Algebraic Solution for Geometry from Dimensional Constraints

J.C. Owen

D-Cubed Ltd.
68 Castle Street
CAMBRIDGE
England

We investigate general configurations of distance and angle dimensions between points, lines and circles on a plane. A simple graphical representation is described for the system of coupled quadratic equations which results from treating the geometries as variables and the dimensions as defining equations. For many configurations of practical interest we show that these equations are poorly suited to numerical solution.

We describe an algorithm for computing the solution to a subset of all possible configurations of geometry and dimensions using purely algebraic methods (in fact the usual arithmetic operations plus square roots). We show that this algorithm solves for all configurations of a practically useful type and that it solves for any configuration which can in principle be solved using these algebraic operations. Specifically, we use the Galois theory of equations to show that the following statements are equivalent.

1. The geometry can be constructed in principle on a drawing board using a ruler and compasses.

2. The coordinates of the geometries can be computed algebraically using only arithmetic operations plus square root.

3. The coordinates of the geometries lie in a normal field extension over the dimension values of degree 2^n for some n.

4. For general (i.e. algebraically independent) dimension values the algorithm described will compute the geometries.

We also describe a working implementation of the algorithm and describe some extensions to the basic ideas which are necessary to make it a practically useful way to specify geometry by means of dimensional constraints.

1. INTRODUCTION

The idea of defining geometries from the dimensions and constraints which the geometry should satisfy is as old as computer graphics (Sutherland, Hillyard and Braid, Light and Gossard, Serrano and Gossard).

The early approaches considered only points as variables. They used standard numerical iteration techniques to solve the system of non-linear equations which are derived from distance dimensions between the points. Later work added new types of geometry and other variables to the equations and also introduced more sophisticated numerical techniques. The main disadvantage with the numerical methods is their unreliability, especially near singularities and their poor performance.

An alternative approach has been developed using ideas from constraint based computer languages (Freeman-Benson et al, Todd, Leler). These methods involve manipulating the equations so that the variables can be computed sequentially starting from known variables. The main disadvantage with this method is the limitation and poor characterisation of the sorts of equations which can be solved.

The aim of this work is to consider in detail the equations which are obtained when points, lines and circles on a plane are defined by relative distance and angle constraints. The main result is that these equations can be solved algebraically for a significant class of configurations. These are all configurations which could in principle be constructed, by some means or other, on a drawing board (or computer screen) using standard drafting construction techniques. We describe an algorithm for computing the solutions and we show that this algorithm finds the solutions whenever they could in principle be found alge-

braically using only arithmetic operations plus square root. We describe such configurations as quadratically solvable or solvable by quadratics.

The theoretical results are achieved using the Galois theory of equations. A secondary aim of this work is to show how this theory can be usefully applied in computational geometry. There are many problems in computational geometry which can be represented as the solution of simultaneous polynomial equations. In many circumstances, especially interactive or 'real time' computer programs it is highly desirable to avoid numerical iteration techniques and indeed to limit oneself to simple operations such as arithmetic and square roots. The Galois theory provides the framework within which it is possible to determine whether a problem can be solved exactly using these operations and if so proving that a particular algorithm will always find the solutions.

We should emphasise that the proof of the completeness of our algorithm for quadratically solvable configurations holds only for general dimension values i.e. where there are no special relationships between the dimension values. For specific dimension values (where we may make use of relations between the values) a wider class of constraint configurations may be quadratically solvable. However the assumption of general dimension values is appropriate for an algorithm which is required to compute the solutions for whatever values are given.

The paper is organised in six sections. Section 2 describes the variables that are used to represent points, lines and circles. A set of simultaneous quadratic equations in these variables result from specifying the relative distances and angles between the geometries. These equations are shown to have an exponentially large number of discrete solutions and we argue that this fact alone makes them badly conditioned for numerical iterative solutions.

In section 3 we develop a simple graphical representation scheme for the equations and we describe an algorithm for solving them. It consists of breaking the graph into pieces each of which is solvable quadratically and then combining these together to get a final solution.

In Section 4 we introduce the Galois theory of equations and use it to classify all configurations which can in principle be solved quadratically. The same theory allows us to show that the algorithm given in Section 3 will always find these solutions. The demonstrations given are not intended to be fully rigorous since this would assume a familiarity with Galois theory which is inappropriate for this audience. Rather the intention is to demonstrate the general method and to make the line of argument clear. Detailed proofs will be published elsewhere.

In Section 5 we describe a fully implemented software module whose function is to compute geometry from dimensions and constraints. We describe some of the extensions which are necessary to the basic algorithm of Section 3 in order to make it a useful system. These include managing underconstrained and over constrained systems and providing users with useful feedback when abnormal conditions occur. Finally Section 6 gives our conclusions.

2. THE CONSTRAINT EQUATIONS

2.1 Representation of points and circles

A point is represented by a two component vector

$$\boldsymbol{p} = (p.x,\ p.y) \tag{2.1}$$

A circle is represented by a three component vector describing its centre point $\boldsymbol{p}$ and radius r.

$$c = (\boldsymbol{p},\ r) \tag{2.2}$$

2.2 Representation of lines

A line on a plane has two degrees of freedom and so can in principle be represented by two real numbers, for example a distance from the origin and an angle to the x-axis. However this form is not very convenient for deriving the equations which result from constraining points or other lines to the line. Provided the line does not pass through the origin it is represented uniquely by the single point on the line nearest to the origin and this is the form we will use. In order to show the quadratic structure of the constraint equations it is convenient to assume that the distance from the line to the origin is a third independent variable and to introduce an extra constraint equation. Thus a line is represented by

$$1 = (\boldsymbol{p},\ r) \tag{2.3a}$$

$$r^2 = \boldsymbol{p}.\boldsymbol{p} \tag{2.3b}$$

where $\boldsymbol{p}$ is the point on the line nearest to the origin and r is the distance to the origin.

