Elimination in generically rigid 3D geometric constraint systems

Jörg Peters, Meera Sitharam, Yong Zhou, and JianHua Fan

University of Florida jorg@cise.ufl.edu

Summary. Modern geometric constraint solvers use combinatorial graph algorithms to recursively decompose the system of polynomial constraint equations into generically rigid subsystems and then solve the overall system by solving subsystems, from the leave nodes up, to be able to access any and all solutions. Since the overall algebraic complexity of the solution task is dominated by the size of the largest subsystem, such graph algorithms attempt to minimize the fan-in at each recombination stage.

Recently, we found that, especially for 3D geometric constraint systems, a further graph-theoretic optimization of each rigid subsystem is both possible, and often necessary to solve wellconstrained systems: a minimum spanning tree characterizes what partial eliminations should be performed before a generic algebraic or numeric solver is called. The weights and therefore the elimination hierarchy defined by this minimum spanning tree computation depend crucially on the representation of the constraints. This paper presents a simple representation that turns many previously untractable systems into easy exercises. We trace a solution family for varying constraint data.

1 Introduction and Motivation

Specifying geometry via constraints is an elegant and succinct approach to defining geometric composites in applications such as computer aided design, robotics, molecular modeling and teaching geometry (see e.g. [3, 22, 21]). However, being able to navigate to and access *all* valid configurations satisfying the constraints is a difficult, ongoing challenge, especially in the practically relevant 3D case. This paper develops an automated strategy for resolving generically rigid configurations by developing an interesting connection between the algebraic complexity of the underlying polynomial system and the topology of its set of its subsystems.

A *geometric constraint system* relates a finite set of geometric objects by a finite set of constraints. Constraints are represented 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 $(x_2 - x_1)^2 + (y_2 - y_1)^2 = d^2$. The *solution or realization* of a geometric constraint system is the (set of) *real* zero(es) of the algebraic system, each representing a valid choice of position, orientation and any other parameters of the geometric elements in the Euclidean plane, on the sphere, or in Euclidean 3D space. Wellconstrained systems have a finite but potentially very large number of zero-dimensional solutions. Underconstrained systems have infinitely many solutions and overconstrained systems have no solution unless they are consistently overconstrained. Wellconstrained or consistently overconstrained systems are called *rigid*.

Modern geometric constraint solvers recursively decompose, solve and recombine the polynomial geometric constraint system according to an optimized partial ordering, called DR-plan. As graph algorithms, they address *generically rigid* systems, i.e. systems that are rigid except possibly for polynomial dependencies arising from specific geometric inputs. A subgraph in a DR-plan corresponding to generically rigid subsystems is called *cluster* (see Section 2 for the formal definition). *Resolving a cluster* $C = \cup_{i=1}^{n} C_i$ means computing position and orientation of all C_i in the common coordinate system of C so that all constraints are satisfied.

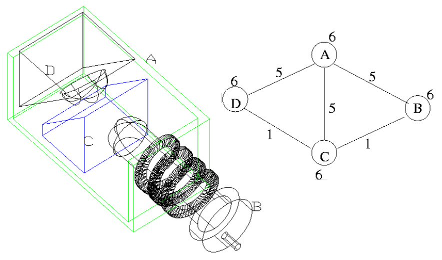

Fig. 1. An underconstrained 3D example and its dof constraint graph. Each object (housing A, screw B, clipped prism C and prism D) has 6 dofs. Incidence of B, C, D with A removes 5 dofs each. Then each of the pairs AB, AC, AD is underconstrained with one extra degree of freedom and the remaining two pairs BC and CD are connected by distance constraints that remove 1 dof each, leaving it with a density of -7.

This paper develops an automated strategy for resolving generically rigid clusters by further developing an interesting connection between the algebraic complexity of a cluster's polynomial system and the topology of its set of child clusters. It shows how a careful formulation of the problems can reduce the algebraic complexity so that many previously untractable systems turn into easy exercises.

