Point-Set Embeddings of Plane 3-Trees

Rahnuma Islam Nishat, Debajyoti Mondal, and Md. Saidur Rahman

Graph Drawing and Information Visualization Laboratory,
Department of Computer Science and Engineering, Bangladesh University of
Engineering and Technology (BUET), Dhaka - 1000, Bangladesh.
nishat.buet@gmail.com, debajyoti_mondal_cse@yahoo.com, saidurrahman@cse.buet.ac.bd

Abstract. A straight-line drawing of a plane graph \( G \) is a planar drawing of \( G \), where each vertex is drawn as a point and each edge is drawn as a straight line segment. Given a set \( S \) of \( n \) points in the Euclidean plane, a point-set embedding of a plane graph \( G \) with \( n \) vertices on \( S \) is a straight-line drawing of \( G \), where each vertex of \( G \) is mapped to a distinct point of \( S \). The problem of deciding if \( G \) admits a point-set embedding on \( S \) is NP-complete in general and even when \( G \) is 2-connected and 2-outerplanar. In this paper, we give an \( O(n^2) \) time algorithm to decide whether a plane 3-tree admits a point-set embedding on a given set of points or not, and find an embedding if it exists. We prove an \( \Omega(n \log n) \) lower bound on the time complexity for finding a point-set embedding of a plane 3-tree. We then consider a variant of the problem, where we are given a plane 3-tree \( G \) with \( n \) vertices and a set \( S \) of \( k > n \) points, and present a dynamic programming algorithm to find a point-set embedding of \( G \) on \( S \) if it exists. Furthermore, we show that the point-set embeddability problem for planar partial 3-trees is also NP-complete.

Keywords. Point-set embedding, Plane 3-tree, Lower bound, NP-complete.

1 Introduction

A straight-line drawing \( \Gamma \) of a plane graph \( G \) is a planar drawing of \( G \), where each vertex is drawn as a point and each edge is drawn as a straight line segment. The problem of computing a straight-line drawing of a graph, where the vertices are constrained to be located at integer grid points is a classical problem in graph drawing literature [7, 16]. One of the variants of this problem is to compute a planar embedding of a graph on a set of points, where the points are located in the Euclidean plane [3, 10, 15].

Let \( G \) be a plane graph with \( n \) vertices and let \( S \) be a set of \( n \) points in the Euclidean plane. A point-set embedding of \( G \) on \( S \) is a straight-line drawing of \( G \), where each vertex of \( G \) is mapped to a distinct point of \( S \). We do not restrict the points of \( S \) to be in general position. In other words, three or more points in \( S \) may be collinear. Figure 1(a) depicts a plane graph \( G \) of ten vertices. Figures 1(b) and (c) depict two sets \( S \) and \( S' \) with ten points each. \( G \) admits a point-set embedding...
on \( S' \) as illustrated in Figure 1(d). But \( G \) does not admit a point-set embedding on \( S \) shown in Figure 1(b) since the convex hull of \( S \) contains four points whereas the outer face of \( G \) has three vertices.

A rich body of literature has been published on point-set embeddings when the input graph \( G \) is restricted to trees or outerplanar graphs. Given a tree \( T \) with \( n \) nodes and a set \( S \) of \( n \) points in general position, Ikebe et al. [9] proved that \( T \) always admits a point-set embedding on \( S \). In 1991 Gritzmann et al. gave a quadratic-time algorithm to obtain an embedding of an outerplanar graph \( G \) with \( n \) vertices on a set of \( n \) points in general position. Later, Castaneda and Urrutia [5] rediscovered the result in 1996. A recent improvement on this problem is given by Bose [3]; who presented an \( O(n \log^3 n) \) time algorithm to compute a straight-line embedding of an outerplanar graph \( G \) with \( n \) vertices on a set of \( n \) points. Cabello [4] proved that the problem is NP-complete for planar graphs in general and even when the input graph is 2-connected and 2-outerplanar. Recently, Garcia et al. gave a characterization of a set \( S \) of points such that there exists a 3-connected cubic plane graph that admits a point-set embedding on \( S \) [8].

![Figure 1.](image)

In this paper, we consider the problem of obtaining point-set embeddings of plane 3-trees. A plane 3-tree \( G \) with \( n \geq 3 \) vertices is a plane graph for which the following hold: (a) \( G \) is a triangulated plane graph; (b) if \( n > 3 \), then \( G \) has a vertex whose deletion gives a plane 3-tree \( G' \) with \( n-1 \) vertices. Any spanning subgraph of a plane 3-tree is a planar partial 3-tree. We give an \( O(n^2) \) time algorithm that decides whether a plane 3-tree \( G \) admits a point-set embedding on a given set \( S \) of \( n \) points or not; and computes a point-set embedding of \( G \) if such an embedding exists. We prove an \( \Omega(n \log n) \) lower bound on the time complexity for obtaining a point-set embedding of a plane 3-tree with \( n \) vertices on a set of \( n \) points. We then consider a variant of the problem, where we are given a plane 3-tree \( G \) with \( n \) vertices and a set \( S \) of \( k > n \) points, and present a
dynamic programming algorithm to find a point-set embedding of $G$ on $S$ if it exists. We describe an $O(nk^4)$-time implementation for that algorithm given by Moosa et al. [12]. Furthermore, we prove that the NP-completeness of point-set embeddability problem holds for planar partial 3-trees.

The rest of this paper is organized as follows. Section 2 presents some definitions and preliminary results. Section 3 gives an $O(n^2)$ time algorithm to obtain a point-set embedding of a plane 3-tree with $n$ vertices, if it exists. Section 4 shows an $\Omega(n \log n)$ lower bound on the running time for computing point-set embeddings of plane 3-trees. Section 5 gives a dynamic programming algorithm to decide whether a plane 3-tree $G$ with $n$ vertices admits a point-set embedding on a set of $k > n$ points. Section 6 proves that point-set embeddability problem is NP-complete for planar partial 3-trees. Finally, Section 7 concludes the paper suggesting future work. An early version of the paper has been presented at the 18th International Symposium on Graph Drawing (GD 2010) [14].

