Solving Geometric Constraint Systems

I. A Global Propagation Approach[1]

Xiao-Shan Gao
Institute of Systems Science
Academia Sinica
Beijing 100080, P.R. China
e-mail: xgao@mmrc.iss.ac.cn

Shang-Ching Chou
Department of Computer Science
The Wichita State University
Wichita, KS 67208, USA
e-mail: chou@cs.twsu.edu

Abstract. We describe a geometric constraint solving method/system that takes the declarative description of geometric diagrams or engineering drawings as input and outputs a sequence of steps to construct the diagram with ruler and compass if succeeds. We extend the local propagation to a global one. Like the local propagation, the global propagation tries to determine the position of a geometric object from the set of geometric objects whose positions are already known. However, our global propagation uses not only the constraints involving this object but also *implicit information* derived from other constraints. The algorithm can be used to build intelligent CAD and interactive computer graphic systems.

Keywords. Geometric constraint system solving, global propagation, intelligent CAD.

1 Introduction

1.1. Motivation. Solving geometric constraints is the central topic in much of the current work of developing intelligent CAD systems [1, 3, 12, 14, 17, 19, 22] and interactive constraint-based graphic systems [2, 11, 20]. The main advantage of using the constraint approach is that the resulting systems accept declarative descriptions of diagrams or engineering drawings, while for conventional systems the users need to specify how to draw the diagrams.

Based on our previous work on automated geometry theorem proving, this paper describes a geometric constraint solving system/method that takes the declarative description of a geometric diagram as input and finds a procedure of how to draw the diagram using ruler and compass. A diagram that can be drawn with ruler and compass is called *an rc-configuration.* Rc-configurations, used in most commercial CAD softwares, are easy to compute and cover most of the practical design problems.

The main difficulty of finding a construction procedure for a declaratively described diagram is the existence of loops in the description, i.e., more than one geometric objects are

[1]This work was supported in part by the NSF Grant CCR-9420857 and the Chinese National Science Foundation.

constrainted or referred simultaneously. We use a *global propagation* to solve constraint problems with loops. Like the *local propagation method* [18, 15], it tries to determine the position of a geometric object from the set of geometric objects already drawn. But our propagation uses not only the constraints involving this object but also *implicit information* derived from other constraints. The global information needed in the propagation comes from a *geometry information base (GIB)* built before the construction begins. The GIB for a configuration is a database containing all the properties of the configuration that can be deduced using a fixed set of geometric axioms [7].

Other distinctive features of our method are as follows. (1) Besides the *dimensional constraints* specifying a distance or an angle, we use many *implicit constraints*, such as the congruence of line segments or angles and ratio of line segments, to specify relations among geometric objects. Using these implicit constraints, we can describe more constraint problems than using dimensional constraints alone. (2) We use methods of *automated geometry theorem proving* from [4, 6, 7] to build the GIB, to solve the over-constraint problem, and to give proper conditions under which auxiliary points can be added.

Our program based on the global propagation method is implemented in Prolog. The program is used to solve 130 constraint problems with different type of loops. Statistics on the running time for the 130 constraint problems can be found in Section 5.

1.2. Related Work. Three main approaches have been developed to solve geometric constraints: the construction approach, the numerical approach, and the symbolic approach.

In the construction approach, a pre-treatment is carried out to transform the constraint problem into a new form that is easy to draw. A majority of the work is to transform the constraint problem to an rc-configuration [12, 14, 17, 19, 21]. Once a constraint diagram is transformed into constructive form, all its solutions can be computed efficiently. The *graphic-constructive method* is used to give complete methods of finding construction procedures for constraint problems without loops [21]. Effective methods of "breaking" loops are presented by Hoffmann, Kramer, Owen, and Verroust [12, 14, 17, 23]. The basic idea is to divide the configuration into components such that each component itself has no loops and all the components can be put together in some way. This approach is powerful enough to solve many difficult constraint problems. Another approach is to use techniques from AI such as searching and rule-based systems to solve constraint problems [1, 3, 19, 22].

In the numerical approach, geometric constraints are translated into algebraic equations and various numerical techniques are used to solve these equations. The first generation constraint-based systems, such as Sketchpad [20] and ThingLab [2], used numerical relaxation as the last resort of solving constraint problems. Newton iteration methods were used in Juno-2 [11] and in the Variational Geometry [16]. The main advantage of numerical methods is their generality. On the other hand, it also needs further improvements in several aspects:

like finding stable solutions during animation.

In the symbolic approach [13, 24, 5], we also transform the geometric constraints into algebraic equations. Instead of using numerical methods to solve the algebraic equations, in the symbolic approach, we first use general symbolic methods such as Wu-Ritt's characteristic method or the Groebner basis method to change the equation set to new forms which are easy to solve, and then solve the new equations numerically. Kramer's approach can also satisfy the geometric constraints in a symbolic way [14]. In this approach, there are no set of explicit equations created. Other related approaches can be found in [25].

