Pakistan Journal of Applied Sciences 2 (1): 17-23, 2002

Drawing Free Trees Inside Convex Regions Using Polygon Skeleton

Alireza Bagheri and Mohammadreza Razzazi
Software Systems Research and Development Laboratory
Department of Computer Engineering, Amir-Kabir University of Technology, Iran

Abstract : Using Simulated Annealing (SA) method for drawing graphs has appeared in the literature. Using this method directly to draw graphs inside a convex polygon leaves considerable amount of edge crossing. This is partly because it does not consider geometrical properties of polygons. In this paper we introduce a new algorithm, which by using geometrical properties of polygons guides SA method, and achieves a drawing that has much fewer edge crossings than the drawings of the previous known algorithms.

Keywords : Graph Drawing, Simulated Annealing, Polygon Skeleton

Introduction

Trees are known structures that have many applications. Hence drawing trees "nicely" has been investigated by many researchers. There are some aesthetics for nice drawing of graphs (trees) that are mentioned in the literature. Some of the most important aesthetics are: *minimizing number of edge crossing, minimizing number of bends per edge, increasing symmetry of drawing, maximizing amount of angles between two edges,* and *distributing vertices all over the region* (Purchase 1997) and (Chan 1999).

Most of the graph and tree drawing algorithms do drawing inside a rectangle. But in some applications it is needed to do drawing inside a polygon. For example consider we have a text which has some pictures of trees. Some times it will have especial effects to draw trees inside some polygons in the middle of the text than drawing them inside a rectangle. In this paper we consider drawing of trees inside a convex polygon. When size of trees or graphs increases some of the drawing algorithms can not do well, and drawing result is not so good. For example, in such case they can not guarantee planarity of drawing of planar graphs. The drawing result of algorithms that use clustering is much better than the drawing result of ones that do not use it (Behrens *et al.* 1997), (Brandenburg 1997), (Brandenburg *et al.* 1999), (Edachery *et al.* 1999), (Roxborough *et al.* 1997), (Sablowski *et al.* 1996) and (Tan *et al.* 1997). The algorithms that use clustering divide vertices of graphs into some groups based on some parameters, and draw vertices of the same groups near each other (Eades *et al.* 1996, 1st), (Eades *et al.* 1996, 2nd) and (Eades *et al.* 1999). Since our algorithm uses clustering, it has fewer problems when size of trees increases. Our algorithm draws edges of trees as a straight line, so we do not have any bend and this increases the readability of the drawing. In addition, our algorithm spreads vertices of trees all over the region by using polygon skeleton. To our knowledge, there is not any previous work which used polygon skeletons for graph drawing.

In section 2, polygon skeletons are explained briefly. In section 3, the simulated annealing method is described. In section 4, our algorithm is introduced. In section 5, drawing results of our algorithm is compared to results of a known algorithm which uses the simulated annealing method too. In section 6, conclusion is stated.

Polygon Skeletons: There are two types of skeleton for simple polygons, Medial Axis, and Straight Skeleton. Medial Axis of a given simple polygon, P, consists of all interior points whose closest point on the boundary of P is not unique (Chin 1995). While Medial Axis is a *voronoi-diagram-like* concept, Straight Skeleton is not defined using a distance function but rather by an appropriate shrinking process. Straight Skeleton is defined as the union of the pieces of angular bisectors traced out by polygon vertices during the shrinking process. (Aichholzer *et al.* 1996), (Aichholzer *et al.* 1995, 1st), (Aichholzer *et al.* 1995, 2nd), (Eppstein *et al.* 1998) and (Felkel *et al.*)

Straight Skeleton, in general, differs from Medial Axis, if P is convex then both structures are identical. Otherwise, the medial axis contains parabolically curved segments around reflex vertices of P, which are avoided by the straight skeleton. In this paper, we consider drawing trees inside convex polygons, and since Straight Skeleton and Medial Axis of convex polygons are identical, it does not make any difference which one we use. Since computing Medial Axis is more efficient than computing Straight Skeleton, we use Medial Axis as the skeleton of polygons. From now on every where we say polygon skeleton we mean Medial Axis of the polygon.

The skeleton of a given polygon, P, partitions the interior of P into n connected areas which we call them faces. Each face is related to just one edge of P. Bisector pieces are called arcs, and their endpoints which are not vertices of P are called nodes of the skeleton. When P is convex the structure is tree and arcs are straight line segments.

