Heuristic Algorithms for Geometric Embedding of Complete Binary Trees onto a Point-set

Mehrab Norouzitallab, Alireza Bagheri, and Mahmoodreza Jahanseir

***Abstract*— In the geometric graph embedding problem, a graph with n vertices and a set of *n* points in the plane are given, and the aim of embedding is to find a mapping between vertices of the graph to these points in such a way that minimizes the length of the embedded graph on the point set. Since the travelling salesman problem is a special case of the graph embedding problem, therefore, the problem is an NP-hard problem. In this paper, we consider a particular case where the given graph is a binary tree. We present four heuristic approaches, then we compare the time complexity, and the resulted embedding length of these algorithms.**

***Index Terms*— point-set embedding, geometric embedding, computational geometry, graph algorithms**

I. Introduction

Given a set P of n points in the plane where $n = 2^k-1$, $k = \{0, 1, \dots, m\}$ and a complete binary tree T with n nodes, each node of T is mapped onto a distinct point of P. The problem is called point-set embedding of complete binary trees. The length of each edge in T is equal to the Euclidean distance between the points on which its two end nodes are mapped. Consequently, the length of T is equal to the total length of its edges. The problem as a special case of geometric embedding problem addresses the issue of embedding T onto P with minimum length.

Embedding a free tree onto a point-set is NP-hard, because it is a generalization of TSP [5]. A path is a special case of a tree; therefore, in TSP, one should embed a given cycle (or path) onto a given set of points in the plane such that the total length of the cycle (or path) is minimized.

Bern et al. [5] developed approximation algorithms for embedding a complete tree onto a point-set on the line and in the plane. Their principal techniques were a notion of approximate geometric sorting, and approximation schemes for the minimum spanning tree problem in the plane. A $O(nloglogn)$ time $O(\Delta logn)$-approximation algorithm was proposed for embedding a complete n-node tree T of maximum vertex degree Δ onto a set P of n points in the plane. When the points of P were on a line, a 3-approximation algorithm was given for embedding of a complete n-node binary tree. They also developed a $O(n^{5.76})$ exact algorithm for this problem.

Hansen [14] gave an algorithm with $O(logn)$ approximation factor which embedded a hypercube, butterfly, or shuffle-exchange graph into a set of points with triangle inequality property. He also showed that we can embed a d-dimensional grid graph into a point-set in $O(nlogn)$ time with $O(log^2n)$ approximation factor for $d = 2$,and $O(logn)$ approximation factor for $d > 2$. Furthermore, he presented a polynomial time algorithm with $O(log^2n)$ approximation factor which embedded an arbitrary graph into a uniformly distributed point set.

Bagheri and Razzazi [2] have given a linear time $O(\Delta ph(T))$-approximation algorithm for geometric embedding of a n-node free tree T onto a point-set in the plane. In their algorithm, $ph(T)$ is the path height [3] of the given tree T. Their algorithm runs in linear time and works for general free trees. The approximation factor provided by them is better than the approximation factor of Bern et al. For n-node trees, $ph(T)$ is usually much less than $logn$.

There are many related problems that have been investigated in the literature, where the embedding length is not important. Cabello [8] showed that planar embedding of a graph onto a point-set in the plane is an NP-hard problem. Bose et al. [7] presented a $\theta(nlogn)$ algorithm to optimal embedding of a tree into a set of points. Bose [6] gave an algorithm to embed an outer planar graph onto a point set with $O(nlog^3n)$ time complexity. Di Giacomo et al. [11,12] and Bagheri and Razzazi [4] investigated point set embedding of trees with given partial drawings. Point set embedding of colored graph on colored points was studied in [1,10,13].

In this paper, two sets of heuristic algorithms are proposed to address the problem of geometric embedding of a complete binary tree onto a point-set, which is itself a special case of embedding a free tree in a point-set with minimum length. The rest of the paper is organized as follows. The embedding algorithms and their complexity analysis are given in section 2. Empirical results are discussed in section 3, and finally section 4 contains some concluding remarks.

