Hof89]. In [Kon92], Gröbner bases are used in constraint solving.

The second approach is based on Ritt's construction of characteristic sets (also referred as triangular sets) [Rit32, Rit50], a technique rediscovered and extended by Wu in the context of mechanical geometry theorem proving [Wu86, Wu94, Cho88]. This method decomposes the solution set of an algebraic system into set-expressions involving the solutions of simpler systems. It is argued in [Wan91] that the method can be used to solve a large number of systems found in the current literature. Wang ([Wan98]) 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 ([Laz91]) and Kalkbrener ([Kal93]) 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 [Laz99].

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 [Gel94, Stu97]. Sederberg uses this method in the context of curve and surface modeling [Sed83]. In [Man93, MC93], sparse resultants are used to compute the solutions of polynomial systems. Emiris and Mourrain [EM96, EV97] 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 [BFH+95] 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 [FH93]. In [FH96], Fudos and Hoffmann describe how to construct conic blending arcs from constraints using the same approach. In [HJA97], a method that combines graph-based and numerical techniques is presented.

DCM [D-C94] 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 [Owe91].

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.

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 [Bor81], where the rules are classes in Smalltalk associated with methods that can be invoked to solve the constraints. Brüderlin [Brü87] 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 [Ald88] 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 [VSR92] describes an approach capable of modeling dimensional, tangency and radius constraints. The sketch is represented by a set of mutually

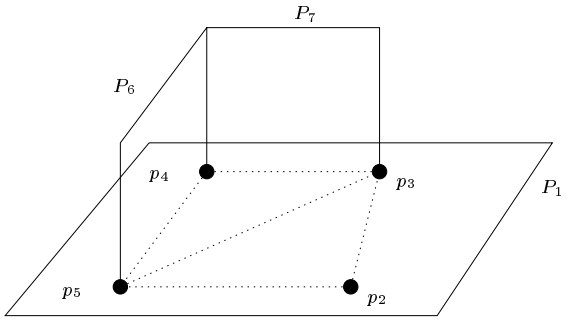

Figure 1: Sketch involving planes P_1, P_6, and P_7, and points p_2, p_3, p_4, and p_5.

constrained distances (CD sets) and angles (CA sets) which are evaluated simultaneously. Joan-Arinyo and Soto [JAS97a, JAS97b] provide a correctness proof of a method based on an extension of the repertoire of the rules presented in [VSR92].

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 $p, p_1, ...$ and planes by $P, P_1, ...$ 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 p_2, p_3, p_4, and p_5. Figure 2 shows the corresponding constraint graph. Edges labeled d_{ij} or a_{ij} denote distance or an angle constraints 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 4 points p_1, p_2, p_3, and p_4, constrained by distances. The labels on the edges

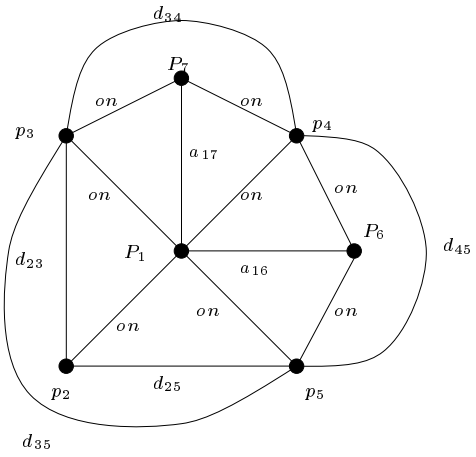

Figure 2: Constraint graph for the sketch of figure 1.

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)

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

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 *over-constrained*. If the solutions satisfy the deleted constraints, the over-constrained 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 [Dur98].

In the plane, Laman's theorem [Lam70] provides a basis for such a test, but it is restricted to primitives with 3 degrees of freedom, like points, planes, and circles with fixed radii. Another characterization is based on Henneberg n-sequences [Hen11], which also leads to an algorithm for finding realizations for a restricted class of graphs, called 2-*simple* or *sequentially constructible*.

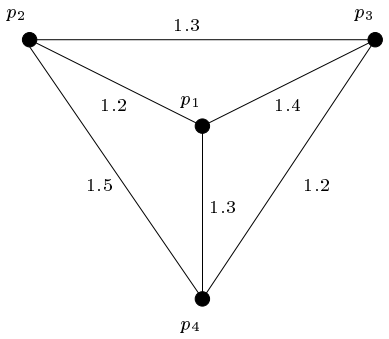

Figure 3: Constraint graph involving 4 points p_1, p_2, p_3, and p_4, constrained by distances.

In [BFH+95], 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 3 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 [CH88].

Hoffmann *et al.* [HLS97a, HLS97b] 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 [HV94], the algorithm from [BFH+95] 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 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 paper [HV94] 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 tetra-

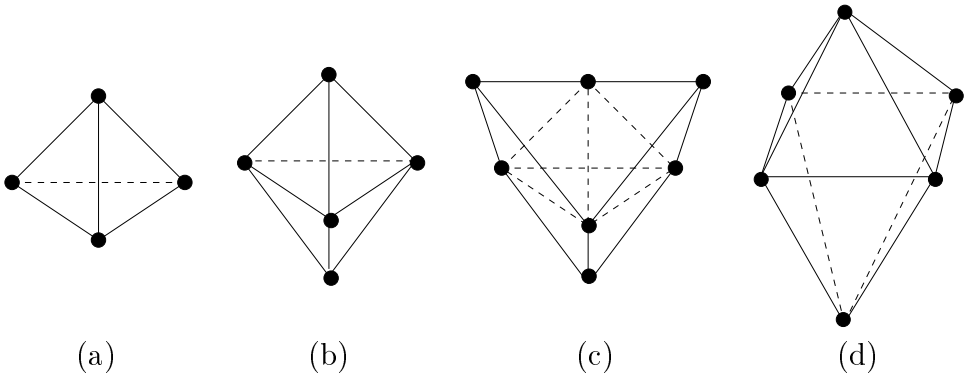

Figure 4: Tetrahedron (a), Double tetrahedron (b), Decahedron (c) and Octahedron (d).

hedra. 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 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 3 planes are under-constrained. The *Tetra* family of problems can be solved directly by using many analytical methods (see [Dur98]).