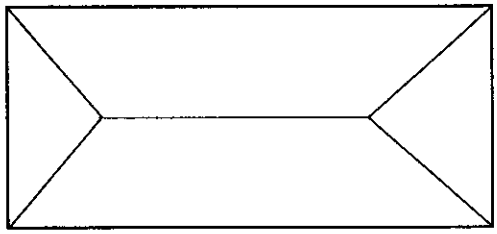

Fig. 1: Skeleton of a rectangle

Simulated Annealing: Simulated Annealing (SA) is a

flexible optimization method, suited for large-scale combinatorial optimization problems. SA differs from standard iterative improvement methods by allowing "uphill" moves - moves that spoil, rather than improve, the temporary solution. The problems for which SA is useful are characterized by a very large discrete configuration space, too large for an exhaustive search, over which an objective cost function is to be minimized (or maximized). After picking some initial configuration, most iterative methods continue by choosing a new configuration at each step, evaluating it and possibly replacing the previous one with it. This action is repeated until some termination condition is satisfied (e.g., no move reduces the objective function). The procedure ends in a minimum configuration, but generally it is a local minimum, rather than the desired global minimum. SA method tries to escape from these local minima by using rules that are derived from an analogy to the process in which liquids are cooled to a crystalline form, a process called *annealing*.

It is well known that when a liquid is cooled slowly, it reaches a totally ordered form, called *crystal,* which represents the minimum energy state of the system. In contrast, rapid cooling results in amorphous structures that have higher energy, representing local minima. The difference lies in the fact that when a liquid is cooled slowly, the atoms have time to reach a thermal equilibrium in each and every temperature. In this state, the system obeys the Boltzmann distribution:

$$P(E) \approx e^{-E/KT}$$

Here, P(E) specifies the probability distribution of the energy values of the states E, as a function of the temperature T and the Boltzmann constant K. On the one hand, for every temperature, each energy E has non-zero probability, and thus the system can change its state to one with higher energy. On the other hand, at low temperatures, the system tends to be in states with very low energy, with the global minimum achieved at temperature zero. (In the sequel, we shall assume that k=1, since our temperatures are artificial anyway.)

Metropolis *et al.* in 1953 devised an algorithm for simulating this annealing procedure by a series of sequential moves. The basic rule is that the probability with which the system changes its state from one with energy E1 to one with energy E2 is:

$$e^{-(E2-E1)/KT}$$

This rule implies that whenever the energy E2 of the new candidate state is smaller than the current energy E1 the system will take the move, and if it is larger the state change is probabilistic. Kirkpatrick *et al.* in 1983 were apparently the first to realize that the above procedure could be used for general optimization problems.

The schematic form of SA method is as follows:

Choose an initial configuration S and an initial temperature T;

Repeat the following (usually some fixed number of times):

(a) Choose a new configuration S' from the neighborhood of S;

(b) Let E and E' be the values of the cost function at S and S' respectively;

if E'<E or random(0,1) < $e^{(E-E')/T}$

then set S ← S';

Decrease the temperature T;

If the termination rule is satisfied, stop; otherwise go back to step 2.

SA method has been applied successfully to many problems. Its first application was to the traveling salesman problem. It has been used for many problems in the design and testing of VLSI, including the placement problem, channel routing, and the folding of PLAs, as well as for problems in image processing, coding theory, floor planning, graph and number partitioning, and graph coloring (Davidson *et al.* 1996). In (Davidson *et al.* 1996) SA method is used for drawing of graphs and trees nicely inside a rectangle. A cost function is defined which includes some aesthetics, and for each aesthetic one factor is dedicated in the cost function. Minimizing the cost function by SA method results in the aesthetics be satisfied.

The drawing results of the algorithm introduced in (Davidson *et al.* 1996), which we call it SA algorithm, are suitable for graphs and trees which are not large. But when number of vertices increases number of edge crossings increases too. For example, drawing of a complete binary tree with 63 vertices, by SA algorithm, usually has two edge crossings, while there are some algorithms which provide drawings with no edge crossing. Our algorithm does not have this weak point of SA algorithm

The Algorithm

In this section, after giving some definitions we introduce our algorithm (For simplicity, from now on we say trees to free trees, and polygon to convex polygons).

Definition 1: Let *FaceSet(j)* be set of all faces which include node j of the skeleton.

Definition 2: Let *CloserSet(j, k)* be set of all nodes of a tree whose paths to node j is shorter than to node k.

