A Time-Saving Region Restriction for Calculating the Crossing Number of K_n

Judith R. Fredrickson, Bei Yuan, Frederick C. Harris, Jr.
Department of Computer Science and Engineering
University of Nevada
Reno, NV 89557
{fredrick, bei, fredh}@cs.unr.edu

Abstract

In 1996 Harris and Harris presented an algorithm for calculating the minimum crossing number of a graph. This algorithm was implemented in parallel by Tadjiev and Harris in 1997. Since the algorithm presented builds huge search trees, the need to restrict the search space is important as graph sizes grow. This paper presents a technique for search space reduction for an algorithm calculating the crossing number of a complete graph. We introduce a restriction on the placement of a new edge in a "good" graph construction. Graph region examination plays a roll in the restriction process. Results show execution time for calculating the crossing number of K_7 and K_8 has decreased by an order of magnitude when employing this restriction.

Keywords: crossing number, complete graph, region restrictions

1 Introduction

The year was 1944, the location Hungary. Paul Turán documented a personal experience with the following description [4]:

> We worked near Budapest, in a brick factory. There were some kilns where the bricks were made and some open storage yards where the bricks were stored. All the kilns were connected by rail with all the storage yards. The bricks were carried on small wheeled trucks to the storage yards. All we had to do was to put the bricks on the trucks at the kilns, push the trucks to the storage yards, and unload them there. We had a reasonable piece rate for the trucks, and the work itself was not difficult; the trouble was only at the crossings. The trucks generally jumped the rails there, and the bricks fell out of them; in short this caused a lot of trouble and loss of time which was precious to all of us. We were all sweating and cursing at such occasions, I too; but nolens volens the idea occurred to me that this loss

of time could have been minimized if the number of crossings of the rails had been minimized. But what is the minimum number of crossings?

Turán had conceptualized and documented the crossing number problem. Known as Turán's Brick Factory Problem, Turán's query is considered the birth of the crossing number problem. Over the years, many have studied this easily stated problem, but little is known about general solutions. Garey and Johnson [6] proved that finding the minimum crossing number of a graph was NP-complete. Because of the difficulty of this problem, work turned away from finding the crossing number of a graph to sub-problems and other related works. Pach and Tóth [12] is a good resource for an overview of references and current open problems on crossing numbers for various graph families. An interesting discussion of the many different interpretations of the term 'graph drawing' can be found in [13].

Almost twenty years before the proof of the crossing number problem as NP-complete, Guy initiated the pursuit of the crossing number of the complete graph K_n [7]. A graph with n vertices is complete if every two of its vertices are adjacent. Our current work joins in Guy's pursuit. We pursue complete graphs because they are a well studied graph family. Some known answers exist upon which to test our theories.

We employ the algorithm proposed by Harris and Harris in [10]. The algorithm uses an exhaustive search to find the crossing number of a graph. As graph size increases, exhaustive searches can become computationally intensive–taking months or even years to complete on a single processor. Using a medium to large Beowulf cluster and a parallel queuing system, along with an updated definition of a good drawing, we have efficiently solved the crossing number problem for a complete graph less than nine vertices. Parallel implementation of the Harris and Harris algorithm has gone through several iterations and improvements in recent times [11, 14, 15, 18]. By harnessing the power of computer clusters and tightening the definition of a good graph drawing from a region perspective, the crossing number problem may be solved for larger graphs.

The remainder of this paper is laid out as follows: Section 2 presents notation and definitions, background information on crossing numbers for complete graphs. Section 3 gives an overview of Edmonds' Rotational Embedding Scheme and the Harris and Harris algorithm. Section 4 describes the new region restriction we use to aid in greatly reducing the search space size and presents results. Section 5 contains our conclusions and current and future work, including a proposed radical region restriction.

2 Background

2.1 Terminology and Definitions

We now informally define some terms from graph theory which will be used in the rest of the paper. These definitions are based on [4].

For our purposes, a *compact-orientable 2-manifold*, or simply a *surface*, may be thought of as a sphere or a sphere with some number of handles placed on it.

