A Proposed Algorithm for Calculating the Minimum Crossing Number of a Graph

Frederick C. Harris, Jr.
Department of Computer Science

Cynthia R. Harris
Department of Mathematics

University of Nevada
Reno, NV 89557
fredh@cs.unr.edu

Abstract

In this paper we present a branch-and-bound algorithm for finding the minimum crossing number of a graph. We begin with the vertex set and add edges by selecting every legal option for creating a crossing or not. After each edge is added we determine if the resulting partial graph is planar. We continue adding edges until either all edges have been added or we reach a point where the graph cannot be completed as started. At this point we backtrack to see if the graph can be drawn with fewer crossings by selecting other options when adding edges.
keywords: Crossing Number, Algorithm

1 Introduction

Determining the crossing number of a graph is an important problem with applications in areas such as circuit design and network configuration [17]. It is this importance that has driven our work in finding the minimum crossing number of a graph.

Informally, the *crossing number* of a graph G, denoted $\nu(G)$, is the minimum number of crossings among all good drawings of G in the plane, where a good drawing has the following properties:

(a) No edge crosses itself
(b) No pair of adjacent edges cross
(c) Two edges cross at most once
(d) No more than two edges cross at one point

Although the problem is easily stated and has been well studied, not much is known about the general solutions. Erdös and Guy [8] put together a survey of what was known in 1973, and Turan discussed it via his Brick Factory Problem [25]. However, it was not until 1983 that Garey and Johnson [9] proved that finding the minimum crossing number of a graph was an NP-Complete problem. Because of the difficulty of this problem, work turned away from finding the minimum crossing number of a graph to sub-problems and other related problems.

With regard to the sub-problems, there has been work on product graphs ranging from the product of C_n and graphs of order four [1] to $C_3 \times C_n$ [22], $C_4 \times C_4$ [5], $C_5 \times C_5$ [21], and $C_5 \times C_n$ [16]. There has also been work in the area of bipartite graphs with the results of $K_{3,n}$ [20], $K_{5,n}$ [15], and the torroidal crossing number of $K_{m,n}$[12] having been found. In 1991 Beinstock published work [2] relating the crossing number of a graph to the arrangement of pseudolines,a topic well studied by combinatorialists.

There have also been several related problems that have been studied over the years. Some of these range from the rectilinear crossing problem [11, 23, 24] to the maximum crossing number of a graph or subgraph [10, 13] to the thrackle conjecture [4].

We now wish to turn the attention of this paper back to the minimum crossing number of a graph. In Section 2 we review some notation and definitions we build on throughout the paper. In Section 3 we outline Edmonds' Rotational Embedding Scheme which gives us the foundation for our proposed algorithm. In Section 4 we discuss depth-first search and branch-and-bound algorithms and in Section 5 present our proposed algorithm. Conclusions and future work are presented in Section 6.

2 Notation and Definitions

We now define some terms from topological graph theory which will be used through the rest of this paper. These definitions are as in [3].

For our purposes, a *compact-orientable 2-manifold*, or simply a surface, may be thought of as a sphere or a sphere with handles. The *genus* of the surface is the number of handles.

An *embedding* of a graph G on a surface S is a drawing of G on S in such a manner that edges intersect only at a vertex to which they are both incident.

A region in an embedding is called a *2-cell* if any simple closed curve in that region can be continuously deformed or contracted in that region to a single point. An embedding is called a *2-cell embedding* if all the regions in the embedding are 2-cell.

An algebraic description of a 2-cell embedding was given by Dyck [6] and Heffter[14]. This description is referred to as a Rotational Embedding Scheme which will be covered in Section 3.

Finally, the relationship between the number of regions of a graph and the surface on which it is embedded is described by the well-known generalized *Euler's Formula* [3]:

> Let G be a connected graph with p vertices and q edges with a 2-cell embedding on the surface of genus n having r regions. Then $p - q + r = 2 - 2n$.

3 Rotational Embedding Scheme

With these definitions as background, we now look at the Rotational Embedding Scheme, first formally introduced by Edmonds [7] in 1960 and then discussed in detail by Youngs [26] a few years later. The following is the formal statement of the Rotational Embedding Scheme as given in [3] on pages 130–131.