Here, we describe our algorithm briefly. The main layout of the algorithm is as follows:

Algorithm Draw-Free Trees

a. Computing the polygon skeleton and area of the faces.
b. Computing weight of the nodes of the skeleton.
c. Computing weights of the edges of the skeleton.
d. Computing weights of the edges of the tree.
e. Mapping the tree onto the skeleton.
f. Drawing the tree using SA method.

In the following we describe each step separately.

a. Computing the Polygon Skeleton and Area of the Faces: We Compute the polygon skeleton by using [Chin 1995]. Since all faces are simple polygons we can use the following formula to compute the area of the faces (Bourke 1988).

$$\frac{1}{2}\sum_{i=0}^{N-1}(X_i Y_{i+1} - X_{i+1} Y_i)$$

Here (Xj, Yj) is the coordination of vertex j (j=0..N-1) of the given face; Xn=X0 and Yn=Y0. To use the above formula, vertices of the faces should be ordered clockwise or counter clockwise.

b. Computing Weight of the Nodes of the Skeleton: For each node j of the skeleton, we compute the following sum as weight of node j:

In fact, this weight approximately represents amount of

$$Weight(j) = \sum_{f \in FaceSet(j)} Area(f)$$

area that is around the node.

c. Computing Weights of the Edges of the Skeleton: For each edge (j, k) of the skeleton, we compute the following pair of sums as weight of the edge:

$$Wjk = \sum_{m \in CloserSet(j,k)} Weight(m)$$

$$Wkj = \sum_{m \in CloserSet(k,j)} Weight(m)$$

$$Weight\ (j,k) = (Wjk, Wkj)$$

The weight of each endpoint approximately represents the amount of area that is near this endpoint.

d. Computing Weights of the Edges of the Tree: For each edge (j, k) of the tree, we compute the following pair as weight of the edge:
The weight of each endpoint represents the number of vertices of its Closer-set.

e. Mapping the Tree on the Skeleton: We consider

$$Weight\ (j,k) = (|\ CloserSet\ (j)\ |, |\ CloserSet\ (k)\ |)$$

the weighted skeleton and the weighted tree, and find an edge of the skeleton and an edge of the tree such that the difference of the weights of their endpoints is minimum (i.e. we find the middle edges). We map the middle edge of the skeleton to the middle edge of the tree. Then we omit these two middle edges from the skeleton and the tree, this divides the tree and the skeleton respectively into two sub-trees and two sub-skeletons. Then we update the weights of the edges of the two sub-skeletons and the two sub-trees. To do this, we decrease one of the weights of the edges of each sub-skeleton (sub-tree) by the total weight of the other sub-skeleton (sub-tree). In order to determine that the weight of which side of each edge should be decreased, consider each sub-skeleton (sub-tree) as a directed tree whose root is the endpoint of the omitted middle edge that is connected to this sub-skeleton (sub-tree). (Consider the direction of each edge from the father to the child). The weight of the beginning point of each edge should be decreased. After updating the weights of the sub-skeletons (sub-trees), we do the matching procedure recursively on the sub-skeletons and the sub-trees. The termination condition is that the number of the nodes of the sub-trees or the number of the edges of the sub-skeletons is less than one.
Some times, a skeleton or a tree may have more than one middle edge; it is easy to find that in this case these edges are common in one of their endpoint. We call this endpoint, the middle node. In this case, from the tree, we omit the middle node and its incident edges, so the tree is divided into two or more sub-trees. To update the weights of the edges of each sub-tree, we should decrease one of the weights of each edge of the sub-tree by the sum of the weights of all the other sub-trees plus one. But, from the skeleton, we omit just the middle node, so the skeleton is divided into two or more sub-skeletons. Then we update the weights of the edges of each sub-skeleton. To do this, we should decrease one of the weights of the edges of the sub-skeleton by sum of the weights of all the other sub-skeletons. In order to determine that the weight of which side of each edge should be decreased, consider each sub-skeleton (sub-tree) as a directed tree whose root is the middle node. (Consider the direction of each edge from the father to the child). The weight of the beginning point of each edge should be decreased. In any case, we divide the sub-skeletons and sub-trees into some possibly balanced groups, and for each group of the sub-skeletons, and its related group of the sub-trees, we call the matching procedure recursively. Therefore, we map the middles of the tree and the skeleton in four ways:

1. The middle node of the tree is mapped to the middle node of the skeleton .
2. The middle node of the tree is mapped to the middle edge of the skeleton.
3. The middle edge of the tree is mapped to the middle edge of the skeleton.
4. The middle edge of the tree is mapped to the middle node of the skeleton.