2.3 Constraint Equations

Distance and angle dimensions give constraint equations which are quadratic in the coordinates of the geometries.

2.3.1 Distance Dimensions

Distance (d) from point $(\boldsymbol{p}_1)$ to point $(\boldsymbol{p}_2)$

$$(\boldsymbol{p}_1 - \boldsymbol{p}_2).(\boldsymbol{p}_1 - \boldsymbol{p}_2) = d^2 \tag{2.4}$$

Distance (d) from point $(\boldsymbol{p})$ to line (l, r)

$$\boldsymbol{p}.l = r^2 \pm dr \tag{2.5}$$

Distance (d) from line $(l_1,\ r_1)$ to line $(l_2,\ r_2)$ This specifies that the lines are parallel and a certain distance apart and gives two constraint equations

$$(l_1.x)(l_2.y) - (l_1.y)(l_2.x) = 0$$

$$(r_1 - r_2)^2 = d^2 \tag{2.6}$$

The fact that a distance dimension between two lines leads to two constraint equations rather than one leads to complications in the presentation of some of the arguments in

the following sections but does not change any of the results. We will therefore assume for simplicity that a distance dimension between two lines does not occur.

Distance (d) from circle $(\boldsymbol{c}_1,\ r_1)$ to point $(\boldsymbol{p}_2)$

A dimension to a circle specifies the distance to the circumference of the circle.

$$(\boldsymbol{c}_1 - \boldsymbol{p}_2).(\boldsymbol{c}_1 - \boldsymbol{p}_2) = (d \pm r_1)^2 \tag{2.7}$$

Distance (d) from circle $(c_1,\ r_1)$ to circle $(c_2,\ r_2)$

$$(\boldsymbol{c}_1 - \boldsymbol{c}_2).(\boldsymbol{c}_1 - \boldsymbol{c}_2) = (d \pm r_1 \pm r_2)^2 \tag{2.8}$$

Distance (d) from circle $(c_1,\ r_1)$ to line $(l_2,\ r_2)$

$$\boldsymbol{c}.\boldsymbol{l} = r_2^2 \pm (d \pm r_1)r_2 \tag{2.9}$$

A $\pm$ sign appears in many of these equations showing the possibility of multiple interpretations of the constraint equations. These could be eliminated by squaring the equation but this would raise the degree of the problem and it would no longer be clear that we had a system of quadratic equations. We therefore prefer to regard these signs as input into the system of equations. They describe which side of a line a point lies on or whether a dimension to the circumference of a circle measures across the circle.

2.3.2 Radius Dimensions

The radius of a circle may be specified by giving the circle a radius dimension. This simply sets the radius to a fixed value and it can be considered as a coefficient in the equations and not a variable.

A constraint equation which involves a circle with a dimensioned radius is the same as the equation for the centre point of the circle with the radius added or subtracted from the dimension value. Thus, if all circle radii are dimensioned we may transform to a system consisting only of points and lines.

2.3.3 Angle Dimensions

We assume that angle dimensions are always given as the cosine (a) of the angle between a pair of lines $(\boldsymbol{l}_1,\ r_1)$ and $(\boldsymbol{l}_2,\ r_2)$.

$$\boldsymbol{l}_1.\boldsymbol{l}_2 = ar_1r_2 \tag{2.10}$$

2.3.4 Logical constraints

We are restricting attention to dimensional constraints containing a numerical value. Many logical constraints such as coincidence, tangency, parallel and perpendicular occur as dimensions with a special value (zero or a multiple of $\pi/2$).

2.4 Discussion of the Constraint Equations

It should now be clear that any dimensioning scheme between points, circles and lines where the distance or angle between the various geometries are specified can be written as a system of simultaneous quadratic equations using the variables given in equations (2.1) - (2.3a) and the constraint equations given in (2.3b) - (2.10).

Reference geometries can be treated by regarding their coordinates as coefficients in the equations and not as variables. This does not change the quadratic nature of the equations.

2.4.1 Rigid body freedom

In the absence of fixed geometries the fact that these equations depend only on the relative positions of the geometry means that the solutions are always degenerate under rigid body transformations. This means that once a solution has been found a rigid body transformation can be applied to position both the coordinates of one geometry (and one coordinate of a second). In the following discussion we will assume that the degeneracy under rigid body motion has been removed.

2.4.2 Description of the solution space

The constraint equations form a set of polynomial equations with the coordinates of the geometries as variables. If we allow these variables to become complex we may describe the equations by the dimensionality of the solution space.

If there are no solutions the equations are said to be overconstrained.

If the solution space has dimension zero i.e. the solutions are discrete points the equations are said to be well constrained.

If the dimensionality is greater than zero the solutions lie in a continuum and the equations are said to be underconstrained.

In most of the rest of the paper we will assume that the equations are well constrained. In the description of the algorithm however we will point out how under- and over constrained systems can be recognised.

