Drawing Straight Lines

Jeremy J. Carroll
Publishing Systems and Solutions Laboratory
HP Laboratories Bristol
HPL-2000-72
December 21st , 2001*

lines, pseudo-lines, polar coordinates, trilinear coordinates, non-linear programming, geometry

We present a graph-drawing algorithm which respects extended co-linearity constraints. These are expressed as a set of 'straight' paths in a planar graph. Constraints of this sort are translated into a set of inequalities over polar coordinates of lines. These inequalities provide necessary and sufficient conditions for the lines to be an appropriate drawing of the graph. Combined with a formula expressing some aesthetic parameter these inequalities then express the graph drawing problem as a classic non-linear program. These inequalities are solved by first solving linear inequalities in the angles, and then tackling the full non-linear program. The solution is a drawing of the graph. The algorithm has been used to draw all the solutions to the 6-Venn triangle problem.

* Internal Accession Date Only Approved for External Publication

1. Introduction

This work is part of a project that is looking at the relationship between sets of paths in planar graphs, and drawings of the graphs in which those paths are straight lines.

We note that this problem is significantly harder than that solved in the well-known paper of Fáry [10]. He draws planar graphs such that every edge is a straight line. Our problem is to have every edge a straight line, *and* to have selected sequences of edges (paths) to be the same straight line.

Shor [19] shows that this problem is NP-hard.

In Carroll [2] it is shown that for sets of lines (not line segments or semi-lines) that their betweenness properties are equivalent to a set of inequalities over their polar coordinates.

The approach taken to drawing graphs with co-linearity constraints here is to construct such a set of inequalities, and then solve them.

1.1 Venn Triangles

A sample graph drawing application requiring this functionality is that of drawing Venn diagrams (see [17]) formed from polygons. This problem motivated this work; and is discussed in more detail in Carroll [3].

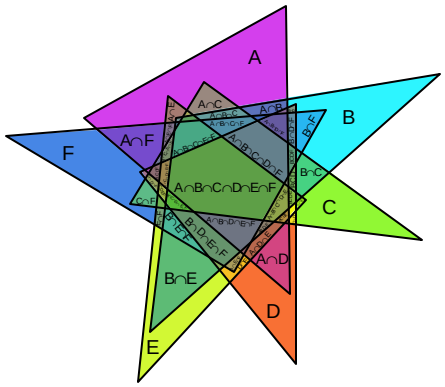

Figure 1 A Venn Diagram of Six Triangles

In 1975, Grünbaum [12] first posed the problem of drawing a simple Venn diagram of six triangles. This is six triangles such that:

- No three lines are concurrent
- The triangles divide the plane into 64 regions each of which is inside its own unique subset of the set of triangles.

1.2 Linear and Non-Linear Programming

Linear and non-linear programming are methods for finding optimal solutions to sets of inequalities. “Optimal” means either minimizing or maximizing an objective function.

Linear programming concerns linear inequalities and objective functions over a number of variables. It is a mature field, initiated in 1826 by Fourier [11]. Farkas [8] furnished the fundamental theorem of linear inequalities in

1894; and with the rise of computers in the 1940's practical methods were developed, (see Schrijver [18]).

Large soluble sets of linear inequalities can be solved except where numerical instability overwhelms the floating-point precision used.

Non-linear programming is the generalization of linear programming to non-linear inequalities and objectives. Much of the theory of non-linear programming is restricted to convex inequalities. This technical restriction is particularly irksome in our context since our inequalities are not convex. The main theoretical work of interest is the Kuhn-Tucker theorem [13] (see also Stoer and Witzgall [23]). This forms the foundations for the non-linear programming library that we use: Spellucci's **donlp2** [20],[22].

1.3 Angles

From graph theoretic constraints and by a consideration of elementary Euclidean geometry [7], we derive a linear program over the angle ordinates.

This is a good starting point for non-linear analysis over the polar coordinates.

1.4 Unaddressed Issues

This paper does not address the important theoretical question of when graphs have such straight line drawings and when not. We do little more than note the obvious that two straight lines intersect at most once. In some sense, this is a practical paper, detailing a working (if somewhat fragile) algorithm. Future theoretical work may show the domain of applicability of the algorithm and give an indication of its numerical stability and computational complexity.

The system described in this paper has only been used for the limited goal of drawing Venn triangles. We occasional mention where variation might be expected for harder examples, but the techniques have not been tested on such examples.

The key simplifications that the limited area of application affords are:

- There are exactly 18 lines considered, forming exactly 63 internal faces, with 80 vertices.
- Every vertex has only two lines meeting.
- There are no pairs of lines that are explicitly parallel.
- The Venn triangles are expressed as line segments, not lines. Each end of a line segment meets an end of one other line segment.

