A Linear Algorithm for Bipartite Drawing with Minimum Edge Crossings of Complete Binary Trees

Eliezer A. Albacea
Institute of Computer Science
University of the Philippines Los Baños
College, Laguna
eaalbacea@uplb.edu.ph

ABSTRACT
In this paper, we present a simple linear algorithm for bipartite drawing with minimum edge crossings of complete binary trees. Also obtained with the drawing algorithm is the formula for computing the bipartite crossing numbers of complete binary trees.

Keywords
Bipartite drawing, bipartite crossing number, graph drawing, complete binary tree.

1. INTRODUCTION
The problem of redrawing graphs on multiple layers with a small or minimum number of edge crossings frequently arises in graph drawing and design of VLSI as mentioned in Di Battista, et.al. [3] and Eades and Wormald [4]. When the number of layers is two, then the problem is known as bipartite drawing. Let $G = (V_0, V_1, E)$ be a connected bipartite graph, where V_0, V_1 is the bipartition of vertices into two independent sets. A bipartite drawing of G consists of placing the vertices V_0 and V_1 into distinct points on two parallel lines x and y, respectively, and then drawing each edge with one straight line segment which connects the points of x and y where the endvertices of the edge where placed. It is unavoidable in some cases that these edges cross. One of the aesthetic criteria for graph drawing is one with minimum edge crossings.

With regards to graph drawing, several efforts have been done in this area. Eades and Kelly [5] presented heuristics for drawing 2-layered networks. Eades, et.al. [7] presented an algorithm for drawing a hierarchical graph. Mutzel and Weiskircher [14] studied two-layer planarization in graph drawing. Yamaguchi and Toh [22] reported an algorithm for two-layered genetic network drawing with minimum edge crossings. Peng [15] presented algorithms for drawing graphs of bounded treewidth/pathwidth.

Related to the problem of bipartite drawing is the problem of computing the number of edge crossings in a bipartite drawing. The bipartite crossing number of a graph G, denoted by bcr(G), is the minimum number of edge crossings over all bipartite drawing of G. The problem of finding the bipartite crossing number was first studied in [9,10], and independently proposed by Watkins [21] as cited in Shahrokhi, et.al. [18]. The bipartite crossing number problem was shown to be NP-complete by Garey and Johnson [8]. However, it was shown to be solvable in polynomial time for bipartite permutation graphs by Spinrad, et. al. [14] and for trees by Shahrokhi, et. al. [19]. Several approaches have been proposed for computing the bipartite crossing number problem. Shahrokhi, et.al. [16] relates bipartite crossing numbers to edge isoperimetric inequalities and Laplacian eigenvalues of graphs. They applied their method to hypercubes and 2-dimensional meshes. Shahrokhi, et. al. [19] showed an intimate relationship between bipartite crossing number problem and the optimal linear arrangement. This was used to produced an $O(n^{1..6})$ time algorithm for computing the bipartite crossing number of trees. Shahrokhi, et.al. [18] developed a new lower bound argument using the Menger's Theorem which relates the bipartite crossing number of a graph to the edge connectivity properties of the graph.

When it comes to bipartite drawing, most of the efforts have been on edge crossing minimization using heuristics [5, 6, 11, 12, 13], although there are exact algorithms that draws bipartite drawing with minimum edge crossings [11, 12]. In this paper, we present a simple linear algorithm that draws directly a bipartite drawing with minimum edge crossings for the case of complete binary trees. We also obtained a formula for the computation of the bipartite crossing numbers of complete binary trees.

2. NOTATIONS AND DEFINITIONS
A *complete binary tree (CBT)* is a binary tree in which all leaves have the same level and all internal nodes have degree 2. Assume that the root has level equal to 1 and that the binary tree has height (maximum level) h. The number of nodes at level d is 2^{d-1} and the total number of nodes in a complete binary tree is $2^h - 1$. As a convention, the root is indexed 1. The left child of the root is indexed 2 and the right child indexed 3. Every even-indexed node v in the subtree rooted at 2 is positioned to the left of its sibling, i.e., v is a left child. On the other hand, every even-indexed node v in the subtree rooted at 3 (right child of the root), is

located to the right of its sibling, i.e. *v* is a right child. See Figure 1 for an example.