2.4.3 Multiple discrete solutions

For a well constrained system we can estimate the number of discrete solutions which the constraint equations may have. For simplicity we assume a system with P points and fix three coordinates. There must be $2P-3$ distance constraints and Bezout's theorem (Kendig) states that a system of $2P-3$ quadratic equations has $2^{(2P-3)}$ solutions. In general we must expect that the number of solutions is an exponential function of the number of geometries.

A word of caution is in order. Bezout's theorem gives the number of solutions in complex, projective space. We are interested only in solutions in real affine space and so Bezout gives only an upper bound. In many practical schemes most of these solutions will no doubt turn out to be imaginary. This is not guaranteed however. Figure 1 shows a fully constrained dimensioning scheme which for P points has $2^{(P-3)}$ distinct, real, finite solutions.

This multiplicity of solutions has two effects. First we must have some algorithm for determining which one of the many solutions is required. Algebraic methods can reference any of the many possible solutions and allow us in principle to locate a solution with certain characteristics

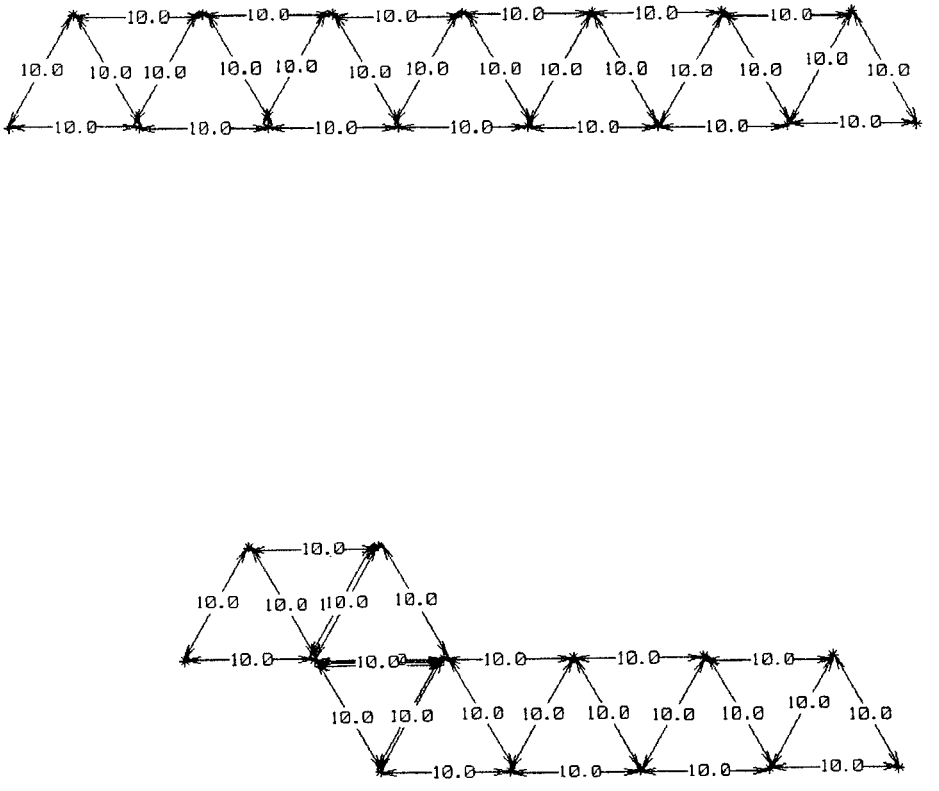

Figure 1 A constraint configuration of P points which has 2**(P-3) distinct real solutions. The configuration may be 'folded' over any number of the non-horizontal dimensions to generate a new solution.

from among all possible solutions. Numerical iteration schemes on the other hand will normally find the solution which is nearest (in some sense) to an initial starting configuration.

Second, this huge multiplicity of solutions might be expected to make the equations very poorly conditioned for solution by numerical iteration. Most numerical methods involve seeking a minimum in the sum of the squares of the residuals of the equations. A solution is found when a zero minimum is reached. The methods become inefficient whenever the residual function has a non-zero minimum or a saddle point with respect to the variation of the variables. Since a saddle point occurs on each path connecting a pair of minima the example in Figure 1 has of the order of $2^{(2P)}$ saddle points where the iteration is badly behaved.

3. GRAPHICAL REPRESENTATION AND SOLUTION OF THE CONSTRAINT EQUATIONS

3.1 Graphical Representation

We use an undirected graph consisting of nodes and pairwise bonds to represent the constraint equation. The geometries are represented by the nodes and the constraints (which always connect two geometries) are represented by the bonds. This form of graph has been widely used in many mathematical applications (Van Leeuwen et al, Owen and Ripka) and its properties have been studied extensively in computer science (Aho, Hopcroft and Ullman).

We often find it convenient to label the vertices according to whether they represent points, lines or circles. Notice that each node represents several degrees of freedom or variables in the equations. Each bond represents a constraint equation.

Figure 2 shows a simple dimensioned drawing together with the graph representing the constraint equations. A finite piece of line is represented as an (unbounded) line with two points constrained to lie on it.

3.2 Solving the Equations

3.2.1 General Considerations

It is clear that any node which is connected into the graph only by a number of constraint bonds equal to its number of degrees of freedom can be computed once the geometries to which it is connected have been computed. This observation underlies the propagation method for solving these equations (Leler, Todd). It is straightforward to check that for points, lines and circles with a dimensioned radius the unknown geometry can always be computed using only square roots plus arithmetic operations. When the unknown geometry is a variable radius circle there are three supporting geometries and the problem of computing the circle is known as the problem of Apollonius. Its solution involves only square roots (Pedoe).

