Geometric Spanning Trees and Hypergraphs that Minimizes the Wiener Index

Atishaya Maharjan

University of Manitoba
Geometric, Approximation, and Distributed Algorithms (GADA) lab

June 23, 2025

Wiener Index in Graphs

- Let $G = (V, E)$ be a weighted undirected graph and let $\delta_G(u, v)$ denote the **shortest (minimum-weight) path** between vertices u and v in G.

Wiener Index in Graphs

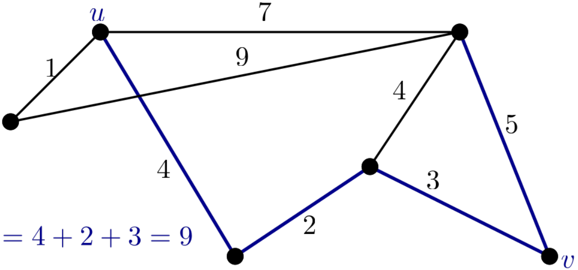
- Let $G = (V, E)$ be a weighted undirected graph and let $\delta_G(u, v)$ denote the **shortest (minimum-weight) path** between vertices u and v in G.
- The Wiener index is defined as:

$$W(G) = \sum_{u,v \in V} \delta_G(u, v)$$

Wiener Index in Graphs

- Let $G = (V, E)$ be a weighted undirected graph and let $\delta_G(u, v)$ denote the **shortest (minimum-weight) path** between vertices u and v in G.
- The Wiener index is defined as:

$$W(G) = \sum_{u,v \in V} \delta_G(u, v)$$



$\delta_G(u, v) = 4 + 2 + 3 = 9$

Problem Statement: Geometric Spanning Trees

Reference: WADS 2023 [Abu-Affash et al., 2023]

1. Input: A set P of n points in the plane.

Problem Statement: Geometric Spanning Trees

Reference: WADS 2023 [Abu-Affash et al., 2023]

1. Input: A set P of n points in the plane.
2. Goal: Construct a **spanning tree** on P that minimizes the **Wiener index**.

Problem Statement: Geometric Spanning Trees

Reference: WADS 2023 [Abu-Affash et al., 2023]

1. Input: A set P of n points in the plane.
2. Goal: Construct a **spanning tree** on P that minimizes the **Wiener index**.
3. The weight function is defined as the Euclidean distance between points.

Results Summary: Geometric Spanning Trees

- Spanning tree of P that minimizes the Wiener index is planar.

Results Summary: Geometric Spanning Trees

- Spanning tree of P that minimizes the Wiener index is planar.
- When P is in convex position, this can be solved in polynomial time.

Results Summary: Geometric Spanning Trees

- Spanning tree of P that minimizes the Wiener index is planar.
- When P is in convex position, this can be solved in polynomial time.
- The hamiltonian path of P that minimizes the Wiener index is not necessarily planar.

Results Summary: Geometric Spanning Trees

- Spanning tree of P that minimizes the Wiener index is planar.
- When P is in convex position, this can be solved in polynomial time.
- The hamiltonian path of P that minimizes the Wiener index is not necessarily planar.
- Computing such a hamiltonian path is NP-Hard.

Minimum Spanning Tree (MST)

Minimum Spanning Tree (MST)

A **minimum spanning tree** of a weighted undirected graph is a spanning tree that has the minimum total edge weight.

- The MST connects all vertices with the minimum total edge weight.
- It is unique if all edge weights are distinct.

Euclidean Minimum Spanning Tree

Euclidean Minimum Spanning Tree

The **Euclidean minimum spanning tree** (EMST) of a set of points in the Euclidean space is the minimum spanning tree where the edge weights are the Euclidean distances between points.

- The EMST can be computed in $O(n \log n)$ time.
- The EMST is unique if all points are distinct.

Euclidean Minimum Spanning Tree

Euclidean Minimum Spanning Tree

The **Euclidean minimum spanning tree** (EMST) of a set of points in the Euclidean space is the minimum spanning tree where the edge weights are the Euclidean distances between points.

- The EMST can be computed in $O(n \log n)$ time.
- The EMST is unique if all points are distinct.

Note: It can be shown that the EMST is a subgraph of special geometric graphs called **Delaunay triangulations** which are a dual of **Voronoi Diagrams** [de Berg et al., 2000].