2 Preliminaries

In this section, we give definitions that will be used throughout the paper and present some preliminary results.

Let $G = (V, E)$ be a connected simple graph with vertex set $V$ and edge set $E$. The degree of a vertex $v$ is the number of edges incident to $v$ in $G$. We denote by $\text{degree}(v)$ the degree of the vertex $v$. A subgraph of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. If $G'$ contains all the edges of $G$ that join vertices in $V'$, then $G'$ is called the subgraph induced by $V'$. A graph $G$ is connected if for any two distinct vertices $u$ and $v$ there is a path between $u$ and $v$ in $G$, otherwise, $G$ is a disconnected graph. The connectivity $\kappa(G)$ of a graph $G$ is the minimum number of vertices whose removal results in a disconnected graph or a single-vertex graph. We say that $G$ is $k$-connected if $\kappa(G) \geq k$.

A tree is a connected graph without any cycle. A rooted tree $T$ is a tree in which one of the vertices, called root, is distinguished from the others. Every edge of $T$ is directed away from the root. If $v$ is a vertex in $T$ other than the root, the parent of $v$ is the vertex $u$ such that there is a directed edge from $u$ to $v$. When $u$ is the parent of $v$, $v$ is called a child of $u$. A vertex in $T$ is called a leaf if it has no children, any other vertex is internal. A descendant of $u$ is a vertex $v$ other than $u$ such that there is a directed path from $u$ to $v$. Let $i$ be any vertex of $T$. We then define a subtree $T(i)$ rooted at $i$ as the subgraph of $T$ induced by the vertex $i$ and all the descendants of $i$. A rooted tree $T$ is an ordered rooted tree if the children of any vertex of $T$ are ordered counter-clockwise.

A graph is planar if it can be embedded in the plane without edge crossings except at the vertices, where the edges are incident. We call such an embedding a plane embedding of the graph.
A plane graph is a planar graph with a fixed plane embedding. A plane graph divides the plane into some connected regions called the faces. The unbounded region is called the outer face and all other faces are called inner faces. The vertices on the outer face are called outer vertices and all other vertices are called inner vertices. If all the faces of a plane graph $G$ are triangles, then $G$ is called a triangulated plane graph.

We denote by $G_n$ a plane 3-tree with $n$ vertices. Examples of plane 3-trees are shown in Figure 2. Here, $G_6$ is obtained from $G_7$ by removing the inner vertex $c$ of degree three. Then $G_5$ is obtained from $G_6$ by deleting the inner vertex $b$ of degree three. $G_4$ is obtained by deleting the outer vertex $g$ of degree three and $G_3$ is obtained in a similar way.

![Figure 2. Examples of Plane 3-trees.](image)

We now have the following fact.

**Fact 1.** Every plane 3-tree $G_n$ with $n > 3$ vertices has exactly one inner vertex $p$ which is the common neighbor of all the three outer vertices of $G_n$.

We call $p$ the representative vertex of $G_n$. Let $a$, $b$, $c$ be the outer vertices of $G_n$. The vertex $p$, along with the three outer vertices $a$, $b$ and $c$, form three triangles $abp$, $bcp$ and $cap$. We call those three triangles the nested triangles around $p$. For a cycle $C$ in $G_n$, we denote by $G_n(C)$ the plane subgraph of $G_n$ inside $C$ (including $C$). We now define a tree $T_{n-3}$ with the inner vertices of $G_n$ which we call the representative tree of $G_n$. The representative tree of $G_n$ is an ordered rooted tree $T_{n-3}$ satisfying the following conditions.

(a) If $n = 3$, $T_{n-3}$ consists of a single vertex.
(b) If $n > 3$, then the root $p$ of $T_{n-3}$ is the representative vertex of $G_n$ and the subtrees rooted at the three counter-clockwise ordered children $p_1$, $p_2$ and $p_3$ of $p$ in $T_{n-3}$ are the representative trees of $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$, respectively, where $C_1$, $C_2$ and $C_3$ are the three nested triangles around $p$ in counter-clockwise order.

Notice that the subscript $n - 3$ in $T_{n-3}$ is the number of internal vertices in $T_{n-3}$. Figure 3(a) depicts a plane 3-tree $G_n$ and Figure 3(b) depicts the representative tree $T_{n-3}$ of $G_n$. 
The following lemma describes the properties of a representative tree [11].

**Lemma 1.** Let \( G_n \) be any plane 3-tree with \( n \geq 3 \) vertices. Then \( G_n \) has a unique representative tree \( T_{n-3} \) with exactly \( n-3 \) internal vertices and \( 2n-5 \) leaves. Moreover, \( T_{n-3} \) can be found in time \( O(n) \). Let \( T(i) \) be an ordered subtree rooted at a vertex \( i \) of \( T_{n-3} \). Then there exists a unique triangle \( C \) in \( G_n \) such that \( T(i) \) is the representative tree of \( G_n(C) \).

By Lemma 1, for any vertex \( p \) of \( T_{n-3} \), there is a unique triangle \( C \) in \( G_n \) such that \( T(p) \) is the representative tree of \( G_n(C) \). We denote this triangle \( C \) as \( C_p \) to show its correspondence with \( p \). For the rest of this paper, we shall often use an internal vertex \( p \) of \( T_{n-3} \) and the representative vertex of \( G_n(C_p) \) interchangeably.

### 3 Point-Set Embeddings of Plane 3-Trees

In this section, we give an \( O(n^2) \) time algorithm to decide whether a plane 3-tree \( G \) with \( n \) vertices has a point-set embedding on a set \( S \) of \( n \) points or not, and obtain a point-set embedding of \( G \) on \( S \) if it exists.

Before presenting the details of our algorithm we focus on some properties of the point-set embeddings of plane 3-trees. Let \( S \) be a set of \( n \) points in the Euclidean plane. The **convex hull** of \( S \) is the smallest convex polygon that encloses all the points in \( S \). Let \( G \) be a plane 3-tree with \( n \) vertices. In any point-set embedding of \( G \) the outer face of \( G \) is drawn as a triangle and hence the following fact holds.