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 4 planes are under-constrained 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

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

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. Notice 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

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

then the algebraic system associated with $Tetra_3$ is

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

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.

3.5 Placement Rules

The solutions of well-constrained problems are in general rigid realizations with 6 degrees of freedom (3 translational and 3 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 3 primitives constrained with respect to each other, i.e. forming a triangle on the constraint graph [HV94]. 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 3 points (Rule *ppp*)

Let p_1, p_2 and p_3 be 3 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}.

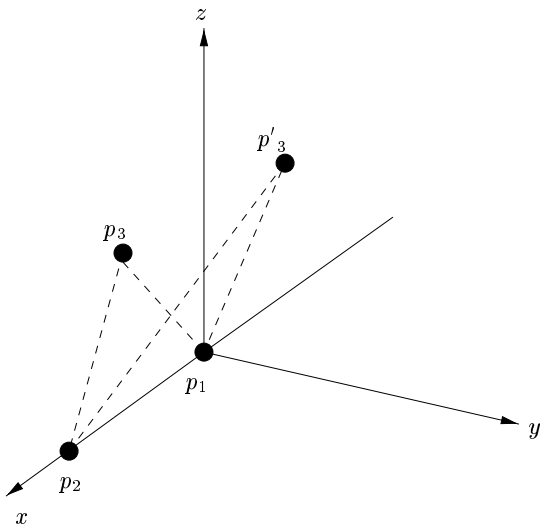

Figure 5: Placement of 3 points.

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

$$\begin{aligned} p_1 &: (0,0,0) \\ p_2 &: (x_0,0,0) \\ p_3 &: (x_1,0,x_2). \end{aligned}$$

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 2 points and 1 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

$$\begin{aligned} P_1 &: (0,0,1:0) \\ p_2 &: (0,0,x_0) \\ p_3 &: (x_1,0,x_2). \end{aligned}$$

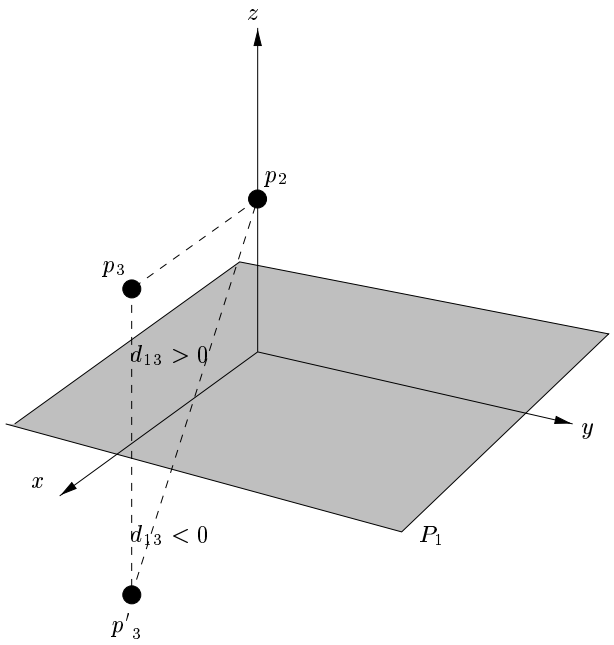

Figure 6: Placement of 2 points and 1 plane.

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 1 point and 2 planes (Rule pPP)

Let P_1 and P_2 be 2 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

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

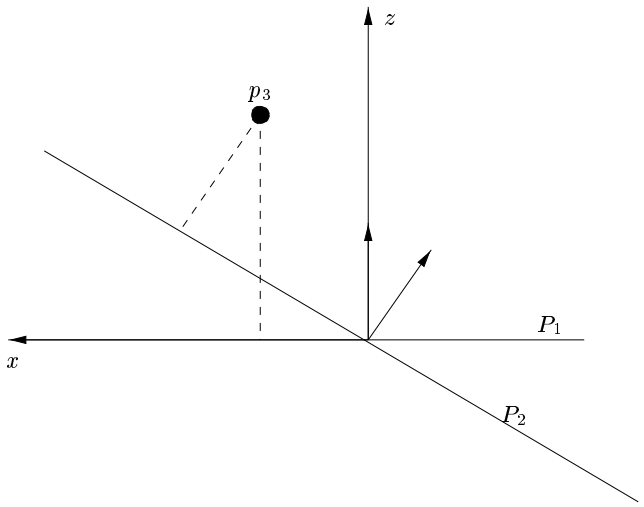

Figure 7: Placement of 2 planes and 1 point. Only the projection on the plane xy is shown.

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

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

Placement of 3 planes (Rule PPP)

Let P_1, P_2 and P_3 be 3 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 satisfy the angle constraints a_{13} and a_{23}.