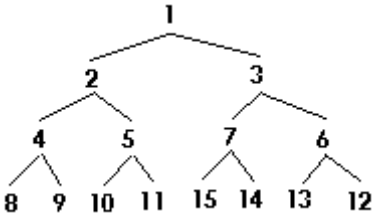

Figure 1. A labeled complete binary tree.

A *pseudotripod* is a set of four distinct nodes connected as in Figure 2a and 2b.

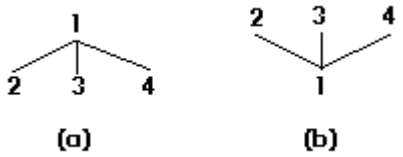

Figure 2. Pseudotripods.

In Figure 2, nodes 1, 2, 3 and 4 are the head, left foot, central foot and right foot, respectively. Edges (1,2), (1,3) and (1,4) are the left leg, central leg and right leg, respectively. Note that a pseudotripod is a bipartite graph.

To *pseudotripod(v)* means to construct a pseudotripod headed at node *v* taking the other nodes directly connected to *v* as the feet aligned apposite *v*. Given Figure 2a, doing a pseudotripod on each foot assuming each foot is directly connected to two nodes will result to a bipartite graph given in Figure 3.

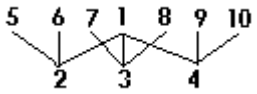

Figure 3. *Pseudotripod(v),* v = 2,3,4.

A pseudotripod is said to be *optimal* if its legs cross the minimum number of other legs. For example, proceeding from Figure 3, the resulting bipartite graph in Figure 4 is a result of an *optimal pseudotripod(7)*.

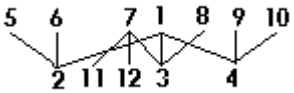

Figure 4. *Optimal pseudotripod(7).*

To minimize edge crossings, the algorithm processes the nodes from left to right. If a node *v* is a left foot, *optimal pseudotripod (v)* makes one of the nodes connected to *v* as the central foot and the other the left foot. The new feet are positioned in between the existing foot and the node found at the left (if it exists), that is, nearest the existing foot. If *v* is a central foot, then one of the nodes becomes the left foot and the other the right foot position. The left foot of *v* is positioned in between the existing foot and the node nearest the existing foot at the left and the right foot is positioned in between the existing foot and the node nearest the existing foot at the right. If *v* is a right foot, then one of the nodes connected to *v* becomes the central foot and the other the right foot. The new feet are positioned in between the existing foot and the node found at the right (if it exists) that is nearest the existing foot.

3. THE ALGORITHM

3.1 Description

The algorithm is an attempt to redraw any complete binary tree of height *h* into a two-layered network or bipartite graph with minimum number of edge crossings. The algorithm is outlined below.

Algorithm: CBT_to_Bipartite

1. Draw the children of the root such that one becomes the left foot and the other the right foot. See Figure 5

for this.

Figure 5. Drawing of the root and its children.

2. For each level of the tree from 2 to *h-1*

   For each vertex *v* do

   optimal pseudotripod(v)

Let us illustrate the execution of the algorithm on the CBT given in Figure 1.

First the root and its children are laid on the graph. This produce a configuration similar to Figure 5.

The for loop in the algorithm will be entered and the current level is 2. Doing optimal pseudotripod on each vertex will produce the bipartite graph in Figure 6.

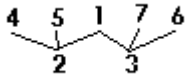

Figure 6. Optimal pseudotripod on level 2.

We repeat this process on level 3 producing the bipartite graph in Figure 7.

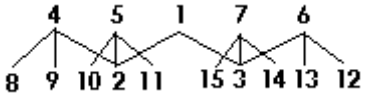

Figure 7. Optimal pseudotripod on level 3.

The resulting network has four edge crossings.

3.2 The Number of Edge Crossings

To count the number of edge crossings, each edge in the CBT will be associated with a label corresponding to the number of edges it crosses when the CBT is redrawn as a bipartite graph. For instance, the labels of the edges of the CBT in Figure 1 is given in Figure 8.

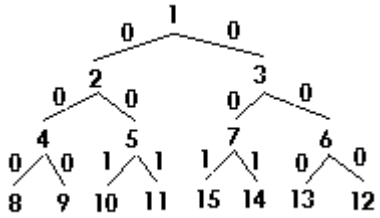

Figure 8. CBT with edge labels.

Let the level of an edge be equal to the level of the child node in the edge.