**Fact 2.** Let \( G \) be a plane 3-tree with \( n \) vertices and \( S \) be a set of \( n \) points. If \( G \) admits a point-set embedding on \( S \), then the convex hull of \( S \) contains exactly three points of \( S \).
By Fact 2, \( G \) has no point-set embedding on \( S \) if the convex hull of \( S \) does not contain exactly three points of \( S \). We thus assume that the convex hull of \( S \) contains exactly three points. One may observe that the outer vertices of \( G \) can be mapped to the three points on the convex hull of \( S \) in six ways, and hence we need to check whether \( G \) admits a point-set embedding on \( S \) or not for each of those six mappings. In the remaining of this section we give an algorithm to check whether \( G \) admits a point-set embedding on \( S \) for a given mapping of the outer vertices of \( G \) to the three points on the convex hull of \( S \).

Let \( T_{n-3} \) be the representative tree of \( G_n \) and let \( p \) be the root of \( T_{n-3} \). Let \( C_{p_1}, C_{p_2}, C_{p_3} \) be the three nested triangles around \( p \). By Lemma 1, \( G_n(C_{p_1}), G_n(C_{p_2}) \) and \( G_n(C_{p_3}) \) are three plane 3-trees which have the corresponding unique representative trees \( T_{n_1}(p_1), T_{n_2}(p_2) \) and \( T_{n_3}(p_3) \), where \( n_1, n_2 \) and \( n_3 \) are the number of internal vertices of \( T_{n_1}(p_1), T_{n_2}(p_2) \) and \( T_{n_3}(p_3) \), respectively. Let \( C \) be a cycle in \( G_n \). We denote by \( \Gamma(C) \) the embedding of \( C \) on some points of \( S \). We call a mapping of \( p \) to a point \( x \in S \) a valid mapping of \( p \) if the proper interiors of \( \Gamma(C_{p_1}), \Gamma(C_{p_2}) \) and \( \Gamma(C_{p_3}) \) contain \( n_1, n_2 \) and \( n_3 \) points, respectively. Let \( R_{x_1}, R_{x_2} \) and \( R_{x_3} \) be the proper interiors of \( \Gamma(C_{p_1}), \Gamma(C_{p_2}) \) and \( \Gamma(C_{p_3}) \), respectively. Figures 4(a) and (b) illustrate a valid mapping of \( p \). We now have the following lemma.

**Lemma 2.** Let \( G \) be a plane 3-tree with \( n \) vertices, let \( a, b \) and \( c \) be the three outer vertices of \( G \), and let \( p \) be the representative vertex of \( G \). Let \( S \) be a set of \( n \) points such that the convex hull of \( S \) contains exactly three points. Assume that \( G \) has a point-set embedding \( \Gamma(G) \) on \( S \) for a given mapping of \( a, b \) and \( c \) to the three points on the convex hull of \( S \). Then \( p \) has a unique valid mapping for the given mapping of \( a, b \) and \( c \).

**Proof.** Since \( \Gamma(G) \) is a point-set embedding of \( G \) on \( S \), \( p \) has a valid mapping to a point \( x \in S \) such that \( R_{x_1}, R_{x_2} \) and \( R_{x_3} \) contain \( n_1, n_2 \) and \( n_3 \) points, respectively. Suppose for a contradiction that the mapping of \( p \) is not unique. Then \( p \) must have a valid mapping to a point \( z \in S, z \neq x \), such that \( R_{z_1}, R_{z_2} \) and \( R_{z_3} \) contain \( n_1, n_2 \) and \( n_3 \) points, respectively. Observe that for any subset of points that is collinear with one of the outer vertices, only the vertex that is closest to the respective outer vertex can be a choice for \( z \). It is now straightforward to observe that among \( R_{x_1}, R_{x_2} \) and \( R_{x_3} \), one must contain the point \( z \). Without loss of generality we assume that \( R_{x_1} \) contains \( z \). Then \( R_{x_1} \) must be a proper subset of \( R_{x_1} \) as depicted in Figure 4(c). Hence, the number of points in \( R_{x_1} \) and \( R_{z_1} \) cannot be \( n_1 \) simultaneously, a contradiction. \( \square \)

Based on Fact 2 and Lemma 2 we devise Algorithm 1, which we call **Point-set-embedding**. The Algorithm **Point-set-embedding** recursively finds valid mappings of the representative vertices.
Algorithm 1 Point-set-embedding

1: \{S is a set of n points and G_n is a plane 3-tree with n vertices\}
2: \{The representative vertex of G_n is p and \(C_p\) is a mapping of the outer face of \(G_n(C_p)\)\}
3: if \(G_n(C_p)\) is a triangle and the interior of \(C_p\) contains no points of \(S\) then
4: \quad return TRUE
5: end if
6: for each point \(u \in S\) interior to \(C_p\) do
7: \quad \{Assume that \(p\) is mapped to \(u\). Let \(C_{p_1}, C_{p_2}\) and \(C_{p_3}\) be the nested triangles around \(p\). Let \(T_{n_1}(p_1), T_{n_2}(p_2)\) and \(T_{n_3}(p_3)\) be the representative trees of \(G_n(C_{p_1}), G_n(C_{p_2})\) and \(G_n(C_{p_3})\)\}
8: \quad if the number of points interior to \(C_{p_1}, C_{p_2}\) and \(C_{p_3}\) are respectively \(n_1, n_2\) and \(n_3\) then
9: \quad \quad return \(\text{Embed}(C_{p_1}) \land \text{Embed}(C_{p_2}) \land \text{Embed}(C_{p_3})\)
10: \quad end if
11: end for
12: return FALSE

A naive approach to find a valid mapping of a representative vertex \(p\) is as follows. For each point \(u \in S\) interior to \(\Gamma(C_p)\), we assume \(u\) as the representative vertex and check whether \(\Gamma(C_{p_1}), \Gamma(C_{p_2})\) and \(\Gamma(C_{p_3})\) contains the required number of points. One can easily compute a valid mapping of \(p\) in \(O(n^2)\) time. Now, if we compute the mappings of the representative vertices for the plane 3-trees \(G_n(C_{p_1}), G_n(C_{p_2})\) and \(G_n(C_{p_3})\) in a recursive fashion, we can obtain a point-set embedding of \(G\). Since there are \(n - 3\) representative vertices, computation of the final embedding takes \(O(n) \times O(n^2) = O(n^3)\) time. In the rest of this section we give a faster method to find a valid mapping of a representative vertex and use it to implement Algorithm Point-set-embedding in \(O(n^2)\) time.