This paper does not make use of any of these assumptions, but the implementation did.

2. Summary of the Algorithm

We briefly describe the whole algorithm including:

- The expected input into the graph drawing algorithm.
- The analysis the graph to generate sets of inequalities, and objective functions.
- The creation of a sequence of linear and non-linear programs.

There is also a final display step that we omit entirely from this paper.

2.1 The Input Graph

The algorithm starts with a planar graph and some specified paths in it. The objective is to draw the graph such that each of the specified paths is a straight line. Moreover, some other aesthetic criteria should be maximal (e.g. minimal size).

The theoretical foundation for the graph drawing algorithm is a result about lines not line segments.

Hence, we require additional information about the end-points of the line-segments.[1]

2.2 Analysing the Graph

During the analysis phase we:

- Fix a polar frame of reference with reference to the graph.
- Find all convex polygons in the graph using Roy's transitive closure algorithm [16][2].
- Generate linear inequalities in angles around this frame of reference.
- Generate betweenness inequalities for each of the triangles in the graph.
- Generate additional betweenness inequalities for the end-points of line segments.
- Generate formulae for various linear and non-linear objective functions.

2.3 Solving the Programs

We now group the various inequalities into a number of linear and non-linear programs. The solution for one is used as the input to the next.

- Solve the linear angle program.

---

[1] This paper only covers simple corners where two line-segments both end, and not more complex cases like two line-segments meeting at a vertex where one ends but not the other.

[2] Often attributed to Warshall [24].

- Solve the non-linear betweenness program with a simple linear objective. (This furnishes an interior starting point for the subsequent harder programs).
- Solve the non-linear betweenness program with the actual objective(s).

The last two programs both produce polar coordinates of a drawing of the Venn diagram. The last is optimised with the user's objective, the penultimate gives the NLP system an easier initial task.

3. The Problem Statement: the Input Graph

We assume as input, a planar graph G expressed as an edge disjoint union of designated paths: i.e. a planar graph which is graph-partitioned into paths $\{P_1, P_2, \cdots P_n\}$ such that:

$$G = \bigcup_1^n P_i \qquad 3.0.1$$

$$\left| P_i \cap P_j \right| \leq 1 \qquad 3.0.2$$

We will refer to these designated paths as *straight* paths.

The theoretical framework for the graph drawing algorithm is about lines not line segments. Hence we need additional information concerning the end-points of the straight paths. Here we only consider simple corners. A simple corner c is a vertex of degree two where two straight paths P and Q both end.

The additional information required is:

> For each straight path R in the graph, which does not intersect with either P or Q, we permit the identification of a vertex v not on R such that in the desired drawing, v and c will lie in the same half plane of R. v should be selected so that there are two straight v-R paths. (A v-R path that is also straight).

If we do not have this information then in some cases the algorithm may produce drawings in which R intersects with P or Q.

We also need to identify the faces in the plane graph well before we draw it. We use facial cycle information, currently specified in the problem statement. See Diestel [6] p78 for a discussion about facial boundaries.

4. Polar Coordinates

Polar coordinates can be used to describe lines as well as points.

We note that a polar reference system consists of an origin O, a semi-line (the axis) starting at O, and an orientation around the origin. Lines in a polar system are given a pair of coordinates (d,q). d is the perpendicular

distance from the line to the origin. d is always non-negative. q is the oriented angle (between 0 and 2π)[3] from the axis to the perpendicular dropped by the line through O. The line can be given a direction following the polar orientation around the origin. Both the choice of polar reference system and this act of orienting lines around it depend only on graph theoretic considerations.

We select a preferred face of G in which we choose to locate the origin. We also choose an orientation around this face.

Using these we then give a direction to all the straight paths in the graph, so that they all follow the chosen orientation around the chosen face.

With these assumptions we are now looking for n sets of pairs (d_i, q_i) such that the plane graph formed by drawing the n lines with these coordinates contains G as a minor in such a way that for each i, P_i corresponds to the line with coordinates (d_i, q_i).

Polar coordinates have the distinct advantage over Cartesian or homogenous coordinates that a line, which has two degrees of freedom, is described with two numbers and not three or four. Moreover, these two numbers separate out two natural and distinct properties of the line, rather than the somewhat artificial Cartesian properties.

4.1 The linear angle inequalities

We consider any two non-parallel lines P and Q, and their point of intersection v. We consider the quadrilateral F formed by the two lines and their two perpendiculars to O. We separate two cases: the normal case in which the polar axis does not intersect the interior of F, and the wrap-around case in which the polar axis does intersect the interior of F.