Let the left to right labels of the edges at level *d* on the subtree rooted at node 2 plus the edge ending at 2 be called the *left strand[d]* and the left to right labels of the edges at level *d* on the subtree rooted at node 3 plus the edge ending at 3 be called the *right strand[d]*.

left strand[1] = [-]

left strand[2] = [0]

left strand[3] = [0 0]

left strand[4] = [0 0 1 1]

The right strands are the reverse of the left strands.

The left strand at level *d* may be computed as follows:

1. copy the *left strand[d-1]*
2. add one to every entry in the *left strand[d-2]* and append to this the reverse of the resulting strand.
3. append the resulting strand in (2) to the resulting strand in (1).

For instance, to compute *left strand[5]*

[0 0 1 1]

[1 1] + [1 1] = [1 1 1 1]

[0 0 1 1 1 1 1 1]

and *left strand[6]*

[0 0 1 1 1 1 1 1]

[1 1 2 2] + [2 2 1 1]

[0 0 1 1 1 1 1 1 1 1 2 2 2 2 1 1].

Thus, the complete strands from levels 1 to level 6 are:

level	strand
1	[-]
2	[0 0]
3	[0 0 0 0]
4	[0 0 1 1 1 1 0 0]
5	[0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0]
6	[0 0 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 0 0]

From the algorithm for computing the strands, we can easily formulate the recurrence relation for the number of edge crossings for an *h*-level CBT when it is converted to a bipartite graph. This recurrence relation is as follows:

$$C(h)=\begin{cases}0, & h=1,2\\ C(h-1)+2C(h-2)+2^{h-1}-4, & h>2\end{cases}$$

The C(h-1) is the copying of the strand from level h-1, the 2C(h-2) is the strand for level h-2 and its reverse and the 2^{h-1} is sum of the ones added to each entry in level h-2 and its reverse.

This recurrence relation simplifies to:

$$C(h) = 2^h\left(\frac{h}{3}+\frac{1}{3}\right) - \left(\frac{14}{9}\right)2^h + (-1)^h\left(\frac{2}{9}\right) + 2$$
$$= 2^h\left(\frac{h}{3}-\frac{11}{9}\right) + (-1)^h\left(\frac{2}{9}\right) + 2$$

.

3.3 Edge Crossings is Minimum

Shahrokhi, et.al. [19] have shown the relationship between the linear arrangement problem and the bipartite crossing number problem. Let us use their result in proving that our drawing produces one with minimum edge crossings.

Given a graph G = (V,E) and a function f: V $\rightarrow$ R, define the length of f to be

$$L(f)=\sum_{uv\in E}|f(u)-f(v)|.$$

The linear arrangement problem is to determine a bijection f:V $\rightarrow$ {1,2, …, |V|} of minimum length. We denote this minimum length by L(G).

Shahrokhi, et.al. [19] showed that for a tree T, the bcr(T) can be expressed in terms of L(T). Specifically, they showed that

$$bcr(T) = L(T) - n + 1 - \sum_{v\in T}\left\lfloor \frac{d_v}{2} \right\rfloor\left\lceil \frac{d_v-2}{2} \right\rceil$$

where d_v is the degree of vertex v and n is the number of vertices in the tree. For a complete binary tree with the level of the root equal to 1 and the maximum level is h, this can be simplified to:

$$bcr(T) = L(T) - \left(2^h - 1\right) + 1 - \left(2^{h-1} - 2\right)$$
$$= L(T) - 2^h - 2^{h-1} + 4$$

.

Chung [1], on the other hand, have shown that for an h-level complete binary tree, where the root is level 1,

$$L(T) = 2^h\left(\frac{h}{3}+\frac{5}{18}\right) + (-1)^h\left(\frac{2}{9}\right) - 2.$$

Combining this with the result of Shahrokhi, et.al. [19], we obtain

$$bcr(T) = 2^h\left(\frac{h}{3}-\frac{13}{18}\right) - 2^{h-1} + (-1)^h\left(\frac{2}{9}\right) + 2$$
$$= 2^h\left(\frac{h}{3}-\frac{11}{9}\right) + (-1)^h\left(\frac{2}{9}\right) + 2$$

.

Note that

$$C(h) = bcr(T).$$

Hence, this proves our claim that our drawing produces minimum edge crossings.