We first compute a convex hull of \(S\). If \(G\) admits a point-set embedding on \(S\), then the convex hull contains exactly three points of \(S\). For a given mapping of the three outer vertices \(a, b, c\) of
to the three points on the convex hull of $S$, we sort the points interior to the triangle $abc$ by increasing polar angle around $a$ in clockwise order. We then put the sorted list in an array $A_a$ as illustrated in Figure 5. For the points with the same slope, we keep a pointer to the point closest to $a$ from the other points with the same slope. The point closest to $a$ is the parent of the other points with the same slope. Let $x$ and $y$ be any two points, where $y$ has a pointer to $x$. Then $y$ is a child of $x$. We also maintain a count of the children of a parent in $A_a$. In a similar way we construct $A_b$ and $A_c$ sorted by polar angle around $b$ and $c$. Clearly, the construction of these three arrays takes $O(n \log n)$ time.

We next take any point $u \in S$, interior to the triangle $abc$, and draw straight lines through $(u, a)$, $(u, b)$ and $(u, c)$ which intersect $bc$, $ac$, $ab$ at $p$, $q$ and $r$, respectively as illustrated in Figure 6. Thus the region inside the triangle $abc$ gets split into six disjoint regions which we denote as $x_1, x_2, \ldots, x_6$. The regions $x_1, x_2, x_3, x_4, x_5, x_6$ are bounded by the triangles $aur, auq, cuq, cup, bup, bur$, respectively. Moreover, by $x_7, x_8$ and $x_9$ we denote the three lines shared by region $x_1$ and $x_6, x_2$ and $x_3, x_4$ and $x_5$, respectively. In the remaining of this section we also denote by $x_i$, $1 \leq i \leq 9$, the number of points of $S$ in region $x_i$.

Let $T_{n-3}$ be the representative tree of $G$ and let $p$ be the root of $T_{n-3}$. Let $n_1, n_2$ and $n_3$ be the number of vertices of the three subtrees rooted at the three children of $p$. We now formulate a set of nine linear equations and solve them under three constraints to check whether $u$ is a valid
mapping for $p$. The three constraints are $x_2 + x_3 + x_5 = n_1$, $x_4 + x_5 + x_6 = n_2$ and $x_1 + x_7 + x_6 = n_3$ which can be obtained easily. For the graph of Figure 5, the constraints are $x_2 + x_8 + x_3 = 4$, $x_4 + x_9 + x_5 = 3$ and $x_1 + x_7 + x_6 = 6$.

The nine equations can be obtained using the three straight lines $(u, a)$, $(u, b)$ and $(u, c)$ as follows. The straight line $(u, a)$ splits the triangle $abc$ into two disjoint regions $x_1 + x_5 + x_6 + x_7$ and $x_2 + x_3 + x_4 + x_8$. The number of points in those regions are $x_1 + x_5 + x_6 + x_7 = u_i - 1$ and $x_2 + x_3 + x_4 + x_8 = |A_a| - u_i - u_a$, where $u_i$ is the index of $u$ in $A_a$ and $u_a$ is the number of children of $u$ in $A_a$. The number of points on $x_9$ is equal to $u_a$ which gives another equation $x_9 = u_a$. Let $u_j$ and $u_k$ be the indices of point $u$ in $A_b$ and $A_c$, respectively and let $u_b$ and $u_c$ be the counts of children of $u$ in $A_b$ and $A_c$, respectively. We then can derive six other equations using $(u, b)$ and $(u, c)$ in a similar way as described above. The nine equations for any point $u$, interior to the triangle $abc$, are given below.

$$x_1 + x_3 + x_4 + x_8 = |A_a| - u_i - u_a, \quad x_9 = u_a,$$
$$x_5 + x_9 + x_4 + x_3 = u_j - 1, \quad x_2 + x_1 + x_7 + x_6 = |A_b| - u_j - u_b, \quad x_8 = u_b,$$
$$x_3 + x_8 + x_2 + x_1 = u_k - 1, \quad x_4 + x_9 + x_5 + x_6 = |A_c| - u_k - u_c, \quad x_7 = u_c.$$

When the vertex $g$ of Figure 5 is mapped to the point $u$ of Figure 6, the equations become $x_1 + x_5 + x_6 + x_7 = 8$, $x_2 + x_3 + x_4 + x_8 = 5$, $x_9 = 0$, $x_5 + x_9 + x_4 + x_3 = 3$, $x_2 + x_1 + x_7 + x_6 = 10$, $x_8 = 0$, $x_3 + x_8 + x_2 + x_1 = 7$, $x_4 + x_9 + x_5 + x_6 = 5$ and $x_7 = 1$.

If we get a unique solution of the set of linear equations, then $u$ is a valid mapping of $p$ by Lemma 2. A simple Gaussian elimination to solve a system of $t$ equations for $t$ unknowns requires $O(t^3)$ time. Here $t = 9$ and hence we can verify whether $u$ is a valid mapping of $p$ in $O(1)$ time.
Similarly, we check the points inside the triangle \(abc\), other than \(u\), to obtain a valid mapping of \(p\). Since there are \(O(n)\) points inside the triangle \(abc\), this step takes \(O(n) \times O(1) = O(n)\) time.

Finally, we find the valid mappings of representative vertices for the smaller plane 3-trees in a recursive manner and obtain a point-set embedding of \(G\). At each recursive step we construct three sorted arrays in \(O(n \log n)\) time and find the mapping of the representative vertex in \(O(n)\) time. Since there are \(O(n)\) representative vertices, computation of the final embedding takes \(O(n) \times (O(n \log n) + O(n)) = O(n^2 \log n)\) time.