Therefore their coordinates can be represented generically by

$$\begin{aligned} P_1 &: (0, 0, 1 : 0) \\ P_2 &: (x_0, 0, x_1 : 0) \\ P_3 &: (x_2, x_3, x_4 : 0). \end{aligned}$$

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.

$$x_1 = cos(a_{12}) \qquad x_0 = \sqrt{1 - x_0} = \sin(a_{12}) \qquad x_4 = cos(a_{13})$$

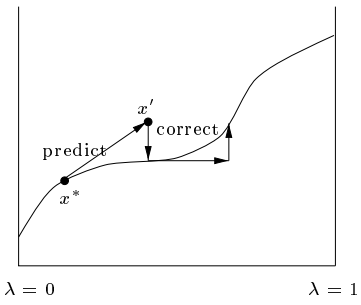

Figure 8: Predictor-corrector scheme. x^* is a point on the homotopy path and x' is the predicted point.

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

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 [Lah34].

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 under-determined and implicitly defines a curve in $C^n \times [0, 1)$, the *homotopy path*. The term *homotopy continuation* refers to a set 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 [AG90, AG97].

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 $F(x) = 0$ [Mor87]. 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. 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* theorem [Mor87], which states that the number of isolated solutions of $F(x) = 0$ is bounded above by its total degree. Morgan [Mor86a] describes how to build a generic start system, whose number of solutions is equal to the total degree of the target system. In [Mor86b], 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* [Ver96]. 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 [DGH98, Kho78, Kho77, Kus75]). Mixed volumes can be computed by several methods, in particular [DGH98, HS95, EC95, VGC96]. For additional theoretical background and standard tools refer to [Sch93, Bet92, BF87, DGH98].

Sparse elimination also provides the basis for solving systems of equations by continuation [HS95, VGC96, HS97]. 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 [Ver96].

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 [MS89] 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* [LSY89, Ver98], 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 [MS89] for further detail.

4.3 Implementations Used

We use two software packages that implement different flavors of homotopy: *Continuum* [DH98], which uses the projective approach, and *PHC* [Ver97b, Ver97a], which implements polyhedral homotopy.

5 Solving the Octahedron

Some octahedral problems have been studied in different contexts, from kinematics [NWM90] to computational chemistry [EM96]. 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.

Because of its topological symmetry, any of the 8 triangular faces of the octahedron can be selected to be placed, and this gives some flexibility choosing a placement order that leads to the simplest associated system. We found that placing the faces with most 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 4-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 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 by their corresponding parametric expressions.

   (d) Parameterize the variables appearing in all bivariate quadratic equations (using *sin* and *cos*) and replace them by 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 3 points according rule *ppp* defined in section 3.5. The primitives can be represented in terms of coordinates by:

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

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 1 second, more than one hour would be required to solve the problem.

We apply Gaussian Elimination to system (2) (Step 2a). 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 2b) 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 2c). 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 6 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}. Notice 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 2d). Finally, we perform the standard trigonometric substitution (Step 2e). 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 3 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 [Dur98]. The core systems of problems Octa_2,...,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 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 [HV95, NWM90, EM96].

As pointed out by one of the referees, computing the BKK bound can be as hard as solvinig 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 2 seconds. System (3) has 8 real and 8 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 [Dur98].

Figures 9, 10, 11, and 12 show 4 realizations of an instance of $Octa_1$ where:

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

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

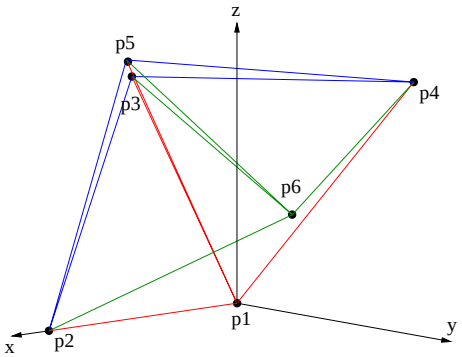

Figure 9: Realization #1 of a typical instance of $Octa_1$.

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

The other 4 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 process of identifying a solution that realizes

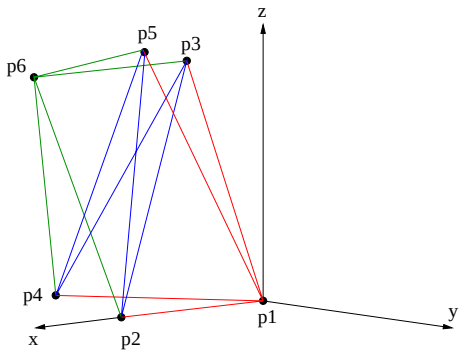

Figure 10: Realization #2 of a typical instance of $Octa_1$.

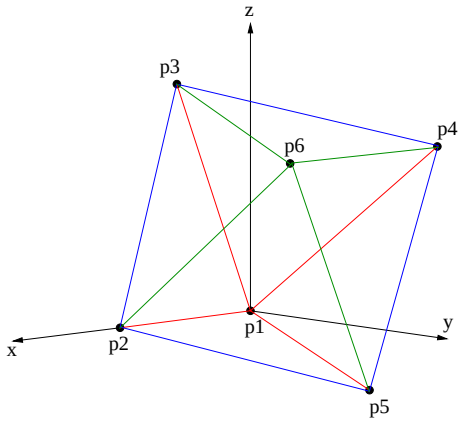

Figure 11: Realization #3 of a typical instance of $Octa_1$.

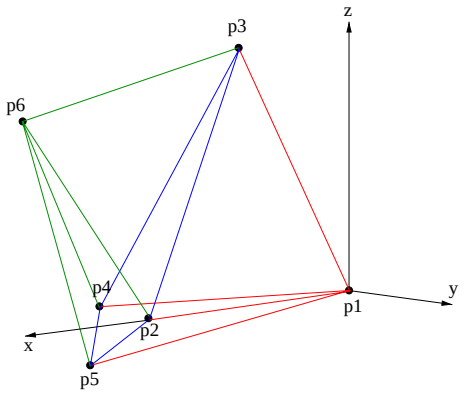

Figure 12: Realization #4 of a typical instance of $Octa_1$.