There is however another class of graphs for which the geometry can be solved exactly. This occurs when the complete graph can be broken up into two or more subgraphs which have at most two nodes in common. If an algebraic solution can be found separately for one of the subgraphs then the distance or angle between the common geometries can be determined and used in the computation of the remaining subgraph. The subgraphs are then positioned with respect to one another so that the common geometries in both pieces coincide. This positioning involves applying the same rigid body transformation to all geometries in the subdiagram and so does not destroy the solution to the constraint equations.

It is straightforward to check that for all combinations of the common geometry pair the transformation needed to position the subgraphs can be computed using only square roots.

3.3 Graph theoretical description of the algorithm

We begin with a few definitions describing the decomposition of a graph into components (Hopcroft and Tarjan, Van Leeuwen).

A graph is connected if every node is connected to every other node by at least one path of bonds.

An articulation node is a node such that the graph falls into two or more disconnected pieces if it is split at that node. More formally an articulation node is a node which lies in the interior of all paths between some pair of nodes.

A graph with more than two nodes and no articulation nodes is biconnected. Thus for a biconnected graph every node is connected to every other node by at least two distinct paths.

A connected graph may be decomposed uniquely into single bonds and biconnected components by splitting it at articulation nodes (the articulation node is duplicated and placed in each piece).

We remark that the graph corresponding to a well constrainted system has no articulation nodes since otherwise the two components could be given a relative rotation around an articulation node representing a point or circle or a translation along an articulation node representing a line.

An articulation pair is a pair of nodes in a biconnected graph such that the graph falls into two or more pieces, each containing at least one other node, if it is split at both nodes of the articulation pair. More formally an articulation pair is a pair of nodes at least one of which lies in the interior of all paths between some pair of nodes.

A graph with more than three nodes and no articulation pairs is called triconnected.

A biconnected graph can be decomposed into split components by splitting it at articulation pairs. The articulation nodes are duplicated into each subgraph and an extra virtual bond is inserted between the new nodes (except if one of the components is a single bond in which case this bond is duplicated). The split components will consist of a single bond, triangles, and triconnected graphs.

A technical remark is that in order to get a unique decomposition into split components it is necessary to recombine certain triangles to form polygons (Hopcroft and Tarjan). However for well constrained configurations such polygons

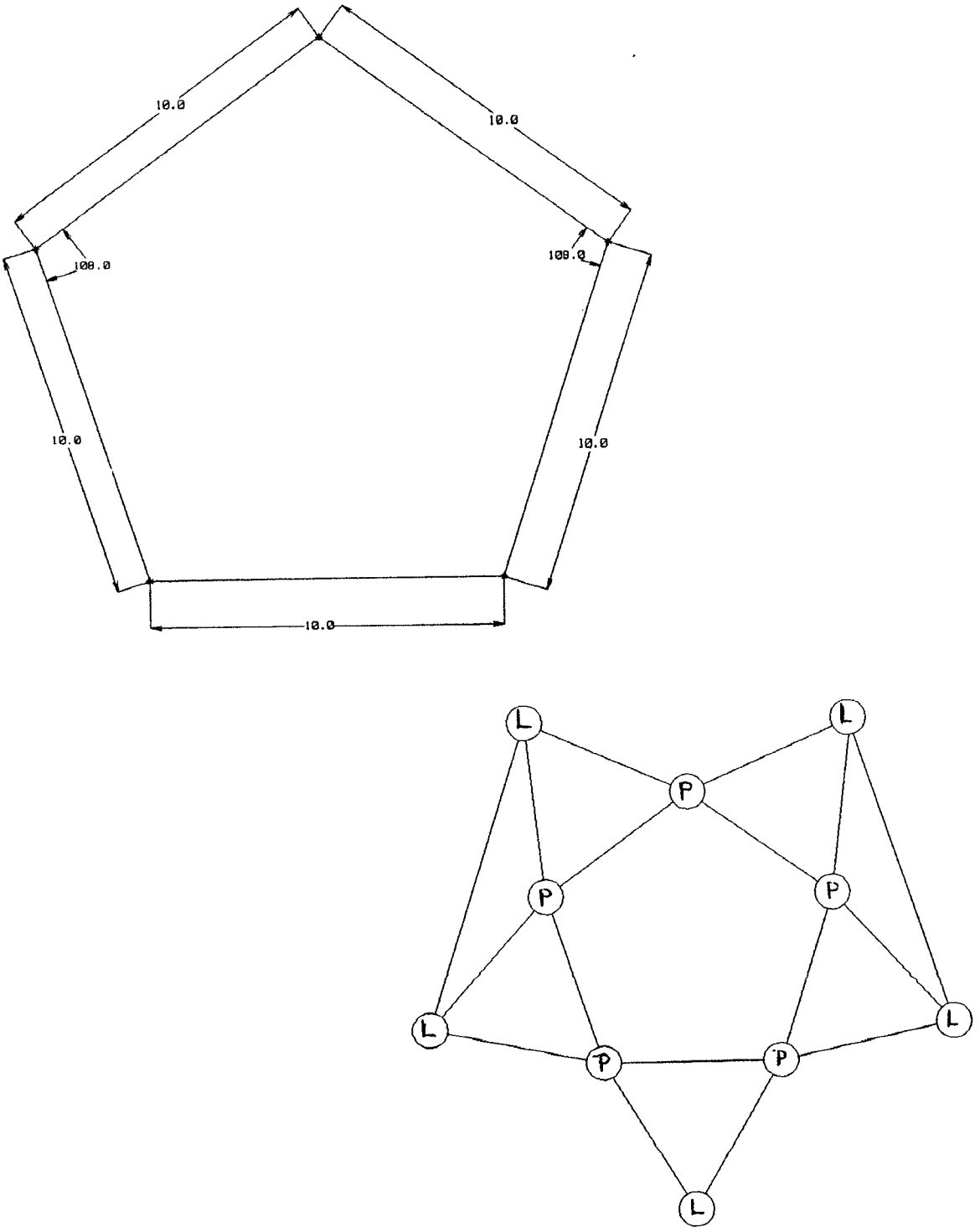

Figure 2 A simple dimensioned drawing with its constraint graph. The one upper point and the lower two points give three different articulation pairs.