Definition 1 *The genus of a surface S is the number of handles on the surface.*

Definition 2 *The genus of a graph G, gen(G), is the smallest genus of all surfaces on which G can be embedded.*

Definition 3 *A graph G is embeddable on a surface S if it can be drawn on S such that any two edges that intersect do so only at a vertex of G mutually incident with them.*

Definition 4 *A graph is a planar graph if it can be embedded on the plane.*

Definition 5 *A planar graph that is embedded on the plane is called a plane graph.*

Definition 6 *A region of the plane graph G is a connected portion of the plane remaining after all curves and points corresponding to edges and vertices of G have been deleted.*

Definition 7 *A region is called a 2-cell region if any simple closed curve in that region can be continuously deformed or contracted in that region to a single point.*

Definition 8 *An embedding is a 2-cell embedding if all the regions in an embedding are 2-cell.*

Definition 9 *A good drawing of a graph G is one with the following properties:*

- *adjacent edges never cross,*
- *two nonadjacent edges cross at most once,*
- *no edge crosses itself,*
- *no more than two edges cross at a point of a plane, and*
- *the (open) arc in the plane corresponding to an edge of the graph contains no vertex of the graph.*

Definition 10 *The crossing number of a graph G, denoted $\nu(G)$, is the minimum number of edge crossings among all the good drawings of G in the plane.*

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[4]:

> Let G be a connected graph with n vertices and m edges with a 2-cell embedding on the surface of genus g and having r regions, then
>
> $$n - m + r = 2 - 2g$$

The definitions related to embeddings and Euler's Formula are vital to the Harris and Harris algorithm as it employs Edmonds' Rotational Embedding Scheme. We discuss both in Section 3.

2.2 Minimum Crossing Number

As stated, we are concentrating on the crossing number of K_n. First, we differentiate between the two general categories of crossing numbers for graphs. The crossing number, denoted $\nu(G)$, allows for edges that are nonlinear. Another category of crossing numbers is the rectilinear crossing number, denoted $\bar{\nu}(G)$, which is defined similarly but with the added constraint that each edge is a straight line segment.

For complete graphs of size 10 and less, a simple formula provides the crossing number [8]:

$$\nu(K_n) = \frac{\lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor \lfloor \frac{n-2}{2} \rfloor \lfloor \frac{n-3}{2} \rfloor}{4} \tag{1}$$

The rectilinear crossing number also holds for this formula, with the notable exception of graphs with 8 vertices. For K_8, the crossing number is 18; while the rectilinear crossing number is 19. As the crossing problem increases in size (11 vertices or more), a different formula appears to hold true for the rectilinear case:

$$\bar{\nu}(K_n) \leq \lfloor \frac{(7n^4 - 56n^3 + 128n^2 + 48n\lfloor \frac{(n-7)}{3} \rfloor + 108)}{432} \rfloor \tag{2}$$

Guy proposed this formula in 1972 [8] and conjectured that equality held. The equality conjecture was shown to be false when Thorpe and Harris [16] generated a rectilinear drawings of K_{12} with 155 crossings and K_{13} with 229 crossings. Recent work by Brodsky *et al.* [3] and Aichholzer *et al.* [1] proves that $\bar{\nu}(K_{10}) = 62$, and [2] proves $\bar{\nu}(K_{11}) = 102$ and $\bar{\nu}(K_{12}) = 153$.

Also in 1972, Guy [8] conjectured that equation 1 holds true for all n. To date this conjecture still stands. In private communication Guy anticipates that divergence around n of size 12 or 13 is possible as was seen in the rectilinear case [9].

3 An Overview of the Algorithm Used

As noted, we employ an exhaustive-search, branch-and-bound algorithm proposed by [10]. An overview of it is presented to allow for understanding of the region restriction presented in Section 3.

3.1 Edmonds' Rotational Embedding Scheme

We start with a look at the Rotational Embedding Scheme, first formally introduced by Edmonds [5] in 1960 and then discussed in detail by Youngs [17] a few years later. The following is the formal statement of the Rotational Embedding Scheme as presented in [4].