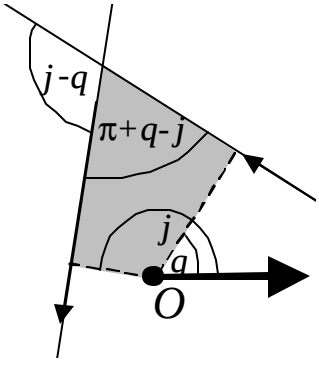

Figure 2: Normal intersection

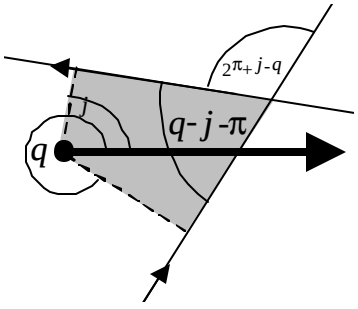

Figure 3: Wrap around intersection

[3] We express angles in radians.

The graph theoretic information alone (including the facial cycles) allows us to distinguish which one of the two cases apply, and which of P and Q play the role of the line at q or j. We also note that the angles marked at v in both figures are positive.

We generate a pair of linear inequalities.

In the normal case:

$$j - q > 0 \qquad 4.1.1$$

$$p + q - j > 0 \qquad 4.1.2$$

In the wrap-around case:

$$2p + j - q > 0 \qquad 4.1.3$$

$$q - j - p > 0 \qquad 4.1.4$$

By considering every pair of intersecting straight paths in G in turn, we generate a set of no more than $n(n\text{-}1)$ linear inequalities. Any drawing of the graph will have lines with polar coordinates whose angles satisfy these inequalities.

Moreover, values solving these inequalities satisfy Euclid's propositions [7] concerning angles in a straight line drawing. In particular, they satisfy the supplementary angle theorem (13), the exterior angle theorem (16) and the angle sum of a triangle (32).

To turn this into a linear program, we add an extra free variable *obj* which we use as the RHS of each inequality. We also change the inequality from a strict inequality to a non-strict inequality e.g. for inequalities like 4.1.3 we generate

$$2p + j - q \geq obj \qquad 4.1.5$$

The linear angle program is the set of modified inequalities with objective function *obj* to be maximised. For graphs which can be drawn as required, this program is soluble with positive *obj* and its solution is a sensible starting point for more complex computations over the polar coordinates.

4.2 Circular Saws

Unfortunately, given a solution to the linear angle inequalities, it is not always possible to draw the graph with lines at those angles. The simplest counterexample is the 3-circular saw below.

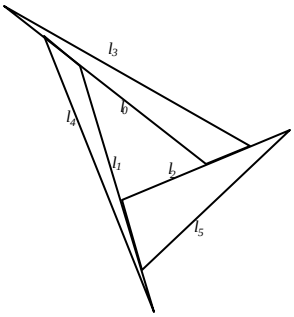

Figure 4: A 3-Circular Saw

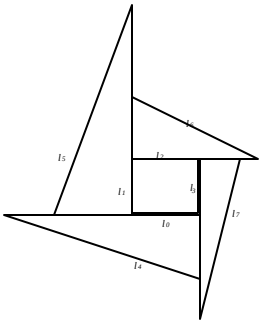

Figure 5: A 4-Circular Saw

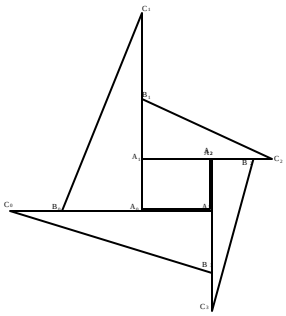

Figure 6 A 4-Circular Saw with labelled vertices

Following Carroll [4] we define a circular saw as:

Defintion: An *n-circular saw* is $3n$ points $A_0, B_0, C_0, A_1, B_1, C_1, \ldots A_{n-1}, B_{n-1}, C_{n-1}$ for some $n = 3$, where:

$A_0, A_1, \ldots A_{n-1}$ form a convex polygon. 4.2.1

For all i: A_{i-1}, A_i, B_i, C_i are collinear, 4.2.2

with[4] $A_{i-1} = A_i - B_i = C_i$ 4.2.3

Here the notation $A_{i-1} = A_i - B_i = C_i$ is an extension of the standard betweenness notation (see e.g. Millman and Parker [14]). The symbol '=' indicates that the points either side of it may be identical.