> Let G be a nontrivial connected graph with $V(G) = \{v_1, v_2, \ldots, v_p\}$. For each 2-cell embedding of G on a surface there exists a unique p-tuple $(\pi_1, \pi_2, \ldots, \pi_p)$, where for $i = 1, 2, \ldots, p$, $\pi_i : V(i) \rightarrow V(i)$ is a cyclic permutation that describes the subscripts of the vertices adjacent to v_i. Conversely, for each such p-tuple $(\pi_1, \pi_2, \ldots, \pi_p)$, there exists a 2-cell embedding of G on some surface such that for $i = 1, 2, \ldots, p$ the subscripts of the vertices adjacent to v_i and in the counterclockwise order about v_i, are given by π_i.

For example, consider Figure 1 which gives a planar embedding of a graph. From this graph we obtain the following counterclockwise permutations associated with each vertex:

$$\begin{array}{ll} \pi_1 = (6,4,2) & \pi_2 = (1,4,3) \\ \pi_3 = (2,4) & \pi_4 = (3,2,1,5) \\ \pi_5 = (4,6) & \pi_6 = (5,1) \end{array}$$

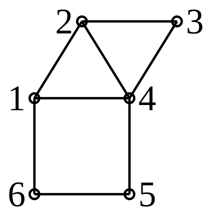

Figure 1: A planar embedding of a graph

From these permutations we can obtain the edges of the graph and the number of regions of the graph. For instance, this graph has 4 regions. The edges for one of these regions can be traced as follows:

1) Start with edge (1,2)
2) From permutation π_2 determine which vertex follows 1; it is 4. Therefore the second edge is (2,4).
3) From permutation π_4 determine which vertex follows 2; it is 1. Therefore the third edge is (4,1).
4) From permutation π_1 determine which vertex follows 4; it is 2. This yields edge (1,2) which was the original edge, so we are finished.

The region we considered is bounded by the edges (1,2), (2,4), and (4,1). The other regions and associated edges can be found in a similar manner.

The important thing to note is the converse portion of the Rotational Embedding Scheme - that every collection of vertex permutations corresponds to an embedding on some surface. Given a set of permutations, we can trace the edges of the regions and determine the genus of the surface on which the graph is embedded.

One of the authors developed a computer program to generate all vertex permutation schemes for a graph [19]. The program then counts the regions of the resulting embedding and, using Euler's formula, determines if a given graph has a planar embedding. The code for this program can be found in [18] and has motivated us to keep track of regions in the construction of a graph drawing, which is the basis for our proposed algorithm presented in Section 5.

4 Depth-First Search with Branch-and-Bound

If the solution space of a problem can be mapped to a tree, where each interior vertex is a partial solution, edges toward the leaves are options that refine the partial solution, and the leaves are complete solutions, then there are various algorithms that can search the tree to find the optimal solution. A depth-first search (DFS) algorithm is one such algorithm which, as its name implies, searches more deeply into the tree for a solution whenever possible. Once a path is found from the root to a leaf representing a solution, the search backtracks to explore the nearest unsearched portion of the tree. This continues until the entire tree has been traversed.

The branch-and-bound portion allows us to change one simple part of the DFS algorithm. When the cost to get to a vertex v exceeds the cost of the current optimal solution, we then tell the DFS algorithm not to traverse the subtree having v as its root.

This method exhaustively covers the entire search space even after finding an initial solution without covering those sections of the search space that lead to solutions that are guaranteed to cost more than the current optimal solution. When the entire tree is covered, the current optimal solution is the globally optimal solution.

5 Proposed Algorithm

For our proposed algorithm we map the solution space from the crossing number problem onto a tree and search for the minimum crossing number with a DFS with branch-and-bound. The root of our tree corresponds to the vertex set. The root has $|E|$ (the number of edges) branches coming out of it. Each branch corresponds to the first edge that is added to the graph. The next level of the tree has $|E| - 1$

branches coming out of each node. The tree is $|E|$ levels deep, and the path to each leaf corresponds to a unique permutation ordering for the edges.

We begin by selecting edge (i, j). We use Euler's Formula to determine if the edge can be added from vertex i to vertex j without any crossings. (We do this by keeping track of the number of regions in the graph as it is constructed.) If so, we add the edge and select the next edge. If not, we choose an edge that has already been added to cross. (This may be the only edge crossed, or the first edge in a long list of edges to be crossed). Once we decide that edge (i, j) is to cross edge (k, l) we create a new vertex m, remove edge (k, l), add partial edges (k, m), (m, l), and (i, m), and then draw edge (m, j) in the same fashion.