This mapping procedure fixes some nodes of the tree on some points of the skeleton. It is clear that if polygon is non-convex, there is the possibility of crossing between edges of the tree and edges of the polygon. This is the reason that our algorithm works only for convex polygons.

f. Drawing the tree Using SA Method: Now, the middle edges and the middle nodes of the skeleton, the sub-skeletons, the tree and the sub-trees have been found and have been mapped. The previous step divides the tree and the skeleton into some clusters, and maps the tree clusters on the skeleton clusters. Now, we use SA method to draw the tree inside the given convex polygon. During the running of SA method, we fix each middle node of the tree on the related middle node or the middle point of the related middle edge of the skeleton. We substitute each middle edge of the tree with a path of length 2 whose endpoints are the endpoints of the middle edge and its middle vertex is a virtual vertex. We fix each virtual vertex of the tree on the related middle node or the middle point of the related middle edge of the skeleton. After termination of SA method, we remove all the virtual vertices and return the substituted edges, then draw the tree. This matching procedure spreads the vertices of the tree all over the skeleton and so all over the drawing region. To clarify the algorithm, we follow an example. We want to draw the tree in Fig. 3 inside the rectangle in Fig. 2. Consider the rectangle and its skeleton in Fig. 2. In this example, the areas of the faces are:
Area(F0) = 2500, Area(F1) = 7500
Area(F2) = 2500, Area(F3) = 7500

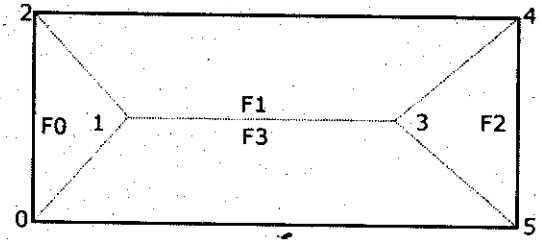

Fig. 2: A rectangle and its skeleton

Then the weights of the nodes are:
Weight(0) = Area(F0) + Area(F3) = 10000
Weight(1) = Area(F0) + Area(F3) + Area(F1) = 17500
Weight(2) = Area(F0) + Area(F1) = 10000
Weight(3) = Area(F1) + Area(F2) + Area(F3) = 17500
Weight(4) = Area(F1) + Area(F2) = 10000
Weight(5) = Area(F2) + Area(F3) = 10000
If we select the node zero as the root of the skeleton and apply DFS, the weights of the edges of the skeleton are:
Weight(1, 2) = (65000, 10000)
Weight(3, 4) = (65000, 10000)
Weight(3, 5) = (65000, 10000)
Weight(1, 3) = (37500, 37500)
Weight(0, 1) = (10000, 65000)
Where *Weight(i, j) = (k, m)* means the weight of endpoints i and j of edge (i, j) are respectively k and m.
Then, consider the tree in Fig. 3. If we select the vertex zero as the root of the tree, and apply DFS, the weights of the edges of the tree are:
Weight(0, 2) = (1, 6) Weight(2, 1) = (6, 1)
Weight(2, 5) = (3, 4) Weight(5, 4) = (4, 3)
Weight(4, 3) = (6, 1) Weight(4, 6) = (6, 1)

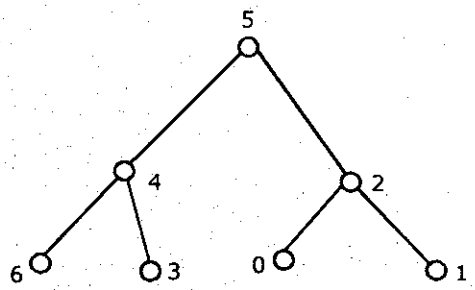

Fig. 3: A simple binary tree

Then, we should map the tree onto the skeleton. If we compute the difference of the weights of the two sides of each edge of the skeleton, we have:
DeltaW(1, 2)= 55000
DeltaW(3, 4)= 55000
DeltaW(3, 5)= 55000
DeltaW(0, 1)= 55000
DeltaW(1, 3) = 0
As can be seen, among these edges, just the edge (1, 3) has the minimum difference of weights. Therefore, we choose it as the middle edge of the skeleton.
Now, we compute the difference of the weights of the two sides of each edge of the tree:
DeltaW(0, 2)= 5 DeltaW(2, 5)= 1
DeltaW(4, 3)= 5 DeltaW(2, 1)= 5
DeltaW(5, 4)= 1 DeltaW(4, 6)= 5
Among these edges, two edges (2, 5) and (5, 4) have the minimum difference of weights. Because here there are two middle edges, we select their common nodes, node 5, as the middle node of the tree.
Therefore, edge (1, 3) of the skeleton is mapped to node 5 of the tree. We omit edge (1, 3) from the skeleton, so the skeleton is divided into two sub-skeletons, which is shown in Fig. 4.