never occur and the recombination is not necessary.

The algorithm for computing an arbitrary graph splits into two phases.

3.3.1 Analysis Phase

1. Split the graph into biconnected components. A well constrained system has only one biconnected component.

2. Split each biconnected component into split components, inserting virtual bonds as necessary.

The split components are either single bonds, triangles, polygons or more complex subgraphs. A well-constrained system has no polygons.

3. At any articulation pair with no single bond and exactly one more complex subgraph, remove the virtual bond from the subgraph.

The removal of virtual bonds may allow further decomposition and removal of virtual bonds. Apply the algorithm recursively from stage 2.

4. The algorithm terminates when the graphs cannot be split further. The equations cannot be solved quadratically if any triconnected subgraphs remain.

At the end of the analysis phase the original graph has been decomposed into a set of triangles whose bonds are either the original bonds or virtual bonds whose dimension values will be obtained from subgraphs composed of triangles.

3.3.2 Computation Phase

The computation proceeds in reverse order to the analysis phase. First triangles with no virtual bonds are computed (using only quadratics) and these are positioned to form the subgraphs (again using only quadratics). The dimension value to use for virtual bonds is obtained from these subgraphs and the procedure is iterated until all the geometries have been computed.

4. GALOIS THEORY

In the preceding section we have given an algorithm which will compute the solution to certain configurations of geometry and dimensions. It is clearly important to know what kinds of configurations it will solve, whether they correspond to some practically interesting class of problems and whether the algorithm could be improved to solve perhaps for a wider class of problems. We will introduce the Galois theory of equations to address these issues.

Galois theory was invented around 1830 by Galois, building on earlier work by Gauss, Abel and others. Its basic idea is to classify the kinds of numbers that can result as the roots of polynomial equations and to look at invariants with respect to permutations among the roots of polynomials.

The theory led to the solution of several classical problems such as the proof of the impossibility of certain geometrical constructions (such as trisecting an arbitrary angle) and was used to show that a general polynomial equation of degree higher than four cannot be solved algebraically.

We will use standard results from Galois theory (Stuart, Herstein) to describe the configurations of geometries and dimensions that can in principle be solved using only quadratic equations. These include all configurations which can be constructed using a ruler and compass. Almost all construction techniques that we are familiar with from computer aided design can be reduced to ruler and compass constructions. Exceptions would be some freeform curve construction methods.

Galois theory can also be used to show that the algorithm given in the previous section will compute the geometries whenever they could in principle be found by solving quadratic equations.

We will show the equivalence of the following statements

1. The geometry can be constructed in principle to satisfy the dimensions using a straight edge and compasses and ruler with spacings equal to the distance dimension values and the cosine of the angle dimension values.

2. The configuration can in principle be solved algebraically using only arithmetic operations plus square root.

3. The coordinates for the geometries all lie in a normal field extension over the dimension values of degree 2^n for some n.

4. For general dimension values and assuming all circles have a dimensioned radius, the algorithm described above computes the solutions.

We first show the equivalence of the first three statements. These describe the type of configurations which can be solved in principle using quadratics.

1 implies 3

It is a standard result (Stuart) that straight edge and compass constructions yield only points whose coordinates lie in field extensions of degree a power of two over the coordinates of a set of initial points. This is because the intersections of lines and circles can always be found by solving quadratic equations. The ruler with distance and angle dimension values can be used to define this initial set of points. The coordinates of the lines as given in equation (2.3) lie either in the same field or in a field extension of degree 2 over the coordinates of any two points on the line.

3 implies 2

A normal field extension of degree 2^n can be decomposed into a sequence of field extensions each of degree 2. Thus all coordinates in the configuration can be obtained by solving a sequence of quadratic equations.

2 implies 1

It is a standard result that points with coordinates in a normal field extension of degree 2^n can be constructed using a straight edge and compass starting from points with coordinates in the base field. These points can be constructed with the ruler measuring dimension values. Lines can be constructed by constructing their nearest point to the origin.

4 implies 3

The algorithm described finds coordinates by solving quadratic equations whose coefficients are either the dimension values or previously computed coordinates. Thus all coordinates which are computed lie in a field extension of degree 2^n over the dimension values.

3 implies 4

This is the step which states that the algorithm given will always find solutions when they exist as the solutions to quadratic equations. We will not give a complete proof here but will indicate the essential ingredients. The arguments are easier to present if we assume that the geometries are always points.

The first ingredient is to specify what is meant by general dimension values. Roughly speaking this means that we may not assume any special relationships among the dimension values. Mathematically this is translated into the statement that there are no algebraic relations between the dimension values. Of course this will never be the case in practice since we use rational values and all rational numbers obey an algebraic relation. It does however correspond with what is normally meant by general values. The statement that a polynomial with general coefficients and degree higher than four cannot be solved algebraically assumes a similar meaning for the term general.