Thus $A_{i-1} = A_i - B_i = C_i$ expands to:

$A_{i-1} - A_i - B_i - C_i$ 4.2.4

Or:

$A_{i-1}=A_i \,\&\, A_i - B_i - C_i$ 4.2.5

Or:

$A_{i-1} - A_i - B_i \,\&\, B_i = C_i$ 4.2.6

Or:

$A_{i-1}=A_i \,\&\, B_i = C_i$ 4.2.7

And also from [4], the circular saw theorem:

Theorem: The circular saw theorem. In a circular saw we have:

$$\prod_{i=0}^{n-1} \sin\left(\angle A_i C_{i+1} B_i\right) \le \prod_{i=0}^{n-1} \sin\left(\angle C_{i+1} B_i C_i\right) \qquad 4.2.8$$

with equality if and only if every betweenness condition 4.2.3 of the circular saw is satisfied with double equality 4.2.7.

[4] Subscript arithmetic modulo n.

This gives non-linear inequalities like:

$$\left|\sin(q_4-q_0)\sin(q_5-q_1)\sin(q_3-q_2)\right|-\left|\sin(q_3-q_0)\sin(q_4-q_1)\sin(q_5-q_2)\right|\geq 0 \qquad 4.2.9$$

(for Figure 4) and:

$$\left|\sin(q_5-q_0)\sin(q_6-q_1)\sin(q_7-q_2)\sin(q_4-q_3)\right| - \left|\sin(q_4-q_0)\sin(q_5-q_1)\sin(q_6-q_2)\sin(q_7-q_3)\right|\geq 0 \qquad 4.2.10$$

(for Figure 5).

In each of 4.2.9 and 4.2.10 q_i is the angle ordinate of the line l_i in the corresponding figure. Moreover, we use the absolute value of each product to avoid the need to worry about the sign (the circular saw theorem is stated with all angles lying between 0 and π). We can replace the comparison operator with either > or = by following the circular saw theorem.

Hence, the angles we have found by solving the linear angle program are usually not possible angles for drawing the graph. However, they are a suitable starting point for the non-linear analysis that follows. This analysis can be thought of as drawing an initial guess at how the lines will go using these angles, and then refining the guess until other (sufficient) conditions hold.

5. Triangles in polar coordinates

We will review some of the theory from Carroll [2].

We consider a triangle formed by three lines: (d_1,q_1), (d_2,q_2) and (d_3,q_3). Each of the three sides of the triangle divides the plane into two half-planes, giving six half planes to consider in total. Of these the triangle lies in three, and O lies in three. If O lies inside the triangle then these two sets of half-planes are identical. Alternatively their intersection may have one or two members. We show all three cases in the following figures.

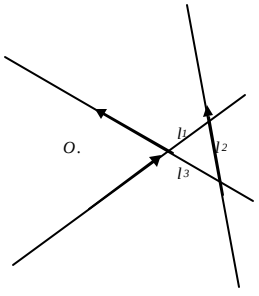

Figure 7: Position 1

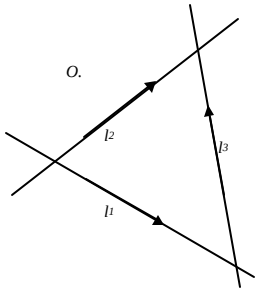

Figure 8: Position 2

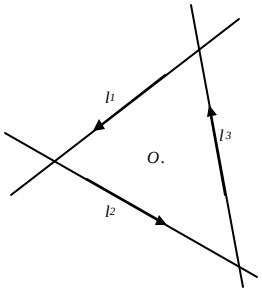

Figure 9: Position 3

We refer to the relative position of the origin to the triangle by saying that the triangle is in position 1, 2 or 3 depending on whether it shares one, two or three half-planes with the origin.

5.1 The Trilinear Constant

Coxeter [5] (p378) asserts[5] that for any point P in the plane and any triangle that the following equation holds between the lengths of the sides of the triangles and the exact trilinear coordinates of P. (See Weisstein [25] for an introduction to trilinear coordinates).

$$ax + by + cz = 2\Delta \qquad 5.1.1$$

Here, a b and c are the lengths of the sides of the triangles and each of x, y and z is the signed perpendicular distance from P to the corresponding side of the triangle. Δ is the area of the triangle.

We prefer to express this in terms of the angles A, B, and C of the triangle as:

$$x\sin(A) + y\sin(B) + z\sin(C) = \frac{4\Delta^2}{abc} \qquad 5.1.2$$

Moreover, we don't care about area, and so transform the equation 5.1.2 into an inequality:

$$x\sin(A) + y\sin(B) + z\sin(C) > 0 \qquad 5.1.3$$

For three concurrent lines we can take 5.1.2 to the limit and find that:

$$x\sin(A) + y\sin(B) + z\sin(C) = 0 \qquad 5.1.4$$

5.2 The Betweenness Inequalities

Translating 5.1.3 and 5.1.4 into our framework of polar coordinates we note that:

- The sine function has period 2π; so we can ignore the wrap around issue.
- The distance ordinate is the unsigned distance of O to the line, whereas the exact trilinear coordinates use signed distance. The sign of the distance depends on whether O lies in the same half plane as the triangle or not, and hence on our notion of position 1, 2 or 3.