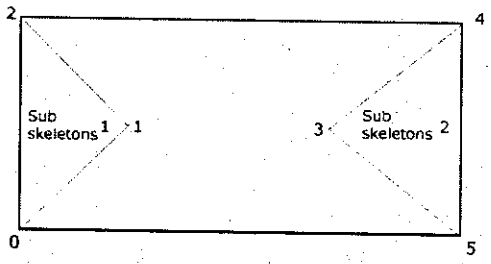

Fig. 4: sub-skeletons of a skeleton

Then, we omit the middle node of the tree and its incident edges, so the tree is divided into two sub-trees, which is shown in Fig. 5.

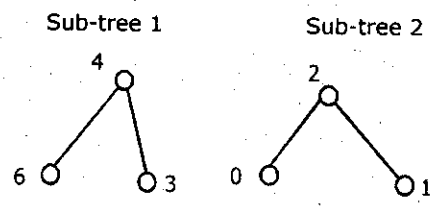

Fig. 5: Sub-trees of a tree

Now, we update the weights of the edges of the two sub-trees and the two sub-skeletons. To update the weights of two sub-skeletons, we consider the middle edge, edge (1, 3), and the two weights of its two sides. We decrease one of the weights of the edges of the sub-skeleton that is connected to node 1 by the weight of side 3, and one of the weights of the edges of the sub-skeleton that is connected to node 3 by the weight of side 1.
So, we update the weights of the edges of the sub-skeletons as follows:
Weight(1, 3) = (37500, 37500) _
W13=37500, w31 =37500
Weight(0, 1) = (10000, 27500)
Weight(1, 2) = (27500, 10000)
Weight(3, 4) = (27500, 10000)
Weight(3, 5) = (27500, 10000)
To update the weights of the edges of the two sub-trees, we consider the middle node, node 5. The total weight of these two sub-trees (the number of nodes) is 3. Then, for each sub-tree we decrease the weight of one side of each edge by sum of the weights of all the other sub-trees plus one (it is 4). So, we update the weights of the edges of the sub-trees as follows:
Weight(4, 6) = (2, 1)
Weight(4, 3) = (2, 1)
Weight(0, 2) = (1, 2)
Weight(2, 1) = (2, 1)

Then, we call the matching procedure recursively for the sub-tree connected to the node 4, and the sub-skeleton connected to the node 1, and also for the sub-tree connected to the node 2, and the sub-skeleton connected to the node 3. Then, node 4, and node 2 of the sub-trees are mapped respectively to the node 1, and node 3 of the sub-skeletons. If we continue, nodes 6, 3, 0, and 1 of the sub-trees are mapped respectively to edges (1, 2), (0, 1), (4, 3), and (3, 5) of the sub-skeletons, and the matching procedure terminates at this point.
At the end, we should draw the tree inside the rectangle using SA method. During running SA method, fixed nodes of the tree are node 5, which is fixed on the middle point of edge (1, 3) of the skeleton, nodes 4 and 2, which are fixed on nodes 1 and 3 of the skeleton respectively, nodes 6, 3, 0, and 1, which are fixed on the middle points of edge (1, 2), (1, 0), (3, 5) and (3, 4) of the skeleton. As can be seen all nodes of the tree are fixed and SA method does nothing here. The drawing result is as shown in Fig. 8.

Drawing Results
In this section, we compare the drawing results of our algorithm and SA algorithm, by some examples. First, consider the simple tree in Fig. 6.

