Solving minimal, wellconstrained, 3D geometric constraint systems: combinatorial optimization of algebraic complexity

Meera Sitharam [*†‡] Jörg Peters[*] [§]Yong Zhou[*,†]

June 20, 2004

Abstract

Many geometric constraint solvers use a combinatorial or graph algorithm to generate a decomposition-recombination (DR) plan. A DR plan recursively decomposes the system of polynomial equations into small, generically rigid subsystems that are more likely to be successfully solved by algebraic-numeric solvers.

In this paper we show that, especially for 3D geometric constraint systems, a further optimization - of the algebraic complexity of these subsystems - is both possible, and often necessary to successfully solve the DR-plan. To attack this apparently undocumented challenge, we use principles of rigid body manipulation and quaternion forms and combinatorially optimize a function over the minimum spanning trees of a graph generated from DR-plan information. This approach follows an interesting connection between the algebraic complexity of the system and the topology of the corresponding constraint graph. The optimization has two secondary advantages: in navigating the solution space of the constraint system and in mapping solution paths in the configuration spaces of the subsystems.

We formally compare the reduction in algebraic complexity of the subsystem after optimization with that of the unoptimized subsystem and illustrate the practical benefit with a natural example that could only be solved after optimization.

Keywords: Variational geometric constraint solving, Cyclical and 3D geometric constraint systems, Decomposition of geometric constraint systems, User navigation of solution conformations, Feature-based and assembly modeling, Conceptual design, Parametric constraint solving, Underconstrained and Overconstrained systems, Degree of Freedom analysis, Constraint graphs.

1 Introduction and Motivation

Geometric constraint systems are used as succinct, conceptual, editable representations of geometric composites in many applications including mechanical computer aideed design, robotics, molecular modeling and teaching geometry. For recent reviews of the extensive literature on geometric constraint solving see, e.g, [11, 15, 7].

A *geometric constraint system* consists of a finite set of geometric objects and a finite set of constraints between them. See Figure 1. The constraints can usually be written as algebraic equations and inequalities whose variables are the coordinates of the participating geometric objects. For example, a distance constraint of d between two points (x_1, y_1) and (x_2, y_2) in 2D is written as $(x_2 - x_1)^2 + (y_2 - y_1)^2 = d^2$

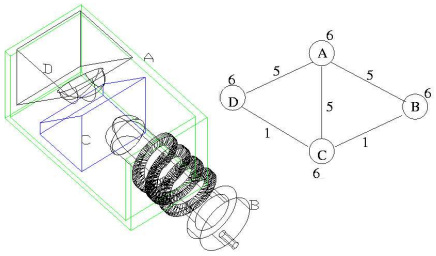
A *solution or realization* of a geometric constraint system is the (set of) real zero(es) of the corresponding algebraic system. In other words, the solution is a class of valid instantiations of (the position, orientation and any other parameters of) the geometric elements such that all constraints are satisfied. Here, it is understood that such a solution is in a particular geometry, for example the Euclidean plane, the sphere, or Euclidean 3 dimensional space. A constraint system can be classified as *overconstrained*, *well-constrained*, or *underconstrained*. Well-constrained systems have a finite, albeit potentially very large number of *rigid* solutions; their solution space is a zero-dimensional variety. Underconstrained systems have infinitely many solutions; their solution space is not zero-dimensional. Overconstrained systems do not have a solution unless they are *consistently overconstrained*. Well or overconstrained systems are called *rigid* systems.

[*]University of Florida

[†]Work supported in part by NSF Grant CCR 99-02025, NSF Grant EIA 00-96104

[‡]*corresponding author: sitharam@cise.ufl.edu*

[§]Work supported in part by NSF Grant DMI-0400214



,

Figure 1: An underconstrained 3D example with 1 extra dof and its dof constraint graph: each of 4 rigid objects has 6 dofs, B, C, D are constrained by incidence with the housing A, which removes 5 dofs in each case, so that each of the pairs AB, AC, AD is underconstrained and has one extra degree of freedom. The remaining 2 pairs BC and CD are connected by distance constraints, which remove 1 dof each

Technically, geometric constraint solving straddles combinatorics, algebra and geometry. Specifically, it overlaps combinatorial rigidity theory, geometric methods for kinematics, robotics and mechanisms, automated geometry theorem proving, geometric invariant theory, and computational algebraic geometry (solving specific classes of polynomial systems arising from geometric constraints).

Many geometric constraint solvers generate a *decomposition-recombination* (DR) plan (formally defined in Section 2) that recursively decomposes the geometric constraint system, - a system of polynomial equations - into small *generically rigid* subsystems. The subsystems are then to be solved by algebraic or numeric solvers. The DR-plan is typically a directed acyclic graph or partial ordering of subsystems. Solving a subsystem C in the DR-plan corresponds to determining the positions and orientations of the rigid bodies already solved as 'child' subsystems of C in the DR-plan. That is the subsystems satisfy their internal active geometric constraints. Since C itself represents a rigid body, these positions and orientations are computed in C's coordinate system, which is appropriately fixed during the solution process.

The size of the largest subsystem in a DR-plan overwhelmingly dominates the complexity of constraint solving, since the complexity of finding real zeroes of even sparse polynomial systems is exponential in the number of variables. DR-planners try to optimize the size of the subsystems solved. They are usually graph-based algorithms that use a combinatorial degree of freedom analysis to isolate *clusters* (formally defined in Section 2 or subgraphs corresponding to generically rigid subsystems and they typically use the number of child clusters that are needed for solving the parent cluster as a measure of the size of the parent cluster.

In this paper we show that, especially for 3D systems, this is a coarse measure of size: crucial further optimization of the algebraic complexity of the subsystems - i.e., number of variables and degree of equations - is necessary to successfully solve many subsystems that commonly arise. Due to the difficulty of the issues involved, literature on 3D geometric constraint solving is sparse, and we are unaware of previous investigations of this natural problem.