We now recall that initially we computed a convex hull that takes \(O(n \log n)\) time and there are six ways for mapping the three outer vertices of \(G\) to the three points on the convex hull of \(S\). Therefore, the time taken for deciding whether \(G\) admits a point-set embedding on \(S\) is \((O(n \log n) + 6 \times O(n^2 \log n)) = O(n^2 \log n)\). However, we can remove the factor \(\log n\) from the time complexity as explained below.

One can observe that the bottle-neck of Algorithm Point-set-embedding is to construct three sorted arrays at each recursive step; which takes \(O(n \log n)\) time at each step and \(O(n^2 \log n)\) time in total. But if we assume that the points are in general position, we can construct the sorted arrays for all the points collectively at the initial step of the algorithm in \(O(n^2)\) time\(^1\) using arrangements [1]. At each recursive step we can update those arrays in such a way that the total number of updates after all the recursive steps becomes \(O(n^2)\). Thus, the time required to decide whether \(G\) admits a point-set embedding on \(S\) becomes \(O(n^2 + O(n \log n) + 6 \times O(n^2) = O(n^2)\).

The above argument holds even when we do not restrict the points to be in general position. Recall that while computing a sorted array \(A_a\), in case of a tie we only need to find the point closest to \(a\), where the closest point \(y\) is the parent of all other points with the same slope. For our algorithm, it suffices to place the children of \(y\) in \(A_a\) consecutively after \(y\) in any order. Since we do not need to sort the children of \(y\), each sorted array can be computed using arrangements in \(O(n)\) time. Hence, all the arrays can be computed in \(O(n^2)\) time. Even in this case, one can update those arrays at each recursive step in such a way that the total number of updates after all the recursive steps becomes \(O(n^2)\). Therefore, we have the following theorem.

\[^1\text{Let } S = \{p_1, p_2, \ldots, p_n\} \text{ be a set of } n \text{ points. Each point } p \in S \text{ in the primal space corresponds to a line } p'^* \text{ in the dual space. Consequently, } S \text{ corresponds to an arrangement of } n \text{ lines in the dual space, where each line is intersected by } n - 1 \text{ other lines. The order of the intersection points on } p'^* \text{ correspond to the sorted order of the slopes of the segments between } p \text{ and } S \setminus \{p\} \text{ in the primal space. This sorted slope sequence can be used to find the required sorted array for } p \text{ in } O(n) \text{ time.}]

10
Theorem 1. Given a plane 3-tree $G$ with $n$ vertices and a point-set $S$ of $n$ points, Algorithm **Point-set-embedding** computes a point-set embedding of $G$ on $S$ in $O(n^2)$ time if such an embedding exists.

We now analyze the time complexity of Algorithm **Point-set-embedding** in some restricted cases. Let $G$ be a plane 3-tree with $n$ vertices and let $T$ be the representative tree of $G$. Suppose that, for each internal vertex $u$ of $T$ the ratio of the number of vertices in the three subtrees rooted at three children of $u$ is $x : y : z$. Without loss of generality we assume that $x \geq y \geq z$. One can easily find that running time of the Algorithm **Point-set-embedding** can be written as $O(n) + T(n)$, where the term $O(n)$ is to obtain the embedding of the outer face of $G$ and the term $T(n)$ is the time to obtain the embedding of $n-3$ internal vertices of $G$. Since at each recursive step we take $O(n \log n)$ time to obtain a valid mapping of a representative vertex, $T(n)$ can be defined recursively as follows. $T(n) \leq T(x+y+z) + T(x+y+z) + cn \log n \leq 3T(n) + cn \log n$. Here $c$ is a constant hidden in $O(n \log n)$ term and $b = \frac{x+y+z}{x}$. We observe that, for $b = 3$, $T(n) = 3T(\frac{n}{3}) + cn \log n = O(n^2)$; for $b = 2$, $T(n) = 3T(\frac{n}{2}) + cn \log n = O(n^{1.58})$; and for $b = 3$, $T(n) = 3T(\frac{n}{3}) + cn \log n = O(n \log^2 n)$. Therefore, we obtain the following theorem.

Theorem 2. Let $G$ be a plane 3-tree with $n$ vertices and $S$ be a set of $n$ points. If the representative tree of $G$ is a complete ternary tree, it can be decided whether $G$ admits a point-set embedding on $S$ in $O(n \log^2 n)$ time.

4 Lower Bound

In this section, we reduce the sorting problem to prove an $\Omega(n \log n)$ lower bound on time complexity for computing a point-set embedding of a plane 3-tree with $n$ vertices on a set of $n$ points.

Theorem 3. The lower bound on the running time of the problem of computing point-set embeddings of plane 3-trees with $n$ vertices is $\Omega(n \log n)$.

**Proof.** We reduce sorting problems into the problem of computing point-set embeddings of plane 3-trees in the sense that point-set embedding algorithm can be used to solve sorting problems with little additional work.

Let $L = (x_1, x_2, \ldots, x_n)$ be a list of $n$ unsorted numbers to be sorted and let the smallest number in $L$ be $x_{\text{min}}$. Without loss of generality we assume that $x_i > 0$, $1 \leq i \leq n$, since if there exists an $x_i \leq 0$ we can obtain a list $L'$ of nonzero positive numbers by adding $1 - x_{\text{min}}$ to all $x_i$, $1 \leq i \leq n$. One can observe that the sorted order of the numbers in $L'$ yields a sorted order of the numbers
in $L$. Suppose that we have an algorithm $X$ that computes a point-set embedding of a plane 3-tree with $n$ vertices in $f(n)$ time. We show that Algorithm $X$ can be used to solve a sorting problem of $n$ numbers in time $f(n) + O(n)$, where the $O(n)$ represents additional time to convert the solution of $X$ to the solution of the sorting problem.

Let $x_{\text{max}}$ be the maximum number in $L$ which can be found in $O(n)$ time. We make a set $S$ of two-dimensional points $(x_i, x^2_i)$, $1 \leq i \leq n$, and let $S' = S \cup \{(x_{\text{max}} + 1, 0), (0, 0)\}$. Let $G$ be a plane 3-tree of $n + 2$ vertices such that its representative tree $T$ has the following properties. (a) The left child and the right child of each internal vertex of $T$ are leaves. (b) The subgraph induced by the internal vertices of $T$ is a path of $n-1$ vertices. Figures 7(a), (b) and (c) illustrate $S'$, $G$ and $T$, respectively. We now use Algorithm $X$ to compute a point-set embedding of $G$ on $S'$.