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, 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

[AG90] E.L. Allgower and K. Georg. *Numerical Continuation Methods, an Introduction*. Springer-Verlag, 1990.

[AG93] E.L. Allgower and K. Georg. Continuation and Path Following. *Acta Numerica*, pages 1–64, 1993.

[AG97] E.L. Allgower and K. Georg. Numerical Path Following. In P.G. Ciarlet and JL. Lions, editors, *Techniques of Scientific Computing (Part 2)*, volume 5 of *Handbook of Numerical Analysis*, pages 3–203. North-Holland, 1997.

[Ald88] B. Aldefeld. Variation of Geometries Based on a Geometric-Reasoning Method. *Computer Aided Design*, 20(3):117–126, 1988.

[Bet92] U. Betke. Mixed Volumes of Polytopes. *Archiv der Mathematik*, 58:388–391, 1992.

[BF87] T. Bonnesen and W. Fenchel. *Theory of Convex Bodies*. BCS Associates, 1987.

[BFH+95] W. Bouma, I. Fudos, C.M. Hoffmann, J. Cai, and R. Paige. A Geometric Constraint Solver. *Computer Aided Design*, 27(6):487–501, June 1995.

[Bor81] A.H. Borning. The Programming Language Aspects of ThingLab, a Constraint Oriented Simulation Laboratory. *ACM TOPLAS*, 3(4):353–387, 1981.

[Brü87] B.D. Brüderlin. *Rule-Based Geometric Modeling*. PhD thesis, Swiss Federal Institute of Technology, 1987. ETH No. 8382.

[Buc65] B. Buchberger. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (in German)*. PhD thesis, Institute of Mathematics, University of Innsbruck, Austria, 1965.

[Buc85] B. Buchberger. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In N.K. Bose, editor, *Recent Trends in Multidimensional Systems Theory*, pages 184–232. D. Reidel, 1985.

[CH88] G.M. Crippen and T.F. Havel. *Distance Geometry and Molecular Conformation*. Research Studies Press, 1988.

[Cho88] S.C. Chou. An Introduction to Wu's Method for Mechanical Theorem Proving in Geometry. *Journal of Automated Reasoning*, 4:237–267, 1988.

[CLO92] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms*. Springer-Verlag, 1992.

[D-C94] D-Cubed. *The Dimensional Constraint Manager*. Cambridge, England, 1994. Version 2.7.

[DGH98] M. Dyer, P. Gritzmann, and A. Hufnagel. On The Complexity of Computing Mixed Volumes. *SIAM Journal on Computing*, 17(2):356–400, 1998.

[DH98] C. Durand and C. M. Hoffmann. Continuum: A Homotopy Continuation Solver for Systems of Algebraic Equations. Technical Report TR 98-028, Department of Computer Sciences, Purdue University, 1998.

[DS83] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[Dur98] C. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA, 1998.

[EC95] I.Z. Emiris and J.F. Canny. Efficient Incremental Algorithm for the Sparse Resultant and the Mixed Volume. *Journal of Symbolic Computation*, 20:117–149, 1995.

[EM96] I.Z. Emiris and B. Mourrain. Polynomial System Solving and the Case of the Six-Atom Molecule. Technical Report 3075, INRIA, 1996.

[EV97] I.Z. Emiris and J. Verschelde. How to Count Efficiently all Affine Roots of a Polynomial System. Technical Report 3212, INRIA, 1997.

[FH93] I. Fudos and C.M. Hoffmann. Correctness of a Geometric Constraint Solver. Technical Report CSD 93-076, Department of Computer Sciences, Purdue University, 1993.

[FH96] I. Fudos and C. M. Hoffmann. Constraint-Based Parametric Conics for CAD. *Computer Aided Design*, 28(2):91–100, 1996.

[Fud95] I. Fudos. *Constraint Solving for Computer Aided Design*. PhD thesis, Department of Purdue University, Purdue University, December 1995.

[Gel94] I.M. Gelfand. *Discriminants, Resultants, and Multidimensional Determinants*. Birkhauser, 1994.

[Hen11] L. Henneberg. *Die Graphische Statik der Starren Systeme*. Liepzig, 1911. Johnson reprint, 1968.

[HJA97] C.M. Hoffmann and R. Joan-Arinyo. Symbolic Constraints in Constructive Geometry. *Journal of Symbolic Computation*, 23:287–300, 1997.

[HLS97a] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding dense subgraphs of constraint graphs. In G. Smolka, editor, *Constraint Programming '97, Lecture Notes in Computer Science 1330*, pages 463–478. Springer-Verlag, 1997.

[HLS97b] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding Solvable Subsets of Constraint Graphs. In G. Smolka, editor, *Springer LNCS 1330*, pages 463–477, 1997.

[Hof89] C.M. Hoffmann. *Solid and Geometric Modeling*. Morgan Kaufmann, 1989.

[HS95] B. Huber and B. Sturmfels. A Polyhedral Method for Solving Sparse Polynomial Systems. *Mathematics of Computation*, 64(212):1541–1555, October 1995.

[HS97] B. Huber and B. Sturmfels. Bernstein's Theorem in Affine Space. *Discrete and Computational Geometry*, 17:137–141, 1997.

[Hub96] B. Huber. *Solving Sparse Polynomial Systems*. PhD thesis, Cornell University, 1996.