The rest of the paper is organized as follows. Section 2 presents the problem formulation. Section 3 presents the method. Section 4 presents more details of the method. Section 5 is a summary of the paper.

2 Problem Formulation

2.1. Geometric Constraint Systems. Some of the main constraints used by us are in Table 1. Besides the *dimensional constraints* specifying a distance or an angle, we use many *implicit constraints*, such as CONG, EQANGLE, MID, and SRATIO (see Table 1), specifying relations among geometric objects. Using these implicit constraints, we can describe more constraint problems than using dimensional constraints alone. The problems in Figures 1 and 4 are such examples. Without using algebraic equations, we cannot describe this diagram with dimensional constraints.

constraints	geometric meaning
DIS(A, B, d)	$\|AB\| = d$
ANGLE(A, B, C, a)	$\angle(ABC) = a$
PLDIS(P, A, B, d)	distance from P to line AB is d
CONG(A, B, C, D)	$\|AB\| = \|CD\|$
EQANGLE(A, B, C, P, Q, R)	$\angle(ABC) = \angle(PQR)$
COLL(A, B, C)	A, B, C are collinear
PARA(A, B, P, Q)	AB is parallel to PQ
PERP(A, B, P, Q)	AB is perpendicular to PQ
MID(M, A, B)	M is the midpoint of AB
SRATIO(A, B, P, Q, r_1, r_2)	$\|AB\|/\|PQ\| = r_1/r_2$

Table 1. The Geometric Constraints

Our method uses points as the only basic geometric objects. Lines and circles are defined as functions of their characteristic points. The main advantage of this approach is that the number of constraints can be minimized. Fewer constraints usually lead to an easy solution.

A constraint problem is defined as follows

$$[[Q_1, \cdots, Q_m], [P_1, \cdots, P_n], [C_1, \cdots, C_m]]$$

where Q_i are the points whose construction order, $Q_1, \cdots, Q_m$, is given by the user; P_i are the points whose construction order is going to be determined by the program; and C_i are the constraints.

An Example. Consider the constraint problem: *"In a solid object, there is a hollow triangle tunnel. We want to put a prism with a square cross section into the tunnel in a position as shown in Figure 1a."* Note that all the collinear constraints in Figure 1 are not explicitly given. We only need to consider the normal cross section. Then the problem is reduced to a plane constraint problem: to put a square into a triangle ABC. In this example, points A, B, C, are known. So we can described the problem as follows: $[[A, B, C], [P, Q, S, R]$, [the constraints]].

The constraints:
|AB|=d1, |BC|=d2, |CA|=d3
PQ // SR, SP // QR, |RS| = |RQ|
RQ is perpendicular to BC

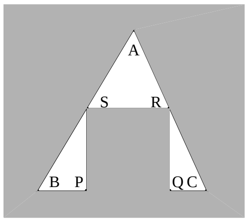


Figure 1a

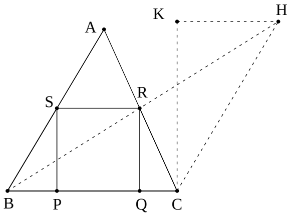


Figure 1b

2.2. Construction by Ruler and Compass. A geometric configuration that can be constructed using a ruler and a compass is called an *rc-configurations.* Precisely, a diagram is an *rc-configuration* if the points in the diagram can be listed in an order $(P_1, \cdots, P_k)$ such that each point P_i can be constructed using the following three basic *constructions*: (1) construction (POINT, P) takes a free point P in the plane; (2) construction (ON, P, o) takes a free point P on a line or circle o; (3) construction (INTERSECTION, P, o_1, o_2) takes the intersection P of a line or a circle o_1 and another line or circle o_2. So for an rc-configuration, associated with each point construction order $(P_1, \cdots, P_k)$ there is a *construction sequence* $(C_1, \cdots, C_k)$ such that point P_i is determined by construction C_i.

Lines and circles are represented as functions of their characteristic points. For instance, the following are some of the lines used in the method: (LINE, P, Q) is the line passing through points P and Q; (PLINE, R, P, Q) is the line passing through R and parallel to line PQ; (TLINE, R, P, Q) is the line passing through R and perpendicular to line PQ; (BLINE, P, Q) is the perpendicular bisector of segment PQ; (DLINE, P, Q, d) is the locus of the points having distance d to PQ, i.e. it represents two lines. We also use the following forms of circles. (PCIR, O, P) is the circle with center O and passing through P; (RCIR, O, r) is the circle with center O and radius r; (ACIR, P, Q α) is the circle (or an arc) consisting of the points R satisfying $\angle PRQ = \alpha$; (CCIR,P,Q,R) is the circle passing through three points P,Q, and R; (DCIR,P,Q) is the circle with PQ as a diameter.