When drawing an edge, we remember that an edge cannot cross itself, that no pair of adjacent edges cross, and that two edges cross at most once. We also keep track of all partial edges and remember that they are actually *part* of an edge. In the scenario described in the previous paragraph, we would have to remember that edge (i, m) is part of edge (i, j).

Let us demonstrate the algorithm with K_5 as our example. K_5 has 10 edges. The first 9 are added directly without crossing another edge. This partial graph is shown in Figure 2. When the last edge is considered, Euler's Formula indicates that the resulting graph cannot be embedded on the plane. Therefore, we must add the edge (i, j) with

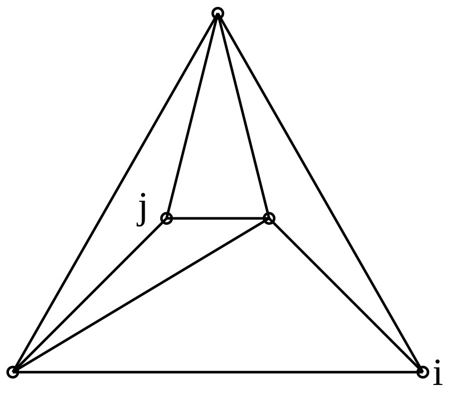

Figure 2: First 9 edges of K_5 with no crossings

at least one crossing.

At this stage we select edge (k, l) to cross, insert vertex m, remove edge (k, l), and add partial edges (k, m), (m, l), and (i, m). The graph is as shown in Figure 3. Once these partial edges have been drawn the algorithm tries to draw the rest of edge (i, j) without any crossings and succeeds. Figure 4 shows this drawing of K_5 with one crossing.

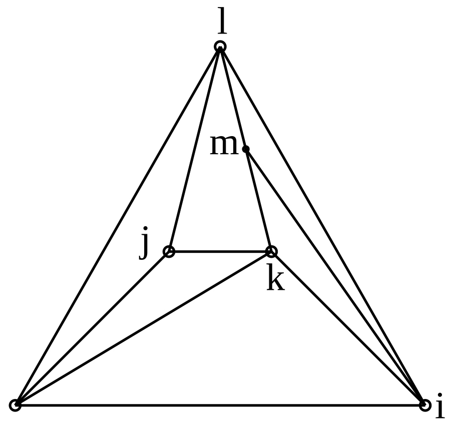

Figure 3: The first part of the last edge is drawn for K_5

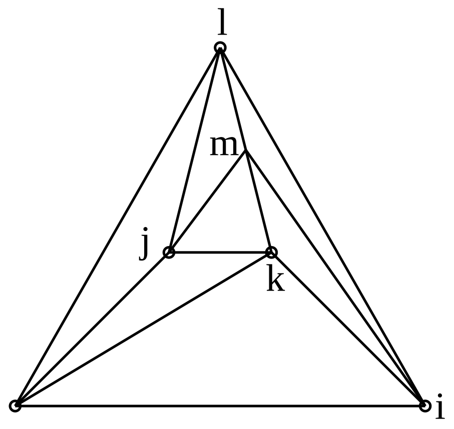

Figure 4: One drawing of K_5 with 1 crossing at m

The algorithm then backtracks with a new bound of one crossing. It will then back up through the rest of the tree and try all other parts of the tree. However, once the algorithm has determined that it cannot add an edge without any crossings, it will not try any further down that branch since the best drawing so far has one crossing. It will finish when it has exhausted all possibilities. In the case of K_5, no possibility will result in a drawing with zero crossings.

6 Conclusions and Future Work

We have presented a proposed algorithm for calculating the minimum crossing number of a graph. This could also be extended to finding the maximum crossing number by making minor modifications to the search algorithm. We could also find all possible drawings with the minimum crossing number, or we could remove the bound and enumerate all possible drawings of a graph.