The fact that there are no algebraic relations between the dimension values has important consequences. It means that there can be no algebraic relations between the coordinates of the geometries apart from those implied by the constraints themselves. For if there were we could add this relation to the constraint equations to get an over constrained set. Elimination Theory would then give an algebraic relation among the dimension values. We may thus assume for example that in a general solution three points are not colinear.

The second ingredient is the following theorem

Theorem: The constraint equations represented by a triconnected graph do not have solutions in a normal field extension of degree 2^n over the dimension values for any finite n.

Proof: We first show that if the theorem were false then none of the geometries can have both coordinates in a field extension of degree smaller than 2^n. We then argue that this is absurd.

Lemma: If the solutions to the constraint equations represented by a triconnected graph have coordinates in a normal field extension of degree 2^n then none of the geometries can have both coordinates in a smaller field.

Proof: A normal field extension of degree 2^n can be decomposed into a sequence of field extensions each of degree 2 over its predecessor. Call these $F0....Fn$ where $F0$ is the field spanned by the dimension values. We may say that a geometry lies in a field if and only if both its coordinates are in the field.

Suppose that at least one geometry lies in $Fn-1$ and the rest lie in Fn but not in $Fn-1$. We will show that each connected set of geometries not in $Fn-1$ is connected by bonds to at most two geometries in $Fn-1$. Then these geometries are an articulation node or pair for the graph and it cannot be triconnected.

The geometries that lie in Fn and not in $Fn-1$ can be written as

$g_j = \boldsymbol{u}_j + y\boldsymbol{v}_j$, g_j not in $Fn-1$

where $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are vectors with coordinates in $Fn-1$, $\boldsymbol{v}_j$ is not the zero vector and y is the new basis coordinate for the field extension of Fn over $Fn-1$. The coordinate y is in fact the square root of an element in $Fn-1$ and the transformation of y into $-y$ yields another solution to the constraint equations

$g_j = \boldsymbol{u}_j - y\boldsymbol{v}_j$, g_j not in $Fn-1$

g_j unchanged for g_j in $Fn-1$

The transformation for each g_j not in $Fn-1$ can be represented as a reflection about a line perpendicular to $\boldsymbol{v}_j$. It is the element of the Galois group corresponding to the field extension of Fn over $Fn-1$.

Now this line of reflection must be the same for all the g_j not in $Fn-1$ which are connected by a constraint equation since the reflection must conserve the value of the constraint in order to satisfy the equations. If the reflection lines were not the same this would imply an algebraic relation between the lines of reflection and this would ultimately yield an algebraic relation between the dimension values. Thus by a chaining argument the reflection lines for all connected geometries are the same.

A similar argument shows that any geometries in $Fn-1$ which are connected to geometries not in $Fn-1$ must lie on this reflection lines.

But the algebraic independence of the dimension values means that at most two geometries can lie on a line and so the geometries which are connected but not in $Fn-1$ are connected to at most two geometries in $Fn-1$. These two geometries are an articulation node or pair and the graph cannot be triconnected. This proves the lemma.

An obvious extension of this lemma shows that any division of the coordinates into different normal field extensions corresponds to a separation of the graph at an articulation node or pair. We remark that articulation nodes do not occur for a well constrained configuration.

The theorem follows since it is impossible for all the coordinates in a triconnected graph to be in the same field extension. For example we may apply a rigid transformation to return the coordinates of one geometry to the base field and this would imply that all coordinates were in the same field as the dimensions.

We have the following simple picture. Whenever the constraint equations are solvable quadratically the solutions lie in a normal field extension of degree 2^n over the dimension values and the elements of the Galois group of the extension represent reflection symmetries of subgraphs about articulation pairs in a larger subgraph.