Section 2 defines key terms, such as constraint graph, degree of freedom analysis and generic rigidity. And it outlines a general graph-based algorithm for partial elimination within one cluster. Section 3 defines the representation and the weights that drive the partial elimination. Section 4 discusses the solution of the remaining, active system and Section 5 illustrates the findings by tracing the solutions of a 3D geometric constraint system for a varying parameter.

2 Graph Structures for Elimination

The decomposition and ordering of elimination of the constraint system is optimized using three graphs: the constraint graph, the DR-plan and the overlap graph. The constraint graph represents the overall system of constraints, the DR-plan is a directed acyclic graph that represents the decomposition (and later recombination) of the constraint system into clusters. The overlap graph is the focus of this paper. It guides the partial resolution of the cluster constraints until only a small active system of active constraints remains to be solved by a general purpose solver.

Rather than focussing directly on elimination within a cluster (addressed in the subsection on overlap graphs, last in this section), this section sketches the larger picture inclusing the DR-plan to show that, once we can solve clusters, we can solve the whole constraint system. Moreover, there are some subtle issues concerning well-posed constraint systems that need to be discussed for completeness.

Constraint Graph, density, cluster and rigidity: A geometric constraint graph $G = (V, E, w)$ corresponding to geometric constraint system is a weighted graph with n vertices V representing geometric objects 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 dofs removed by the constraint represented by e. Figure 1 illustrates a small 3D constraint system and its dof constraint graph. A subgraph $A \subseteq G$ that satisfies

$$d(A) := \sum_{e \in A} w(e) - \sum_{v \in A} w(v) \geq -d_0$$

is called *dense* and $d(A)$ is called *density* of A. Here d_0 is a constant, typically $\binom{\mathrm{D}+1}{2}$, i.e. the dofs of a rigid body in D dimensions. For 2D Euclidean geometry, we expect $d_0 = 3$ and for 3D geometry $d_0 = 6$. For a rigid body fixed with respect to a global coordinate system, $d_0 = 0$.

The following purely combinatorial notions for computing a good decomposition are based on density.

A *dof-rigid cluster*, short cluster, is a constraint graph all of whose subgraphs, including itself, either have density at most $-d_0$ (wellconstrained) or can be replaced by well-constrained subgraphs so that G remains dense (well-overconstrained).

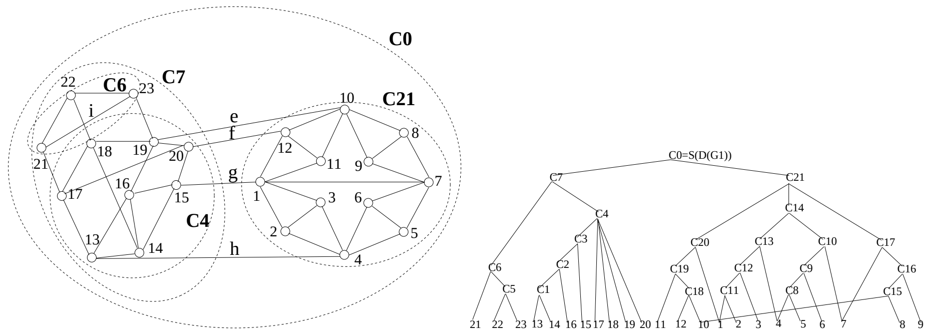

Fig. 2. Fan-in minimizing DR-plan (*right*) of a constraint graph (*left*) ; all vertex weights are 2, all edge weights 1.

A dense graph is *minimal* if it has no dense proper subgraph. Minimal dense subgraphs are clusters, but a dense graph that is not minimal can be underconstrained, i.e. not a cluster: the apparently correct density can be the result from summation with another subgraph of density greater than $-d_0$.

A constraint system is *generically rigid* if it is rigid (does not flex or has only finitely many non-congruent, isolated solutions) for all nondegenerate choices of coefficients of the system. A generically rigid system yields a cluster, but the converse is not always the case. In 3D, there are wellconstrained, generically non-rigid clusters due to the presence of generic 'hidden' *constraint dependencies* such as the 'banana' and 'hinge' configurations [6, 16, 17, 15]. These can result in overconstraints that are not detectable by a dof count.