[HV94] C.M. Hoffmann and P.J. Vermeer. Geometric Constraint Solving in R^2 and R^3. In *Computing in Euclidean Geometry*. World Scientific Publishing, 2nd edition, 1994.

[HV95] C.M. Hoffmann and P.J. Vermeer. A Spatial Constraint Problem. In *Prooceedings of the Computational Kinematics Workshop*, Nice, France, September 1995. ACM.

[JAS97a] R. Joan-Arinyo and A. Soto. A Correct Rule-Based Geometric Constraint Solver. *Computer and Graphics*, 21(5):599–609, 1997.

[JAS97b] R. Joan-Arinyo and A. Soto. A Ruler-and-Compass Geometric Constraint Solver. In M.J.Pratt, R.D.Sriram, and M.J.Wozny, editors, *Product Modeling for Computer Integrated Design and Manufacture*, pages 384–393. Chapman and Hall, London, 1997.

[Kal93] M. Kalkbrener. A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties. *Journal of Symbolic Computation*, 15:143–167, 1993.

[Kho77] A.G. Khovanskii. Newton Polyhedra and Toroidal Varieties. *Functional Analysis and its Applications*, 11:289–296, 1977.

[Kho78] A.G. Khovanskii. Newton Polyhedra and the genus of Complete Intersections. *Funktsional'nyi Analiz i Ego Prilozheniya*, 12(1):51–61, 1978.

[Kon92] K. Kondo. Algebraic Method for Manipulation of Dimensional Relationships in Geometric Models. *Computer Aided Design*, 24(3):141–147, 1992.

[Kus75] A.G. Kushnirenko. A Newton Polytope and the Number of Solutions of a System of k Equations in k Unknowns. *Uspekhi Matematicheskikh Nauk.*, 30(2):266–267, 1975.

[Lah34] E. Lahaye. Une Méthode de Resolution d'une Categorie d'Equations Transcendantes. *Comptes Rendus des Séances de l'Acadmie des Sciences*, 198:1840–1842, 1934.

[Lam70] G. Laman. On Graphs and the Rigidity of Plane Skeletal Structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.

[Laz91] D. Lazard. A New Method for Solving Algebraic Systems of Positive Dimension. *Discrete Applied Mathematics*, 33:147–160, 1991.

[Laz99] D. Lazard. On Theories of Triangular Sets. *Journal of Symbolic Computation*, 28:105–124, 1999.

[Li97] T.Y. Li. Numerical Solutions of Multivariate Polynomial Systems by Homotopy Continuation Methods. *Acta Numerica*, 6:399–436, 1997.

[LM95] H. Lamure and D. Michelucci. Solving Geometric Constraints by Homotopy. In *Third Symposium on Solid Modeling and its Applications*, pages 263–269, Salt Lake City, Utah, 1995. ACM.

[LSY89] T.Y. Li, T. Sauer, and J.A. Yorke. The Cheater's Homotopy: an Efficient Procedure for Solving System of Polynomial Equations. *SIAM Journal of Numerical Analysis*, 26(5):1241–1251, 1989.

[Man93] D. Manocha. Efficient Algorithms for Multipolynomial Resultant. *The Computer Journal*, 36(5):485–496, 1993. Special issue on Quantifier Elimination.

[MC93] D. Manocha and J. Canny. Multipolynomial Resultant Algorithms. *Journal of Symbolic Computation*, 15(2):99–122, 1993.

[Mor86a] A.P. Morgan. A Homotopy for Solving Polynomial Systems. *Applied Mathematics and Computation*, 18:87–92, 1986.

[Mor86b] A.P. Morgan. A Transformation to Avoid Solutions at Infinity for Polynomial Systems. *Applied Mathematics and Computation*, 18:77–86, 1986.

[Mor87] Alexander Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. Prentice-Hall, 1987.

[Mor92] A.P. Morgan. Polynomial Continuation and its Relationship to the Symbolic Reduction of Polynomial Systems. In B.R. Donald, D. Kapur, and J.L. Mundy, editors, *Symbolic and Numerical Computation for Artificial Inteligence*. Academic Press, 1992.

[MS89] A.P. Morgan and A.J. Sommese. Coefficient-Parameter Polynomial Continuation. *Applied Mathematics and Computation*, 29:123–160, 1989.

[NWM90] P. Nanua, K.J. Waldron, and V. Murthy. Direct Kinematic Solution of a Stewart Platform. *IEEE Transactions on Robotics and Automation*, 6(4):438–443, August 1990.

[OR70] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.

[Owe91] J. Owen. Algebraic Solution for Geometry from Dimensional Constraints. In *ACM Symposium on the Foundations of Solid Modeling*, pages 397–407, Austin, Texas, 1991.

[Pat92] N.M. Patrikalakis. Surface-to-Surface Intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, 1992.

[PR86] H.O. Peitgen and P.H. Ritcher. *The Beauty of Fractals, Images of Complex Dynamical Systems*. Springer-Verlag, 1986.

[Rit32] J.F. Ritt. *Differential Equations from the Algebraic Standpoint*. American Mathematical Society, 1932.

[Rit50] J.F. Ritt. *Differential Algebra*. American Mathematical Society, 1950.

[Roj99] J. M. Rojas. Solving Degenerate Sparse Polynomial Systems Faster. *Journal of Symbolic Computation*, 28(1–2):155–186, 1999.

[SB93] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 2nd edition, 1993.

[Sch93] R. Schneider. *Convex Bodies*. Cambridge University Press, 1993.

[Sed83] T.W. Sederberg. *Implicit and Parametric Curves and Surfaces*. PhD thesis, Mechanical Engineering, Purdue University, August 1983.