II. The Algorithms

In this section, four heuristics for geometric embedding of a complete binary tree onto a point-set are presented. The first three heuristics are based on finding the median of the points in the plane. It should be noticed that the first two heuristics are almost identical and the third one is a combination of them. Separation of these algorithms in this

M. Norouzitallab is with the Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran(e-mail: m.norouzitallab@gmail.com).

A. Bagheri is with the Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran(e-mail: ar_bagheri@aut.ac.ir).

M. Jahanseir is with the Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran(e-mail: mr.jahanseir@aut.ac.ir).

ISBN: 978-988-19253-3-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

paper is only because of some presentation reasons. In section A, the median based algorithms are posed. Then in section B, an algorithm that works based on finding nearest neighbors of the points is proposed.

A. The Median Based Algorithms

The aim of these algorithms is partitioning P into two subsets with identical number of points, where the points of each subset have fewer differences in their x-coordinates or y-coordinates while the root is also mapped onto a point. With a same policy, the whole tree would be embedded onto the point set. In our first algorithm, the root of T is mapped onto the point $p_m=(x_m, y_m)$ of P that its x-coordinate is the median of all the x-coordinates of the points of P. Furthermore, the point-set P is partitioned into two subsets P_1 and P_2. All of the points in P_1 have x-coordinates less than x_m and all the points in P_2 have x-coordinates greater than x_m. The two sub trees T_1 and T_2 of T are recursively embedded onto P_1 and P_2. The following pseudo-code shows how this algorithm works.

X_MEDIAN_EMBEDDING(P, T)

1 if $|T| = 0$ then
2 return
3 p_m = FIND_X_MEDIAN(P)
4 let P_1 is the set of all points of P that have x-coordinates less than x-coordinate of p_m
5 let P_2 is the set of all points of P that have x-coordinates greater than x-coordinate of p_m
6 let T_1 and T_2 be the two sub-trees of T
7 MEDIAN_X_EMBEDDING(P_1, T_1)
8 MEDIAN_X_EMBEDDING(P_2, T_2)
9 map the root of T onto p_m, connect p_m to the points associated with the roots of T_1 and T_2, if T_1 and T_2 are not null.

Theorem 1. The X_MEDIAN_EMBEDDING algorithm has the time complexity $O(nlogn)$ and it requires $O(n)$ space.

Proof. The time complexity of the algorithm is strictly depended on the time complexity of the median finder algorithm. We can find median of the points in $O(n)$ time [9]. It can be seen simply that the time complexity of the algorithm has the following recursion:

$$T(n)=\begin{cases}2T(n/2)+O(n) & n>0\\ O(1) & n=0\end{cases}$$

From the master theorem [9], it can be concluded that the time complexity of the algorithm is $O(nlogn)$. Similarly, the space complexity of the algorithm can be found by the following recursion:

$$S(n)=\begin{cases}S(n/2)+O(n) & n>0\\ O(1) & n=0\end{cases}$$

Which leads $O(n)$ space complexity.

One can choose y-coordinates instead of x-coordinates, and this will lead to y median embedding algorithm. Similarly, in the y median algorithm the same results are expected.

In the following, "x-y median algorithm" is described which is a combination of x median and y median algorithms. In this algorithm, there are two choices to embed the root of T, where one choice is the x median and the other is the y median of the points. Furthermore, this procedure recursively embeds the other nodes of the tree onto the other points in the plane. For the selection, between the x median and the y median, the one that minimizes the length of the embedding is chosen.

The following pseudo code illustrates the x-y median algorithm. In this algorithm, E keeps the embedding of the points. Furthermore, E_x and E_y maintain the x-median and the y-median embedding of the points, respectively.

X_Y_MEDIAN_EMBEDDING(P, T)

1 if $|T| = 1$ then
2 map the root of T on the remained point in P into E
3 return E
4 p_{m_x} = FIND_X_MEDIAN(P)
5 let P_{1_x} is the set of all points of P that have x-coordinates less than x-coordinate of $p_{m\ x}$
6 let P_{2_x} is the set of all points of P that have x-coordinates greater than x-coordinate of $p_{m\ x}$
7 p_{m_y} = FIND_Y_MEDIAN(P)
8 let P_{1_y} is the set of all points of P that have x-coordinates less than x-coordinate of $p_{m\ y}$
9 let P_{2_y} is the set of all points of P that have x-coordinates greater than x-coordinate of $p_{m\ y}$
10 let T_1 and T_2 be the two subtrees of T
11 E_x = X_Y_MEDIAN_EMBEDDING(P_{1_x}, T_1) ∪ X_Y_MEDIAN_EMBEDDING($P_{2\ x}$, T_2)
12 $E_x(r) = p_{m\_x}$, where r is the root of T
13 E_y = X_Y_MEDIAN_EMBEDDING(P_{1_y}, T_1) ∪ X_Y_MEDIAN_EMBEDDING($P_{2\ y}$, T_2)
14 $E_y(r) = p_{m\_y}$, where r is the root of T
15 if the length of embedding in E_x is less than the length of embedding in E_y then
16 return E_x
17 else
18 return E_y