We use principles of rigid body manipulation and quaternion forms to define the objective function(s) being optimized and provide an efficient combinatorial algorithm to perform the optimization. The algorithm simultaneously optimizes a combination of choices:
(a) the set of child clusters used for solving the parent cluster;
(b) the parent cluster's coordinate system;
(c) the rotation and translation parameters of the child clusters to be resolved and
(d) the partial order of their resolution.
The algorithm effectively optimizes a function over the minimum spanning trees of a graph generated for the parent cluster being solved using the structure of overlaps between child clusters. This gives an interesting connection between the algebraic complexity of the polynomial system corresponding to a cluster and the topology of its set of child clusters. We formally compare the reduction in algebraic complexity of the subsystem after optimization, with that of the unoptimized subsystem. And we illustrate with a natural example that could only be solved after optimization.

This optimization algorithm has two secondary advantages which we illustrate by example. First, the optimization yields a sequence of key points on a solution path in the combined configuration space of the

child clusters. This path has a physical interpretation as a sequence of translations and rotations: it starts from an arbitrary initial position and orientation of the child clusters and ends with their solved position and orientation, i.e., a point in the configuration space of the parent cluster. Second, the optimized subsystem yields a more intuitive classification of the realizations of the parent cluster C - i.e., the finite set of real roots of the polynomial system of geometric constraints that are active at C – given specific realizations of the child clusters. This is useful for the well-documented problem of controlling the combinatorial explosion and navigating the solution space of rigid geometric constraint systems.

In Section 2, we formally define constraint graphs, degree of freedom analysis to determine generic rigidity, DR-plans and the properties of DR-plans that we shall assume in this manuscript. In Section 3, we use a natural example to illustrate how a 3D constraint solver would formulate the subsystem of polynomial equations that correspond to a cluster of a DR-plan. Both the algebraic and the numeric solver in MAPLE are unable to solve this subsystem. We then use quaternion forms from rigid body mechanics to formulate an equivalent subsystem of dramatically lower algebraic complexity in a formal sense. The quaternion forms provide a sequence of points on a physically meaningful solution path for the cluster. The example sets the stage for Section 4 which gives a formal definition of the objective function being optimized, a description of the (purely combinatorial) optimization algorithm, and a formal comparison of the algebraic complexity of optimized and unoptimized systems in the general case. In Section 5, we show the solutions and realizations of the example in Section 3. This suggests that the method can be used to assist in describing and navigating the solution space of wellconstrained geometric constraint systems.

2 Constraint Graphs, Degree of Freedom Analysis and DR-plans

A geometric constraint graph $G = (V, E, w)$ corresponding to geometric constraint system is a weighted graph with n vertices (representing geometric objects) V and m edges E (representing constraints). The weight $w(v)$ of a vertex v counts the degrees of freedom (*dof*s) of the object represented by v and the weight $w(e)$ of an edge e counts the degrees of freedom removed by the constraint represented by e. For example, Figure 1, shows a 3D constraint systems and their respective dof constraint graphs. Note that the constraint graph could be a *hypergraph* with each hyperedge involving any number of vertices. A subgraph $A \subseteq G$ that satisfies

$$\sum_{e \in A} w(e) + D \geq \sum_{v \in A} w(v)$$

is called *dense*, where D is a dimension-dependent constant, to be described below. Function $d(A) = \sum_{e \in A} w(e) - \sum_{v \in A} w(v)$ is called *density* of a graph A.

The constant D is typically $\binom{d+1}{2}$ where d is the dimension. The constant D captures the degrees of freedom of a rigid body in d dimensions. For 2D contexts and Euclidean geometry, we expect $D = 3$ and for spatial contexts $D = 6$, in general. If we expect the rigid body to be fixed with respect to a global coordinate system, then $D = 0$.

Next, we give some purely combinatorial properties of constraint graphs based on density. These will be later shown to be related to properties of the corresponding constraint systems.

A dense graph with density strictly greater than $-D$ is called *overconstrained*. A graph that is dense and all of whose subgraphs (including itself) have density at most $-D$ is called *wellconstrained.* A graph G is called *well-overconstrained* if it satisfies the following: G is dense, G has at least one overconstrained subgraph, and has the property that on replacing all overconstrained subgraphs by wellconstrained subgraphs (in any manner), G remains dense. A graph that is wellconstrained or well-overconstrained is called a *cluster*. A dense graph is *minimal* if it has no dense proper subgraph. Note that all minimal dense subgraphs are clusters but the converse is not the case. A graph that is not a cluster is said to be *underconstrained*. If a dense graph is not minimal, it could in fact be an underconstrained graph: the density of the graph could be the result of embedding a subgraph of density greater than $-D$.

To discuss how the graph theoretic properties based on *degree of freedom (dof) analysis* described above relate to corresponding properties of the corresponding constraint *system*, we need to introduce the notion of *genericity*. Informally, a constraint system is generically *rigid* if it is physically rigid (does not flex or has only finitely many non-congruent, isolated solutions) for most of choices of coefficients of the system. More formally we use the

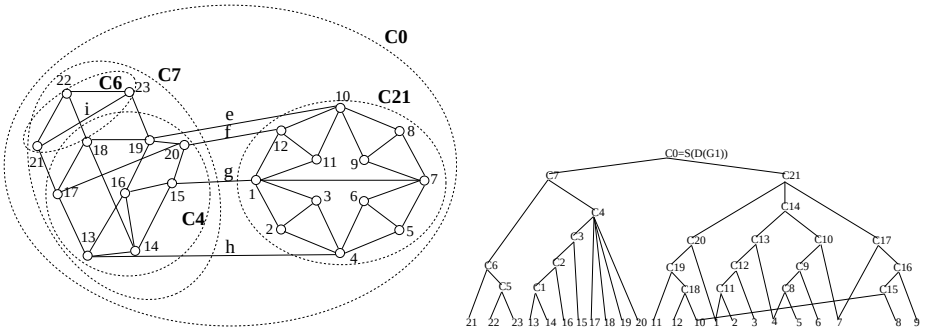


Figure 2: 2D constraint graph G1 and the corresponding DR-plan; all vertices weights are 2, all edges weights 1.

notion of genericity of e.g, [2]. A property is said to hold *generically* for polynomials $f_1, \ldots, f_n$ if there is a nonzero polynomial P in the coefficients of the f_i such that this property holds for all $f_1, \ldots, f_n$ for which P does not vanish.

