Title:

Constraint Representation of 2 Dimensional Models with Respect to Avoiding Cycles

Author:
Manfred Rosendahl, ros@uni-koblenz.de, University Koblenz, Germany

Keywords:
Modeling Geometric Constraints in 2D, Graph Oriented Declarative Modeling

DOI: 10.14733/cadconfP.2016.17-22

Introduction:

A geometric constraint system which models 2-dimensional geometries in a form that often no multidimensional equation systems are necessary, when solving a given constraint system, is described. This is achieved because not only constraints between points are used, but also circles and lines are introduced as objects.

A geometric constraint system can be described by a bipartite graph. The nodes are divided into two disjoint sets V and C; V is representing the geometric objects and the scalar dimensional values; C is representing the constraints. The edges in the graph are linking then V and the C nodes. Mostly, therefore also in [1], the geometry is described by points as the geometrical objects. Here it is shown that by also using circles and lines as geometric objects, often a solution without cycles exists and therefore a sequential computation is possible. For an overview on modeling geometric constraints see [2].

Main Sections:

When using only points and scalar values, we could have the following constraints:

- Dp (P1, P2, d): Distance between points P1 and P2 is d
- A3(P1, P2, P3, alpha): Angle between the lines (P1,P2) and (P1,P3) is alpha
- RW(P1, P2, P3):The lines (P1,P2) and (P1,P3) are perpendicular
- Dl (P, Pa, Pe, d): the distance between the point P and the line from Pa to Pe is d.
- Vertical (P1, P2): P1 and P2 have the same x coordinate.
- Horizontal (P1,P2): P1 and P2 have the same y coordinate.
- Equ (<expression>): The value of the expression is 0. It must be possible to compute one variable if the others are known.

All these constraints consume at least one degree of freedom.

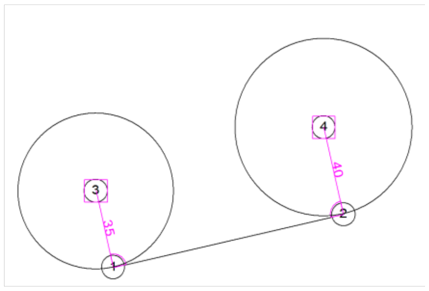
The points have 2 degrees of freedom; the scalar values have 1. But points and values can be fixed. If only one dimension of a point is fixed, it still has 1 degree of freedom. Otherwise, the point's degree of freedom is 0. The constraint system itself is represented by an undirected graph. To find a solution of the system, the graph must be directed and must fulfill the following conditions.

- Each node of V has no more incoming edges than the node's degree of freedom.
- Each node of C has as many outgoing edges as the constraint consumes degrees of freedom. All other edges of each node of C are incoming.

Each incoming edge of a point determines one locus for the position of the point.

For a point, we have the following loci: x coordinate, y coordinate, on line, on circle. For each combination of up to 2 loci, a procedure is defined to compute the point.

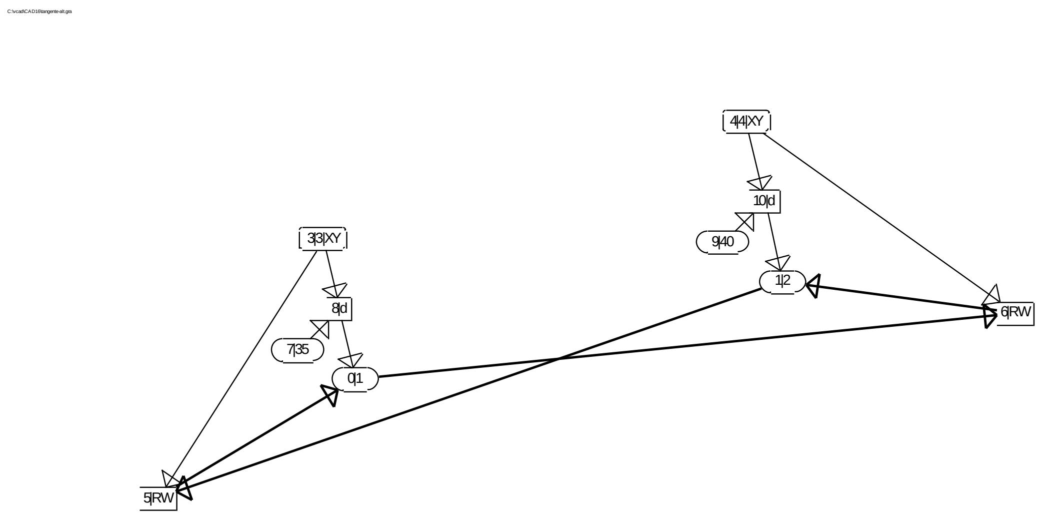
As an example, we consider the line tangent to two circles.