In a point-set embedding $\Gamma(G)$ of $G$ on $S'$, the outer vertices of $G$ are mapped to the convex hull of $S'$ by Fact 2. The convex hull of $S'$ contains the points $(0, 0)$, $(x_{\text{max}} + 1, 0)$ and $(x_{\text{max}}, x^2_{\text{max}})$ which we denote by $a$, $b$ and $c$, respectively. Let the representative vertex $p$ of $G$ be mapped to a point $z \in S'$ in $\Gamma(G)$ and let the proper interiors of the triangles $abz = C_{p_1}$, $bcz = C_{p_2}$, $caz = C_{p_3}$ be $R_{z_1}$, $R_{z_2}$, $R_{z_3}$, respectively. Since $p$ is also the root of $T$ with two leaves, two of the regions $R_{z_1}$, $R_{z_2}$ and $R_{z_3}$ do not contain any point of $S$. One can observe that such two regions can be obtained if $z$ is the second smallest (or the second largest) number of $L$. Suppose that the region $R_{z_2}$ (or $R_{z_1}$) contains all the points of $S'$ other than $a$, $b$, $c$ and $z$. We now consider the point-set embedding of $G(C_{p_2})$ (or $G(C_{p_1})$). Let $p_1$, $p_2$ and $p_3$ be the left, middle and right child of $p$ in $T$. If the children are leaves, we have no vertices left to consider. Otherwise, $p_2$ is an internal vertex. Since the left child and the right child of $p_2$ are leaves, $p_2$ must be mapped to the next smallest number or (next largest number) to ensure two regions which do not contain any points of $S'$, maintaining the plane embedding of $G$.

Thus the sequence of $x$-coordinates of the mappings of the internal vertices of $T$ from the root gives an increasing (or decreasing) order of the numbers in $L$. Moreover, we can check whether the obtained order is increasing or decreasing in constant time. Thus, we can use Algorithm $X$ to solve the sorting problem which implies that the lower bound of Algorithm $X$ is equal to the lower bound $\Omega(n \log n)$ on the running time of sorting problems. $\square$

5 Generalized Case

In this section, we consider the problem of computing a point-set embedding of a plane 3-tree $G$, when the number of given points is greater than the number of vertices of $G$. 
A brute force approach to solve this problem is to try all possible mappings of the vertices of \( G \) to the given points and to check whether for each mapping the embedding has any edge crossing or not. If the total number of vertices is \( n \) and the number of given points is \( k > n \), then at most \( kC_n \times n! \) different mappings are possible. Clearly, this approach is impractical for large \( n \) and \( k \).

We now present a dynamic programming technique to solve the problem. Here we formally define the input and output of the problem _Feasibility Checking_.

**Problem:** Feasibility Checking.

**Input:** A plane 3-tree \( G \) and a mapping of the three outer vertices \( a, b \) and \( c \) of \( G \) to three different points of \( S \).

**Output:** If \( G \) is drawable with the given mapping of \( a, b \) and \( c \), then the output is _True_. Otherwise, the output is _False_.

We denote the mapping of a vertex \( v \) to a point \( w \in S \) by \( v_w \). We denote by \( F_p(a_x, b_y, c_z) \) the Feasibility Checking problem of any vertex \( p \) of \( T \), where the three outer vertices \( a, b \) and \( c \) of \( G(C_p) \) are mapped to points \( x, y \) and \( z \), respectively. We solve this decision problem by showing that the optimal solution of the problem consists of the optimal solutions of the subproblems. The following lemma proves this optimal substructure property.

**Lemma 3.** Let \( G \) be a plane 3-tree with the representative tree \( T \). Let \( p \) be any internal vertex of \( T \) with the three children \( p_1, p_2, p_3 \) in \( T \) and let \( a, b, c \) be the outer vertices of \( G(C_p) \). Then the solution to the Feasibility Checking problem of \( p \) is _True_ if and only if the solutions to the Feasibility Checking problems of \( p_1, p_2 \) and \( p_3 \) are _True_.

---

*Figure 7. Illustration for the proof of Theorem 3.*
Proof. Let \( G(C_{p_1}), G(C_{p_2}) \) and \( G(C_{p_3}) \) be three plane 3-trees with the representative vertices \( p_1, p_2 \) and \( p_3 \) and the outer vertices \( \{a, b, p\}, \{b, c, p\} \) and \( \{c, a, p\} \), respectively. By definition, \( F_{p_1}(a_x, b_y, p_w), F_{p_2}(b_y, c_z, p_w) \) and \( F_{p_3}(c_z, a_x, p_w) \), for some \( \{w, x, y, z\} \in S \), are the Feasibility Checking problems of \( p_1, p_2 \) and \( p_3 \), respectively. The vertex \( p \) is an inner vertex of \( G \) and therefore, \( p \) must be mapped to a point of \( S \) inside the triangle \( xyz \). Since the mappings of \( a, b, c \) are preassigned and the mapping of \( p \) is the same in the embeddings \( \Gamma'_p, \Gamma''_p \) and \( \Gamma'''_p \), those three embeddings can be combined to get the embedding \( \Gamma'_p \) of \( G(C_p) \). Thus the solution of the Feasibility Checking problem of \( p \) consists of the solutions of the Feasibility Checking problems of \( p_1, p_2 \) and \( p_3 \) and hence, the solution to the Feasibility Checking problem of \( p \) is \text{True} if and only if the solutions to the Feasibility Checking problems of \( p_1, p_2 \) and \( p_3 \) are \text{True}. \( \square \)

We can readily find the overlapping subproblems property of the Feasibility Checking problem. Overlapping subproblem occurs when a recursive algorithm visits the same problem more than once. Figure 8 illustrates an example where the same subproblem \( F(e_x, f_y, b_z) \), for some \( \{x, y, z\} \in S \), occurs in two different embeddings.