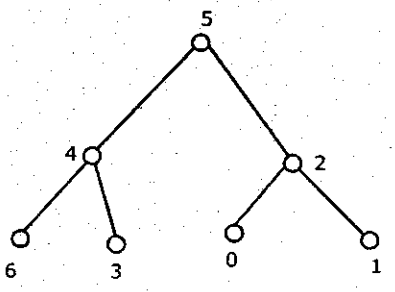

Fig.6: A complete binary tree with 7 node

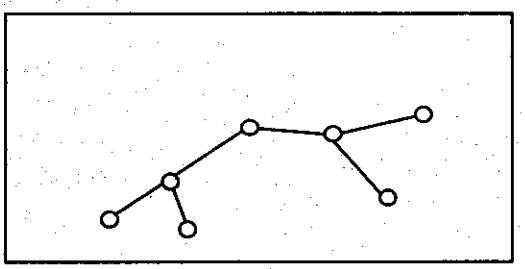

Fig. 7: Drawing of a tree with 7 nodes by SA algorithm

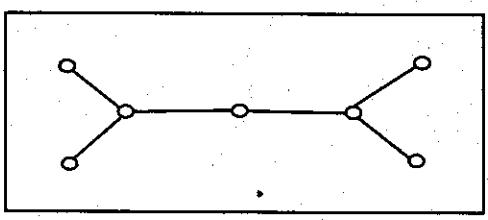

Fig. 8: Drawing of a tree with 7 nodes by our algorithm

If we draw this tree by SA algorithm inside a rectangle, the result is as shown in Fig. 7; the drawing result of our algorithm is as shown in Fig. 8. The drawing result of the same tree by SA algorithm inside a square is shown in Fig. 9, and the drawing result of our algorithm is shown in Fig. 10.
Now, consider the tree in Fig. 11. The drawing result of this tree by SA algorithm inside: a square is as shown in Fig. 12; the drawing result of our algorithm is shown in Fig. 13.

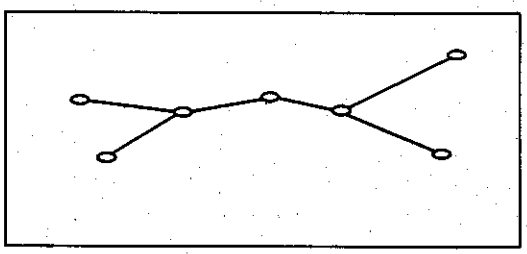

Fig. 9: Drawing of a tree with 7 nodes by SA algorithm

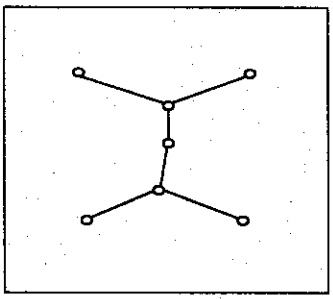

Fig. 10: Drawing of a tree with 7 nodes by our algorithm

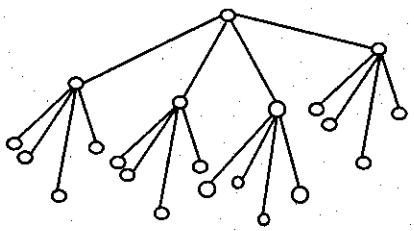

Fig. 11: A tree with 21 nodes

Now, let us to increase number of vertices of trees, for example, consider a complete binary tree with 63 vertices. The drawing result of this tree by SA algorithm inside a square is shown in Fig. 14.

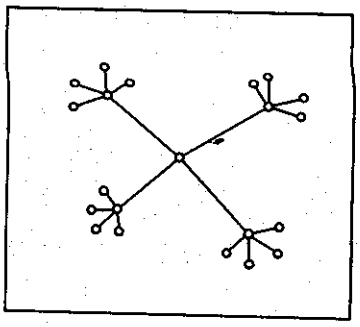

Fig. 12: Drawing of a tree with 21 nodes by SA algorithm

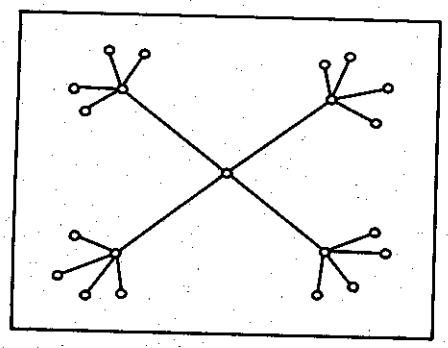

Fig. 13:Drawing of a tree with 21 nodes by our algorithm

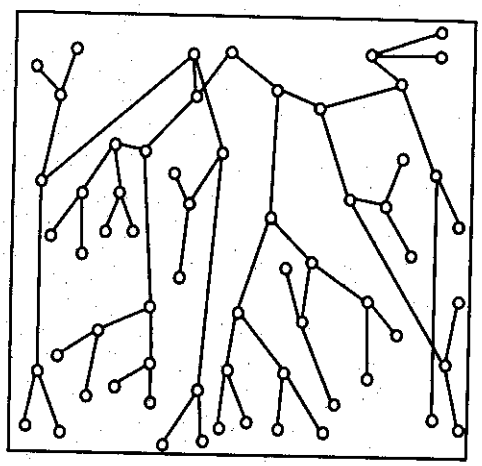

Fig. 14: Drawing of a complete binary tree with 63 nodes by SA algorithm

As can be seen from the figures, however the tree is planar, the drawing of it by SA algorithm is not planar. The drawing result of this tree by our algorithm is shown in Fig. 15.