Theorem 2. The X_Y_MEDIAN_EMBEDDING algorithm has time complexity $O(n^2)$ and it needs $O(n)$ space.

Proof. In the above algorithm, we can find the x-median and y-median of the points in linear time [9]. Furthermore, the length of an embedding can be found in $O(n)$ time. Therefore, we can obtain time complexity of X_Y_MEDIAN_EMBEDDING algorithm from the following recursive equation:

$$T(n)=\begin{cases}4T(n/2)+O(n) & n>0\\ O(1) & n=0\end{cases}$$

Which is $O(n^2)$. Moreover, the space complexity of the algorithm follows from the below recursion:

ISBN: 978-988-19253-3-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$S(n)=\begin{cases} S(n/2)+O(n) & n>0 \\ O(1) & n=0 \end{cases}$$

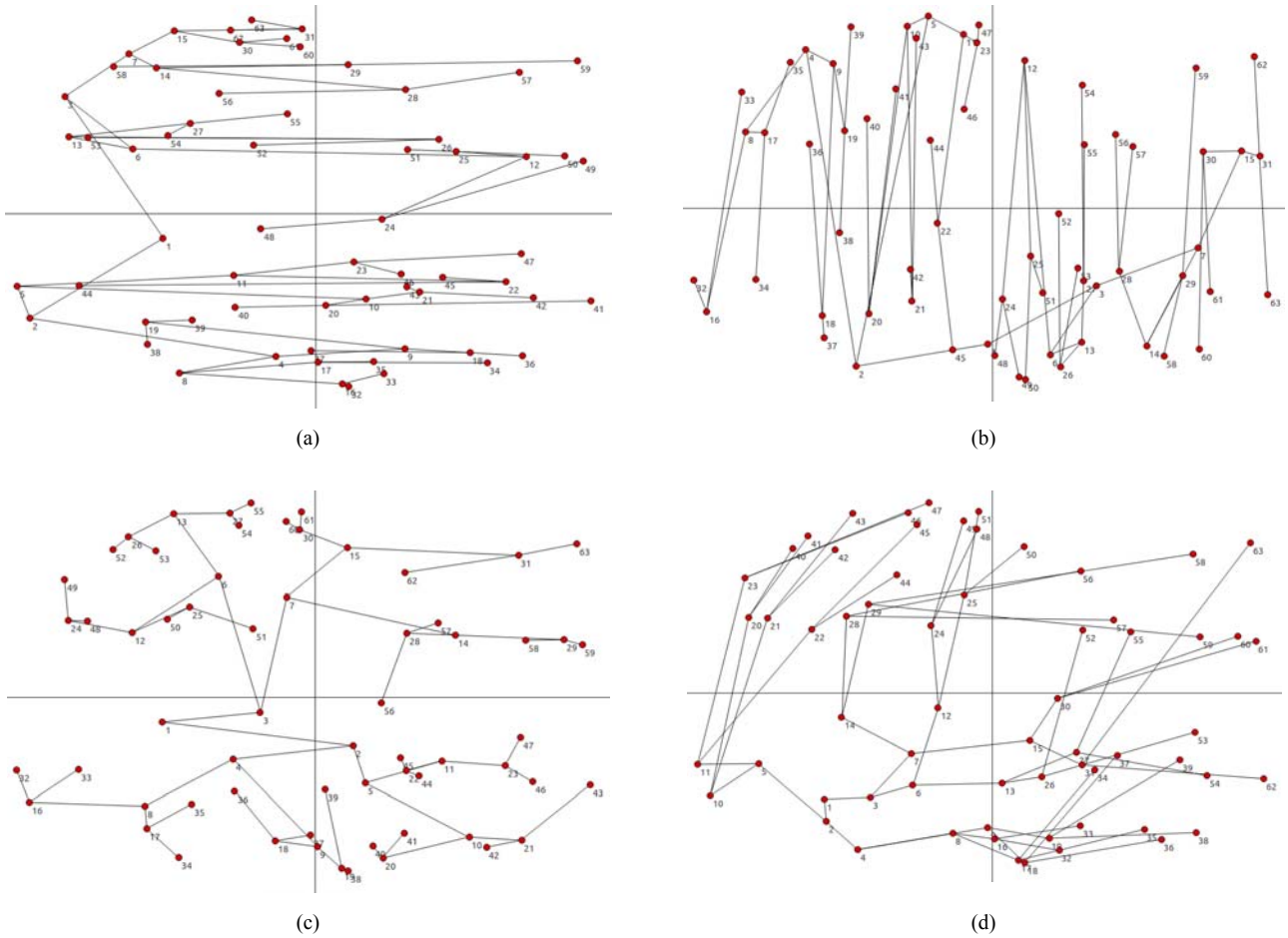
Which leads $O(n^2)$.