The tangent line must fulfill the following conditions.

- The distance between the center and the tangent point must be equal to the radius of the circle.
- The angle with the origin at one tangent point and the corners at the other tangent point and the center of the circle must be a right angle.

The corresponding constraint graph and the possible orientation (when the center and the radii of the circles are fixed) are shown below.



Note: Oval or rounded nodes represent variables (scalars, points, circles and lines). If points are rounded, they are fixed in the coordinates annotated (here X and Y). The rectangular nodes represent the constraints. The first number in the notation is the number of the node in the underlying implementation. The second symbol in the notation is either the type of the constraint node or the value or notation of the variable node. For totally or partially fixed points, the third symbol in the notation identifies what is fixed.

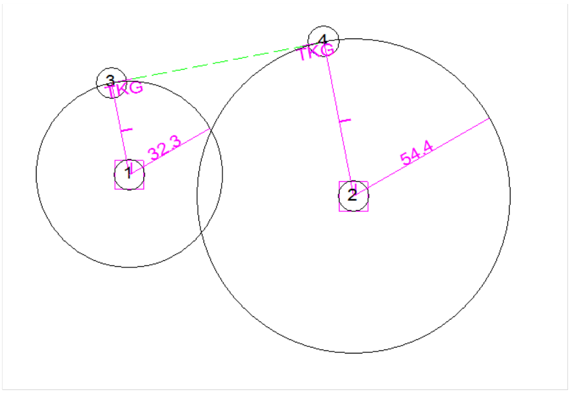
To compute one tangent point, the other tangent point must be known. Therefore, it is not possible to compute the solution sequentially. But if the two circles are known, it is possible to compute the tangent line. If we introduce both circles and lines as geometric objects with corresponding constraints, a sequential solution is possible.

The class circle is derived from the class point (its center) and has 3 degrees of freedom (the X, Y coordinates of the center and the radius). A line has 2 degrees of freedom. As parameters for the line, we take the length and the direction of the perpendicular line from the origin to the line.

As new constraints we introduce:

- 2 lines or circles are tangential

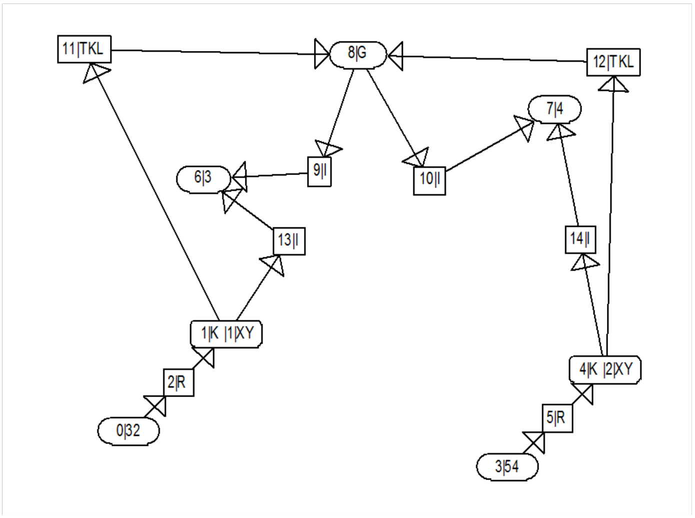
- Intersection point of lines or circles
- Point on line or circle
- 2 lines parallel or rectangular
- Distance point to line
- Direction of line
- Radius of a circle