[Stu97] B. Sturmfels. Introduction to Resultants. Lecture notes at the AMS short course on Applications of Computational Algebraic Geometry, San Diego, January 1997.

[Ver96] J. Verschelde. *Homotopy Continuation Methods for Solving Polynomial Systems.* PhD thesis, Katholieke Universiteit Leuven, 1996.

[Ver97a] J. Verschelde. PHCPACK, 1997. Available at http://www.math.msu.edu/~ jan/.

[Ver97b] J. Verschelde. PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation. Technical Report TW 265, Department of Computer Science, Katholieke Universiteit Leuven, 1997.

[Ver98] J. Verschelde. Numerical Evidence for a Conjecture in Real Algebraic Geometry, 1998. Available at http://www.math.msu.edu/~ jan/.

[VGC96] J. Verschelde, K. Gatermann, and R. Cools. Mixed-volume Computation by Dynamic Lifting Applied to Polynomial System Solving. *Discrete Computational Geometry*, 16(1):69–112, 1996.

[VSR92] A. Verroust, F. Schonek, and D. Roller. Rule-Oriented Method for Parametrized Computer-Aided Design. *Computer Aided Design*, 24(10):531–540, 1992.

[Wan91] D. Wang. On Wu's Method for Solving Systems of Algebraic Equations. RISC-Linz Series 91-52.0, Johannes Kepler University, Austria, 1991.

[Wan98] D. Wang. Decomposing Polynomial Systems into Simple Systems. *Journal of Symbolic Computation*, 25:295–314, 1998.

[WMS90] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical Continuation Methods for Solving Systems arising in Kinematics. *Journal of Mechanical Design*, 112:59–68, 1990.

[Wu86] W. Wu. Basic Principles of Mechanical Theorem Proving in Elementary Geometries. *Journal of Automated Reasoning*, 2:221–252, 1986.

[Wu94] W. Wu. *Mechanical Theorem Proving in Geometries: Basic Principles.* Springer-Verlag, 1994.

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$_2$

$$\left\{\begin{array}{lcl}(\alpha_1\,y_2^2+\alpha_2)\,y_1^2+\alpha_3\,y_2\,y_1+\alpha_4\,y_2^2+\alpha_5 &=& 0\\ (\beta_1\,y_3^2+\beta_2)\,y_1^2+\beta_3\,y_3\,y_1+\beta_4\,y_3^2+\beta_5 &=& 0\\ (\gamma_1\,y_3^2+\gamma_2)\,y_2^2+\gamma_3\,y_3\,y_2+\gamma_4\,y_3^2+\gamma_5 &=& 0\end{array}\right. \tag{4}$$

Octa$_3$

$$\left\{\begin{array}{lcl}(y_2^2+\alpha_1)\,y_1^2+\alpha_2\,y_2\,y_1+y_2^2+\alpha_3 &=& 0\\ (\beta_1\,y_3^2+\beta_2)\,y_1^2+\beta_3\,y_3\,y_1+\beta_4\,y_3^2+\beta_5 &=& 0\\ (y_3^2+\gamma_1)\,y_2^2+\gamma_2\,y_3\,y_2+y_3^2+\gamma_3 &=& 0\end{array}\right. \tag{5}$$

Octa$_4$

$$\left\{\begin{array}{lcl}(\alpha_1\,y_2^2+\alpha_2)\,y_1^2+\alpha_3\,y_2\,y_1+\alpha_4\,y_2^2+\alpha_5 &=& 0\\ (\beta_1\,y_3^2+\beta_2)\,y_1^2+\beta_3\,y_3\,y_1+\beta_4\,y_3^2+\beta_5 &=& 0\\ (\gamma_1\,y_3^2+\gamma_2)\,y_2^2+\gamma_3\,y_3\,y_2+\gamma_4\,y_3^2+\gamma_5 &=& 0\end{array}\right. \tag{6}$$

Octa$_5$

$$\left\{\begin{array}{lcl}\alpha_1+\alpha_2\,y_1+\alpha_3\,y_2+\alpha_4\,y_1^2+\alpha_5\,y_2^2+\alpha_6\,y_2\,y_1 &=& 0\\ \beta_1+\beta_2\,y_1+\beta_3\,y_3+\beta_4\,y_1^2+\beta_5\,y_3^2+\beta_6\,y_3\,y_1 &=& 0\\ \gamma_1+\gamma_2\,y_2+\gamma_3\,y_3+\gamma_4\,y_2^2+\gamma_5\,y_3^2+\gamma_6\,y_3\,y_2 &=& 0\end{array}\right. \tag{7}$$

Octa$_6$

$$\left\{\begin{array}{lcl}(y_2^2+\alpha_1)\,y_1^2+\alpha_2\,y_2\,y_1+y_2^2+\alpha_3 &=& 0\\ (\beta_1\,y_3^2+\beta_2)\,y_1^2+\beta_3\,y_3\,y_1+\beta_4\,y_3^2+\beta_5 &=& 0\\ \gamma_1\,y_2^2+\gamma_2\,y_3\,y_2+\gamma_3 &=& 0\end{array}\right. \tag{8}$$

Octa$_7$

$$\left\{\begin{array}{lcl}(\alpha_1+\alpha_2y_3+\alpha_3y_1)y_2+\alpha_4y_1+\alpha_5 &=& 0\\ \beta_1y_2^2+\beta_2y_2+\beta_3 &=& 0\\ \gamma_1y_3^2+\gamma_2y_3+\gamma_3 &=& 0\end{array}\right. \tag{9}$$