Thus the constraint system E is generically rigid if there is a nonzero polynomial P in the coefficients of the equations of E - or the parameters of the constraint system - such that E is solvable when P does not vanish. For example, if E consists of distance constraints, the parameters are the distances. Even if E has no overt parameters, i.e, if E is made up of constraints such as incidences or tangencies or perpendicularity or parallelism, E in fact has hidden parameters capturing the extent of incidence, tangency, etc., which we consider to be the parameters of E.

A generically rigid system always gives a cluster, but the converse is not always the case. In fact, there are well-constrained clusters whose corresponding systems are not generically rigid and are in fact generically not rigid due to the presence of generic "hidden" *constraint dependencies*. (These dependencies include the so-called "bananas" and "hinge" situations [9], [3], [4] and could also result in overconstraints that are not detectable by a dof count).

However, it should be noted that in 2 dimensions, according to Laman's theorem [16] if all geometric objects are points and all constraints are distance constraints between these points then any minimal dense cluster represents a generically rigid system. But there are no known combinatorial characterizations of 2D rigidity, when other constraints besides distances are involved.

In fact, there is no known, tractable characterization of generic rigidity of systems for 3 or higher dimensions, based purely on combinatorial properties of the constraint graph [29], [9], although several conjectures exist.

2.1 DR-plans and their relevant properties

The DR-planner is a graph algorithm that outputs a *decomposition-recombination plan (DR-plan)* of the constraint graph. In the process of combinatorially constructing the DR-plan in a bottom up manner, at stage i, it locates a wellconstrained subgraph or cluster S_i in the current constraint graph G_i, and uses an abstract, unevaluated *simplification* of S_i to to create a transformed constraint graph G_{i+1}.

Decomposition algorithms based on constraint graphs have been proposed since the early 90's based on recognition of subgraph patterns such as triangles [8, 21, 22, 20] [17, 19]; and based on Maximum Matching [23, 1]. However, prior to [12], the DR-planning problem and appropriate performance measures for the planners were not formally defined and many DR-planners before [13] were not able to handle general constraint systems. That paper also gives a table comparing 3 main types of DR-planners, with respect to these performance measures some of which are relevant here.

Formally, a DR-plan of a constraint graph G is a directed acyclic graph (DAG) whose nodes represent clusters in G, and edges represent containment. The leaves or sinks of the DAG are all the vertices (primitive clusters) of G. The roots or sources are all the maximal clusters of G. For rigid graphs, the DR-plans have a single source. There could be many DR-plans for G. See Figure 2. An *optimal* DR-plan is one that minimizes the maximum fan-in. The *size* of a cluster in a DR-plan is its fan-in (it represents the size of the corresponding subsystem, once its children are solved).

We would additionally like the *width* i.e, *number* of clusters in the DR-plan to be small, preferably linear in the size of G: this reflects the complexity of the planning process and affects the complexity of the solving process that is based on the DR-plan. Next we give other properties of DR-plans that we assume in this paper.

2.1.1 Assumption of Rigidity

As mentioned earlier in the section, no tractable, combinatorial method of determining generic rigidity is known. The DR-planner that we have in mind implemented in the opensource FRONTIER geometric constraint solver [28] (2003 version) relies heavily on the definition of a cluster given earlier, albeit augmented to check for most types of troublesome constraint dependencies [27] before deciding generic rigidity. We will restrict ourselves to constraint graphs for which rigidity is verified using this augmented definition of a cluster. This augmented definition requires resolving some clusters first, i.e., it imposes dependences between clusters that do not contain one another as well as a a *solving priority order* on the DR-plan (see [27], [24], [25]). However, since the augmented definition of a cluster is involved (see [27]), we will use the terms *generically rigid system* and the above definition of *cluster* interchangeably. In Section 4, we sketch how to generalize our proposed algorithm when the augmented definition is used.

2.1.2 Complete decomposition into maximal clusters

We assume that for each cluster C in the DR-plan, the set of its children satisfies one of two criteria. In one case, they form a complete set of clusters C_i each of which is maximal in that the only proper cluster of C that contains C_i is one that intersects every other cluster in the set on a non-trivial subgraph; in the second case, they consist of a pair of clusters whose intersection is a non-trivial cluster and whose union induces all of C.

Trivial subgraphs are subgraphs that correspond (after resolving incidence constraints) to a single point in 2D or 3D and to a fixed or variable length line segment in 3D. (Whenever 2 rigid clusters overlap on a non-trivial cluster, the cluster induced by their union is automatically rigid.) The FRONTIER constraint solver's DR-planner [18] [28] (2003 version) has this property. In fact, it relies on this property to determine rigidity of clusters. In the former case, the cluster C is associated with several *covering sets* of maximal clusters within C. (The union of a covering set of clusters induces all of C).

2.1.3 Removal of Overconstraints and Stability Assumption

Each covering set of maximal clusters results in a corresponding system of active constraints for C. In general, this system could be overconstrained by the presence of explicit overconstraints or by overconstraints that arise as a consequence of generic "hidden" constraint dependences (that affect generic rigidity as mentioned earlier) and are not detected by a simple degree of freedom count. We assume that the DR-plan contains complete information about both these types of constraints. The FRONTIER constraint solver's DR-planner [18] [28] (2003 version) satisfies this property. The former type of overconstraints are detected and corrected by a method given in [10] that gives a complete description of the set of *reducible* overconstraints, i.e., those can be removed without making the cluster underconstrained. The latter types of constraint dependences are detected by the FRONTIER's DR-planner [28] (2003 version) in the process of determining rigidity of clusters (as pointed out earlier, we restrict ourselves to the class of graphs for which rigidity is thus correctly determined).

However, even after these types of overconstraints have been removed and C is a wellconstrained cluster, it is still a nontrivial task to pick a guaranteed stable, independent system of polynomial equations corresponding to C. The difficulty of this task is compounded by shared objects between clusters, rotationally symmetric trivial clusters etc. See [26] for a description of and solution to this problem, which is not our focus here. We will assume that C is wellconstrained and that any system of polynomial equations constructed from the active constraints in C is independent and stable.