The tangent line is defined by the following constraints:

1. Line tangential to circle 1
2. Point 3 on Circle 1
3. Point 4 in circle 2
4. point 3 on line
5. point 4 on line
6. Line tangential to circle 2

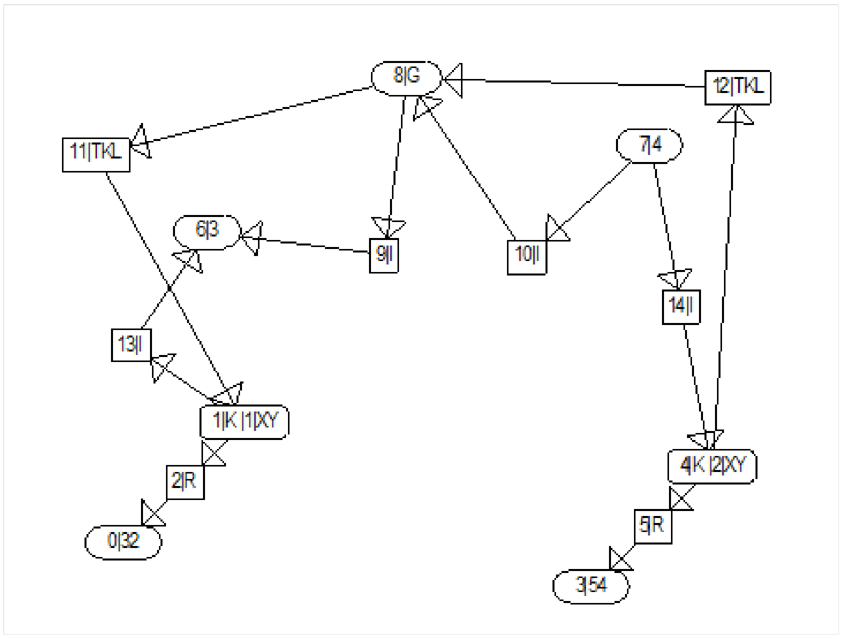
For the line and the 2 tangent points we have 6 degrees of freedom. These degrees are consumed by the 6 constraints. So the model is fully defined if the two circles are fixed. Starting with the fixed circles the following graph can be propagated.



For each type of node, variable and constraint, a computation method is defined. For constraints, the method sets the loci to the geometric objects which are reached by the outgoing edges. For geometric objects, the method computes the parameters of the object with respect to the given loci. If the constraint graph has no cycles, the computation can be done sequentially in topological order.
A geometric object can have at most as many incoming edges as its degree of freedom. But not all combinations are possible. E.g., a line cannot be determined twice by a constraint which fixes the line's direction. For a circle only 1 constraint can fix the radius, and mostly 2 constraints can fix the center

point. Points may have only one constraint which can be satisfied only by either the X coordinate or the Y coordinate. Each constraint has as its property the coordinates, which are influenced by the constraint.

Given two circles, we can construct not only a common tangent line. For instance, we can also construct the second tangent point if the centers of the circles and the first tangent point are given. The variation is done over the radii of the circles. The directed graph is shown below.

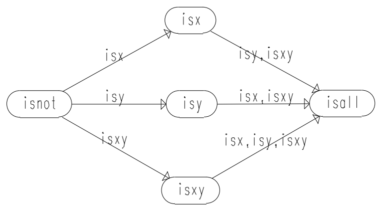


With this modeling all other variations can be solved sequentially, too.