Euclidean Minimum Spanning Tree VS Wiener Index

Question: Is the EMST a good approximation for the spanning tree that minimizes the Wiener index?

- **Answer:** No, see the code examples.

Divide and Conquer Algorithm for Approximating Hamiltonian Paths

Algorithm 1 DivideAndConquerWiener(P)

procedure DivideAndConquerWiener(P, $depth = 0$, $maxDepth = 10$)
    if $|P| \leq 4$ or $depth > maxDepth$ **then**
        return BruteForceHamiltonianPath(P)
    end if
    $(a, b, c) \leftarrow$ FindBisectingLine(P)
    $(P_L, P_R) \leftarrow$ PartitionPoints(P, a, b, c)
    $\pi_L \leftarrow$ DivideAndConquerWiener(P_L, $depth + 1$)
    $\pi_R \leftarrow$ DivideAndConquerWiener(P_R, $depth + 1$)
    return ConnectPaths(π_L, π_R)
end procedure

Supporting Procedure: FindBisectingLine

Algorithm 2 FindBisectingLine(P)

procedure FINDBISECTINGLINE(P)
  $minX \leftarrow \min_{p \in P} p.x$, $maxX \leftarrow \max_{p \in P} p.x$
  $minY \leftarrow \min_{p \in P} p.y$, $maxY \leftarrow \max_{p \in P} p.y$
  $width \leftarrow maxX - minX$, $height \leftarrow maxY - minY$
  if $width \geq height$ **then**
    $midX \leftarrow (minX + maxX)/2$
    return $(1.0, 0.0, -midX)$ ▷ Vertical line: $x = midX$
  else
    $midY \leftarrow (minY + maxY)/2$
    return $(0.0, 1.0, -midY)$ ▷ Horizontal line: $y = midY$
  end if
end procedure

Supporting Procedure: PartitionPoints

Algorithm 3 PartitionPoints(P, a, b, c)

procedure PartitionPoints(P, a, b, c)
    $L \leftarrow [\ ]$, $R \leftarrow [\ ]$
    for each $p \in P$ **do**
        $v \leftarrow ap.x + bp.y + c$
        if $v \leq 0$ **then**
            $L \leftarrow L \cup \{p\}$
        else
            $R \leftarrow R \cup \{p\}$
        end if
    end for
    if $L = \emptyset$ **then**
        Move one point from R to L
    else if $R = \emptyset$ **then**
        Move one point from L to R
    end if

Supporting Procedure: ConnectPaths

Algorithm 4 ConnectPaths(π_1, π_2)

procedure ConnectPaths(π_1, π_2)
    options $\leftarrow$ empty list
    Append $(\pi_1 + \pi_2, W(\pi_1 + \pi_2))$ to *options*
    Append $(\pi_1 + \text{reverse}(\pi_2), W(\pi_1 + \text{reverse}(\pi_2)))$ to *options*
    Append $(\text{reverse}(\pi_1) + \pi_2, W(\text{reverse}(\pi_1) + \pi_2))$ to *options*
    Append $(\text{reverse}(\pi_1) + \text{reverse}(\pi_2), W(\text{reverse}(\pi_1) + \text{reverse}(\pi_2)))$ to *options*
    return path with minimum Wiener index from *options*
end procedure

Approximation Quality and Efficiency

#Pts	Approx Ratio	Range	D&C Time (s)	Speedup
6	1.0206 ± 0.0501	[1.0000, 1.3021]	0.0001	72.4x
7	1.0212 ± 0.0480	[1.0000, 1.3014]	0.0001	481.8x
8	1.0427 ± 0.0731	[1.0000, 1.3307]	0.0002	2800.6x
9	1.0625 ± 0.0873	[1.0000, 1.3318]	0.0003	6785.5x
10	1.0520 ± 0.0776	[1.0000, 1.4271]	0.0003	80249.2x

Table 1: Approximation quality and speed comparison with optimal algorithm.

References I

Abu-Affash, A. K., Carmi, P., Luwisch, O., and Mitchell, J. S. B. (2023).
Geometric spanning trees minimizing the wiener index.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000).
Computational Geometry: Algorithms and Applications.
Springer-Verlag, second edition.

The end.

maharjaa@umanitoba.ca