We now prove Theorem 4 which states a recursive solution of Feasibility Checking problem.

**Theorem 4.** Let \( G \) be a plane 3-tree with the representative tree \( T \) and let \( p \) be any vertex of \( T \). Let \( a, b, c \) be the three outer vertices of \( G(C_p) \) and \( p_1, p_2, p_3 \) be the three children of \( p \) when \( p \) is an internal vertex of \( T \). Let \( F_p(a_x, b_y, c_z) \) be the Feasibility Checking problem of \( p \), where \( x, y \) and \( z \) are three points in \( S \). Then \( F_p(a_x, b_y, c_z) \) has the following recursive formula.

\[
F_p(a_x, b_y, c_z) = \begin{cases} 
\text{False} & \text{if } x, y, z \text{ are collinear;} \\
\text{True} & \text{if } x, y, z \text{ are not collinear and } p \text{ is a leaf;} \\
\text{False} & \text{if } p \text{ is an internal vertex and no point of } S \text{ is inside the triangle } xyz; \\
\bigvee_m \{F_{p_1}(a_x, b_y, p_m) \land F_{p_2}(b_y, c_z, p_m) \land F_{p_3}(c_z, a_x, p_m)\}, \text{ where } m \in S \text{ is inside the triangle } xyz, \text{ otherwise.}
\end{cases}
\]

**Proof.** First we consider the case when \( x, y, z \) are collinear. We then assign \( F_p(a_x, b_y, c_z) = \text{False} \) because three non-collinear points are necessary to draw a triangle. The next case is \( x, y, z \) are not collinear and \( p \) is a leaf. We then assign \( F_p(a_x, b_y, c_z) = \text{True} \) since three non-collinear points are sufficient to draw a triangle. In the next case, \( p \) is an internal vertex and no point of \( S \) is inside the triangle \( abc \). We assign \( F_p(a_x, b_y, c_z) = \text{False} \) since \( p \) cannot be placed inside \( C_p \). In the remaining case we define \( F_p(a_x, b_y, c_z) \) recursively according to Lemma 3. \( \square \)

Based on the recursive structure of Theorem 4, Nishat \textit{et al.} \cite{14} devised an \( O(nk^3) \)-time implementation for Feasibility Checking. Recently, Moosa \textit{et al.} \cite{12} showed that a slightly modified
implementation takes only $O(nk^4)$ time, which was also pointed out by an anonymous reviewer in our initial submission of this paper presenting an $O(nk^8)$-time implementation for Feasibility Checking. In the following we first describe the $O(nk^4)$-time implementation technique.

Let $G$ be a plane 3-tree with $n$ vertices and let $S$ be a set of $k$ points. Let $T$ be the representative tree of $G$. Recall that each vertex $i$ of $T$ corresponds to a unique triangle $C_i$ in $G$. Since the number of nodes in $T$ is $O(n)$, there are $O(n)$ such triangles. Each triangle $C_i$ can be mapped to three points of $S$ in $O(k^3)$ possible ways. We now find the solutions to the subproblems by a bottom-up computation. To avoid recomputation of the solutions, we store the solutions to the subproblems in a table $F_C[1:n,1:k,1:k,1:k]$. Each entry of the table initially contains null to denote that the entry is yet to be filled in. During the computation, the solution of $F_i(a_x,b_y,c_z)$ is stored into the entry $F_C[i,x,y,z]$. When a subproblem is first encountered during the computation, its solution is stored in the table. Each subsequent time the subproblem is encountered, the value stored in the table is looked up and returned.

We now use Theorem 4 for the bottom-up computation. For each leaf $i$, we mark each of the $O(k^3)$ possible mappings of $C_i$ as False or True depending on whether the three points corresponding to that mapping are collinear or not in $O(1)$ time. For each internal vertex $i$, we mark each of the $O(k^3)$ possible mappings of $C_i$ as follows. Let $a, b, c$ be the three vertices on the boundary of $C_i$ in anticlockwise order and we want to mark the mapping $a_x, b_y, c_z$, for some $\{x, y, z\} \in S$. Let $i_1, i_2$ and $i_3$ be the three children of $i$ that correspond to the triangles $abi, bci$ and $cai$, respectively. If there exists a mapping of $i$ to a point $m \in S$ interior to $C_i$ such that $F_C[i_1,x,y,m] \land F_C[i_2,y,z,m] \land F_C[i_3,z,x,m] = True$, then $F_C[i,x,y,z]$ is marked True. Otherwise, $F_C[i,x,y,z]$ is marked False. Since the computation is bottom-up, the values of $F_C[i_1,x,y,m], F_C[i_2,y,z,m]$ and $F_C[i_3,z,x,m]$ are already computed and can be retrieved in $O(1)$ time. Therefore, the time required to compute $F_C[i,x,y,z]$ is at most the number of possible choices for the mapping of $i$, which is $O(k)$. Since there are $O(nk^3)$ entries to be computed, the whole computation takes $O(nk^4)$ time.
Observe that this technique only tests the point-set embeddability of $G$ on $S$. To compute an embedding of $G$ on $S$, we either use a table similar to $FC$ or table $FC$ itself to store the computed mappings of the vertices during the bottom-up computation. We then do an $O(n)$-time top-down computation to find an embedding of $G$ on $S$.

6 Point-Set Embeddings of Planar Partial 3-Trees

In this section, we prove that deciding point-set embeddability is NP-complete for planar partial 3-trees. Cabello [4] reduced the NP-complete problem 3-partition to prove that deciding point-set embeddability is NP-complete for 2-outerplanar graphs. We prove that the graph he constructed in his reduction is a planar partial 3-tree.

The input of 3-partition problem is a natural number $B$ and a set $S$ of $3n$ natural numbers $a_1, \ldots, a_{3n}$ with $B/4 < a_i < B/2$. The problem is to decide whether $S$ can be partitioned into $n$ subsets such that each subset contains exactly three elements of $S$ and the sum of the elements of each subset is exactly $B$. For a given instance of 3-partition problem, Cabello constructed a planar graph $G$ as follows.