For convenience, we use another construction: (MRATIO, P, Q, R, r_1, r_2) which constructs a point P on line QR such that the ratio of the directed segments $\frac{\overline{QP}}{\overline{PR}}$ is r_1/r_2, where

r_1 and r_2 could be numbers or algebraic expressions in geometric quantities. It is known that the construction MRATIO can be reduced to the three basic constructions.

3 The Method

The method takes a constraint problem as input and finds a construction sequence for it if succeeds. The main steps of the global propagation method are given below.

1. For a constraint problem

$$[[Q_1, \cdots, Q_m], [P_1, \cdots, P_n], [C_1, \cdots, C_m]],$$

let $CT = \{C_1, \cdots, C_m\}$ be the constraint set, $CS = \emptyset$ the construction sequence, $QS = \{Q_1, \cdots, Q_m\}$ the points with given construction order, and $PS = \{P_1, \cdots, P_n\}$. We assume that the problem is not over-constraint, i.e., we have $|CT| \leq 2 * |PS| - 3$. For our treatment of over-constraint problems, see Section 4.2.

2. Build the GIB as described in Section 4.5. Then repeat the following steps first for QS and then for PS until both QS and PS become empty.

3. Take a point P from QS or PS. For each constraint $T \in CT$ involving P, decide the locus Lc of the points satisfying T, assuming all points constructed in CS are known. We then obtain a set of triples

$$\{(P, T_1, Lc_1), \cdots, (P, T_s, Lc_s)\}.$$

We consider three cases. (1) $s = 0$. Point P might be an arbitrarily chosen (free) point. We add a new construction C = (POINT, P) to CS. (2) There exist $i \neq j$ such that $T_i \neq T_j$ and Lc_i and Lc_j are not parallel lines or concentric circles.[2] We add a new construction C = (INTERSECTION, P, Lc_i, Lc_j) to CS and remove T_i and T_j from CT. (3) Otherwise, point P is a semi-free point: it can move freely on a line or a circle. We add a new construction C= (ON, P, Lc_1) to CS and remove T_1 from CT.

4. Now we check whether the remaining problem is over constraint, that is, whether $|CT| > 2 * |PS|$ (note this inequality is different from the one in Step 1). (1) If it is, the construction sequence is invalid. If P is from QS, the order given by the user can not be constructed and the method terminates. Otherwise, restore the removed constraints and repeat the preceding step for a new point from PS. (2) If it is not over-constraint, point P is constructed. We need to repeat the preceding step for a new point.

[2]We can check these conditions using automated theorem proving methods such as Wu's method[4, 24] and the area method[6] which are quite fast.

The crucial step of the above method is how to determine the locus Lc from a constraint T in Step 3. In what follows, we will discuss this in detail. In the following, we will always use $A, B, C, \ldots$ to represent known or constructed points, and $U, V, W, \ldots$ to represent any points known or not. P is the current point to be constructed and T is the constraint.

3.1. Determine a line. Our method allows partial properties of geometric objects to be propagated. We consider three partial properties for a line L: (L_1) L has a fixed direction; (L_2) L passes through a known point; (L_3) L tangents to a known circle. We thus have six possible forms to determine a line: $(L_i, L_j), i \leq j$.

For case L_1, we first have some *basic situations* which can determine the direction of line L. The most frequently used are (1) L passes through two known points; (2) L cuts a fixed length between a pair of parallel lines; (3) L cuts a pair of segments on the sides of a known angle such that the ratio of the segments is known; (4) L cuts a pair of segments between three known lines passing through a point such that the ratio of the segments is known. Second, any line with known direction will be propagated to a line with known direction by the following *propagations*: (1) parallel, (2) perpendicular, and (3) forming a known angle. So the direction of a line is usually determined by a sequence of lines U_1V_1 τ_1 U_2V_2 τ_2 ... τ_{s-1} U_sV_s, where U_i, V_i are points and τ_i are transformations. If the direction of U_sV_s is determined by one of the basic situations, the direction of line U_1V_1 is known.

The above process clearly demonstrates the global feature of the propagation. It also demonstrates the advantage of using partial properties of geometry objects: even if the geometric object is unknown, some of its partial properties can be propagated.

Let us consider the example from [12] (Figure 2). After constructing B, C, X, we need to construct D. Since $|XD|$ is known, we need only to know the direction of XD which can be determined by global propagation: $XD \perp DE$, $\angle(DEA) = a_2$, $AE \perp AB$, and $\angle(ABC) = a_1$.