Since to date, no tractable, combinatorial method of determining generic rigidity has been proven (see conjectures in [6, 15, 30]), we restrict the scope of our approach to the large (and possibly complete) class of constraint graphs whose rigidity is fully verified by the DR-plan [30] (2003 version) based on [12, 15, 29, 28]. This verification imposes a solving priority on the DR-plan and some dependences between clusters that do not contain one another but covers all known constraint graph dependencies. Therefore, in the following, we may assume that all hidden or explicit overconstraints have been removed.

DR-plan, covering sets, trivial subgraphs, completeness: The DR-plan of a constraint graph G is a directed acyclic graph whose nodes represent clusters in G, and whose edges represent containment. The leaves or sinks of the DR-plan are all the primitive clusters of G and the roots or sources are all the maximal clusters of G. For rigid graphs, the DR-plans have a single source but there could be many DR-plans for G. The DR-planner is a graph algorithm that generates the decomposition-recombination plan by working bottom up: at stage i, the DR-plan picks a wellconstrained cluster C_i in the current constraint graph G_i, and uses an abstract, unevaluated simplification of C_i to create a transformed constraint graph G_{i+1}. Since the complexity of finding real zeroes of sparse polynomial systems is exponential in the number of variables, the size of the largest subsystem in a DR-plan dominates the complexity of constraint solving, and DR-planners use a combinatorial degree of freedom (*dof*) analysis to minimize the number of child clusters [4, 11], or, equivalently, the fan-in, to isolate clusters (Figure 2).

Trivial subgraphs are subgraphs that correspond (after resolving incidence constraints) to a single 2D or 3D point or to a line segment in 3D. Whenever two rigid clusters overlap on a non-trivial subgraph, the cluster induced by their union is rigid. A *covering set* S of a cluster C is a set of child clusters $C_i \neq C$ whose union covers all the vertices of C. A covering set S is a *complete set of maximal clusters* if one of the following two conditions holds. Either it consists of only two clusters whose intersection is a non-trivial subgraph, or it covers all edges in C and every C_i is *maximal*, i.e. the only proper cluster of C that contains C_i is one that intersects every other cluster in S on a non-trivial subgraph. Note that S is not unique and we will use later on that there can be several covering sets of maximal clusters within S.

Even for a wellconstrained cluster C, it is still a nontrivial task to pick a guaranteed stable, independent system of polynomial equations corresponding to C. Moreover, shared objects between clusters may appear to have inconsistent coordinates due to numerical roundoff (see [19] for a description of and solution to this problem, which is not our focus here.)

Overlap Graph, spanning tree and active constraints. Figures 3 and 4 illustrate the concepts to be discussed. The *overlap graph* of a subset S of child clusters of a cluster C is an undirected graph. Its vertices are the clusters in S and there is an edge (i, j) with weight $w(k)$ if child clusters C_i, C_j overlap in a trivial subgraph with k distinct points; $w(k)$ counts the dofs after one cluster is expressed in the coordinate system of the other, e.g. $k = 1$ for a rotation about a common edge between C_i and C_j.

Any spanning tree T of the overlap graph of the clusters in S induces a system of equations for resolving C. 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 [10, 31] (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 which is counted by the sum of weights. Considering all covering sets S of C, we therefore select all spanning trees that minimize this sum.

The ordering expressed in each spanning tree defines a partial elimination of constraints. The remaining constraints (non-tree edges in S) are called *active* and form the active system of equations that must be solved. Active constraints represent, for example distance and point-matching of cluster pairs in S. While the main goal is to reduce the number of variables, the polynomial degree and separation of variables of the active system is a secondary consideration since the time complexity is polynomial in these parameters. The choice of a root of S strongly influences degree and separation as we will see in the next section.

