Drawing Free Trees Inside Rectilinear polygons Using Polygon Skeleton

[Extended Abstract]

Alireza Bagheri
Software Systems Research and Development Laboratory
Department of Computer Engineering
Amir-Kabir University of Technology, Tehran, Iran
098-021-6018208
bagheri@ce.aku.ac.ir

Mohammadreza Razzazi
Software Systems Research and Development Laboratory
Department of Computer Engineering
Amir-Kabir University of Technology, Tehran, Iran
098-021-64542733
razzazi@ce.aku.ac.ir

ABSTRACT
In drawing a graph inside a polygon beside reduction of edge crossing, the avoidance of intersection of edges with sides of the polygon is of primary concern. In this paper we introduce a new algorithm for drawing free trees inside a rectilinear polygon by using properties of polygons to guide Simulated Annealing (SA) method.

General Terms
Algorithms

Keywords
Graph Drawing, Simulated Annealing, Polygon Skeleton required for Proceedings

1. 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 [8 and 17].

Most of graph drawing algorithms do drawing inside a rectangle. But in some texts it may have special effects to draw graphs inside general polygons in the middle of the text. In this paper we consider drawing of free trees inside a rectilinear polygon by means of straight skeleton of the polygon and simulated annealing (SA) method. We assume that the reader is familiar with SA method.

Drawing of large graphs by algorithms that use clustering usually has much fewer edge crossings and are nicer than drawing results of ones that do not use it [4, 6, 7, 14, 18, 19 and 20]. 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 [11, 12 and 13]. Our algorithm uses clustering and spreads vertices of trees all over the inner region of the given polygon by using straight skeleton of the polygon [1, 2, 3, 15, and 16]. We may have some bends in edges of the tree in our drawing.

In section 2, straight skeletons are briefly described. In section 3, our algorithm is introduced. In section 4, some drawing results of our algorithm are illustrated and compared to drawing results of [10], which uses SA method. In section 5, conclusion is stated.

2. STRAIGHT SKELETONS
There are two types of skeleton for simple polygons, medial axis, and straight skeleton. Medial axis of a simple polygon P, consists of all interior points whose closest point on the boundary of P is not unique [9]. 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 [1, 2, 3, 15, and 16].

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 rectilinear polygons, and to avoid parabolically curved segments we use straight skeleton as the skeleton of polygons. In the following, by polygon skeleton we mean straight skeleton 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 simple, the structure is tree and arcs are straight line
3. THE ALGORITHM

In this section, after giving some definitions we introduce our algorithm (From now on, for simplicity, we use trees for free trees, and polygon for rectilinear polygon).

Definition 1. Let FaceSet(j) be set of all faces that include node j of the given skeleton.

Definition 2. Let CloserSet(j, k) be set of all nodes of the given 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:

Free Trees Drawing Algorithm

a. Compute the polygon skeleton and area of the faces.

b. Compute weight of the nodes of the skeleton.

c. Compute weights of the edges of the skeleton.

d. Compute weights of the edges of the tree.

e. Do the mapping of the tree onto the skeleton.

f. Remove the crossings between edges of the tree and the border of the polygon.

f. Draw the tree using SA method.

In the following we describe steps of the algorithm.

a. Computing the polygon skeleton and area of the faces. We obtain straight skeleton of the given polygon by using [9]. Since all faces are simple polygons we can use the following formula to compute the area of the faces [3]:

\[
\frac{1}{2} \sum_{i=0}^{n-1} (X_i Y_{i+1} - X_{i+1} Y_i)
\]

Here \((X_i, Y_j)\) are the coordinates of vertex \(j (j = 3D0...N - 1)\) of the given face; \(X_n = 3DX_0\) and \(Y_n = 3DY_0\). To use the above formula, vertices of the faces should be ordered clockwise or counter clockwise.

b. Computing weights of the nodes of the skeleton. For each node j of the skeleton, we compute the following sum as the weight of node j:

\[
Weight(j) = 3D \sum_{f \in \text{FaceSet}(j)} \text{Area}(f)
\]

In fact, this weight approximately represents the amount of area which includes the node.

c. Computing weights of the edges of the skeleton. For each edge \((j, k)\) of the skeleton, we assign two weights, where each weight is associated to one of its endpoints.