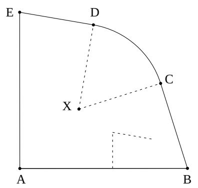


Figure 2

The constraints:
DIS(B,C,d1), DIS(E,D,d2),
DIS(X,C,d3), DIS(X,D,d3)
ANGLE(A,B,C,a1), ANGLE(D,E,A,a2)
PERP(X,C,C,B), PERP(X,D,D,E), PERP(B,A,A,E)

Three cases lead to L_2. (1) There is a known point A on L. (2) If the ratio of the distances of two known points A and B to L is r_1/r_2 then L passes through a fixed point Q which can be introduced by the construction (MRATIO, Q, A, B, r_1, r_2). (3) If a circle is tangent to two known circles, the line connecting the two tangent points passes through the similarity center Q of the two known circles.

Four most frequently used cases for L_3 are given below. (1) Line L is tangent to a known circle. (2) Line L cuts a segment with fixed length in a known circle. In this case, L is tangent to a concentric circle of the known circle, which passes the midpoint of any chord of the known circle with the known length. (3) The chord cut by L on a known circle faces a known inscribed angle on the circle. This case can be reduced to case (2). (4) L cuts a segment with fixed length between two known concentric circles.

3.2. Determine a circle. The most frequently used partial properties of a circle are given below. (C_1) The center is known. (C_2) The center is on a line. (C_3) The center is on a circle. (C_4) The radius of the circle is known. (C_5) The circle passes through a known point. (C_6) The circle is tangent to a known line. (C_7) The circle is tangent to a known circle. To determine a circle, we need three constraints. The most interesting cases are the so-called *Apollonius's drawing problems*, namely, determining the circle if three constraints $(C_i, C_j, C_k), 5 \leq i \leq j \leq k \leq 7$ are given. In [12], this kind of problem is solved using algebraic techniques such as the Gröbner basis method. Our program can provide geometric construction for all of the ten Appolonius problems. Besides the two trivial cases (C5,C5,C5) and (C6,C6,C6), five of them, (C5,C5,C6), (C5,C5,C7), (C5,C6,C6), (C5,C7,C7), and (C7,C7,C7), are included in our collection (problems 79, 103, 100, 105, and 106 in [8]).

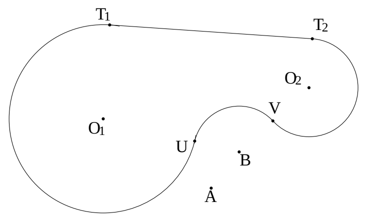


Figure 3a

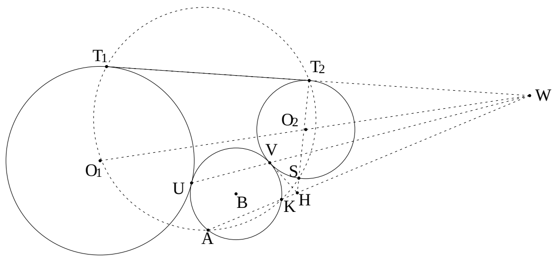


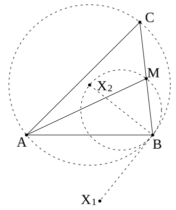
Figure 3b

Figure 3a shows an engineering part which needs the case (C_5, C_7, C_7). Suppose that O_1 and O_2 are two known circles and A is a known point. We need to construct a circle B which tangents to circles O_1 and O_2 and passes through point A. Figure 3b shows the solution given by our method. By case (3) of L_2, line UV passes through the similarity center of circles O_1 and O_2, i.e., the intersection W of line O_1O_2 and a common tangent line T_1T_2 of the two circles (which is easy to construct). Let S be the intersection of circles (CCIR, T_1,T_2,A) and(PCIR,O_2,T_2) and H the intersection of lines AW and T_2S. Then O_2V is perpendicular to HV. We hence can construct point V. Now U can be constructed easily.

The process of determining a circle is similar to that of determining a line. We will omit the details due to limit on the length of the paper.

3.3. LC transformations. An *LC transformation* is a one-to-one map from the Euclidean plane to itself which transforms a line to a line and a circle to a circle. We mainly use two kinds of LC transformations.

The first kind is related to the constraint SRATIO(P, Q, Q, R, r_1, r_2). If points P, Q, and R are collinear and one of them, say Q, is already known, then point P is on a line or a circle if and only if point R is on a line or circle. The method works as follows. Let P be the point to be constructed, Q a known point. Then we will call the main method to see whether point R is on a line or circle. If it is, we will compute the corresponding line or circle that point P is on.

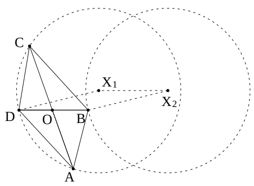


The constraints:
DIS(A,B,d1), DIS(A,M,d2)
MID(M,B,C), ANGLE(A,C,B,a1)

Figure 4

Consider the constraint problem in Figure 4. Suppose we have constructed points A and B. Since $\angle(ACB) = a_1$, point C is on circle Cr =(ACIR, A, B, α_1). We construct the center X_2 of circle Cr[3]. Since M is the midpoint of BC, point M is on another circle (DCIR,X_2,B). Hence M is the intersection of (RCIR,A,d_2) and (DCIR,X_2,B).