We identify the three lines as l_1, l_2 and l_3 as in the diagrams, with polar coordinates (d_1,q_1) etc. Moreover, we order them as in the figures. Then we get one of the following three inequalities depending upon the position of the triangle:

$$-d_1\sin(q_3 - q_2) + d_2\sin(q_3 - q_1) - d_3\sin(q_2 - q_1) > 0 \qquad 5.2.1$$

$$d_1\sin(q_3 - q_2) - d_2\sin(q_3 - q_1) + d_3\sin(q_2 - q_1) > 0 \qquad 5.2.2$$

$$d_1\sin(q_3 - q_2) + d_2\sin(q_1 - q_3) + d_3\sin(q_2 - q_1) > 0 \qquad 5.2.3$$

[5] A proof is given in Carroll [2].

We have written these inequalities so that all of the sine terms are positive, when the linear angle inequalities hold. Particular attention should be paid to the middle term of 5.2.3, where the position 3 triangle differs from the other two.

For concurrent lines, position 3 is not possible, and we find:

$$d_1 \sin(q_3 - q_2) - d_2 \sin(q_3 - q_1) + d_3 \sin(q_2 - q_1) = 0 \qquad 5.2.4$$

We refer to the complete set of inequalities and equalities of the form 5.2.1 through 5.2.4 as the betweenness inequalities of *G*.

In Carroll [2] it is shown that given a straight line drawing *D* and a polar frame of reference, any non-negative solution to the combined linear angle inequalities and betweenness inequalities has the same betweenness relationships as *D*.

Since the betweenness inequalities and the linear angle inequalities of a drawing depend only on its graph theoretic properties, we can write down the inequalities solved by a drawing of *G* before we have drawn it. Thus to draw the graph, it almost suffices to solve the combined set of inequalities. The only issue is that the straight line drawing result is not directly applicable to line segments. So we augment the betweenness inequalities with extra inequalities describing the position of the end-points of the straight paths.

5.3 Corners and Betweenness

Considering a corner *c* and some path *R*, in section 3 we encourage the identification of a half-plane of in which *c* lies. Figure 10 shows one possible drawing consistent with *c* lying in the same half-plane as *v*.

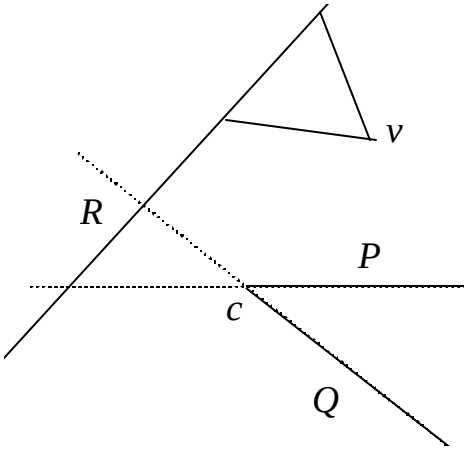

Figure 10 A corner

From this drawing we could identify one of the betweenness inequalities for the triangle formed by *R* and extensions of *P* and *Q*. However, this drawing is only one of the possible drawings. We have not specified the sign of the angles between *R* and *P* and *R* and *Q*. We have not generated linear angle inequalities for the intersection of *R* and *Q*, and *R* and *P*. In particular, we do not know if the intersection of *R* and *P* is as in the figure, or if *R* is parallel to *P*, or if *R* intersects the extended line *P* in

the other semi-line of *P* not shown in the figure.

Thus two of the three terms in the betweenness inequality are of unknown sign, and the roles of *P*, *Q* and *R* in the betweenness inequality are unknown. On close examination it is found that as the signs of the angle differences change, the betweenness inequality changes between 5.2.1, 5.2.2, and 5.2.3 appropriately to express that c lies in the identified half-plane and not the other.[6] Thus, in some sense, there is one betweenness inequality that expresses that either *c* lies in the same half-plane of *R* as the origin or that it lies in the other half-plane.[7]

For each of the additional statements about corners provided in the problem definition we add an appropriate additional betweenness inequality.