Octa$_8$

$$\left\{\begin{array}{lcl}(\alpha_1y_2^2+\alpha_2)y_1^2+\alpha_3y_1y_2+\alpha_4y_2^2+\alpha_5 &=& 0\\ \beta_1y_1^2+\beta_2y_1y_3+\beta_3 &=& 0\\ \gamma_1y_2^2+\gamma_2y_2y_3+\gamma_3 &=& 0\end{array}\right. \tag{10}$$

B Tetrahedral Problems

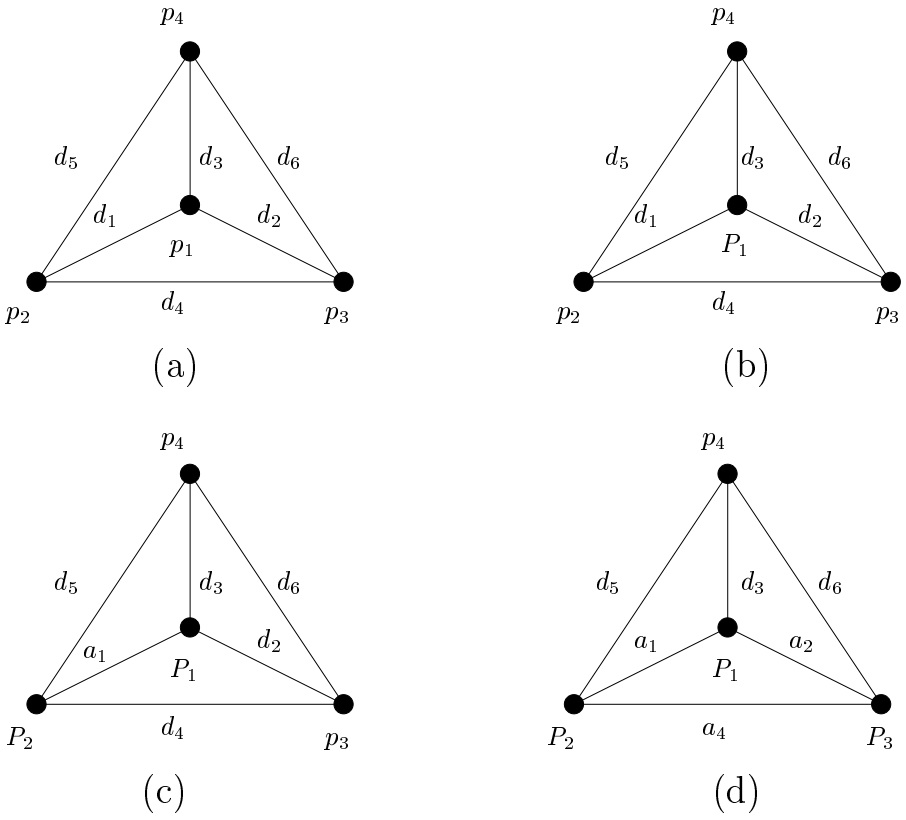

Figure 13: (a) Tetra_1, (b)Tetra_2, (c) Tetra_3 and (d) Tetra_4

C Octahedral Problems

- Octa_1: 6 points.
- Octa_2: 5 points and 1 plane.
- Octa_3: 4 points and 2 planes that are adjacent in the constraint graph.
- Octa_4: 4 points and 2 planes that are not adjacent in the constraint graph.
- Octa_5: 3 points and 3 planes that form a triangle in the constraint graph.
- Octa_6: 3 points and 3 planes that form a path in the constraint graph.

- Octa_7: 4 planes and 2 points that are adjacent in the constraint graph.
- Octa_8: 4 planes and 2 points that are not adjacent in the constraint graph.