The median based algorithms can be extended to higher dimensions. Considering the X_MEDIAN_EMBEDDING algorithm, if the dimension is d, the problem can be divided into two sub-problems based on the median of the k-coordinates of the point-set, where k is less than or equal to d. The time complexity of such an algorithm is only depended on the number of points and is $O(n)$. Similar to the X_Y_MEDIAN_EMBEDDING algorithm, a divide and conquer strategy can be used for higher dimensions so that it changes the algorithm to select between the median of each coordinate in P. Therefore, the recursive equation of the time complexity of this algorithm is $T(n)=2dT(n/2) + O(n) = O(n^{1+\log(d)})$.

B. *The Nearest Neighbors Algorithm*

In this algorithm, the root of T is mapped onto a point p_r from the point-set, then its left child is mapped onto the nearest neighbor of p_r, and its right child is mapped onto the second nearest neighbor of p_r. A same strategy is applied to embed the remained nodes of T onto the remained points of P where the mapping of the nodes to the nearest neighbors is done from the leftmost child to the rightmost child in each level of T. The following pseudo codes show the details of this algorithm

```
Nearest_Neighbors_Embedding(P, T)
1    if |T| = 1 then
2        map the root of T onto the only point of P
3        return
4    BEST_EMBEDDING_LEN ← 0
5    R_BEST ← 0
6    For r ← 1 to |P| do
7        EMBEDDING_LEN ←
         Embed_Based_On_A_Point(P, T, r)
8        If EMBEDDING_LEN <
         BEST_EMBEDDING_LEN then
9            BEST_EMBEDDING_LEN ←
             EMBEDDING_LEN
10           R_BEST ← r
11   Embed_Based_On_A_Point(P, T, R_BEST )
Embed_Based_On_A_Point(P, T, r)
1    EMBEDDING_LEN ← 0
2    Q ← ∅
3    let x be the root of T
4    map x onto p_r
```



(a) (b) (c) (d)

Fig. 1. (a) *X* Median Embedding (b) *Y* Median Embedding (c) *X-Y* Median Embedding (d) Nearest Neighbors Embedding

ISBN: 978-988-19253-3-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

5 ENQUEUE(Q, $<p_r, x>$)
6 While $Q \neq \emptyset$ do
7 $<p_{top}, x> \leftarrow$ DEQUE(Q)
8 let x_l and x_r be the left and right children of x
9 let p_1 and p_2 be the first and the second unvisited nearest neighbors of p_{top}
10 map x_l onto p_1 and x_r onto p_2
11 mark p_1 and p_2 as visited
12 If x_l and x_r have children then
13 ENQUE(Q, $<p_1, x_l>$)
14 ENQUE(Q, $<p_2, x_r>$)
15 EMBEDDING_LEN $\leftarrow$ EMBEDDING_LEN + DISTANCE(p_{top}, p_1) + DISTANCE(p_{top}, p_2)
16 return EMBEDDING_LEN

Theorem 3. The nearest neighbors algorithm has time complexity $O(n^2 log^2 n)$ and needs $O(n)$ space.