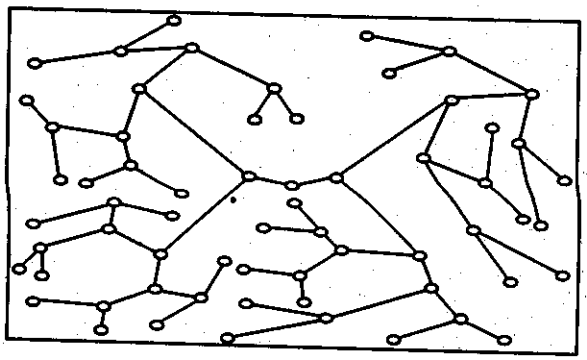

Fig. 15:Drawing of a complete binary tree by our algorithm

The mentioned examples and many other observations show that when size of graphs (trees) increases, SA algorithm does not draw well (i.e. there is some edge crossing in drawing of even planar graphs), while the drawing result of our algorithm is much better.
Now let us to represent some other examples of the drawing of our algorithm. Consider the tree in Fig. 16. The drawing result of this tree inside a square by our algorithm is as shown in Fig. 17.

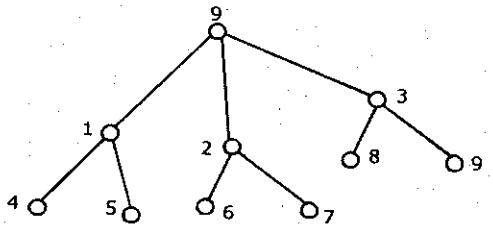

Fig. 16: A tree with 10 nodes

As can be seen, the skeleton of the square has four main branches while this tree has three main branches. The drawing result of the same tree inside a triangle by our algorithm is as shown in Fig. 18. Since, the skeleton has as many branches as the tree, the figure is more symetric than Fig. 17.

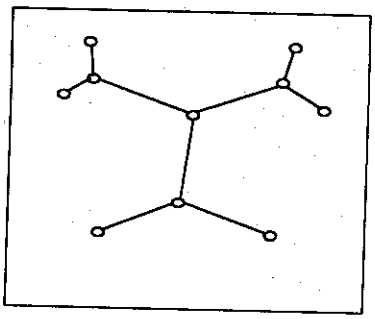

Fig. 17:Drawing of a tree with 10 nodes by our algorithm

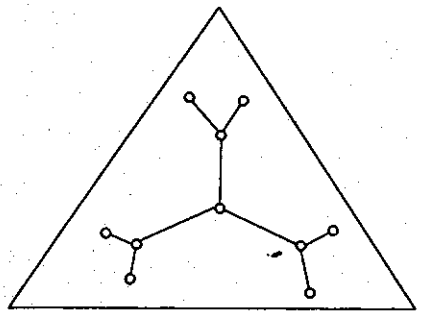

Fig. 18:Drawing of a tree with 10 nodes by our algorithm in a triangular region

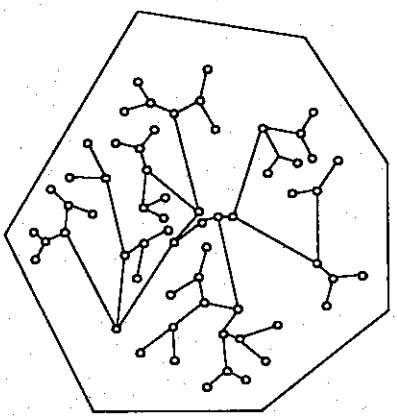

Fig. 19: Drawing of a binary tree with 63 nodes by our algorithm in a convex region

As the last example, the drawing of a complete binary tree with 63 nodes inside a convex region by our algorithm is as shown in Fig. 19. As can be seen, the tree is spreaded all over the convex polygon by our algorithm using matching procedure between the tree and the skeleton of the polygon.