5.4 Modifying the betweenness inequalities

To express the inequalities as a linear or non-linear program requires that we translate them from strict inequalities to non-strict ones. Since we can scale our solution by multiplying all the distances by a constant, it suffices to choose an arbitrary positive RHS for each inequality 5.2.1, 5.2.2 or 5.2.3. We choose 1.

Hence as a linear or non-linear program, each betweenness inequality is expressed as one of:

$$-d_1 \sin(q_3 - q_2) + d_2 \sin(q_3 - q_1) - d_3 \sin(q_2 - q_1) \geq 1 \qquad 5.4.1$$

$$d_1 \sin(q_3 - q_2) - d_2 \sin(q_3 - q_1) + d_3 \sin(q_2 - q_1) \geq 1 \qquad 5.4.2$$

$$d_1 \sin(q_3 - q_2) + d_2 \sin(q_1 - q_3) + d_3 \sin(q_2 - q_1) \geq 1 \qquad 5.4.3$$

$$d_1 \sin(q_3 - q_2) - d_2 \sin(q_3 - q_1) + d_3 \sin(q_2 - q_1) = 0 \qquad 5.4.4$$

6. The Betweenness Inequalities as a Linear Program

As noted in section 1.2, linear programming is a lot easier than non-linear programming. Thus it is desirable to express our drawing problem as an LP problem rather than an NLP problem. While the betweenness inequalities are non-linear in the angle ordinates, they are linear in the distance ordinates. Thus it is tempting to make a convincing guess at the angles (using linear or non-linear analysis) and then to solve the betweenness inequalities as a linear program over the distance ordinates.

Any such solution can be used as an initial interior point to further non-linear analysis of the betweenness inequalities over both coordinates.

---

[6] The parallel case is outside the scope of this paper, but is discussed in Carroll [2].

[7] A different introduction to the betweenness inequalities may have made this clearer.

The author has not succeeded in identifying even experimentally sufficient conditions for the linear program to be soluble.

Clearly we need to find angles that satisfy the linear angle inequalities and the circular saw inequalities 4.2.9 and 4.2.10, which are all necessary conditions. In practice this is not sufficient and the linear program solver will usually complain of numerical instability; possibly because there are some other figures that put constraints on the angles between the lines, not just intersections and circular saws.

Alternative explanations are

- There may be implicit unavoidable circular saws in the graph formed by the extension of the drawings of the straight-paths from line segments to lines.
- Insufficient precision in the floating-point arithmetic of the solver (typically 64 bit).
- A linear program with a sensitivity that exceeds the floating-point capabilities of the solver.[8]

The author has explored the first at length, but unsuccessfully.

7. Drawing the Graph

From the linear angle inequalities we generate a linear program to maximise the smallest angle in the figure.

We use these angles as a starting point to a non-linear program using the linear angle inequalities and the betweenness inequalities with some simple linear objective (e.g. to minimise the sum of the distances, or to maximise the smallest angle in the figure).

A solution to this problem is (the coordinates of) a drawing of the graph satisfying the co-linearity constraints. It can then be used as an interior starting point for a more compute intensive computation trying to maximise the user's real objective, some aesthetic function over the drawing.

An example aesthetic function is to minimize the size of the graph by considering the distance from the origin to selected vertices in the graph.

A solution to such a program will be an optimal drawing of the graph.

8. The Polygon Analysis

We now go back to the beginning of the algorithm to fill in the omitted step of polygon analysis. In this step we find all the convex polygons in the graph, and as a consequence:

[8] These are typically less than machine precision, since the solver needs to treat some sufficiently small numbers as zero.

- All the triangles.
- All the circular saws.

The algorithm is as follows:

1. Fix a polar frame of reference.[9]
2. For each pair of intersecting straight paths in G, allocate four angle identifiers for the angles formed at their intersection. (If one or both of the paths ends at the intersection then two or three of the angle identifiers are unused). The angles can be distinguished using the polar frame of reference, see Figure 11.
3. Allocate a square connectivity matrix indexed by the set of angles (with up to $2n(n\text{-}1)$ members).
4. For each straight path P in G, for every pair of distinct vertices u and v on P, add two entries to the connectivity matrix. We connect angles at u and v along both sides of P, see Figure 12.
5. Apply Roy's [16] transitive closure algorithm[10] over the connectivity matrix, recording every derivation in the connectivity matrix. The algorithm is modified to exclude paths which turn back across themselves.
6. Every r-agon now has r entries in the diagonal of the connectivity matrix.

We store the triangles found in this way separately, and classify them as either in position 1, 2 or 3 with respect to the origin.

For each polygon, we can look for triangles around it that form a circular saw.

We note that since the algorithm described does not use the circular saw information that this last step can be omitted, and the expensive transitive closure algorithm can be replaced by taking the third power of the connectivity matrix (since we are only using the triangle information).