- Start with a 4-cycle with vertices $v_0, \ldots, v_3$, and edges $(v_{i-1}, v_{i \mod 4}), 1 \leq i \leq 4$.
- For each $a_i$ in the input, make a path $B_i$ consisting of $a_i$ vertices, and put an edge between each of those vertices and the vertices $v_0, v_2$.
- Construct $n - 1$ triangles $T_1, \ldots, T_{n-1}$. For each triangle $T_i$, put edges between each of its vertices and $v_2$, and edges between two of its vertices and $v_0$.
- Make a path $C$ of $(B + 3)n$ vertices, and put edges between each of the vertices in $C$ and $v_0$. Furthermore, put an edge between one end of the path and $v_1$, and another edge between the other end and $v_3$.

Figure 9(a) depicts a plane embedding of $G$. To show that $G$ is a planar partial 3-tree, we first construct a triangulated planar graph $G'$ from $G$ by adding edges to $G$. We then prove that $G'$ is a plane 3-tree. In the following we describe the edges that we add to $G$ to construct $G'$.

- Add edges $(v_1, v_3)$ and $(v_0, v_2)$.
- For each $B_i, 1 \leq i \leq 3n$, let the two end vertices be $w'_i$ and $w''_i$. Add the edges $(w''_i, w'_{i+1}), 1 \leq i < 3n$ and $(w'_1, v_1)$.
- For each triangle $T_i, 1 \leq i \leq n - 1$ let the two vertices that are adjacent to both $v_0$ and $v_2$, be $u'_i$ and $u''_i$. Add the edges $(u''_i, u'_{i+1}), 1 \leq i < n - 1$ and $(u''_{n-1}, v_3)$.
Figure 9. Illustration for (a) $G$ and (b) $G'$. 
– For each vertex $v$ on $C$ that is not adjacent to $v_1$ add an edge $(v_1, v)$.

Figure 9(b) depicts a plane embedding of $G'$, where the edges added to $G$ are drawn with dashes. It is straightforward to observe that $G'$ is a triangulated planar graph. We now prove that $G'$ is a plane 3-tree showing that we can delete the vertices of degree three recursively such that at each step the resulting graph remains triangulated. Such a sequence of deletions of degree three vertices is as follows.

– First we delete the $(B+3)n$ vertices on the path $C$ sequentially starting from the end vertex that is adjacent to $v_1$. Observe that such deletions yield a vertex of degree three on $C$ in successive steps.
– Next for each triangle $T_i$, $1 \leq i \leq n - 1$, we delete the vertex of $T_i$ that is not adjacent to $v_0$.
– Then for each triangle $T_i$, $1 \leq i \leq n - 1$, in the order $T_{n-1}, T_{n-2}, \ldots, T_1$, we first delete the vertex $u''_i$ and then the vertex $u'_i$.
– Finally, we delete the vertices of the paths $B_i$, $1 \leq i \leq 3n$, in the order $w''_{3n}, \ldots, w''_{3n}, w''_{3n-1}, \ldots, w''_{3n-1}, \ldots, w''_1, \ldots, w'_1$. We now remove $v_0$ to obtain the base triangle with vertices $v_1, v_2$ and $v_3$.

We now have the following theorem.

**Theorem 5.** It is NP-complete to decide whether a planar partial 3-tree with $n$ vertices has a point-set embedding on a set of $n$ points in the Euclidean plane.

7 Conclusion

We have given an $O(n^2)$ time algorithm that decides whether a plane 3-tree $G$ with $n$ vertices admits a point-set embedding on a set $S$ of $n$ points or not; and computes a point-set embedding of $G$ if such an embedding exists. Since a plane 3-tree $G$ has only a linear number of planar embeddings, we can check point-set embeddability for all the embeddings of $G$ and determine whether $G$ has a planar embedding on the given set of points in polynomial time when the embedding of $G$ is not fixed.

We have proved an $\Omega(n \log n)$ lower bound on the time complexity for obtaining point-set embeddings of plane 3-trees. Beside obtaining a point-set embedding of a plane 3-tree, we have considered a generalized problem, where the given point-set has more than $n$ points. We have given a polynomial-time space-efficient algorithm to solve the problem. It is a challenge to find simpler
algorithms for obtaining point-set embeddings of plane 3-trees both in the restricted and generalized cases.

Moosa et al. [12] tried to give an improved algorithm for computing point-set embeddings of plane 3-trees in terms of running time. Let $abc$ be a triangle with $n$ points in its proper interior and let $n_1, n_2$ be two integers such that $n_1 + n_2 \leq n$. Their algorithm uses binary search to find two points $x$ and $y$ on line $bc$, such that triangles $abx$ and $acy$ contain $n_1$ and $n_2$ points, respectively. If the length of line segment $bc$ is $|bc|$ and the distance between the closest pair of points is $d$, then this approach will take time proportional to $O(\log \frac{|bc|}{d})$ (see Lemma 14.1 in [2]), which is a major problem since we assume that the points are not necessarily in general position. Recently, Moosa et al. [13] and Durocher et al. [6] independently proposed two different modifications to avoid this problem that yields an $O(n^{4/3+\varepsilon})$-time algorithm for computing point-set embeddings of plane 3-trees, for any fixed $\varepsilon > 0$. Both of these algorithms use triangular range search data structure. Finding an algorithm for computing point-set embeddings of plane 3-trees with running time faster than $O(n^{4/3+\varepsilon})$ is still open.

We have shown that the NP-completeness proof of the point-set embeddability problem given by Cabello [4] holds for planar partial 3-trees. Cabello mentioned that the technique of his NP-hardness proof does not seem possible to extend to show the NP-hardness of point-set embeddability problem for 3-connected planar graphs. Therefore, it would be interesting to investigate the complexity of the problem for planar partial 3-trees that are 3-connected.

Acknowledgment

This work is done in Graph Drawing & Information Visualization Laboratory of the Department of CSE, BUET established under the project “Facility Upgradation for Sustainable Research on Graph Drawing & Information Visualization” supported by the Ministry of Science and Information & Communication Technology, Government of Bangladesh. We thank the anonymous referees for their useful suggestions, which helped us to improve the results and the presentation of our paper.

References