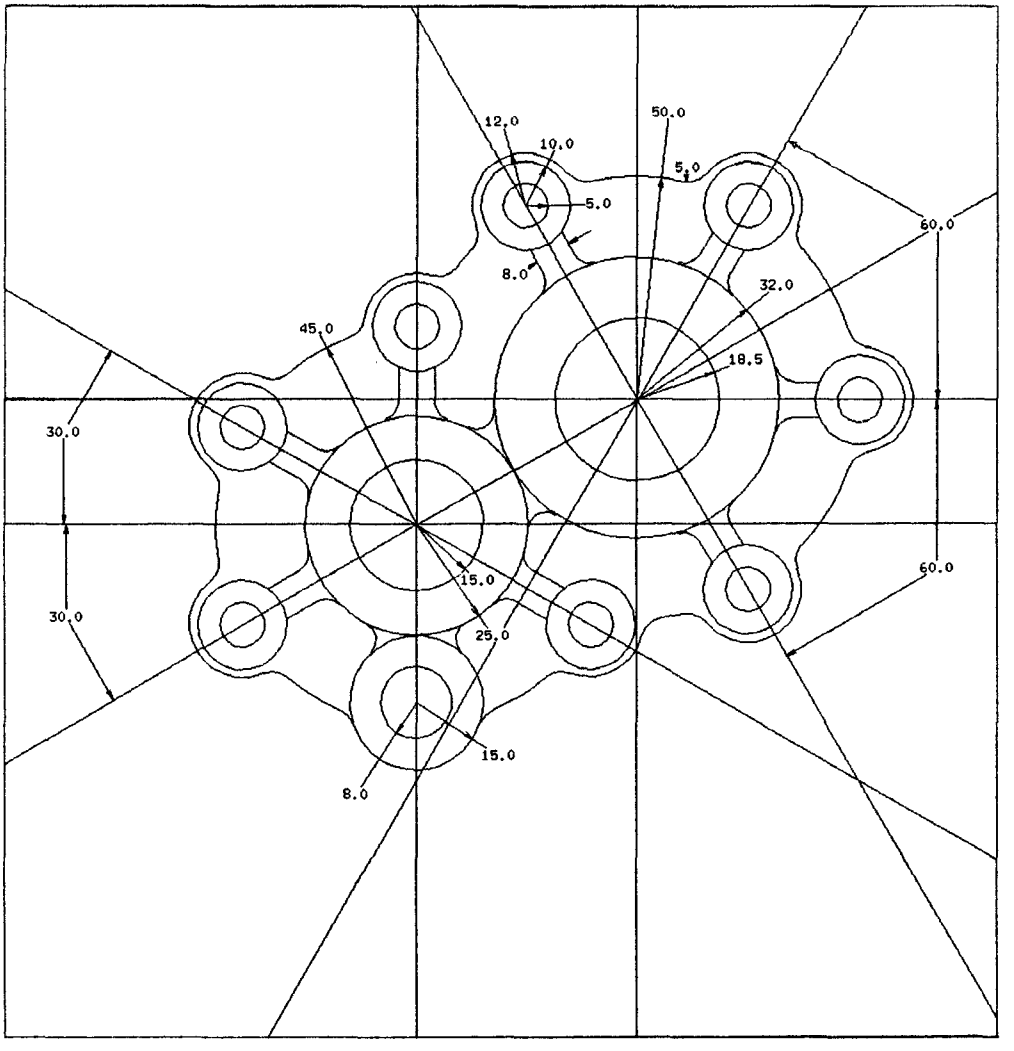

Figure 3 A dimensioned drawing where the geometries have been computed from the explicit dimensions and constraints. The geometry is represented by unbounded lines, complete circles and points. The points are not displayed here. The dimensional constraints consist of the explicit dimensions which are shown, explicit dimensions which are not shown but implied by symmetry (for example similar features have the same dimensions) and implicit constaints such as the fact that end points lie on curves and curves are tangent to one another. There are approximately 200 geometries and 400 dimensional constraint equations. These are fully solved in less than one second on a SUN workstation.