3 An instructive example

In this section, we show three increasingly sophisticated and increasingly efficient approaches to *representing the active constraints* needed to resolve a cluster of the DR-plan satisfying the properties of 2. Since we want to be able to compute any and all solutions, the number of variables and the algebraic degree of the constraints plays a crucial role.

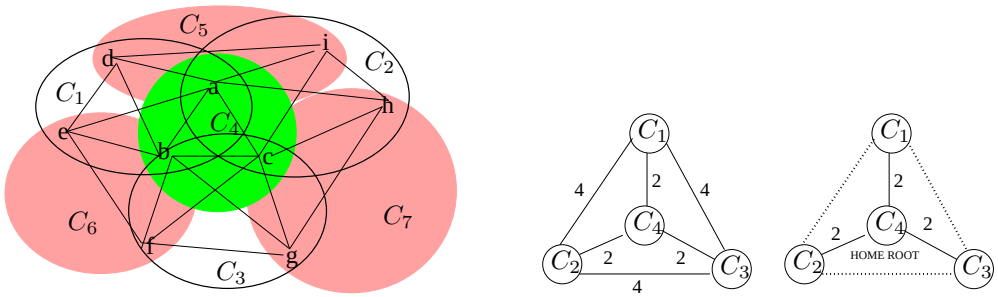


Figure 3: (*left*) The child clusters of `hex_tet` ; (*right*) the weighted overlap graph.

Figure 3 shows the constraint structure and child clusters C_i of a problem we call `hex_tet` , for reasons that will become obvious when we look at specific geometric instances later on. All vertices represent points with 3 degrees of freedom (dof) and all edges are distance constraints removing 1 dof each. The child clusters C_1, C_2 and C_3 are tetrahedra and all other child clusters are triangles. The actual distance values are not yet relevant for the discussion in this section (but will be in Section 5 when we actually solve instances). We may think of them as equal (except for the 3 distances ef, gh and di which need to be larger in order for a solution to exist).

The cluster C satisfies the DR-plan properties of Section 2. Specifically, the child clusters form a complete set of maximal clusters properly contained in C, and their pairwise intersections are trivial. Minimal covering sets are: $\{C_1, C_2, C_3\}$, $\{C_5, C_6, C_7\}$.

In the following, we denote by

$$\mathbf{x}_{a,C_i} := \begin{bmatrix} x_{a,C,C_i} \\ y_{a,C,C_i} \\ z_{a,C,C_i} \end{bmatrix} \in R^3 \text{ the coordinates of Point } a \text{ in } C_i\text{'s local coordinate system and by}$$

$$\mathbf{x}_{a,C,C_i} := \begin{bmatrix} x_{a,C,C_i} \\ y_{a,C,C_i} \\ z_{a,C,C_i} \end{bmatrix} \text{ the coordinates of Point } a \textit{ parametrized} \text{ in the dof's in } C\text{'s coordinate system.}$$

In particular, $\mathbf{x}_{a,C,C_i}$ may depend on rotation angles that will only be resolved to a final position $\mathbf{x}_{a,C}$ in cluster C by solving the constraint system.

3.1 The Unoptimized polynomial system

Current constraint solvers, such as FRONTIER [28], construct the polynomial system corresponding to `hex_tet` as follows:

1 Pick an arbitrary minimal covering set of child clusters. For the example, we choose $\{C_1, C_2, C_3\}$.

2 Set the coordinate system of C to one of the child clusters. This child cluster is called the *home cluster* for C. For the example, we choose C_1.

3 Each of the other two clusters' position and orientation within C's coordinate system can be expressed using a composite 3D rotation matrix $\mathbf{M}$ and a translation vector $\mathbf{t}$. For example, point i in C_2 can be written as

$$\mathbf{x}_{i,C,C_2} = \mathbf{M}_{C_2}\mathbf{x}_{i,C_2} + \mathbf{t}_{C_2}$$

where $\mathbf{t}_{C_i} \in \mathbb{R}^3$ is a translation that positions C_i in C's coordinate system and

$$\mathbf{M}_{C_i} = \begin{bmatrix} s_2 s_3 & s_2 c_3 & -c_2 \\ s_1 c_2 s_3 - c_1 c_3 & c_1 s_3 + s_1 c_2 c_3 & s_1 s_2 \\ s_1 c_3 + c_1 c_2 s_3 & -s_1 s_3 + c_1 c_2 c_3 & c_1 s_2 \end{bmatrix}$$

is a composition of 3D rotations that orient C_i in C's coordinate system. The entries of $\mathbf{M}_{C_i}$ are polynomials of total degree at most 3 in the six variables $s_{1,C_i}, s_{2,C_i}, s_{3,C_i}, c_{1,C_i}, c_{2,C_i}, c_{3,C_i}$. These variables

represent the sines and cosines of the rotation angles and are therefore related by three *trig-relation* equations of the form $s_i^2 + c_i^2 = 1$. (Choosing the rotation angles as variables is not practical; the constraint system would no longer be polynomial in the variables.)

Since C_2 shares the point a with C_1 and the coordinates of C_1 are fixed, $\mathbf{t}_{C_2} = \mathbf{x}_{a,C_1} - \mathbf{x}_{a,C_2}$. Similarly, $\mathbf{t}_{C_3} = \mathbf{x}_{b,C_1} - \mathbf{x}_{b,C_2}$.

4 This leaves six active constraints for C: three distance constraints of type

$$\delta^2_{i,C_j,C_k} = \|\mathbf{x}_{i,C,C_j} - \mathbf{x}_{i,C,C_k}\|^2$$

where δ is the prescribed distance and three overlap constraints

$$\mathbf{x}_{\ell,C,C_j} = \mathbf{x}_{\ell,C,C_k}, \qquad j,k \in \{1,2,3\}.$$

Thus, to resolve C, we need to solve a system of twelve polynomial equations (including the six trig-relation equations) in twelve variables s_{i,C_j} and c_{i,C_j}, $1 \le i \le 3$ $2 \le j \le 3$. The six trig-relations have degree 2. The three distance constraints each have a total degree 6 (since, for example, the rotation matrix results in cubic expressions for the value of x_{i,C,C_2}) and the overlap constraints each have total degree 3.