> Let G be a nontrivial connected graph with $V(G) = \{v_1, v_2, \ldots, v_n\}$. For each 2-cell embedding of G on a surface there exists a unique n-tuple $(\pi_1, \pi_2, \ldots, \pi_n)$, where for $i = 1, 2, \ldots, n$, $\pi_i : V(i) \to V(i)$ is a cyclic permutation that describes the subscripts of the vertices adjacent to v_i. Conversely, for each such n-tuple $(\pi_1, \pi_2, \ldots, \pi_n)$, there exists a 2-cell embedding of G on some surface such that for $i = 1, 2, \ldots, n$ the subscripts 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:

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

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) Go to permutation π_2 and find out who follows 1, and it is 4. Therefore the second edge is (2,4).
3) Go to permutation π_4 and find out who follows 2, and it is 1. Therefore the third edge is (4,1).

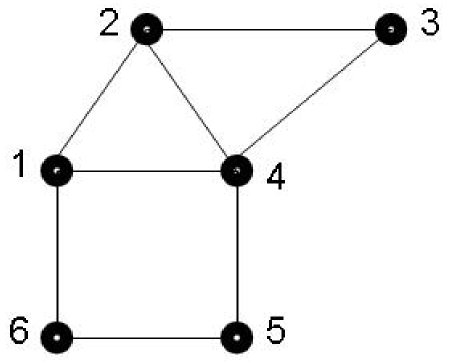

Figure 1: A Planar Embedding of a Graph

4) Go to permutation π_1 and find out who follows 4 and it is 2.
   This yields the original edge (1,2), so we are finished.

The region traced was bounded by the edges (1,2), (2,4), and (4,1). The other regions and edges can be found in a similar manner.

It is important to note the converse portion of the Rotational Embedding Scheme: every collection of vertex permutations corresponds to an embedding on some surface. Given a set of permutations, we can trace the edges and determine the genus of the surface.

3.2 The Harris and Harris Algorithm

In this algorithm [10] the solution space of the crossing number problem is mapped onto a tree. The tree is then searched for the crossing number with a branch-and-bound depth-first search (DFS). A DFS 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. Branch and bound allows us to change one small part of the DFS algorithm. When the cost to get to a vertex v exceeds the current optimal solution, we tell the DFS algorithm not to traverse the subtree having vertex v as its root. This saves us from having to cover a section of the search space that is guaranteed to cost more than the current optimal solution.

First, we begin with the vertex set for the graph in question and begin to add edges. After each edge is added, we determine whether the partial graph is still planar. This is done using the Rotational Embedding Scheme.

Once we cannot add any edges and keep the partial graph planar we are ready to begin our mapping to the tree.

At this stage the partial graph is the root of the tree. The root of the tree has a branch for each possible planar embedding. We select the first embedding and begin to build the rest of its tree. Regions compose the levels of the tree and edge segments through the regions are the branches to the next level. Tree building begins by considering laying down the next edge (which will go from vertex i to vertex j). The first option we have is through which one of the k regions to which vertex i is adjacent to should this edge should leave. These regions represent the next layer of our tree. Once the region is selected, the next option is which of the l edges of that region the edge will cross. When we have made this decision, we will create a cross vertex (degree 4) and place an edge from vertex i to the cross vertex and then try to lay the edge from the cross vertex to vertex j. This may be possible directly, or it may require more cross vertices.

For all the rest of the edges we lay them down in a similar manner, and when we have finally laid them all down we have reached a leaf in the tree. At this point we have a cost for the current solution which is the number of cross vertices we have added. This number of crossings becomes the new bound. The DFS continues by backing up and trying other paths through the tree using the bound as a stopping criteria. We proceed until the entire solution space is traversed. In order to understand this, we will walk through the algorithm with K_5 as our example. In Figure 2 we have the vertex set for the graph with all the edges that can be added to the graph while it remains planar.