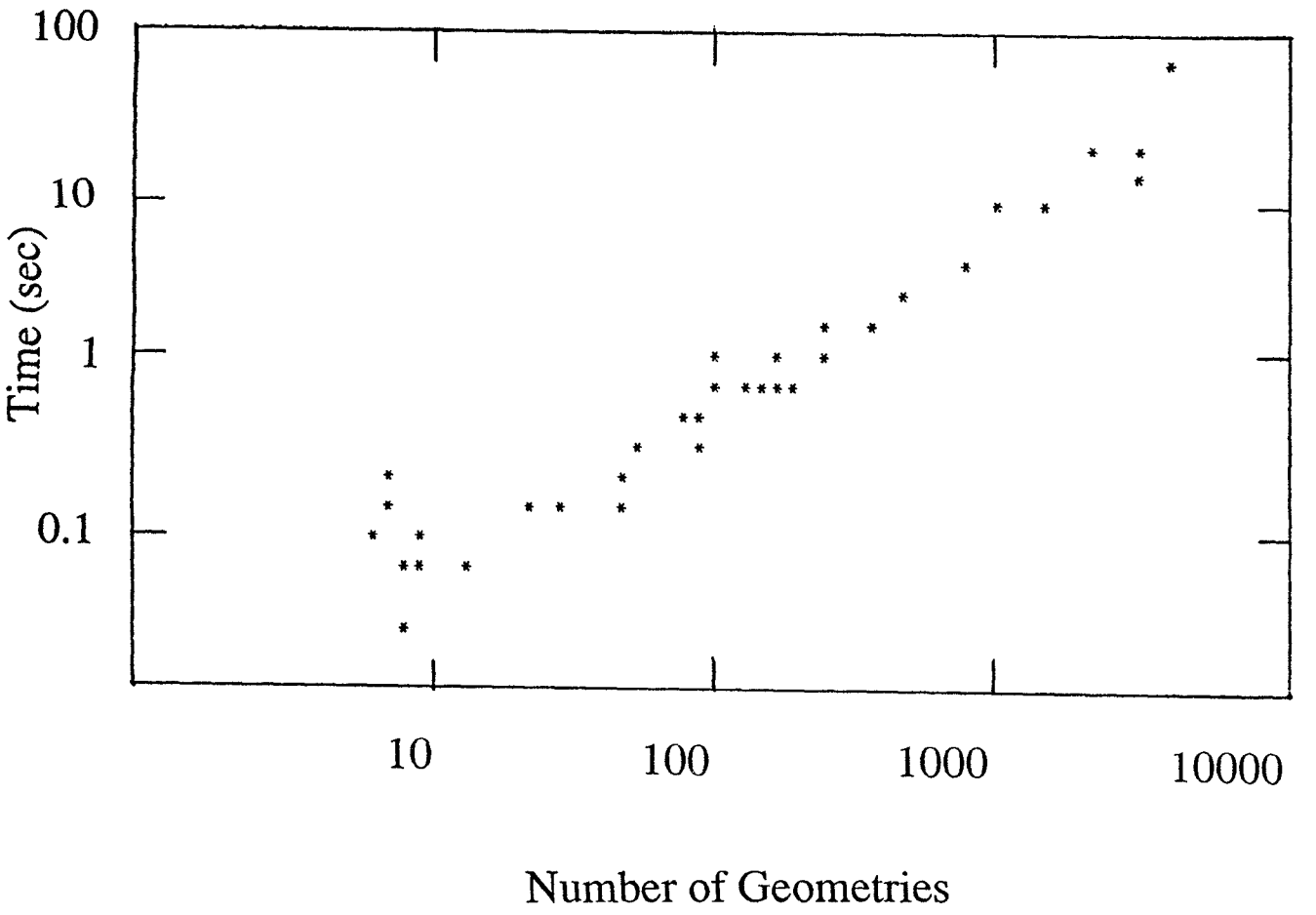

Figure 4 A comparison of execution time versus number of geometries for a variety of well constrained configurations taken from actual dimensioned drawings. The slope of unity on this log-log plot shows that execution time is approximately linear as a function of the number of geometries in the configuration.

5. IMPLEMENTATION

We have developed a software module which accepts as input points, lines, circles (with fixed or variable radius) and various dimensional constraints between them. The algorithm described above is used as the basis for computing the geometries to satisfy the dimensions. Following a computation the module returns information on whether individual geometries are underconstrained, well constrained or overconstrained and whether real solutions could be found. The particular solution which is chosen to the constraint equations is determined by conserving logical relationships between the geometries such as whether three dimensioned points occur in a clockwise or counterclockwise order. By changing these relationships the user may select different discrete solutions.

Well Constrained Configurations

The algorithm is robust and numerically stable. On the SUN sparcstation configurations with several hundred geometries can be computed in less than a second.

Figure 3 shows a fully dimensioned drawing for which all of the geometry has been computed by the algorithm from given dimension values.

Figure 4 shows a log-log plot of the execution time as a function of the number of geometries for a selection of realistic dimensioned drawings. We note that the relationship between configuration size and execution time is approximately linear over three orders of magnitude.

Underconstrained Configurations

For underconstrained configurations the module will attempt to find an acceptable solution by changing as few geometries as possible. Unfortunately it cannot be guaranteed that a solution will be found even though it exists. It is ironic that when there are a finite number of solutions they can all be found, but when there are an infinite number we cannot guarantee to find any.

Overconstrained Configurations

The module will identify geometries in an overconstrained subgraph and solve for the rest if required.

In our experience the main difficulty in using the module for interactive design is related to constraint configurations which are formally overconstrained but which are none the less consistent and solvable. In many circumstances a designer does not distinguish between an overconstrained but solvable configuration and a well constrained one.

We have found it very useful to preprocess the constraint graph and to remove obviously redundant constraints prior to the evaluation stages.

6. CONCLUSION

We have described and implemented an algorithm which will solve for co-planar points, lines and circles so that they satisfy given relative dimensions. The algorithm is algebraic and involves solving only quadratic equations. We have also demonstrated that this algorithm is complete in the sense that any configuration which could in principle be solved using only quadratic operations will be solved by the algorithm.

With the exception of some freeform curves almost all geometry generated by commercial CAD drawing systems is created using algorithms which ultimately solve only quadratic equations. The dimensions are then read off from the geometry. For any design which can be created in these systems the algorithm which we have presented can be used to invert the process and allow the designer to first specify the dimensions and then compute the geometry.

Our main conclusion is that for this type of geometry in two dimensions the inversion process can be done without resorting to numerical techniques. It can also be done in practice in a time which is very nearly linearly proportional to the size of the configuration.

A similar conclusion holds in many cases for three dimensional geometry and we are currently implementing a related algorithm for these cases.

References

Aho, V.A. Hopcroft, J.E. and Ullman, J.D., The design and analysis of computer algorithms, Addison Wesley (1974).

Freeman-Benson, B.N., Maloney, J. and Borning, A., An incremental Constraint Solver, Communications of ACM 33 54 (1990).

Herstein, I.N., Topics in algebra, John Wiley and Sons (1975).

Hillyard, R.C. and Braid, I.C., Analysis of dimensions and tolerances in computer aided mechanical design, 10 161 (1978).

Hopcroft and Tarjan, Dividing a graph into triconnected components, Siam Journal of Computing 3 135 (1973).

Kendig, K., Elementary Algebraic Geometry, Springer-Verlag (1977).

Leler, W., Constraint Programming Languages, Addison Wesley (1988).

Light, R. and Gossard, D, Modification of geometric models through variational geometry, 14 209 (1982).

Owen, J.C. and Ripka, G., The ground state of a strongly interacting fermion system, La Revista del Nuovo Cimento 6 1 (1983).

Pedoe, D. Geometry, Dover (1970).

Serrano, D. and Gossard, D.C., Combining Mathematical Models with Geometric Models in CAE systems, ASME, Proceedings of 1986 International Computers in Engineering conference, Chicago, Ill (1986).

Stuart, I. Galois Theory, Chapman and Hall (1973).

Sutherland, I.E., Sketchpad: a man-machine graphical communication system, Phd thesis MIT (1963).

Todd, P., A k-tree generalisation that characterizes consistency of dimensioned engineering drawings, SIAM Journal of Disc Math 2 255 (1989).

Van Leeuwen, J., Groenveld, J. and de Boer, J., New method for the calculation of the pair correlation function, Physica 2 792 (1959).