3.2 An Alternative using Quaternions

A more careful approach to step 3 of the procedure can reduce the degree of the equations. Each of the two clusters C_2 and C_3 share a point with the home cluster. Then their position and orientation within C's coordinate system can be represented as a rotation about an arbitrary axis pivoting through the shared point.

It is well-known that rotation about an axis $\mathbf{v} \in R^3$ by an angle α can be expressed as

$$\mathbf{q}^{-1} \cdot (0,\mathbf{x}) \cdot \mathbf{q}, \text{ where } \mathbf{q} := (\cos(\alpha/2), \sin(\alpha/2)\frac{\mathbf{v}}{\|\mathbf{v}\|}).$$

Here $\cdot$ is the quaternion multiplication. In fact, any rotation matrix can be expressed in terms of a (unit) quaternion as

$$\mathbf{Q} := \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 1-2q_1^2-2q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 1-2q_1^2-2q_2^2 \end{bmatrix}, \qquad \sum_{i=0}^{3} q_i^2 = 1.$$

For example, the rotation about x-axis by $-\alpha$ is represented in terms of $q = (c,s,0,0)$ where $c := \cos(\alpha/2)$, $s := \sin(\alpha/2)$ and hence

$$\mathbf{Q}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1-2s^2 & 2sc \\ 0 & -2sc & 1-2s^2 \end{bmatrix}, \quad \sin(\alpha) = 2sc, \cos(\alpha) = 1-2s^2.$$

Back to our example, in step **3**, we translate C_2 so that $\mathbf{x}_{a,C,C_2} = \mathbf{0}$, i.e. the point a that it shares with the home cluster becomes the origin. Then we apply to each point i the quaternion matrix $\mathbf{Q}_{C2}$ (with unevaluated q_i). Finally, we translate to the target position $\mathbf{x}_{a,C,C_1}$ of a in the home cluster:

$$\mathbf{x}_{i,C,C_2} = \mathbf{x}_{a,C,C_1} + \mathbf{Q}_{C_2}(\mathbf{x}_{i,C_2} - \mathbf{x}_{a,C_2}).$$

We are left with the three overlap and three distance constraints as in the previous formulation. However, we now need to resolve only eight equations in eight variables! Moreover, the three overlap equations are only quadratic, as are the two trig-relation equations for C_2 and C_3. The distance equations di between C_1 and C_2, and ef between C_1 and C_3 are quadratic and only the distance constraint ih between C_2 and C_3 is of total degree 4.

3.3 An Optimized Alternative

Both the number of unresolved constraints and the degree of the equations can be further reduced. Consider the following contruction for a constraint system expressing `hex_tet` .

1 Choose $\{C_1, C_2, C_3, C_4\}$ as the (not minimal!) covering set.

2 Choose C_4 as the home cluster. Without loss of generality, using basic trigonometry, the coordinate system for C is

$$\mathbf{x}_{a,C,C_4} = \mathbf{0}, \quad \mathbf{x}_{b,C,C_4} = \begin{bmatrix} \text{dist}(a,b) \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_{c,C,C_4} = \begin{bmatrix} \text{dist}(a,c)c_{23} \\ \text{dist}(a,c)s_{23} \\ 0 \end{bmatrix}$$

where c_{23} and s_{23} are the sine and cosine of the angle between the edge a, b and the edge a, c.

3 Since C_4 and each C_k, $k = 1, 2, 3$ share two points, $p_k, \tilde{p}_k \in \{a, b, c\}$, the position and orientation of the cluster C_k are fixed except for rotation about the axis through the two points. The coordinates of an arbitrary point i in cluster C_k can then be represented as

$$\mathbf{x}_{i,C,C_k} = \mathbf{x}_{p_k,C} + \mathbf{R}_{C_k}(\mathbf{x}_{i,C_k} - \mathbf{x}_{p_k,C_k}).$$

where $\mathbf{R}_{C_k}$ is a linear expressions in 2 variables s_{C_k} and c_{C_k}. The variables represent the sine and cosine of rotation about $p_k\tilde{p}_k$ and satisfy $c_{C_k}^2 + s_{C_k}^2 = 1$.

4 This leaves only three distance constraints. To resolve the cluster C, we now only need to solve a system of six constraints in the six variables s_{C_j} and c_{C_j}, $1 \le j \le 3$. All constraints, the three distance constraints and the three trig-relation constraints, are quadratic!

3.4 Interpretation

The reduction in algebraic complexity from the original to the optimized solution is significant. None of the algebraic and numerical solvers, we surveyed, succeeded in solving the original problem formulation but all solutions of the optimized system were obtained in less than one minute. Before we look at individual instances in Section 5, we characterize the principles underlying the optimized solution and give a general algorithm for optimization.

4 Optimizing Algebraic Complexity using Cluster Topology

In this Section, we formulate the problem of optimizing the algebraic complexity of the polynomial system for resolving a parent cluster C. We restrict ourselves to the 3D case – the 2D case is similar and simpler. Our definition of the optimization problem is accompanied by a sketch of an optimization algorithm.

We assume that parent cluster C, for which the optimized polynomial system needs to be constructed, satisfies the DR-plan properties of Section 2. Specifically, we assume that the child clusters form a complete set of maximal clusters properly contained in a well-constrained cluster C; that all overconstraints and hidden constraint dependencies have been removed as required. We may also assume that the pairwise intersections of these clusters are trivial subgraphs since, if C is formed from a pair of child clusters whose intersection is non-trivial, the home cluster fixes the other clusters by fixing three of its points.

Definition 4.1 *The overlap graph of a subset S of child clusters of C is an undirected graph whose vertices are the clusters in S. The edges represent pairs of clusters. If the overlap between the pair C_i, C_j is a trivial subgraph that reduces, after resolving incidence constraints, to k points then the weight of an edge is $w_{ij}(k) := 6 - 2k$.*

The weight of an edge (i, j) agrees with the number of rotation variables needed to represent the cluster C_i, if C_j were to be fixed as the home cluster. The number of variables is calculated using (see Section 3):
R, the two variable rotation of C_i about the axis formed by the two shared points in the overlap;
Q, the four variable quaternion matrix representing rotation of C_i about the shared point in the overlap;
M, the six variable combination of rotation and translation of the unoptimized example.