[9] This step does not appear to have any logical connection with the algorithm, however in practice the author has found it useful, and we are going to need this later anyway.

[10] Usually attributed to Warshall [24].

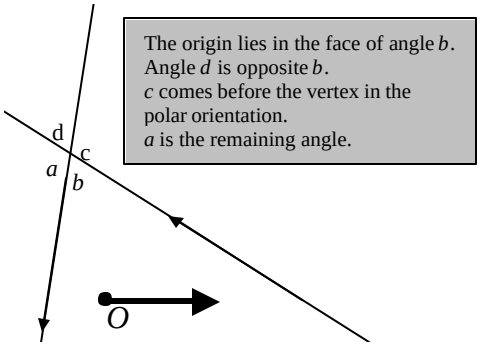

Figure 11: The four angles at a vertex.

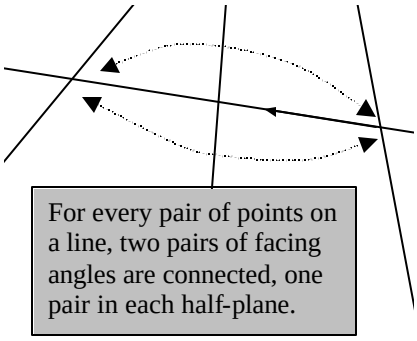

Figure 12: Connecting angles.

9. Redundant Inequalities

The methods described generate hundreds of inequalities per graph. It is fairly easy to eliminate some redundant linear inequalities. With the betweenness inequalities, an optimal implementation would use geometric considerations to eliminate superfluous ones before entering the non-linear programming steps. The author has found the performance of **donlp2** [20] acceptable despite the presence of large numbers of redundant inequalities.

10. Further Practical Considerations

We have seen in section 6 a number of practical considerations concerning linear programming. We now discuss non-linear programming.

First: the key difference between the various linear programming solutions is speed and price. The key difference between the various non-linear programming solutions is effectiveness. Before using **donlp2** [20], the author had unhappy experiences with less useful systems.

Second: the numerical issues raised in section **Error! Reference source not found.** apply to non-linear programming just as much if not more so than to linear programming. It was found necessary to adjust some of the **donlp2** parameters, in particular a smaller value for *epsx* was used, to prolong the search with a large number (a thousand or two) of small steps.

Third: a particular difficulty is how to express the linear angle inequalities. In the betweenness programs, these inequalities are strict inequalities compared with 0. It is, however not possible to express strict inequalities in any well thought out NLP system. In donlp2 for example there is a small constant `delmin` chosen by the user such that "`constraints are satisfied if |h_i(x)|<=delmin, g_j(x)>=-delmin respectively`" (from the Fortran common block description in [21]). This means that any number close to zero counts as zero, and so a formula that should be strictly greater than zero can be satisfied with a (very small) negative value.

Thus when using **donlp2** the RHS of the linear angle inequalities and the value of `delmin` need to be chosen in such a way that the value of the formula really is positive for any acceptable solution.

Fourth: when using NLP it is generally required that the objective function and the constraints are differentiated. It is significantly more efficient if the program is structured to allow analytic differentiation rather than numerical.

Fifth: when specifying objectives in **donlp2** there is considerable practical advantage in having an objective that is well defined in a fairly large surrounding of the desired solution. E.g. instead of minimising the distance from the origin to the intersection of two lines (which is undefined when the lines are parallel), it is better to maximise the reciprocal of this value. **donlp2** does in fact cope quite well with division by zero errors, and does have a method to identify the domain of definition of the functions evaluated, but the author found considerable benefit in not having to use either of these.

A typical 6-Venn triangle diagram took about three minutes to draw on a 700MHz PC. The algorithm successfully draws all 126 6-Venn triangles.

11. Theoretical Summary

The theoretical foundation for this work is provided by Carroll [2] and [4].

The important gaps are:

- Determining the domain of solubility of the graph-drawing problem with straight-line constraints.
- How well does the algorithm apply to other straight line drawing problems? Are there special properties of Venn triangles (like Bultena's directed dual with one source and one sink [1]) that make them particularly easy to draw?
- Understanding the constraints on angles in arbitrary line drawings. This may consist of further diagrams like the circular saw.
- Understanding the constraints on the facial boundary information in the input. Can this be generated as part of the algorithm?
- Providing an adequate account of non-simply intersecting problems. In particular doing justice to Pappus' theorem [15], which can be read as a non-trivial constraint on the angles of a particular line drawing.

12. Conclusion

We have seen how the graph-drawing problem with straight-line constraints can be specified as a sequence of linear and non-linear programs in increasing difficulty. The optimal solution of each is a suitable starting point for the next.