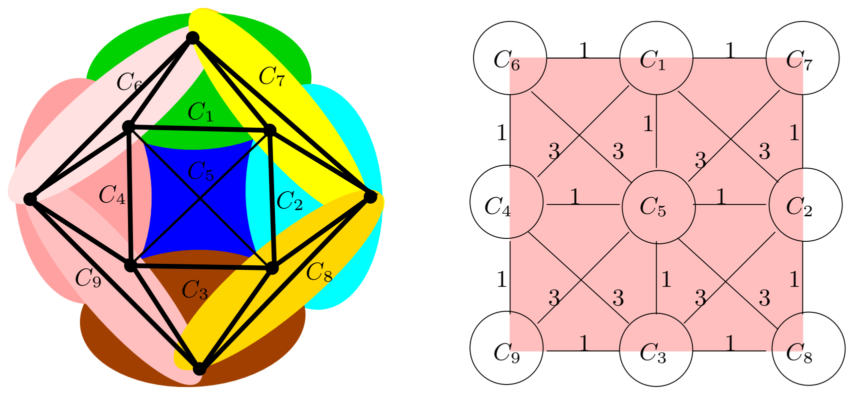

Fig. 3. (*left*) Constraint graph of problem `quad`: given are 8 vertices (indicated as •) and 18 distance constraints (edges). The graph structure is that of a square-base pyramid (see also Figure 5, inset) with the apex split into four vertices and the triangular faces folded down. The vertices are not labeled since they have different labels (and different local coordinates) in different clusters C_i. Incidence of these different instances of a vertex has to be explicity enforced. The edges are not labeled with distances, since geometry does not influence the construction of the overlap graph. In Section 5, specific edge lengths are assigned. (*right*) The corresponding weighted overlap graph (all remaining edges of the complete graph are of weight 6 and are omitted).

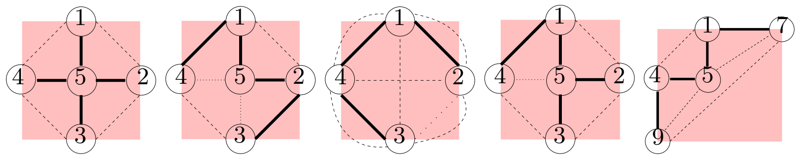

Fig. 4. Five spanning trees (*solid* edges) of covering sets of `quad` (same layout as in Figure 3(*right*) ; label i stands for C_i). The sum of weights are from left to right 4,8,9,6 and 4. These numbers match the number of *dashed* distance constraints (= 1 constraint) between clusters and *dotted* overlaps that are not tree edges (sharing of nodes = 3 constraints). When optimally rooted (with root C_5), the sum of the depths of the first, leftmost tree is 4, and for the rightmost, it is 6. Therefore the first tree is preferred for the partial elimination.

3 Partial Elimination

Based on the characterization in Section 2, we can optimize the partial elimination of a cluster C by the following algorithm

(1) Determine $\mathcal{S}_C$, the set of all minimum weight spanning trees for all covering sets S of C (first and last tree in Figure 4) and

(2) minimize the sum of the depths of all nodes for all choices of rooted trees in $\mathcal{S}_C$.

In general, due to the large number of possible covering sets, the above algorithm requires solving an NP-complete problem. Moreover, keeping track of the spanning trees with the low complexity requires complex data structures [20]. Fortunately, in practical applications, the number of child clusters is no more than ten and even an exhaustive search through all covering sets S is an efficient option.