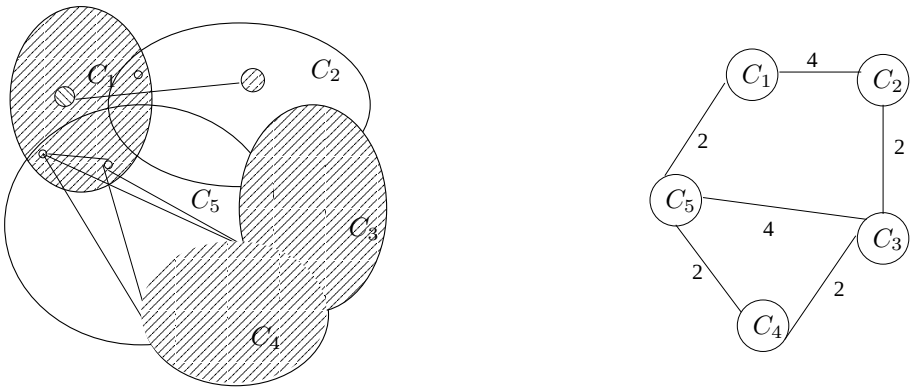


Figure 4: (*left*) Constraint graph of problem `pent_plat` and (*right*) the corresponding weighted overlap graph with edges of weight 6 omitted.

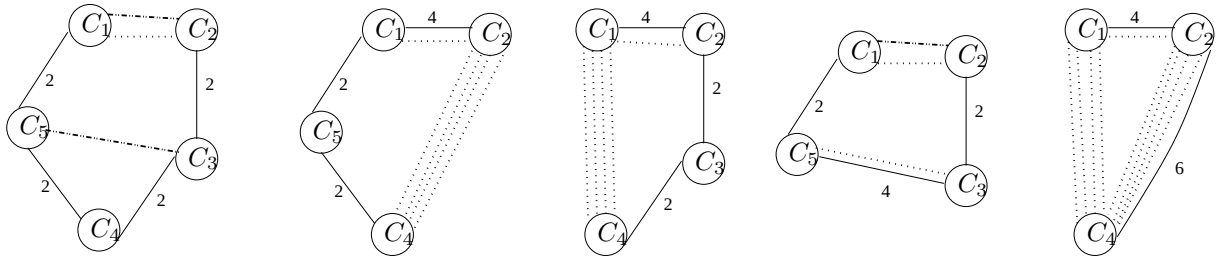


Figure 5: Five spanning trees of the covering sets of the example in Figure 4. Distance constraints between clusters are *dotted*. The overlaps that are not tree edges are *dashed*.

Based on the examples of the previous section, we observe that
(i) any choice of a set S of child clusters that forms a covering set for the parent cluster C, and (ii) any spanning tree T of the overlap graph of the clusters in S induces a system of equations for resolving C.

The choice H of a tree root represents the home cluster to be fixed. Since C is assumed to be wellconstrained, the number of variables and equations equals the total edge weight of the the spanning tree. Most sparse polynomial system solvers such as [5], [14] (geometric constraint systems are sparse) take time exponential in the number of variables. Hence the overwhelming factor in the algebraic complexity of the system is the number of variables. Degree and the number of terms are less important since the time complexity is polynomial in these parameters.

Algorithmically, optimization requires *finding a spanning tree of minimum total weight in the subgraph of the overlap graph induced by S, over all covering sets S*. Alternately, if S are chosen to be *minimal* covering sets, then the optimization requires *finding a tree of minimum total weight containing S, over all covering sets S*.

For general graphs and either arbitrary sets of vertices S of a fixed size or even for a fixed input set S, this problem is called the *Steiner tree* problem and it is NP-complete. Our overlap graphs, however, are special; the sets S are covering sets, and most importantly, since the cluster C is decomposed into its maximal clusters, the number of child clusters in most applications is no more than ten. So, for our purposes, this problem is tractable if we use the original formulation in the previous paragraph, even if we exhaustively search through all covering sets S.

Figure 5 shows covering sets and trees for the example `pent_plat` in Figure 4 where child clusters $C_1, \ldots, C_5$ form the parent cluster C. We are interested in those that have the minimum possible weight (in this case 8). Note the rightmost covering set onlyh has a spanning tree of weight 10.

We can further optimize the degree among the set V_C of polynomial systems for C with minimum number of

variables (and equations). The set V_C can be identified with the set of spanning trees for optimal covering sets S (i.e., those that have spanning trees of minimum weight over all S). Now, following the examples of Section 3, given such an optimal covering set S and a minimum weight spanning tree T for S, the equations needed to resolve C are determined by
(a) the cluster overlaps between pairs of clusters in S, corresponding to non-tree edges clusters in S,
(b) distance constraints between clusters in S and
(c) quadratic trig-relation constraints relating the rotation parameters of each cluster in S except the home cluster.

The degrees of these equations are determined by:

(iii) the choice of the home cluster H or the root of T.

To determine the polynomial degree of equations, let C_j be the direct predecessor of a cluster C_i on the path from the root to C_i. Then an overlap constraint involving a point in cluster C_i that is not in C_j has degree $\deg(C_i)$ at most

$$\deg(C_i) = \begin{cases} \max\{2, \deg(C_j)\} & \text{if } w_{ij} = 4, \\ \deg(C_j) + 1 & \text{if } w_{ij} = 2. \end{cases}$$

A distance constraint has, of course, twice the degree. Thus overall, a good intuitive heuristic is to use, as a root, a home cluster H that minimizes the depth of the rooted tree.

For the example trees in Figure 5 choosing any of the left 4 covering sets, we get 8 equations, but choosing the rightmost gives 10 equations. By choosing different home clusters say C_4 versus C_5 for the left-most tree we get different maximum degrees for the distance constraints. The best choice overall is the left most tree with home cluster C_4, which yields 3 overlap constraints (between C_1 and C_2, see discussion on removal of assumptions as to why the other non-tree overlap can be discarded) of degree 2 and 1 distance constraint of degree 4. Notice also that in the third tree from left, with home cluster C_1 the endpoints of two of the the distance constraints can be viewed as between to C_1 and C_3 in which case we obtain a degree reduction by 2 over treating all 4 distance constraints as between C_1 and C_4.