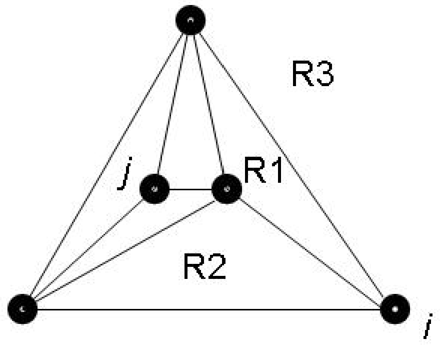

Figure 2: Planar Portion of K_5

At this stage the algorithm states that we are to take all remaining edges and try to lay them down one at a time. This is fairly simple in this case because there is only one edge left to be added, and that is from vertex i to vertex j. Next we choose a region through which to lay the edge. We have three choices, and these are the three regions to which vertex i is adjacent (R1, R2, and R3). We select R1 which has 3 edges. We cannot legally cross 2 of the edges (since they are incident with i) so there is only one choice. We then place a cross vertex on this edge and connect an edge from i to the cross vertex as shown in Figure 3. We then find out if we can draw the edge from the cross vertex to j and keep the graph planar. In this case we can and we are finished with this edge and have one crossing. This solution is shown in Figure 4. The algorithm then backtracks and tries the other regions to which i is adjacent and finds out that there are multiple ways to draw K_5 with one crossing, but none with zero crossings.

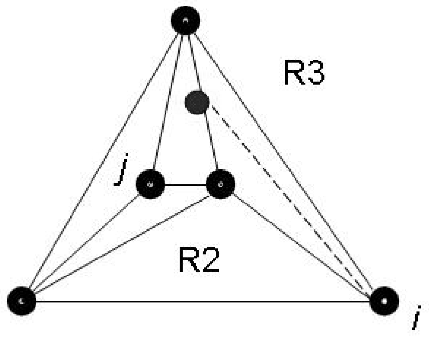

Figure 3: Beginning to Lay Down an Edge for K_5

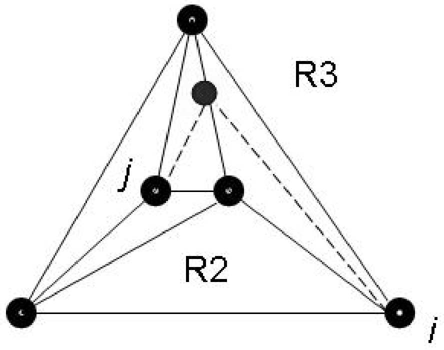

Figure 4: One Drawing of K_5 with One Crossing

4 Restricting a Good Drawing Through Region Analysis

Figure 5 shows four possibilities of how we can lay down an edge from u to v starting through region R1. They are all good drawings based on Definition 9. R1 has five edges, three of which are legal to cross. Recall that each edge adjacent to R1 not incident to vertex u will be used in generating all valid uv edges, so we are assured that all edge placements shown in Figure 5 will be generated. Figure 5(a) and (b) both illustrate placement of edge uv with one crossing, while Figure 5(c) and (d) have two and four crossings, respectively. Note that the new edge in both (c) and (d) exits R1 on the same edge and enters R3 on the same edge, but the edge in Figure 5 (d) travels less efficiently, creating more crossings than in (c). Thus the drawing in Figure 5 (d), though a good drawing, will not aid in producing a complete graph with a minimum number of crossings because the drawing in (c) is more efficient.

With this observation we add a further restriction to the definition of a good drawing to include a region restriction.

Let $f : u = p_0, p_1, ...p_n, p_{n+1} = v$ denote the edge laid down from two non-adjacent vertices u, v of G. Let $H = G + f$.