The optimal solution of the final programs is an optimal drawing of the graph.

We have seen parts of a theoretical framework that may, in the future, offer a complete graph theoretic characterisation of straight lines in the Euclidean plane.

13. References

[1] B. Bultena, B. Grünbaum, and F. Ruskey, *Convex Drawings of Intersecting Families of Simple Closed Curves,* 1998. Presented at the 11th Canadian Conference on Computational Geometry, (1999), 18-21.

[2] J.J. Carroll, *Betweenness in Polar Coordinates*. Hewlett-Packard Laboratories Report, No. HPL-2000-71, Bristol, U.K., 2000.

[3] J.J. Carroll, *Drawing Venn Triangles*. Hewlett-Packard Laboratories Report, No. HPL-2000-73, Bristol, U.K., 2000.

[4] J.J. Carroll, *The Sharpness of Circular Saws*. Hewlett-Packard Laboratories Report, No. HPL-2000-74, Bristol, U.K., 2000.

[5] H.S.M. Coxeter, *Some Applications of Trilinear Coordinates*, Linear Algebra and its Applications, New York, 1995, pp. 375-388.

[6] Reinhard Diestel, *Graph Theory*, New York, 1997.

[7] Euclid, The Elements, Book I, c 300 BC, [English translation: D. Joyce, http://aleph0.clarku.edu/~djoyce/java/elements/bookI/bookI.html, 1996].

[8] Farkas Gy, *A Fourier-féle mechanikai elv alkalmazásai* (Hungarian) [On the applications of the mechanical principle of Fourier], Mathematikai és Természettudományi Értesitö 12, 1894 pp 457-472. (German translation: [9])

[9] Farkas J, *Über die Anwendungen des mechnischen Princeps von Fourier,* Mathematische und naturwissenshaftliche Berichte aus Ungarn 12, 1895 pp 263-281.

[10] Fáry, I. *On Straight Line Representations of Planar Graphs*. Acta Sci. Math. 11, 1948, pp 229-233.

[11] J.B.J. Fourier *Analyse des travaux de l'Académie Royale des Sciences, pendant l'année 1823, Partie mathématique*, Historie de l'Académie Royale des Sciences de l'Institut de France 6 [1823] (1826) xxix-xli [partially reprinted as: Premier extrait, in: *Oevres de Fourier,* Tome II (G. Darboux, ed.), Gauthier-Villars, Paris, 1890, [reprinted: G.Olms, Hildesheim, 1970] pp 317-319].

[12] B. Grünbaum, *Venn diagrams and independent families of sets*. Mathematics Magazine 48, 1975. pp12-22.

[13] H.W. Kuhn and A.W. Tucker. *Non-linear programming*. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability (ed. J. Neyman). Berkeley.1950, pp 481-492

[14] Richard S. Millman and George D. Parker, Geometry: A Metric Approach with Models, 2nd Ed., New York, 1991 (1st Ed. 1981).

[15] Pappus, *Mathematical Collection,* c300.

[16] B. Roy. *Transitivité et connexité*. C.R. Acad. Sci. Paris 249, 1959 pp 216-218.

[17] Frank Ruskey, *A Survey of Venn Diagrams*, The Electronic Journal of Combinatorics 4, DS#5, 1997, http://www.combinatorics.org/Surveys/ds5/VennEJC.html.

[18] Alexander Schrijver. *Theory of Linear and Integer Programming.* Chichester (Wiley) 1986 (paperback: 1998).

[19] Peter W. Shor, *Stretchability of pseudolines is NP-hard.* Applied geometry and discrete mathematics, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 4, Amer. Math. Soc., 1991, pp 531-554.

[20] P. Spellucci, donlp2, ftp://ftp.mathematik.tu-darmstadt.de/ /pub/department/software/opti/donlp2_ansi_c.tar.gz, (ANSI C translation of Fortran due to S. Schoeffert) 1998.

[21] P. Spellucci, *DONLP2 User Guide*, included with [20], 1998.

[22] P. Spellucci, *A new technique for general nonlinear programs using only equality constrained subproblems*. Math. Prog. 82 1998, pp 413-448.

[23] Josef Stoer and Christoph Witzgall, *Convexity and Optimization in Finite Dimensions I,* Berlin, 1970.

[24] S. Warshall, *A theorem on Boolean matrices*, Journal of the ACM, 9(1), 1962, pp11-12

[25] Eric Weisstein, *Trilinear Coordinates*, World of Mathematics, 1996-2000, http://mathworld.wolfram.com/TrilinearCoordinates.html.