The first step in the future work is to implement this algorithm and test it on some known graphs. Once that is accomplished, we would be able to determine if the order of the edges matters in reaching the correct answer more quickly. The order may be important for graphs in general, but it may not for certain classifications (such as K_n). Once that has been determined, we would like to calculate the crossing number for several families which are important in circuit design, such as K_n, $K_{(m,n)}$, and various others. The last item on the list (which would not be done last) would be to parallelize this process. This would greatly speed up the calculations since parallel DFS branch-and-bound algorithms can achieve super-linear speedup.

References

[1] L. Beineke and R.D. Ringeisen. On the crossing number of products of cycles and graphs of order four. *J. Graph Theory*, **4**:145–155, 1980.

[2] D. Bienstock. Some provably hard crossing number problems. *Disc. Comput. Geom.*, **6**:443–459, 1991.

[3] G. Chartrand and L. Lesniak. *Graphs and Digraphs.* Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 2nd. edition, 1986.

[4] J.E. Cottingham. *Thrackles, Surfaces, and Maximum Drawing of Graphs.* PhD thesis, Clemson, University, Clemson, SC 29634, August 1993.

[5] A.M. Dean and R.B. Richter. The crossing number of $C_4 \times C_4$. *J. Graph Theory*, **19**(1):125–129, 1995.

[6] W. Dyck. Beiträge zur analysis situs. *Math. Ann.*, **32**:457–512, 1888.

[7] J. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices Amer. Math. Soc.*, **7**:646, 1960.

[8] P. Erdös and R. Guy. Crossing numbers of graphs. In Y. Alavi, et al., editor, *Graph Theory and Applications*, pages 111–124. Springer-Verlag, New York, 1973.

[9] M.R. Garey and D.S. Johnson. Crossing number is NP-Complete. *SIAM J. of Alg. Disc. Meth.*, **4**:312–316, 1983.

[10] R. Guy, H. Harborth, B. Piazza, R. Ringeisen, and S. Stueckle. Crossings in the complete graph. preprint.

[11] R.K. Guy. A minimal problem concerning complete plane graphs. In Y. Alavi, D.R. Lick, and A.T. White, editors, *Graph Theory and Applications*, pages 111–124, Berlin, 1972. Springer-Verlag.

[12] R.K. Guy and T.A. Jenkyns. The torroidal crossing number of $K_{m,n}$. *J. Combin. Theory*, **6**:235–250, 1969.

[13] F. Harary, P.C. Kainen, and A.J. Schwenk. Torroidal graphs with arbitrarily high crossing numbers. *Nanta Math.*, **6**:58–67, 1973.

[14] L. Heffter. Über das problem der nachbargebiete. *Math. Ann.*, **38**:477–508, 1891.

[15] D.J. Kleitman. The crossing number of $K_{5,n}$. *J. Combin. Theory*, **9**:315–323, 1971.

[16] M. Klesc, R.B. Richter, and I. Stobert. The crossing number of $C_5 \times C_n$. *J. Graph Theory*, **22**(3):239–243, 1996.

[17] F.T. Leighton. *Complexity Issues in VLSI.* MIT Press, Cambridge, MA, 1983.

[18] C. Lovegrove. Crossing numbers of permutation graphs. Master's thesis, Clemson University, Clemson, SC 29634, May 1988.

[19] C. Lovegrove and R.D. Ringeisen. Crossing numbers of permutation graphs. *Congr. Numer.*, **67**:125–135, 1988.

[20] R.B. Richter and J. Siran. The crossing number of $K_{3,n}$ in a surface. *J. Graph Theory*, **21**(1):51–54, 1996.

[21] R.B. Richter and C. Thomassen. Intersection of curve systems and the crossing number of $C_5 \times C_5$. *Discrete Comput. Geom.*, **13**:149–159, 1995.

[22] R.D. Ringeisen and L.W. Beineke. The crossing number of $C_3 \times C_n$. *J. Combin. Theory, Ser. B*, **24**:134–136, 1978.

[23] C. Thomassen. Rectilinear drawings of graphs. *J. Graph Theory*, **12**:335–341, 1988.

[24] J.T. Thorpe and F.C. Harris, Jr. A parallel stochastic optimization algorithm for finding mappings of the rectilinear minimal crossing problem. *Ars Comb.*, **43**:135–148, 1996.

[25] P. Turan. A note of welcome. *J. Graph Theory*, **1**:7–9, 1977.

[26] J.W.T. Youngs. Minimal imbeddings and the genus of a graph. *J. Math. Mech.*, **12**:303–315, 1963.