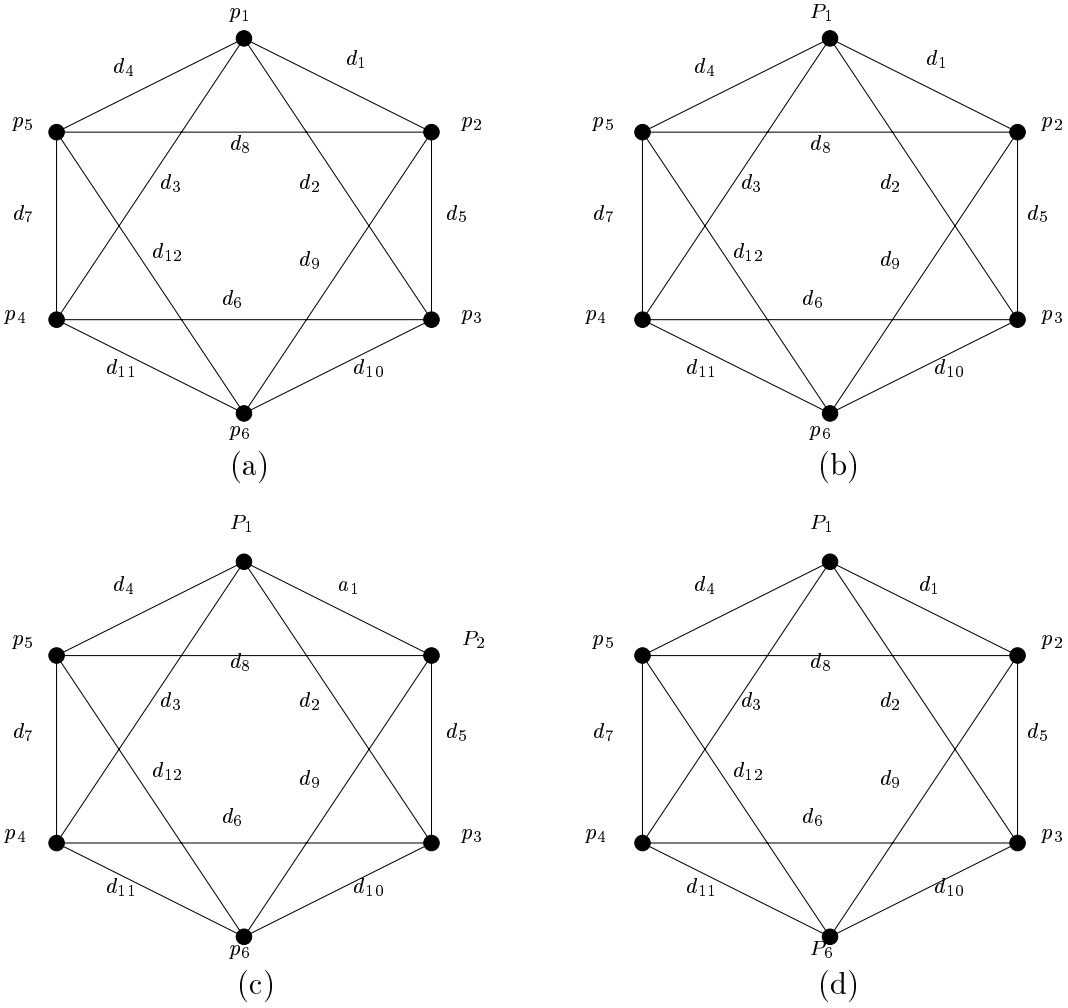

Figure 14: (a) Octa_1, (b) Octa_2, (c) Octa_3, (d) Octa_4

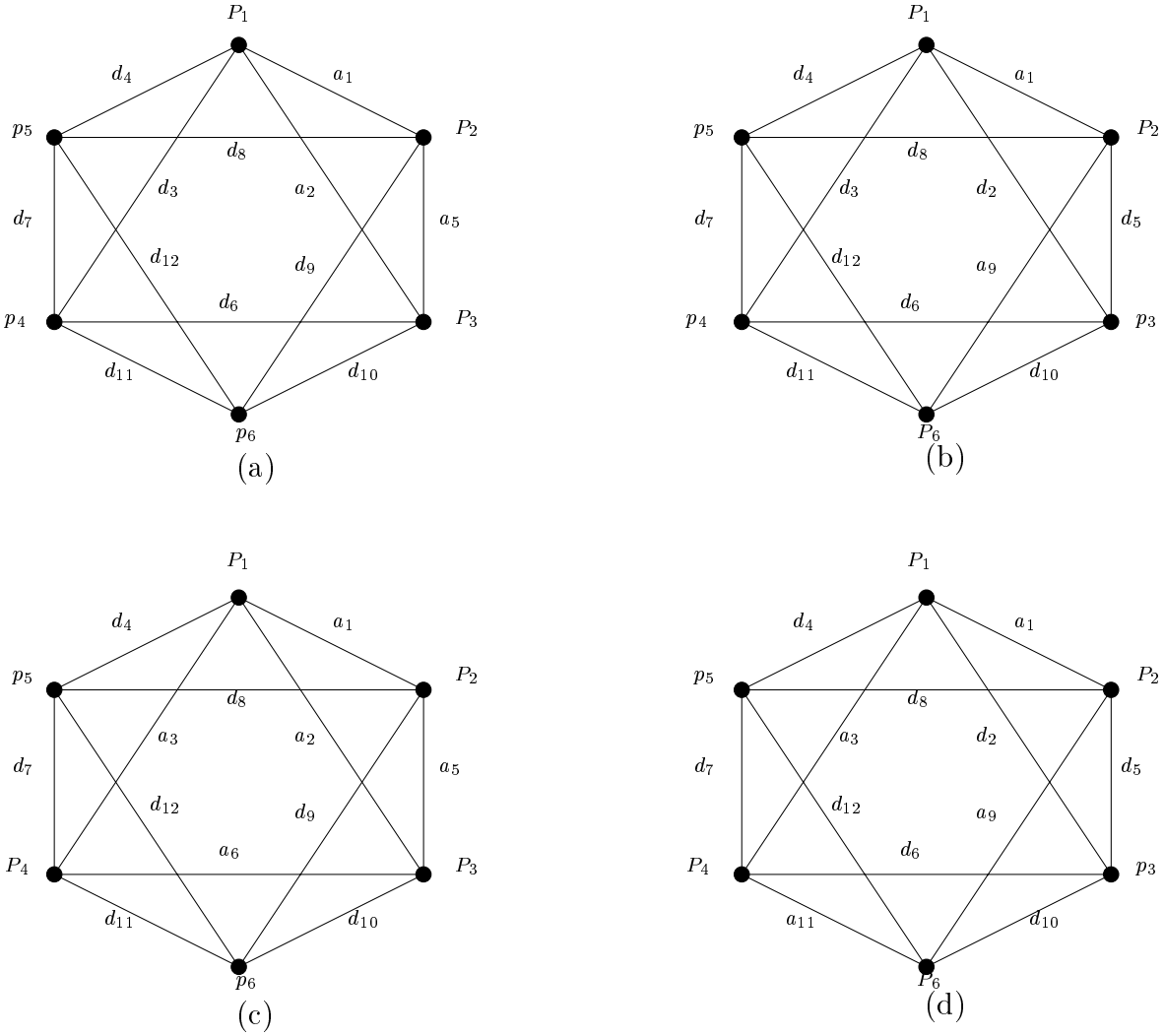

Figure 15: (a) Octa_5, (b) Octa_6, (c) Octa_7, (d) Octa_8