Proof. The time complexity of the Embed_Based_On_A_Point algorithm is $O(nlog^2 n)$, because the time complexity of finding unvisited nearest neighbors in the plane using algorithm [15] for each point of P is $O(log^2 n)$ and the while loop repeats $O(n)$ times. Furthermore, in the Nearest_Neighbors_Embedding algorithm, for each point of P, we call the Embed_Based_On_A_Point algorithm. Therefore, time complexity of the nearest neighbors algorithm will be $O(n^2 log^2 n)$. Furthermore, the amount of needed space for the algorithm is $O(n)$, because we need a queue of size $O(n)$ to store the mapping between the nodes of the tree and the points.

The algorithm can be extended to higher dimensions without any changes. The only difference is the time complexity which will be changed to $O(dn^3)$, where d is the dimension which affects the time complexity of calculating Euclidean distance between the points. It should be noticed that for this time complexity, no nearest neighbors query processing algorithm is considered.

III. Experimental results

To study the results of the algorithms for a huge number of points, some experiments have been conducted. In each experiment, some points have been generated using a random function with uniform distribution in Euclidean space which the points are restricted to have x-coordinates and y-coordinates between [$-C$, C]. The experiments have been repeated for several times with different point-sets and different values for C. Fig. 1 illustrates final embedding of our algorithms on a special point-set.

To compare the algorithms with the optimal embedding, the algorithms have also been compared to the minimum spanning tree of the point-set. Fig. 2 shows that the X-Y median algorithm approximates the optimal solution of the embedding problem properly while the result of X Median or Y Median is not sufficiently appropriate. The length of the embedding which is achieved by X-Y median algorithm is very close to the length of minimum spanning tree of the same point-set. Also, we know that the length of minimum spanning tree is less than the length of optimal embedding. Therefore, the X-Y median algorithm leads a good solution to the geometric embedding problem for complete binary