The example shows that computing the degrees of the equations, that result for a given tree and home cluster, requires care. But the optimization problem and algorithm are straightforward:

over the spanning trees in V_C and the choices of their roots or home clusters H,
we *minimize the maximum degree of the system of equations* and,
in the resulting set of rooted trees V_{DC},
we *find those with the minimum number of equations of maximum degree.*

Keeping track of the spanning trees in V_C and V_{DC} can be achieved in polynomial time using the data structures of [6]. However, in practice, the big O complexity for creation and maintenance of these datastructures hides large constants. Thus for the small sizes of clusters and overlap graphs that are typical in most applications, an exhaustive algorithm, albeit of exponential time complexity works better.

4.1 Reduction of Algebraic Complexity through Optimization

The formal decrease in the *number of variables and equations* obtained through the optimization is easy to quantify. Without optimization, to solve a wellconstrained cluster C using k child clusters that form a minimal convering set, we would need $6(k-1)$ variables, independent of the number of nonempty overlaps between clusters. Using our optimization method, we would typically need two fewer variables for each point overlap and four fewer for each pair overlap. Let k_1 be the number of pair overlap edges, and k_2 the number of point overlap edges in a minimum weight spanning tree T of the overlap graph of the child clusters. Then we need at most $2k_1 + 4k_2 + 6(k - 1 - k_1 - k_2)$ variables. If $k_1 + k_2 = k - 1$, we reduce the number of variables by a factor linear in k, the number of child clusters.

The *total degree* of all the distance equations in the unoptimized system is 6. Let us assume the root in the minimum spanning tree T that maintains a distance of atmost 2 (resp. 1) to each cluster involved in a distance constraint. Then, in the optimized system, the largest degree of a distance equation is at most 4 (resp. 2).

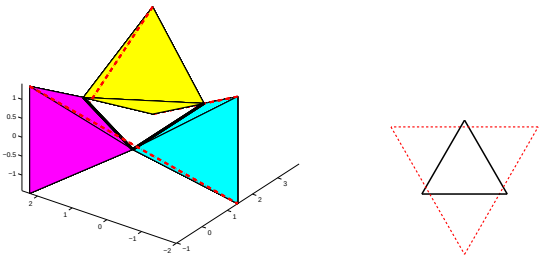


Figure 6: Unique solution to problem `hex_tet` : the home cluster C_4 connecting the three tetrahedron clusters C_1, C_2, C_3 is the central triangle. The *dashed* lines represent distance constraints. (*left*) Spatial view with the tetrahedra displayed. (*right*) Diagrammatic visualization: central triangle and edges satisfying the distance constraints.

4.2 Removing Assumptions made in Section 2

We turn to two assumptions made in Section 2. The first is to ensure that the system of equations constructed by the above algorithm is stable [26]. In the leftmost tree of Figure 5, although the cluster C is wellconstrained and the dof count is six, there are two non-tree edges representing overlaps. Choosing the overlap constraints between $C1$ and $C2$ would result in an independent system for solving C, but choosing the overlap between $C3$ and $C5$ would result in three dependent constraints. This problem is addressed independently and solved in [26]. Here, we do not impose this constraint of stability into our optimization problem, since the feasible region for our optimization problem corresponds to choosing or describing an independent set – and this is nontrivial. We simply assume that any choice of the appropriate number of overlap constraints and other constraints is a valid choice.

The second assumption made in 2 is that there are no clusters (among the child clusters of C) whose clusterhood depends on other clusters. In fact, in order to obtain tractable, approximate characterizations of combinatorial rigidity [27], such dependent clusters are necessary. In our optimization problem, these dependent clusters have to stay with their partner cluster, affecting the set of valid choices for covering sets S. However, while outside the scope of this paper, there is a relatively straightforward extension of the above optimization algorithm that would remove both of these assumptions.

5 Solutions to an Optimized system of Cluster Constraints

We now solve geometric instances of the problem `hex_tet` in section 3. In the first instance, the three clusters are initialized with the identical local vertex coordinates of a regular terahedron:

$$\mathbf{v}_1^i := \left[\begin{smallmatrix}-1\\-1\\-1\end{smallmatrix}\right], \quad \mathbf{v}_2^i := \left[\begin{smallmatrix}-1\\1\\1\end{smallmatrix}\right], \quad \mathbf{v}_3^i := \left[\begin{smallmatrix}1\\-1\\1\end{smallmatrix}\right], \quad \mathbf{v}_4^i := \left[\begin{smallmatrix}1\\1\\-1\end{smallmatrix}\right], \qquad i = 1, 2, 3.$$

The incidence and distance constraints that rearrange the tetrahedra are

$$\mathbf{v}_2^i = \mathbf{v}_1^{i+1}, \qquad (\mathbf{v}_3^i - \mathbf{v}_4^{i+1})^2 = \delta_i^2,$$

where the superscript is understood modulo 3. The underlying geometry is visualized in Figure 6 for one, extremal choice of δ_i. We consider three cases. In each case, it was necessary to use the optimized formulation of the system described in Section 3.

i. The maximal allowable value for $\delta_1^2 = \delta_2^2 = \delta_3^2$ is $\bar{\delta}^2 := 22 + 4\sqrt{2}\sqrt{3}$. Then, as shown in Figure 6, there is exactly one solution. We juxtapose the spatial view with a more abstract, diagramatic view of the solution. In general, the spatial view does not yield a good visualization since the clusters intersect and block the view. Especially when presenting solutions on paper, showing just the home cluster and the active distance constraints often reveals the underlying symmetries.

ii. For $\delta_1^2 = \delta_2^2 = \delta_3^2 = \bar{\delta}$ (no square), we obtain four solutions with evident symmetries (Figure 7).

iii. Finally, by altering the edge lengths of one cluster, we obtain ten solutions, shown in Figure 8.