For a circle we have, beside the loci for the center as with points the following loci: tangential to line or circle, point incident with circle, and radius given. For up to three combinations of these loci, a procedure is defined to compute the circle. In many cases two or more solutions are possible. Then the solution nearest to the actual situation is chosen. For lines we have up to two of the following loci: direction, tangential to circle, point co-incident with the line.
The idea of orienting the edges in the constraint graph to determine the sequence of the computations is well known. When an orientation without cycles is possible, this orientation can be achieved by propagating the constraints and the degree of freedom. With cycles the maximal constraint matching is adapted from the Edmonds method for bipartite graphs [3]. With the constraints used here, we have to consider some special semantics. When orienting the constraint graph each variable node may have at most so many incoming edges as the degree of freedom of the variable is. That is 2 for points and lines and 3 for circles. But this condition is necessary but not sufficient. Some constraints as horizontal or vertical alignment of point can only be satisfied by the x respectively y coordinate. For lines constraints like horizontal or vertical can only be satisfied by the direction. A circle can have only 2 incoming constraints involving the radius. To orient the undirected constraint graph of the model, we need therefore a special function for each variable class. This function determines whether for an adjacent constraint the variable still has a degree of freedom to orient an edge as incoming to the variable. For all classes of variables, a function `function sdf(afix:tfix):integer` is implemented. `afix` can have the following values:
lsy: The constraint can be satisfied by the y coordinate of a point
lsxy: The constraint can be satisfied by the x coordinate or the y coordinate of a point
lsr: The constraint can be satisfied by the radius of a circle resp. the distance of a line from the origin.
lsxr: The constraint can be satisfied by the x coordinate of the center of a circle or the radius of a circle resp. the direction and distance from the origin of a line
lsyr: The constraint can be satisfied by the y coordinate of the center of a circle or the radius of a circle

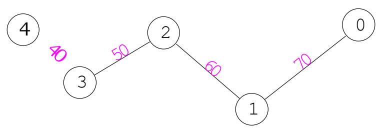
Isxyr: The constraint can be satisfied by the x coordinate or the y coordinate of the center of a circle or the radius of a circle

So the parameter `afix` identifies the parameters of the variable with which the constraint can be satisfied. The function `sdf` examines all already incoming edges, and determines what is already fixed. For scalars this is simple. When an edge is already incoming, `sdf` is 0.
For points, the following transition matrix must be used.

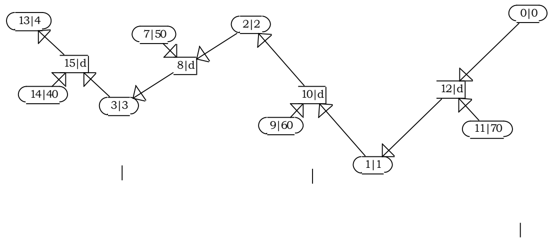


A point has initially 2 degrees of freedom, which corresponds to the state `isnot`. If the X-coordinate is fixed by a constraint, the state changes into `isx`. Now it can only fulfill yet another constraint, which changes the Y-coordinate too. This can be either a constraint only affecting the Y-coordinate (`isy`), or a constraint which specifies a locus (`isxy`).
If no further transition with `afix` is possible, `sdf` is 0. Using the transition matrix when both constraints can only be satisfied by the x coordinate, 2 edges can not be incoming oriented.
For a circle, we have the same requirements for the center point. But an additional requirement is that there can be at most one incoming edge, which can only be satisfied by the radius.
For a line, we can have at most one incoming constraint that can be satisfied only by the direction of the line (which is the case with the constraints parallel and perpendicular and generally for a constraint determining the direction of a line).

With this graph-oriented approach we can also handle under-constrained systems in an adequate manner.



If we move the point 0, the resulting constraint graph is shown below.



The move of point 0 is propagated through all points. So the other points will follow the moving of point 0. To achieve this, a breadth-first search starting with point 0 is done. Then the edges are oriented in the order of the sorting (when possible).

References:

[1] Berling, R.; Rosendahl, M.: Modeling of Geometric Constraints in CAD-Applications. Part of: Geometric Constraint Solving and Applications, Brüderlin, B; Roller, D (Eds.), 151-169, Springer 1998. http://dx.doi.org/10.1007/978-3- 642-58898-3_8

[2] Bettig, B.; Hoffmann, C.M.: Geometric Constraint Solving in Parametric CAD, 2010 www.cs.purdue.edu/homes/cmh/distribution/PapersChron/ConstraintSurvey2010.pdf

[3] McHugh, J.: Algorithmic Graph Theory, Prentice-Hall, New Jersey 1990.