\[
W_{jk} = 3D \sum_{m \in \text{CloserSet}(j, k)} \text{Weight}(m)
\]

\[
W_{kj} = 3D \sum_{m \in \text{CloserSet}(k, j)} \text{Weight}(m)
\]

\[
\text{Weight}(j, k) = 3D(W_{jk}, W_{kj})
\]

The weight of associated to an endpoint approximately represents the amount of area of the polygon which is closer to this endpoint than the other one.

d. Computing weights of the edges of the tree. For each edge \((j, k)\) of the tree, we associate a weight computed as follows:

\[
\text{Weight}(j, k) = 3D(|\text{CloserSet}(j, k)|, |\text{CloserSet}(k, j)|)
\]

e. Mapping the tree onto the skeleton. We consider the weighted skeleton (the weighted tree), and find an edge of the skeleton (the tree) such that the difference of the weights of its endpoints is minimum (i.e. we find the middle edge). 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 which weight of an edge needs to be decreased, we 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). (Considering the direction of each edge from the father to the child). The weight associated to the beginning point of each edge should be decreased. After updating the weights of the sub-skeletons (sub-trees), we do the mapping procedure recursively on the sub-skeletons and the sub-trees. The termination condition is satisfied when the number of the nodes of the sub-trees or the number of the edges of the sub-skeletons becomes less than one.

A skeleton or a tree may have more than one middle edge; it is easy to see that in this case these edges share an endpoint. We call this endpoint, the middle node. In this case, we omit from the tree the middle node and its incident edges, so the
tree is divided into two or more sub-trees. To update the weights associated to the edges of each sub-tree, we should
decrease one of the weights of each sub-tree edge by the
sum of number of nodes of all the other sub-trees plus one.
But, we omit from the skeleton 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 two weights of the
sub-skeleton edges, by sum of the weights of the edges of the
other sub-skeletons, which were incident to the deleted
node.

In order to determine that the weight of which endpoint
of each edge should be decreased, we consider each sub-
skeleton (sub-tree) as a directed tree whose root is the mid-
dle node. (Considering the direction of each edge from the
father to the child). The weight of the beginning point of
each edge should be decreased. In each case, we divide the
sub-skeletons and sub-trees into some possibly balanced
groups, and for each group of the sub-skeletons, and its re-
group of the sub-trees, we call the mapping procedure
recursively. Therefore, we do the mapping by using one of
the following ways:

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

In case 1, the node of the tree is mapped onto the node of
the skeleton. In case 2, the node of the tree is mapped onto
the middle point of the edge of the skeleton. In cases 3 and
4, first we substitute the edge of the tree with a path of
length 2 whose endpoints are the endpoints of the middle
eight and its middle vertex is a virtual vertex. Then the
virtual vertex of the tree is mapped onto the related middle
node or the middle point of the related middle edge of the
skeleton. Then cases 3 and 4 continues respectively as cases
2 and 1.

This mapping procedure provides a mapping list of the nodes
of the tree which are mapped onto the corresponding points
of the skeleton. This mapping list is used by SA method to
spread the vertices of the tree all over the drawing region.
After termination of SA method, all the virtual vertices are
removed and the substituted tree edges are restored.

f. Removing the crossings between edges of the tree
   and the border of the polygon. As the starting config-
tree by our algorithm is shown in Figure 5.

Figure 4. Drawing of a 63-nodes complete binary tree by SA algorithm

Figure 5. Drawing of a 63-nodes complete binary tree by our algorithm

In all the following examples, drawing of a 31-nodes complete binary tree inside some rectilinear polygons is presented. Figures 6, 7, and 8 illustrate the drawings of the tree respectively inside U-shape, T-shape, and S-shape rectilinear polygons by our algorithm.

Figure 6. Drawing of a 31-nodes complete binary tree by our algorithm inside a U-shape rectilinear polygon

Figure 7. Drawing of a 31-nodes complete binary tree by our algorithm inside a T-shape rectilinear polygon

Figure 8. Drawing of a 31-nodes complete binary tree by our algorithm inside a S-shape rectilinear polygon
5. CONCLUSION
In this paper we introduced a new algorithm that uses straight skeletons of polygons and simulated annealing method to draw free trees inside rectilinear polygons. The drawing results show that our algorithm draws trees nicer than the previous known algorithms, even when trees are relatively large, with respect to the aesthetics that we mentioned at the introduction section.

6. REFERENCES