This section now explains the partial elimination implied by the spanning tree and characterizes the degree of the resulting active equations. Once we solve the active system, recombining clusters according to the DR-plan translates solutions of the clusters into solutions of the overall constraint system. The goal of partial elimination is to express, at each level of the spanning tree, all child cluster in the common coordinate system of the current parent cluster. For a, an instance of a point in the cluster C_i, we write

$$\mathbf{x}_{a,C_i} := \left[\begin{smallmatrix} x_{a,C_i} \\ y_{a,C_i} \\ z_{a,C_i} \end{smallmatrix}\right] \in R^3 \text{ the coordinates of } a \text{ in } C_i\text{'s local coordinate system}$$

$$\mathbf{x}_{a,C_j,C_i} := \left[\begin{smallmatrix} x_{a,C_j,C_i} \\ y_{a,C_j,C_i} \\ z_{a,C_j,C_i} \end{smallmatrix}\right] \in R^3 \text{ the coordinates of } a \text{ in } C_j\text{'s coordinate system.}$$

When, as is usually the case, two clusters do not join to form a composite rigid cluster but can moved with respect to one another, the coordinates $\mathbf{x}_{a,C_j,C_i}$ are parametrized in these dofs and the dofs will appear as variables in the active system. These dofs depend, for example, on rotation angles. Once the active system is solved, the parametrized coordinates are resolved to a final position $\mathbf{x}_{a,C}$. We express the transformation experienced by a child node, in terms of the following transformations:

T_a	translation that maps a to the origin. (T_a^{-1} maps the origin to a.)
R_{ab}	rotation that maps $b-a$ to the x-axis.
M_{bc}	the matrix $[b, c, b \times c] \in \mathbb{R}^{3\times3}$ whose columns span a coordinate system in $\mathbb{R}^3$.
T	undetermined translation (three dofs).
R	undetermined rotation about the x-axis (1 dof).
Q	undetermined unit quaternion (orientation, 3 dofs).

If C_j and C_i overlap in k points, mapping each point, say $\mathbf{x}_{a,C_i}$ to $\mathbf{x}_{a,C_j,C_i}$, as required by the overlap leaves degrees of freedom in terms of the undetermined maps T, R and Q as follows.

k .	overlap points	map
3	$(a, b, c) \to (a', b', c')$	$T_{a'}^{-1} M_{b'-a',c'-a'} M_{b-a,c-a}^{-1} T_a$
2	$(a, b) \to (a', b')$	$T_{a'}^{-1} R_{b'-a'}^{-1} R R_{b-a} T_a$
1	$(a) \to (a')$	$T_{a'}^{-1} Q T_a$
0	none	QT

Here $(a, b) \to (a', b')$ means that $\mathbf{x}_{a,C_j,C_i} = \mathbf{x}_{a',C_j}$ and $\mathbf{x}_{b,C_j,C_i} = \mathbf{x}_{b',C_j}$, i.e. a and b are mapped to their two counterparts a' and b', respectively. If there is no overlap, i.e. $k = 0$, then the orientation Q and translation T are completely free.

We assume for now that the overlap points are in general position. According to Section 2, the weight of each edge in the overlap graph corresponds to the dofs of the overlap of the pair of clusters connected by the edge. Since we need zero parameters to describe the position and orientation of a cluster with respect to another if the two share three points (rigid joint), one if they share an edge, three if they share a point and six if they share no point,

$$w(0) = 6, \quad w(1) = 3, \quad w(2) = 1, \quad w(3) = 0.$$

The challenge is now to formulate the active constraints as low degree polynomial constraints whose dofs match the weight $w(k)$. Choosing the rotation angles as dofs matches the weight but does not lead to polynomial equations. With

$$R := \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}, \qquad s^2 + c^2 = 1, \text{ and}$$

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

we do obtain quadratic polynomial equations in the dofs c,s and q_i, $i = 0, 1, 2, 3$. However, the weights are too high: we have two variables for $k = 2$ and four variables for $k = 1$ instead of one and three. A simple, but very effective insight is that we can parametrize the variables c, s, q_j by stereographic projection:

$$c := \frac{1-t_0^2}{1+t_0^2}, \quad s := \frac{2t_0}{1+t_0^2}, \quad q_0 := \frac{1-\sum_{i=1}^3 t_i^2}{1+\sum_{i=1}^3 t_i^2}, \quad q_j := \frac{2t_j}{1+\sum_{i=1}^3 t_i^2}, j = 1, 2, 3.$$