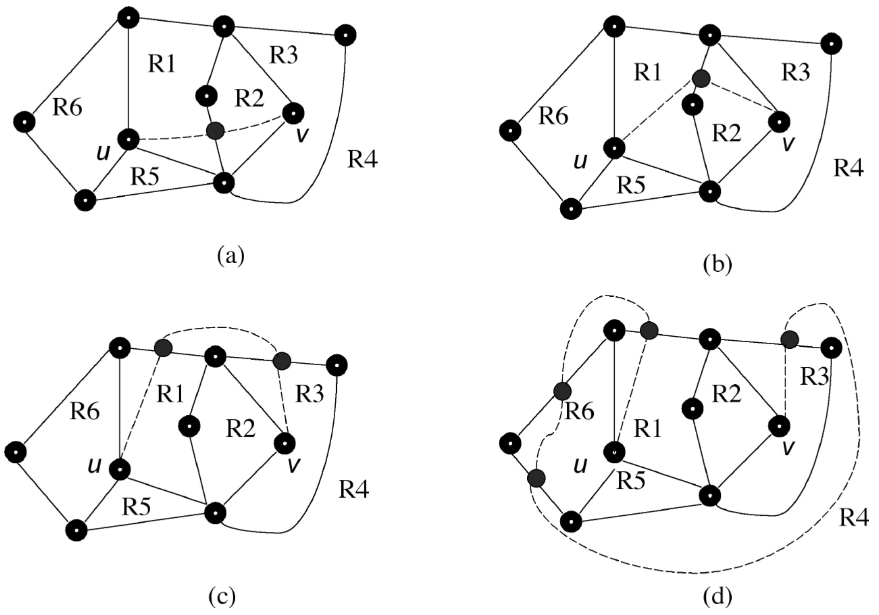

Figure 5: Good Drawings for Placement of Edge uv

The graph H is defined to be a *region restricted good drawing* if every region of H is adjacent to at most two vertices on f. This new restriction eliminates the graph in Figure 5 (d) from being generated.

Figure 6 shows the laying down of the six initial edge segments emanating from vertex u in laying of all valid edges from u to all other nonadjacent vertices. For any of the edges from u to v of G, the edges will be restricted from entering regions R1, R6 and R5 after laying down the initial edge segment. With each additional edge segment placed, the relevant regions will join the list of restricted regions for any given edge.

Modification to the graph generation algorithm was minimal. Whenever an edge segment is laid down, its initiating vertex is used to add to, or create, a list of restricted regions for that edge.

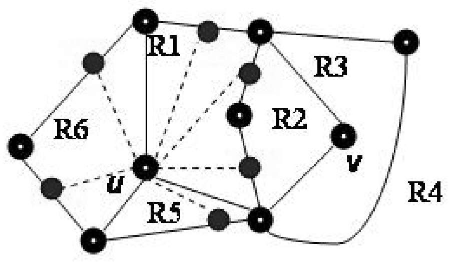

Figure 6: Initial Edge Segments From Vertex u to Other Nonadjacent Vertices

4.1 Results

Table 1 shows a comparison of the number of partial edges laid down when building K_n for $5 \leq n \leq 8$ with and without the restricted good graph. Tests were run on 6 parallel processors. For all n in the table, the restricted good drawing tests generated graphs with the true minimum crossing number.

A substantial search space reduction resulted as can been seen by comparing results for $n = 7$. The search space is reduced by a factor of nearly 75. Using the good drawing definition alone, we were unable to generate any complete graphs for $n = 8$ after over a week of runtime. With the restricted good drawing, we fully processed all jobs in under four hours. This time reduced to just over 2 hours when running on eight processors.

Tests were run on K_9, and results for the crossing number were not achieved. These results indicate that tighter restrictions on the search space

Vertices (n)	$\nu(K_n)$	**Good Drawing**	**Restricted Good Drawing**
5	1	3	3
6	3	203	71
7	9	1,498,775	19,979
8	18	*	46,697,854