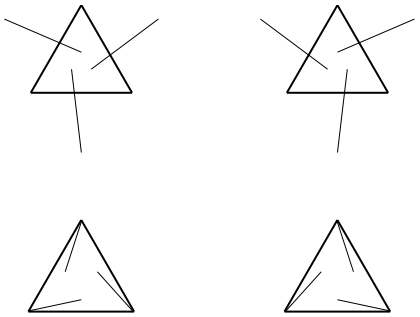


Figure 7: Diagramatic view of the four solutions to problem (ii).

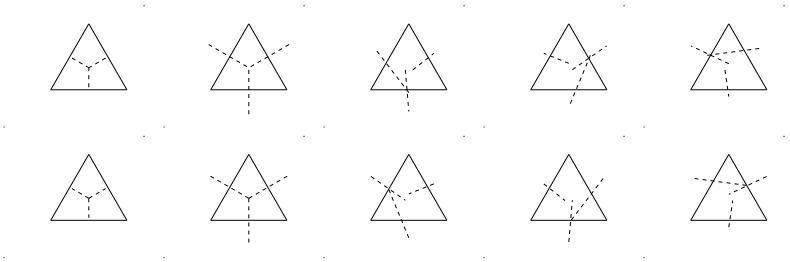


Figure 8: Diagramatic view of the ten solutions to problem (iii).

Beyond the obvious practical usefulness, the optimization method presented here suggests that the choice of home cluster (here the triangle cluster C_4) is a valuable tool to understand the structure of the solution space in terms of symmetries.

In addition, the overall optimization method suggests a visualization of a solution path for the solved cluster C as a sequence of operations on the child clusters. The unoptimized case explicitly resolves overlap constraints between clusters as part of a simultaneous system that includes active distance constraints etc. and does not provide any idea of a solution path. The optimized method uses the chosen rooted spanning tree of Section 4 to define the sequence of solution operations. The method first applies a sequence of unevaluated or abstract rotations to each non-home child cluster, considered in the tree order, and translates it to its overlap points with its ancestor cluster in the tree. This leaves only the abstract rotations to solve for (1 or 2 depending on the size of the overlap with the tree ancestor), using the nontree overlaps and distance constraints.

References

[1] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphics*, pages 83–92, 1993.

[2] D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*. Springer-Verlag, 1998.

[3] Henry Crapo. Structural rigidity. *Structural Topology*, 1:26–45, 1979.

[4] Henry Crapo. The tetrahedral-octahedral truss. *Structural Topology*, 7:52–61, 1982.

[5] Ioannis Emiris and John Canny. A practical method for the sparse resultant. In *International Conference on Symbolic and Algebraic Computation, Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, pages 183–192, 1993.

[6] David Eppstein. Representing all minimum spanning trees with applications to counting and generation. Technical Report 95-50, Univ. of California, Irvine, Dept. of Information & Computer Science, Irvine, CA, 92697-3425, USA, 1995.

[7] I. Fudos. *Geometric Constraint Solving*. PhD thesis, Purdue University, Dept of Computer Science, 1995.

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

[9] Jack E. Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial Rigidity*. Graduate Studies in Math., AMS, 1993.

[10] C Hoffman, M Sitharam, and B Yuan. Making constraint solvers more useable: the overconstraint problem. *to appear in CAD*, 2004.

[11] Christoph M. Hoffmann, Andrew Lomonosov, and Meera Sitharam. Geometric constraint decomposition. In Bruderlin and Roller Ed.s, editors, *Geometric Constraint Solving*. Springer-Verlag, 1998.

[12] Christoph M. Hoffmann, Andrew Lomonosov, and Meera Sitharam. Decomposition of geometric constraints systems, part i: performance measures. *Journal of Symbolic Computation*, 31(4), 2001.

[13] Christoph M. Hoffmann, Andrew Lomonosov, and Meera Sitharam. Decomposition of geometric constraints systems, part ii: new algorithms. *Journal of Symbolic Computation*, 31(4), 2001.

[14] B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial system. *Math. Comp.*, 64:1541–1555, 1995.

[15] G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.

[16] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

[17] R. Latham and A. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28:917–928, 1996.

[18] Andrew Lomonosov and Meera Sitharam. Graph algorithms for geometric constraint solving. In *submitted*, 2004.

[19] A. Middleditch and C. Reade. A kernel for geometric features. In *ACM/SIGGRAPH Symposium on Solid Modeling Foundations and CAD/CAM Applications*. ACM press, 1997.

[20] J. Owen. www.d-cubed.co.uk/. In *D-cubed commercial geometric constraint solving software*.

[21] J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modeling*, pages 397–407, Austin, Tex, 1991.

[22] J. Owen. Constraints on simple geometry in two and three dimensions. In *Third SIAM Conference on Geometric Design*. SIAM, November 1993. To appear in Int J of Computational Geometry and Applications.

[23] J.A. Pabon. Modeling method for sorting dependencies among geometric entities. In *US States Patent 5,251,290*, Oct 1993.

[24] M Sitharam. Frontier, an opensource 3d geometric constraint solver: algorithms and architecture. *monograph, in preparation*, 2004.

[25] M Sitharam. Graph based geometric constraint solving: problems, progress and directions. In Dutta, Janardhan, and Smid, editors, *AMS-DIMACS volume on Computer Aided Design*, 2004.

[26] M Sitharam, A Arbree, Y Zhou, and N Kohareswaran. Solution management and navigation for 3d geometric constraint systems. *submitted, available upon request*, 2004.

[27] M Sitharam and Y Zhou. A tractable, approximate, combinatorial 3d rigidity characterization. *submitted to ADG 2004, available upon request*, 2004.

[28] Meera Sitharam. Frontier, opensource gnu geometric constraint solver: Version 1 (2001) for general 2d systems; version 2 (2002) for 2d and some 3d systems; version 3 (2003) for general 2d and 3d systems. In *http://www.cise.ufl.edu/∼sitharam, http://www.gnu.org*, 2004.

[29] W. Whiteley. Rigidity and scene analysis. In *Handbook of Discrete and Computational Geometry*, pages 893 –916. CRC Press, 1997.