This yields

overlap points k	dofs $w(k)$	(rational) degree in the t_j of the transformation
3	0	0
2	1	2
1	3	4
0	6	4 (1 for T)

The partial elimination now proceeds as follows. We traverse the selected, minimal, rooted spanning tree in depth first order. At each node, we express the child nodes (and all their children which are already expressed in the child's coordinate system) in the parent's coordinate system. The constraints obtained after this partial elimination, are called active. Since we introduce new variables with each step of the partial elimination, the *coordinate degree* of the numerator and of the denominator of the constraint systems is 2 throughout the elimination. (The coordinate degree is an n-tuple listing the degrees for each variable, e.g. (2,2) for x^2y^2.) In our examples, the active system consists of point-matching constraints and distance constraints. After clearing the denominator, the coordinate degree of a point-matching constraint is 2 and that of a distance constraint is 4.

Note that the total degree increases with the depth of the spanning tree. Moreover, since all descendant clusters are transformed with the child cluster, the equations are more separated and sparse if the spanning tree is shallow; and that means less work.

4 Solving the Active System

We are interested in real solutions to the constraint system and currently use a recursive subdivision solver for tensor-product Bézier polynomials [14] which allows us to focus on a finite domain and real solutions. In fact, we encountered the noteworthy problem when we attempted to solve the equations of the example `quad` (see Figure 3) with an algebraic solvers, the online version of Synaps [1]. Since algebraic solvers typically consider both real and complex solutions, we did not receive an answer to `quad`, whose generically rigid constraints are rigid over the reals (isolated roots) but algebraically dependent over the complex numbers, yielding a 1-parameter family of complex solutions. (We did eventually get our results confirmed by Synaps after substituting one correct real value for one of the parameters).

When using a subdivision-based solver on the standard Bézier domain $[0..1]^N$, there is a price to pay for the stereographic parameterization. For numerical stability, we restrict the parametrization to $t_i \in [-1..1]$ and create another active constraint system for $1/t_i \in [-1..1]$. Change of variables then maps each finite domain to the Bézier domain [0..1]. This yields up to 2^N separate constraint systems when there are N variables. However, since every subdivision step generates in principle an exponential number of subproblems (to be efficiently pruned), the cost of starting with an exponential number of systems is negligible and preferable to a higher number of variables and equations. For the particular solver, we also needed to filter out duplicate roots obtained on the multiply covered boundaries of the subdivision domains.

5 Solutions to families of constraint problems

As our main example, we consider the constraint system shown in Figure 3. The eight nodes of `quad` can be viewed as a Stewart mechanism with a quadrilateral fixed base and a quadrilateral work platform. (Below, we will expand and shrink the work platform according to a parameter r, while in typical engineering applications the work platform is moved by adjusting the edge lengths between the two platforms.) Evidently, choosing the minimal covering set (with four clusters, Figure 4, center) does not yield the best partial elimination since the sum of dofs is 9. The minimal dof sum is 4, and the leftmost tree, with root C_5 is preferred to the rightmost tree, because its depth is 1. That is, the algorithm *automatically* generates the reduced, active system that we might have derived, without proof of minimality, by intutive selection: fix cluster C_5 as the quadrilateral base platform of the Stewart mechanism and parametrize the four points of the moving platform of the Stewart mechanism each by one parameter. There are four overlap constraints, between C_5 and $C_j, j = 1, 2, 3, 4$, that imply and hence allow to automatically discard, the overlaps between C_i and $C_j, j := i \mod 4+1$. The coordinates of the root cluster C_5 are

$$\mathbf{x}_{1,C_5} := [\begin{smallmatrix}1\\1\\0\end{smallmatrix}], \quad \mathbf{x}_{2,C_5} := [\begin{smallmatrix}-1\\1\\0\end{smallmatrix}], \quad \mathbf{x}_{3,C_5} := [\begin{smallmatrix}-1\\-1\\0\end{smallmatrix}], \quad \mathbf{x}_{4,C_5} := [\begin{smallmatrix}1\\-1\\0\end{smallmatrix}],$$