Table 1: Good Drawing *versus* Restricted Edges Placed

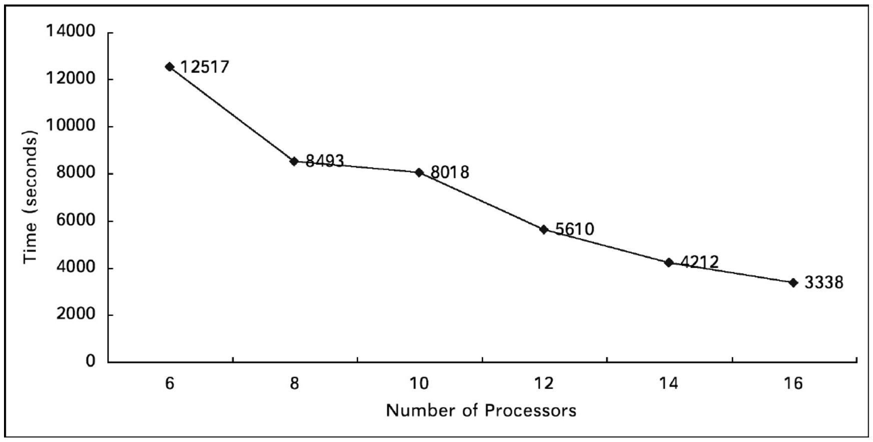

Figure 7: Execution Time for K_8 Run with a Number of Processors

will be needed. Figure 7 shows execution time of our algorithm run on K_8 with a variety of processors.

5 Conclusions and Future Work

5.1 Conclusions

We have presented a tightening of the definition of a good graph drawing for use with a parallel, branch-and-bound, exhaustive-search algorithm for finding the crossing number of complete graphs. The new definition examines graphs not only as sets of vertices and edges but also in terms of a set of regions. Implementation of the region restriction on the laying down of an edge during graph building has greatly reduced the search space of this problem for $n < 9$. This initial step in the direction of region examination for search space reduction is promising as it has prompted new questions related to even tighter region restrictions to be explored.

5.2 Future Work

The success of the restricted good graph for $n < 9$ has prompted a further restriction that is currently an open question. A radical region restriction is a greedy edge placement based on the analysis of the number of regions between vertices. The question we pose: If when laying down all edges from vertex u to vertex v of graph G, can we generate only those edges that are of the minimum region separation from u to v? For example, Figure 8 shows four of the ten possible uv edge placements for the given graph. Figure 8 (a) and (b) are the only two possible uv edge placements that have the minimum number of regions (in this case two) separating u and v. The minimum number of separating regions, being two, indicates that only one crossing will be generated in each of these cases. Figure 8 (c) and (d) show two other uv edge placements of the remaining eight, each of which separates u and v by three regions. A three region separation will introduce two crossings. Can we generate the two shortest path edges and ignore the rest? In other words: Is greedy good?

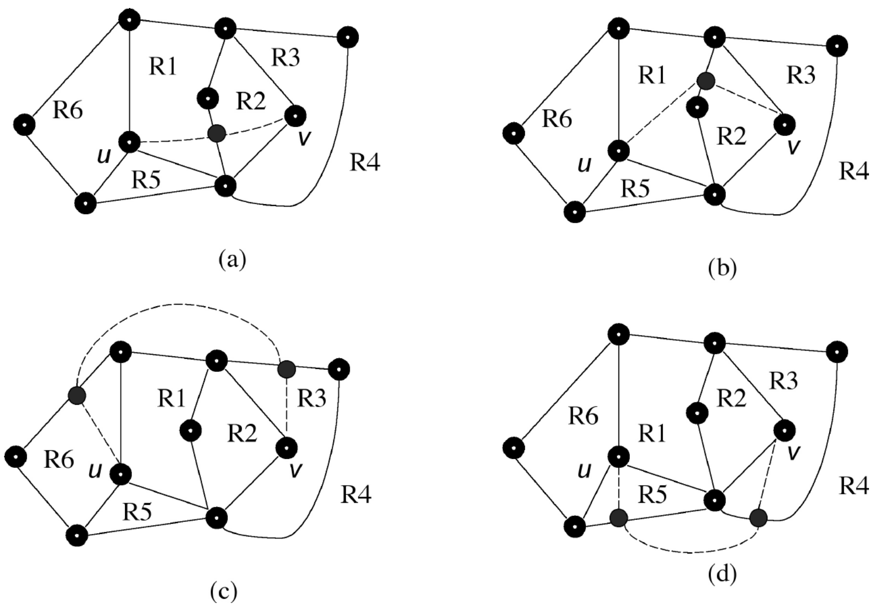

Figure 8: Edge Placement for Edge uv

Another issue is that of order. Does it matter in which order the edges are laid down? If order doesn't matter, we will be able to greatly prune the search tree of possible edge placements. We look to generate complete sets of crossing number graphs for small n and separate the results into non-isomorphic families. At this point, region restrictions and order questions can be tested to see if any of the families are lost. Once we know that non-isomorphic families are not missing, we can fine tune a tree pruning method that will not overlook the minimum crossing number for a graph.