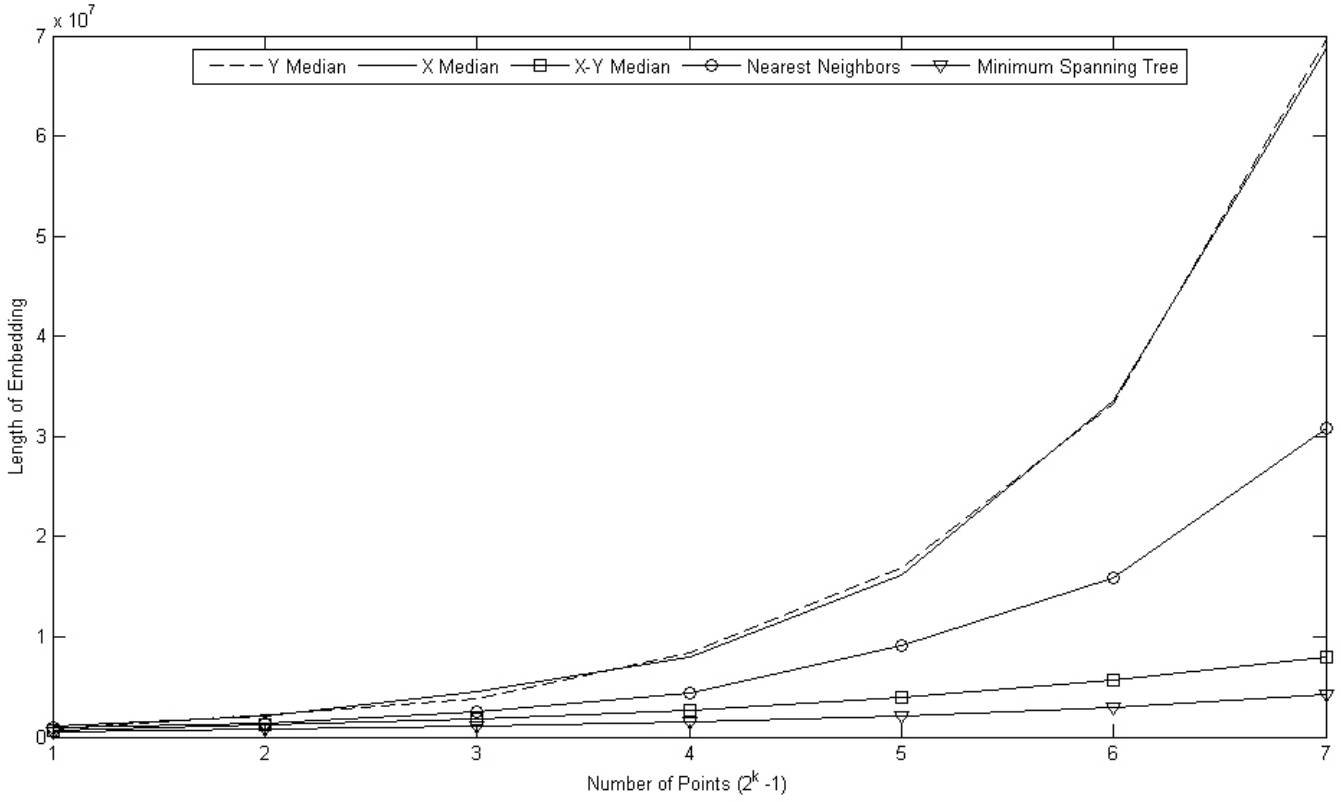


Fig. 2. A comparison of the proposed algorithms.

ISBN: 978-988-19253-3-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

trees. Finally, based on the resulted embedding length and the time complexity of the proposed algorithms, it can be concluded that *X*-*Y* Median algorithm is substantially better than the others.

IV. Conclusion

In this paper, four heuristics for geometric embedding of a complete binary tree into a point-set have been presented. The *X*-median algorithm and the *Y*-median algorithm have time complexity $O(nlogn)$. The third algorithm was the *X*-*Y* median algorithm with time complexity $O(n^2)$. The last algorithm was the nearest neighbor and its time complexity was $O(n^2log^2n)$. Finally, algorithms had been compared and we concluded that the *X*-*Y* median algorithm is a good candidate for the problem. An interesting open problem is finding an upper bound for the length of the embedding in the heuristics.

References

[1] M. Badent, E. Di Giacomo, G. Liotta, "Drawing colored graphs on colored points," *Theoretical Computer Science*, vol 408, pp 129-142, 2008.

[2] A. Bagheri, M. Razzazi, "An approximation algorithm for minimum length geometric embedding of trees," Tech. Rep., 2010.

[3] A. Bagheri, M. Razzazi, "Minimum height path partitioning of trees," *Scientia Iranica*, vol 17, pp. 99-104, 2010.

[4] A. Bagheri, M. Razzazi, "Planar straight line point set embedding of trees with partial embeddings", *Information Processing Letters*, vol 110, pp 521-523, 2010.

[5] M. W. Bern, H. J, Karloff, P. Raghavan, B. Schieber, "Fast Geometric Approximation Techniques and Geometric Embedding Problems," *Theoretical Computer Science*, vol 106 , pp. 265-281, 1992.

[6] P. Bose, "On embedding an outer-planar graph on a point set," *Computational Geometry: Theory and Applications*, vol 23, pp. 303-312, 2002.

[7] P. Bose, M. McAllister, J. Snoeyink, "Optimal algorithms to embed trees in a point set," *Journal of Graph Algorithms and Applications*, vol 1, pp. 1-15, 1997.

[8] S. Cabello, "Planar embeddability of the vertices of a graph using a fixed point set as NP-hard," *Journal of Graph Algorithms and Applications*, vol 10, pp 353-363, 2006.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Intorduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[10] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, F. Trotta, S. K. Wismath, "k-colored point-set embeddability of outer planar graphs," *Journal of Graph Algorithms and Applications*, vol 12, pp 29-49, 2008.

[11] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, S. Wismath, "Point-set embedding of trees with edge constraints," In: *Proceeding of Graph Drawing*, 2008, vol 4875, pp 113-124.

[12] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer, S. Wismath, "Point-set embedding of trees with given partial drawings," *Computational Geometry: Theory and Applications*, vol 42, pp 664-676, 2009.

[13] E. Di Giacomo, G. Liotta, F. Trotta, "On embedding a graph on two sets of points," *International Journal of Foundations of Computer Science*, vol 17 (5), pp 1071-1094, 2006.

[14] M. D. Hansen, "Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems," *30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 604-609.

[15] T. M. Chan, "A Dynamic Data Structure for 3-d Convex Hulls and 2-d Nearest Neighbor Queries", *Journal of the ACM*, vol 57(3):16, 2010.

ISBN: 978-988-19253-3-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)