Conclusion

In this paper we introduced a new algorithm that used skeleton of polygons and the simulated annealing method to draw free trees. The drawing results showed that our algorithm drew trees nicer than the previous known algorithms, even when trees were relatively large. The authors of this paper believe that to continue this work, we can find a better mapping method to map trees onto the skeletons. Perhaps, *pattern matching methods* can be used to improve the mapping and enhance the drawing results.

References

Aichholzer O. and F. Aurenhammer, 1996, "Straight Skeletons for General Polygonal Figures in the Plane", Proc. 2nd Ann. Int. Conf. Computing and Combinatorics (COCOON'96), Lecture Notes in Computer Science (LNCS)1090, Springer , pp. 117-126.

Aichholzer O., F. Aurenhammer, D. Alberts and B. Gartner, 1995, "Straight Skeletons of Simple Polygons", Proceedings of the 4th International Symposium at LIESMARS'95, Wuhan/China, Oct.

Aichholzer O., F. Aurenhammer, D. Alberts and B. Gartner, 1995, "A Novel Type of Skeleton for Polygons", J. Universal Computer Sci.1: 752-761, http://www.iicm.edu/jucs-1-12/ a-novel-type-of/html/paper.html.

Behrens D., E. Barke and R. Tolkiehn, 1997, "Design Driven Partitioning", 2nd Asian and South Pacific Design Automation Conference, Chiba, Japan.

Bourke P., 1988, "Calculating the area and centroid of a polygon", http://www.swin.edu.au/astronomy/pbourke/geometry/polyarea/.

Brandenburg F. J., 1997, "Graph Clustering I: Cycles of Cliques", Proceedings of Graph Drawing 97, LNCS 1353.

Brandenburg F. J. and A. Sen, 1999, "Graph Clustering II: Trees of Cliques with Size Bounds", (http://www.infosun.fmi.uni-passau.de/br/lehrstuhlMitarbeiter/Brandenburg/publikationen/).

Chan T. M., 1999, "A Near-Line Area Bound for Drawing Binary Trees", In Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms, pp. 161-168.

Chin F., J. Snoeyink and C. A. Wang, 1995, "Finding the Medial Axis of a Simple Polygon in Linear Time", Proc. 6th Ann. Int. Symp. Algorithms and Computation (ISAAC 95), Lecture Notes in Computer Science 1004, pp. 383-391.

Davidson R. and D. Harel, 1996, "Drawing Graphs Nicely Using Simulated Annealing", ACM Transactions on Graphics, 15:301-331.

Eades P. and Q. W. Feng, 1996, "Multilevel Visualization of Clustered Graphs", Symposium on Graph Drawing GD'96, ed. S. C. North, LNCS 1190, pp. 101-112.

Eades P., Q. W. Feng and X. Lin, 1996, "Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs", Symposium on Graph Drawing GD'96, ed. S. C. North, LNCS 1190, pp. 113-128.

Eades P., Q. W. Feng and H. Nagamochi, 1999, "Drawing Clustered Graphs on an Orthogonal Grid", J. Graph Algorithms and Applications, http://www.cs.brown.edu/publications/jgaa/, 3: 3-29.

Edachery J., A. Sen and F. J. Brandenburg, 1999, "Graph Clustering Using Distance-K Cliques", 7th International Symposium on Graph Drawing (GD'99), LNCS 1731, pp. 98-106.

Eppstein D. and J. Erickson, 1998, "Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions", 14th ACM Symposium on Computational Geometry, Minneapolis, pp. 58-67.

Felkel P. and S. Obdrzalek, "Straight Skeleton Implementation", Proceedings of Spring Conference on Computer Graphics, ISBN 80-223-0837-4, pp. 210-218, http://www.cgg.cvut.cz/Publications/felkel_sccg_98.ps.gz

Purchase H., 1997, "Which Aesthetic Has the Greatest Effect on Human Understanding?", 5th Symposium on Graph Drawing, Rome, Italy, LNCS 1353, pp. 248-261.

Roxborough T. and A. Sen, 1997, "Graph Clustering Using Multiway Ratio Cut", Proceedings of Graph Drawing Symposium GD'97, LNCS 1353, pp. 291-296.

Sablowski R. and A. Frick, 1996, "Automatic Graph Clustering", Proceedings of Graph Drawing GD'96, Berkeley, California, LNCS 1190, pp. 396-400.

Tan X., J. Tong, P. Tan, N. Park and F. Lombardi, 1997, "An Efficient Multi-Way Algorithm for Balanced Partitioning of VLSI Circuits", Proceedings of the International Conference on Computer Design (ICCD'97), IEEE, pp. 608-613.