Once we know that we are locating the crossing number for K_n with our pruning methods intact, we hope we can generate a counterexample to the conjecture, equation 1, posed by Guy [8] many years ago as the exact solution of this problem. It is anticipated that divergence may take place around $n = 12$ as it did in the rectilinear case[16].

Another avenue of exploration is trying to generate the crossing number of K_n using the non-isomorphic crossing number graph families of K_{n-1} as feeder graphs. A construction-based method for larger n may be attainable in the future.

References

[1] O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. In *Proc. 17th Ann. ACM Symp. Computational Geometry*, pages 11–18, 2001.

[2] O. Aichholzer, F. Aurenhammer, and H. Krasser. On the crossing number of complete graphs. In *Proceeding of the* 18^{th} *annual symposium on Computational Geometry*, Barcelona, Spain, June 2002.

[3] A. Brodsky, S. Durocher, and E. Gether. The rectilinear crossing number of k10 is 62. *The Electronic J. of Combinatorics*, 8, 2001. Research Paper 23.

[4] G. Chartrand and L. Lesniak. *Graphs and Digraphs.* Chapman & Hall/CRC, Boca Raton, FL, 3nd. edition, 1996.

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

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

[7] R.K. Guy. A combinatorial problem. *Nabla (Bulletin of the Malayan Mathematical Society)*, 7:68–72, 1960.

[8] R.K. Guy. Crossing numbers of graphs. In Y. Alavi, D.R. Lick, and A.T. White, editors, *Graph Theory and Applications*, pages 111–124, Berlin, 1972. Springer-Verlag.

[9] R.K. Guy, November 24, 1992. Private Communication with R.P. Pargas.

[10] Frederick C. Harris, Jr. and Cynthia R. Harris. A proposed algorithm for calculating the minimum crossing number of a graph. In Yousef Alavi, Allen J. Schwenk, and Ronald L. Graham, editors, *Proceedings of the Eighth Quadrennial International Conference on Graph Theory, Combinatorics, Algorithms, and Applications*, Kalamazoo, Michigan, June 1996. Western Michigan University.

[11] Sean C. Martin. A parallel queuing system for computationally intensive problems on medium to large beowulf clusters. Master's thesis, University of Nevada, Reno, Reno, NV 89557, December 2003.

[12] J. Pach and G. Toth. Thirteen problems on crossing numbers. *Geombinatorics*, 9:225–246, 2000.

[13] J. Pach and G. Toth. Which crossing number is it, anyway? *J. Combinatorial Theory B*, 80:225–246, 2000.

[14] Umid Tadjiev. Parallel computation and graphical visualization of the minmum crossing number of a graph. Master's thesis, University of Nevada, Reno, Reno, NV 89557, August 1998.

[15] Umid Tadjiev and Frederick C. Harris, Jr. Parallel computation of the minimum crossing number of a graph. In Michael Heath, Virginia Torczon, Greg Astfalk, Petter E. Bjorstad, Alan H. Karp, Charles H. Koelbel, Vipin Kumar, Robert F. Lucas, Layne T. Watson, and David E. Womble, editors, *Proc. of the* 8^{th} *SIAM Conf. on Parallel Process. for Sci. Comput.*, Minneapolis, Minnisota, March 1997. SIAM.

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

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

[18] Bei Yuan, Sean C. Martin, Judith R. Fredrickson, and Frederick C. Harris, Jr. A generic queuing system for computationally intensive problems. *Submitted*, May 2004.