A problem in the construction of point C is: which constraint will be removed from the constraint set. Constraint $\angle ACB = a_1$ must be removed. If we remove constraint (COLL, M, B, C) or $MB = MC$, the further construction of point M will lead to inconsistency. The right choice is to remove constraint $|AM| = d_2$. There is actually *a general rule of removing constraints*: if an LC transformation is used to construct a line or a circle, then the constraint leading to the original line or circle (before transformations) will be removed. Related discussions can be found in Section 4.1.



The constraints:
DIS(A,C,d1), DIS(B,D,d2)
ANGLE(A,B,C,a1), ANGLE(A,D,C,a2)
ANGLE(B,O,C,a3)

Figure 5

The second kind of LC transformations is as follows: suppose that segment AB has a fixed direction and a fixed length then point A is on a line or a circle if and only of point B is on a line or a circle. An example is given in Figure 5. Suppose that points A and C have been constructed. Next, we construct point B. Since $\angle ABC = a_1$, B is on circle C_1 =(ACIR, A, C, a_1). Similarly, point D is on circle C_2 = (ACIR, A, C, a_2). Since BD has a fixed direction and a fixed length, point B must be on another circle C_3, and B is the intersection of circles

[3]The center Q of Cr can be constructed as follows: Take a point X_1 such that $\angle(ABX_1) = \alpha_1$. Since BX_1 is a tangent line of circle Cr, X_2 is the intersection (BLINE, A, B) and (TLINE,B,B,X_1).

C_2 and C_3.

3.4. More construction methods. We just mentioned three major techniques of obtaining a locus from a constraint. In our method, more methods of constructing loci are used. Some of these methods are targeted to certain problems such as those involving the centroid of a triangle, angle bisectors, special properties of squares, etc. Others are quite general. One such case is related to constraint SRATIO. Suppose we wish to construct point P which satisfies constraint $PP_1/PP_2 = r_1/r_2$. One of the eight solutions to this constraint used in our program are given below. If P_1 and P_2 are on two known lines L_1 and L_2 meeting at point A and PP_1 and PP_2 have fixed directions then the locus of P is the line passing through point A and the intersection point of two lines having fixed distances to lines L_1 and L_2, respectively. If $L_1 \parallel L_2$, the locus is also a line.

Solving the constraint problem in Figure 1 needs this rule. Since $RQ/RS = 1$,$RQ \perp BC$, and $RS \parallel BC$, R is on a line BH, where H is the intersection of line (PLINE, C, A, B) and the line (DLINE,B,C,$|BC|$).

Another general technique is the *base triangle method.* The idea is that when the shape of a triangle ABC can be determined while its position is unknown, we can draw a triangle $A'B'C'$ which is congruent to triangle ABC and then transform $A'B'C'$ to the right position. The base triangle method can be extended to the *base polygon method.* One of the special case is very useful. Suppose that points $P_1, P_2, \cdots, P_n$ satisfy the following conditions: $DIS(P_iP_{i+1}) = d_i$, $i = 1, \cdots, n-1$, $ANGLE(P_j, P_{j+1}, P_{j+2}) = a_j$ $j = 1, \cdots, n-2$. Then $P_1P_2 \cdots P_n$ is a known polygon. In particular, the length of P_1P_n is known and if point P_1 is known then point P_n is on a circle.

4 Points of Friction

4.1. Constraint transformation. The basic idea behind the method of breaking "loops" of constraints given in Section 3 is to transform a constraint to a different but equivalent one. A problem related to the constraint transformation is that in some cases a locally equivalent transformation may lead to inconsistency in other constructions. For example, in solving the constraint problem in Figure 4, we must consider $|AM| = d_2$ the constraint to be transformed. Otherwise, inconsistency will occur in the construction of the next point M.

In some cases, such as in using the LC transformations, this problem can be solved by proper selection of constraints. In more complicated cases, we use the following methods. (1) Changing constraints. In some cases, to preserve the consistency, we need not only to change the involving constraint but also to change other constraints. (2) Use new constructions. We have noticed that the inconsistency is caused by the existence of multiple solutions rising from taking the intersection of a circle with a line or a circle. To discard the wrong solution, we just need to collect all the constraints about the constructing point and check which of the intersections satisfy them. We need to introduce the following new construction,

(INTERSECTION, P, l, c, e)

where e is a set of constraints to be satisfied. The construction means that P is the intersection of line l and circle c that satisfies the constraints e. Note that these logic expressions are used during the drawing of the diagram; hence, all related variables in the logic expression are numerical ones. The difficulty here is that if a number is very small, we do not know whether it is zero. Fortunately, most design problems do not contain such cases. Actually such cases occur only when we change the diagrams dynamically, and can be taken care of by examining whether the denominator of an expression vanishes when evaluating it.