a 2-unit square. The local coordinates of each of the cluster C_i, $i = 1, 2, 3, 4$ are

$$\mathbf{x}_{1,C_i} := [\begin{smallmatrix}-1\\1\\0\end{smallmatrix}], \quad \mathbf{x}_{2,C_i} := [\begin{smallmatrix}1\\1\\0\end{smallmatrix}], \quad \mathbf{x}_{3,C_i} := [\begin{smallmatrix}0\\4\\0\end{smallmatrix}].$$

The four distance constraints are

$$\|\mathbf{x}_{3,C_i} - \mathbf{x}_{3,C_j}\| = r. \qquad j := i \mod 4 + 1.$$

Figure 5 illustrates all possible configurations for different choices of r (with $r = 0$ being the minimum and $r = 2\sqrt{10}$ the maximal distance possible for real solutions). The time needed to solve is in the range of seconds.

While the active system corresponding to a similar five-sided platform is not solvable by Maple, Maple `solve` returns an answer to the active constraints of `quad`: three 1-parameter families of solutions and one singleton. Painstaking examination confirms correctness of the output of the subdivision solver: there are only isolated, 0-dimensional real solutions.

References

1. B. Mourrain et al. Synaps Library, web interface *http://www-sop.inria.fr/galaad/logiciels/synaps/html/index.html*
2. R. Latham and A. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28:917–928, 1996.
3. Christoph M. Hoffmann and Andrew Lomonosov and Meera Sitharam. Geometric constraint decomposition. In Bruderlin and Roller Ed.s, editor, *Geometric Constraint Solving*. Springer-Verlag, 1998.
4. Christoph M. Hoffmann and Andrew Lomonosov and Meera Sitharam. Decomposition of geometric constraints systems, part i: performance measures. *Journal of Symbolic Computation*, 31(4), 2001.
5. Christoph M. Hoffmann and Andrew Lomonosov and Meera Sitharam. Decomposition of geometric constraints systems, part ii: new algorithms. *Journal of Symbolic Computation*, 31(4), 2001.