3.4 Running Time

Doing a pseudotripod whether optimal or not can obviously be done in *O(1).* For an n-node CBT, we apply the pseudotripod *n/2* times. Hence, the total running time of the algorithm is *O(n)* which is linear.

4. CONCLUSIONS

We have presented a simple linear algorithm for bipartite drawing of a complete binary tree. The algorithm also produced a drawing with minimum edge crossings. Also presented is the formula for computing the bipartite crossing numbers of complete binary trees.

5. REFERENCES

[1] Chung, F.R.K. A conjectured minimum valuation tree, SIAM Review 20, 1978, 601-604.

[2] Chung, F.R.K. Some problems and results in labelings of graphs, In The theory and applications of graphs, Wiley, New York, 1980, 255-264.

[3] Di Battista, J., Eades, P., Tamassia, R. and Tollis, I.G. Algorithms for drawing graphs: an annotated bibliography, Comput. Geom. 4, 1994, 235-282.

[4] Eades, P. and Wormald, N. Edges crossings in drawings of bipartite graphs, Algorithmica 11, 1994, 379-403.

[5] Eades, P. and Kelly, D. Heuristics for drawing 2-layered networks, Ars Combinatorica 21-A, 1986, 89-98.

[6] Eades, P. and Wormald, N. The median heuristic for drawing 2-layered networks, Tech Report 69, University of Queensland, 1986.

[7] Eades, P., Lin, X., and Tamassia, R. An algorithm for drawing a hierarchical graph, International Journal of Computational Geometry and Applications 6, 1996, 145-155.

[8] Garey, M.R. and Johnson, D.S. Crossing number is NP-complete, SIAM J. Algebraic and Discrete Methods 4, 1983, 312-316.

[9] Harary, F. Determinants, permanents and bipartite graphs, Mathematical Magazine 42, 1969, 146-148.

[10] Harary, F. and Schwenk, A. A new crossing number for bipartite graphs, Utilitas Mathematica 1, 1972, 203-209.

[11] Junger, M and Mutzel, P. Exact and heuristic algorithms for 2-layer straightline crossing minimization, Graph Drawing 1995.

[12] Junger, M. and Mutzel, P. 2-layer straighline crossing minimization: performance of exact and heuristic algorithms, J. of Graph Algorithm and Applications 1, 1997, 1-25.

[13] Li, X.Y. and Stallman, M.F. New bounds on the barycenter heuristics for bipartite drawing, Graph Drawing 2001.

[14] Mutzel, P. and Weiskircher, R. Two-layer planarization in graph drawing, Proceedings ISAAC'98, in Lecture Notes in Computer Science 1533, Springer-Verlag, Berlin, 1998, 69-79.

[15] Peng, Z. Drawing graphs of bounded treewidth/pathwidth, MS Thesis, University of Auckland, 2001.

[16] Shahrokhi, F., Szekely, L.A. and Vrto, I. Bipartite crossing numbers of meshes and hypercubes, Proceedings Graph Drawing 1997 (GD '97), 1997, Rome, Italy, in Lecture Notes in Computer Science 1353, Springer-Verlag, Berlin, 37-46.

[17] Shahrokhi, F., Sykora, O., Szekely, L.A. and Vrto, I. On bipartite crossings, largest biplanar subgraphs, and the linear arrangement problem, Proceedings 5th Workshop on Algorithms and Data Structures (WADS '97), 1997, Halifax, Nova Scotia, Canada, in Lecture Notes in Computer Science 1272, Springer-Verlag, Berlin, 55-68.

[18] Shahrokhi, F., Sykora, O., Szekely, L.A. and Vrto, I. A new lower bound for bipartite crossing number with applications, DIMACS Technical Report 98-52, 1998.

[19] Shahrokhi, F., Sykora, O., Szekely, L.A. and Vrto, I. On bipartite drawings and the linear arrangement problem, SIAM J. Computing 30, No. 6, 2001, 1773-1789.

[20] Spinrad, J., Brandstadt, A. and Stewart, L. Bipartite permutation graphs, Discrete Applied Mathematics 19, 1987, 279-292.

[21] Watkins, M.E. A special crossing number for bipartite graphs: a research problem, Annals of New York Academy of Sciences 175, 1970, 405-410.

[22] Yamaguchi, A. and Toh, H. Two-layered genetic network drawings with minimum edge crossings, Genome Informatics 12, 2001, 456-457.