Consider the constraint problem in Figure 5. After constructing points A, C, B and O, two constraints are left for point D: COLL(D,B,O) and DIS(D,B,d_2). Then point D is the intersection of a line and a circle. But only one of the two intersections is correct and can be constructed as follows: (INTERSECTION, D, (LINE, B, O), (RCIR, B, d_2), (ANGLE, A, D, C, a_2)).

4.2. Over constraint. Following Hoffmann [12], a constraint diagram is called *well defined* if it has n characteristic points and $2 * n - 3$ independent constraints. The shape of a well-defined diagram is determined up to finite possibilities, while its position in the plane is totally free. To decide whether some concrete constraints are independent, generally speaking, we need general purpose algebraic techniques such as Wu-Ritt's characteristic method [24, 5].

To solve over-constraint problems, we first remove the proper number of constraints and let the remaining problem become a well defined one, and then find a construction sequence for the new constraint problem. Suppose that the new constraint diagram is an rc-configuration RC and that a construction sequence is founded. Then there are two possibilities. (1) For an rc-configuration there exist very efficient methods of proving assertions about the configuration such as the coordinate-based method in [4] and the area method in [6]. Using these methods, we can check whether the removed constraints are valid on the configuration RC. We need simply remove those constraints which are valid on RC. (2) For those constraints not valid on RC, we can solve the problem in the following special case. Suppose that the extra constraint is a *directional constraint*, that is , a constraint like parallel or perpendicular. Then we can move RC to satisfy this constraint without changing the shape of RC. In other words, we need only to move RC to a new position. Consider the following example: "Draw a circumscribed triangle ABC in a given circle O, if $|AB|, |AC|$ are known, and BC is parallel to a given line XY."

After removing the constraint $BC \parallel XY$, a triangle $A'B'C'$ can be drawn easily. Let M be the midpoint of $B'C'$. By (2) of case L_3 in Section 3.1, line BC is a tangent line of circle (PCIR, O, M). Since $BC \parallel XY$, we can draw ABC without difficulty.

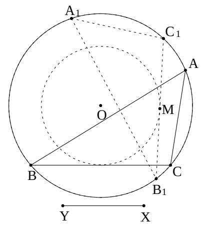


Figure 6

The constraints:
Points X and Y are known
DIS(O,A,d1), DIS(O,B,d1), DIS(O,C,d1)
DIS(A,B,d2), DIS(A,C,d3), PARA(B,C,X,Y)

4.3. Multiple Construction Sequences. The search mechanism used by us may lead to different construction sequences for the same constraint problem. Different construction sequences might occur because for the same construction order of points, certain points can be constructed in more than one way. But the majority of the different construction sequences are caused by different construction orders of points. Though these different construction sequences give the same picture, they might be quite different in computational complexity and have different geometric meanings. This allows us to select the best one among a set of construction sequences. For instance, we can find the construction sequence using the least number of auxiliary points. In other words, the shortest construction sequence, which can be selected using the breadth-first search or the depth-first iterative deepening search [8].

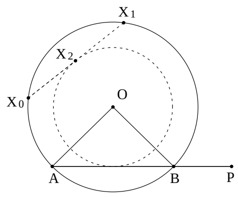


Figure 7

The constraints:
DIS(O,A,d1), DIS(O,B,d1)
DIS(O,P,d2), DIS(A,B,d3)

Consider Figure 7. If the construction order is O, A, B, P, the construction sequence is easy to find. If the construction order is O, P, A, B, the construction sequence can be given using case (2) of partial property L_3 in Section 3.1. Specifically, we need first to draw a chord X_0X_1 of length d_3 of circle (RCIR,O,d_1). Let X_2 be the midpoint of the chord. Then line AB is coincident with the tangent line drawn from point P to circle (PCIR,O,X_2). The second construction sequence is more complicated, but it has a different geometric meaning. Geometrically it means "in a given circle, draw a chord that has a fixed length and passes through a known point."

To construct a diagram in different point order is also useful in the following situation. Suppose that the constraint problem in Figure 7 is a part of a larger constraint problem and in the solution of the larger problem points O and P are already constructed and we need to construct A and B. Then we have to use the second construction sequence. This trick was also used in the example of Section 2.1.

4.4. Multiple Solutions. For a construction sequence, different diagrams can be constructed due to the fact that some constructions such as the intersection of two circles give more than one point. A major advantage of using rc-configuration is that we can skip from one solution to another in the same construction sequence without difficulty.