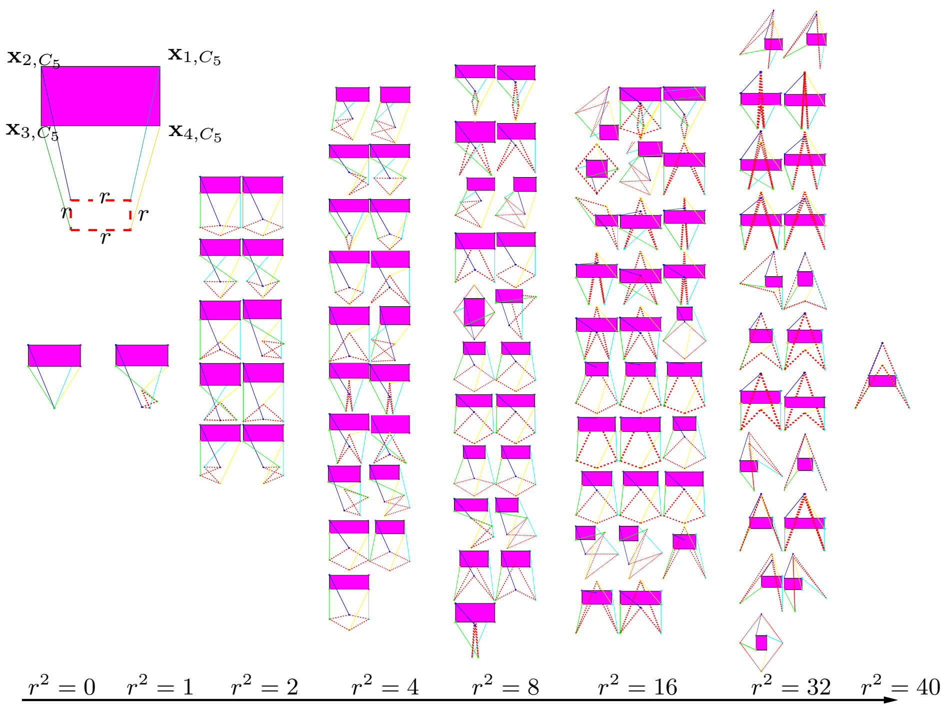

Fig. 5. All possible configurations of `quad` parametrized by r, the value of the four distance constraints displayed as thick *red* dashed lines. The gallery contains both macro information (number of non-isomorphic solutions) and micro information (geometry of the realization). For example, (cf. *upper left* inset). $r = 0$ corresponds to a square-based pyramid (with the solid (*purple*) rectangle representing cluster C_5; for each of C_1, C_2, C_3 and C_4 only one edge is displayed.) $r^2 = 40$ corresponds to a saddle with two opposing triangle faces flipped up and two flipped down. Find the planar configuration!

6. Jack E. Graver and Brigitte Servatius and Herman Servatius. *Combinatorial Rigidity*. Graduate Studies in Math., AMS, 1993.
7. I. Fudos and C. M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16:179–216, 1997.
8. 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.
9. D. Cox and J. Little and D. O'Shea. *Using algebraic geometry*. Springer, 1998.
10. 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.
11. C Hoffman and M Sitharam and B Yuan. Making constraint solvers more useable: the overconstraint problem. *CAD*, 36(4), 377-399, 2004.

12. Andrew Lomonosov and Meera Sitharam. Graph algorithms for geometric constraint solving. In *submitted*, 2004.
13. S. Ait-Aoudia and R. Jegou and D. Michelucci. Reduction of constraint systems. In *Compugraphics*, pages 83–92, 1993.
14. J Gaukel. Effiziente Lösung polynomialer und nichtpolynomialer Gleichungssysteme mit Hilfe von Subdivisionsalgorithmen, PhD thesis, Mathematics, TU Darmstadt, Germany, 2003.
15. M Sitharam and Y Zhou. A tractable, approximate, combinatorial 3d rigidity characterization. *proceedings of ADG 2004*, 2004.
16. Henry Crapo. Structural rigidity. *Structural Topology*, 1:26–45, 1979.
17. Henry Crapo. The tetrahedral-octahedral truss. *Structural Topology*, 7:52–61, 1982.
18. M Sitharam and J. Peters and Y Zhou. Solving minimal, wellconstrained, 3D geometric constraint systems: combinatorial optimization of algebraic complexity *submitted, http://www.cise.ufl.edu/∼sitharam*, 2004.
19. M Sitharam and A Arbree and Y Zhou and N Kohareswaran. Solution management and navigation for 3d geometric constraint systems. *to appear, ACM TOG*, 2005.
20. 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.
21. I. Fudos. *Geometric Constraint Solving*. PhD thesis, Purdue University, Dept of Computer Science, 1995.
22. G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.
23. G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.
24. J. Owen. www.d-cubed.co.uk/. In *D-cubed commercial geometric constraint solving software*.
25. J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modeling*, pages 397–407, Austin, Tex, 1991.
26. 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.
27. J.A. Pabon. Modeling method for sorting dependencies among geometric entities. In *US States Patent 5,251,290*, Oct 1993.
28. M Sitharam. Frontier, an opensource 3d geometric constraint solver: algorithms and architecture. *monograph, in preparation*, 2004.
29. M Sitharam. Graph based geometric constraint solving: problems, progress and directions. In Dutta and Janardhan and Smid, editor, *AMS-DIMACS volume on Computer Aided Design*, 2004.
30. 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.
31. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial system. *Math. Comp.*, 64:1541–1555, 1995.
32. W. Whiteley. Rigidity and scene analysis. In *Handbook of Discrete and Computational Geometry*, pages 893 –916. CRC Press, 1997.