For a well defined constraint problem, we can use the method provided in [9] to show that by introducing some heuristics of selecting roots, the method can construct a unique diagram which is "similar" to a given sketch. Furthermore, the diagrams constructed by using all possible construction sequences are congruent. To prove this fact, first note that that rc-construction for a well defined diagram is a special case of the cluster formation in [9]. For a well defined diagram, we need only taking the intersection of lines and circles, and these constructions can be treated as cluster formation. Therefore, by using the heuristics introduced in [9], all the resulting diagrams will be congruent. Here, the construction of auxiliary points should also follows the heuristics. An interesting problem is to extend the result in [9] to the solutions of general triangular sets [10].

If the constraint problem is *under-constraint*, i.e. $|CT| < 2 * L - 3$ where L is the number of points, the shape of the diagram may change continuously resulting in infinite solutions. In our program, the user is allowed to drag the free (constructed by POINT) or semi-free (constructed by ON) points in the diagram to change the diagram. Under-constrainted problems are rare in real design problems. On the other hand, almost all problems from automated geometry theorem proving are under-constrainted.

4.5. Geometry Information Base (GIB). The geometric information needed in the global propagation comes from the GIB. The GIB for a configuration is actually a database containing all its properties that can be deduced from a fixed set of geometric axioms. To built a GIB for a geometric configuration, first the original constraints should be put into the GIB. Then the program will keep applying a fixed set of geometric rules (or axioms) to all the data in GIB to get new data and to put the new data into the GIB until nothing new can be obtained. The actual implementation is much more complicated [7].

The aim of the GIB here is to find collinear points and cyclic points as much as possible. The key concept is the *full-angles*, which are the directed angles between lines [7]. The basic rules about full-angles are (1) the generalization of the inscribed angle theorem: "full-angle (PA,PB) = full-angle (QA,QB) $\neq 0$ iff P, Q, A, B are on the same circle." (2) full-angle (OA,AB) = full-angle(AB,OB)$\neq 0$ iff OA = OB. (3) full-angle (AB,CD) = full-angle(AB,UV) + full-angle(UV,CD). (4) Triangles ABC and PQR are similar if full-angle(AB,BC) = full-angle(PQ,QR) and full-angle(AC,BC) = full-angle(PR,QR).

To enhance the speed of database construction, we presented the techniques of *data-based search* and *structured database* [7]. With these techniques, the building of GIB is very fast. It costs less than one second for 70% of the 130 difficult geometric problems tested in [7]. The time of building the GIB is tolerable, since we need only do that once for each problem.

4.6. Adding auxiliary points. In most cases of breaking loops, we need to *add auxiliary points*. Before adding an auxiliary point, the program first searches the GIB to see whether there is a point satisfying all the conditions of the auxiliary point. If it is, we need not introduce the new point. Otherwise, a new point is added to CS. After adding an auxiliary point, the GIB is enlarged to include the constraints for the new point.

Another problem is that some extra conditions are needed to guarantee the existence of the auxiliary points. For instance, if the new point is the intersection of two lines, the two lines cannot be logically parallel, which can be checked with the theorem proving method in [4]. A more interesting case is that the two lines are not parallel if the diagram is in "general positions," and in some specific cases they might become parallel. In automated theorem proving these kinds of conditions are critical for the logic correctness of the proofs. Since our main goal is to draw the diagram, this seems not a serious problem: diagrams satisfying extra conditions such as two "generally non-parallel" lines becoming parallel consist of a very small part of the set of all diagrams (actually probability zero). During numerical computation, theses conditions usually correspond to the vanishing of the denominators of some expressions and have been taken care of using some special measures.

5 Discussion

This paper demonstrates that for a large portion of rc-configurations described declaratively with loops, we can find construction sequences for them geometrically. We have searched from more than one thousand constraint problems and picked 130 each having different kinds of loops. Our program has been used to solve all of them efficiently. Checking over-constraints in Step 4 of the method is crucial to the performance of the method. Without this step, the method will search all the possible permutations of the points in the problem. Our experiments show that this step prohibits most of the unsuccessful search at a very early stage.

The method is implemented in Prolog to take advantage of its search engine. After the generation of a construction sequence, we draw the diagram using our C/Xwindow based system *Geometry Expert*[4] which is a geometry theorem prover and a dynamic diagram editor. In our current system, the generation of construction sequences and the drawing of the diagram are two separate phases, so it is not WYSIWYG (what you see is what you get). Our next goal is to integrate the two programs to implement a double viewing and WYSIWYG system.

Table 2 contains statistics on the running times needed for the Prolog based program to find the first solutions of for the 130 constraint problems on a Sparc-20. The average timing

[4]The program is available via anonymous ftp at emcity.cs.twsu.edu: pub/ geometry/ software/ gex_sparc.tar.Z.

is 1.84 seconds. To find the shortest solutions is much more time consuming.

Time	%	Time	%	Time	%	Time	%	Time	%
< 0.5	18%	< 1.0	47%	< 2.0	77%	< 5.0	94%	< 22.0	100%

A problem with all construction approaches including the one in this paper is that it seems impossible to give a complete set of construction rules because of the intrinsic difficulty of solving geometric constraint problems. A challenge problem is to find a set of construction rules which can be used to draw regular polygons with five or seventeen sides, which can be drawn with ruler and compass. An alternative way to achieve completeness is to use symbolic computation techniques. This is the topic of Part II of this paper.

References

[1] F. Arbab & B. Wang, Reasoning About Geometric Constraints, *Intelligent CAD II*, H. Yoshikawa and T. Holden (eds), North-Holland, 93–107, 1990.

[2] A. Borning, The Programming Language Aspect of ThingLab, *ACN Tras. on Programming Language and Systems*, Vol. 3, No.4, 353-387, 1981.

[3] B. Brudelin, Constructing Three-Dimensional Geometric Objects Defined by Constraints, in *Workshop on Interactive 3D Graphics*, p.111-129, ACM, 1986.

[4] S.C. Chou, *Mechanical Geometry Theorem Proving*, D.Reidel Publishing Company, Dordrecht, Netherlands, 1988.

[5] S. C. Chou & X. S. Gao, Mechanical Formula Derivation in Elementary Geometries, *Proc. ISSAC-90*, ACM, New York, 1990, p. 265–270.

[6] S. C. Chou, X. S. Gao, & J. Z. Zhang, *Machine Proofs in Geometry*, World Scientific, 1994.

[7] S.C. Chou, X.S. Gao, & J.Z. Zhang, A Fixpoint Approach To Automated Geometry Theorem Proving, WSUCS-95-2, CS Dept, Wichita State University, 1995.

[8] S.C. Chou, X.S. Gao, & J.Z. Zhang, A Collection of 130 Geometric Constraint Problems and Their Solutions, WSUCS-96-2, CS Dept, Wichita State University, 1996.

[9] I. Fudos and C. Hoffmann, Correctness Proof of a Geometric Constraint Solver, *Intl J of Comp Geometry and Applic*, vol. 6, Dec. 1996, p. 405–420.

[10] X. S. Gao & S. C. Chou Solving Geometric Constraint Problems, II. A Symbolic Approach and Decision of Rc-constructibility, WSUCS-96-3, CS Dept, Wichita State University, 1996.

[11] A. Heydon & G. Nelson, The Juno-2 Constraint-Based Drawing Editor, *SRC Research Report 131a*, 1994.

[12] C. Hoffmann, Geometric Constraint Solving in R^2 and R^3, in "Computing in Euclidean Geometry", eds D. Z. Du and F. Huang, p. 266–298, World Scientific, 1995. ****

[13] K. Kondo, Algebraic method for manipulation of dimensional relationships in geometric models, *Geometric Aided Design*, 24(3), p. 141-147, 1992.

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

[15] W. Leler, *Constraint Programming Languages*, Addison Wesley, 1988.

[16] V.C. Lin, D.C. Gossard, & R.A. Light, Variational Geometry In Computer-Aided Design, *Computer Graphics*, vol. 15, no. 3, 171–177, 1981.

[17] J. Owen, Algebraic Solution for Geometry from Dimensional Constraints, in *ACM symp. Found. of Solid Modeling*, ACM press, p. 397-407, Austin TX, 1991.

[18] G. L. Steele & G. L. Sussman, CONSTRAINTS – A Language for Expressing Almost-Hierarchical Descriptions, *Artificial Intelligence*, p. 1-39, Vol. 14, 1980.

[19] G. Sunde, Specification of Shape by Dimensions and Other Geometric Constraints, in *Geometric Modeling for CAD Applications*, M. J. Wozny, et al eds, 199-213, North Holland, 1988.

[20] I. Sutherland, Sketchpad, A Man-Machine Graphical Communication System, in *Proc. of the Spring Joint Comp. Conference*, North-Holland, 329–345, 1963.

[21] P. Todd, A k-tree Generalization that Characterizes Consistency of Dimensioned Engineering Drawings, *SIAM J. of Disc. Math.*, Vol 2, 255-261, 1989.

[22] R. C. Veltkamp, Geometric Constraint Management with Quanta, *Intelligent Computer Aided Design*, D.C. Brown et al eds, 409–426, North-Holland, 1992.

[23] A. Verroust, F. Schonek, and D. Roller, Rule-oriented method for parameterized computer-aided design. *Geometric Aided Design*, 24(3), p.531-540, October, 1992.

[24] Wu Wen-tsün, *Basic Principles of Mechanical Theorem Proving in Geometries*, Springer-Verlag, 1993.

[25] J. Z. Zhang, L. Yang, & X. C. Yang, The Realization of Elementary Configurations in Eulciddean Space, *Science in China (series A)*, vol. 37, No.1, p.15-26, 1994.