The Wiener Index: From Trees to Graphs with Many Cut-Edges

Anne Christiono

53rd Southeastern International Conference on Combinatorics, Graph Theory, and Computing

March 8, 2022

Introduction to the Wiener Index

1947: Harry Wiener proposed the Wiener Index of a chemical graph.

- Oldest molecular topological index
  - Other topological indices created based on Wiener Index's success
- Correlated with various chemical properties of a molecule
  - Viscosity, density, boiling point

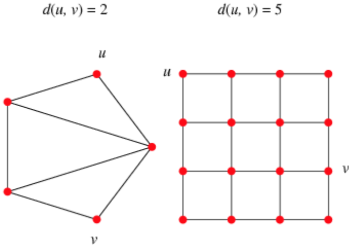
The Wiener Index

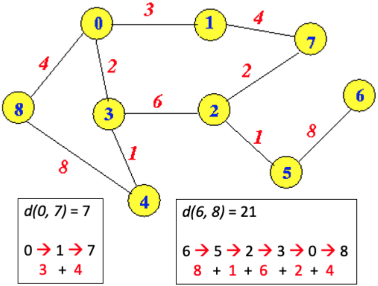
The sum of the distances between all pairs of vertices in a graph.

Introduction to Graphs

Distance: length of the shortest path between two vertices u and v

- denoted as $d(u, v)$
- if graph is weighted, distance refers to **least weighted distance**

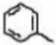




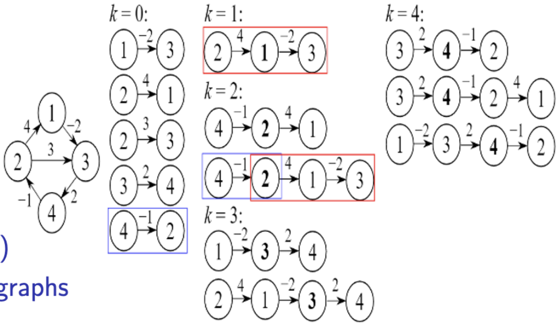
Wiener Index Formula and Examples

Wiener Index of graph G:

$$W(G) = \sum_{(u,v)} d((u,v)).$$

Molecules	Wiener Indices
	4
	10
	9
	27
	30

Existing Algorithms: Floyd-Warshall



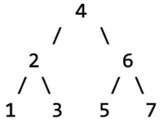
Floyd-Warshall:

- Time complexity- $O(N^3)$
- Weighted and directed graphs
- All pairs shortest path
- Dynamic programming

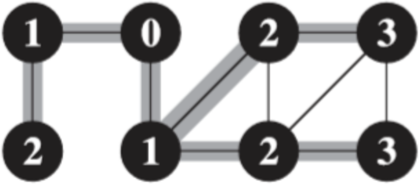
Existing Algorithms: BFS

Breadth-First Search (BFS):

- Time complexity- $O(NE)$
- Performs well on sparse graphs
- Unweighted and directed graphs
- Single source shortest path



Breadth first traversal: 4, 2, 6, 1, 3, 5, 7



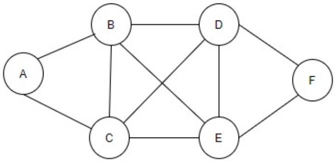
Weighted Graphs

Edge Weighted Graphs

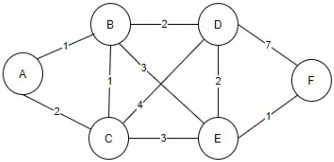
Each *edge* has a weight pre-assigned to it.

Wiener Index:

$$W(G) = \sum_{(u,v)} d((u,v)).$$



UnWeighted Graph



Weighted Graph

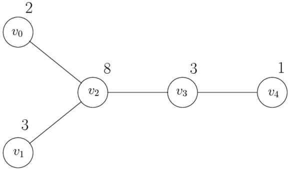
Weighted Graphs

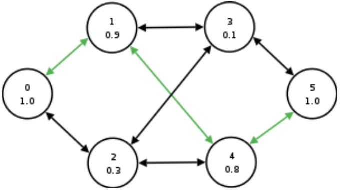
Vertex Weighted Graphs

Each *vertex* has a weight pre-assigned to it.

Wiener Index:

$$VWW(G) = \sum_{(u,v)} d((u,v)) \cdot w(u) \cdot w(v).$$





Algorithms for Weighted Graphs

Floyd-Warshall:

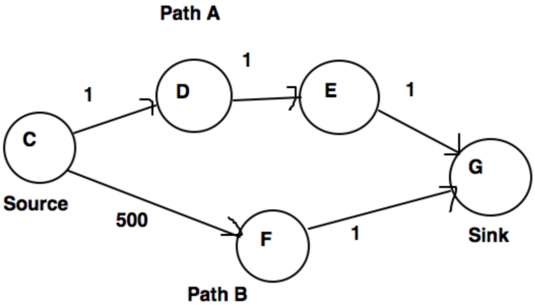
- iterates through all possible paths between vertices
- large complexity: $O(N^3)$

```
for k from 1 to N do
    for i from 1 to N do
        for j from 1 to N do
            if dist[i][j] > dist[i][k] + dist[k][j] then
                dist[i][j] = dist[i][k] + dist[k][j]
        end
    end
end
```

Algorithms for Weighted Graphs

BFS: does not work for weighted cases

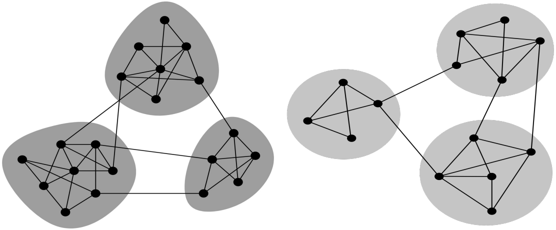
- relies on taking path with least edges
- longer path may be less expensive



Algorithms for Weighted Graphs

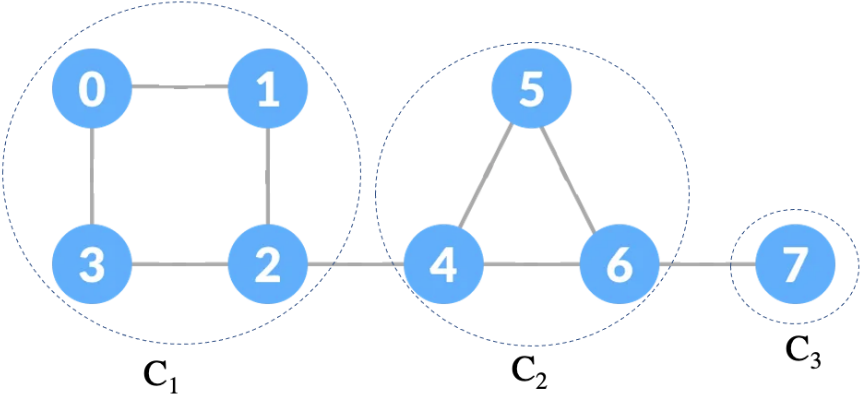
Generalization: trees to graphs with many pseudo-components

- **Cut-edge:** edge such that if removed, the graph becomes disconnected; number of pseudo-components increases.
- **Pseudo-component:** Maximal induced subgraph of graph G with no cut-edges



Pseudo-Component Examples

Consider pseudo-components as more complex nodes

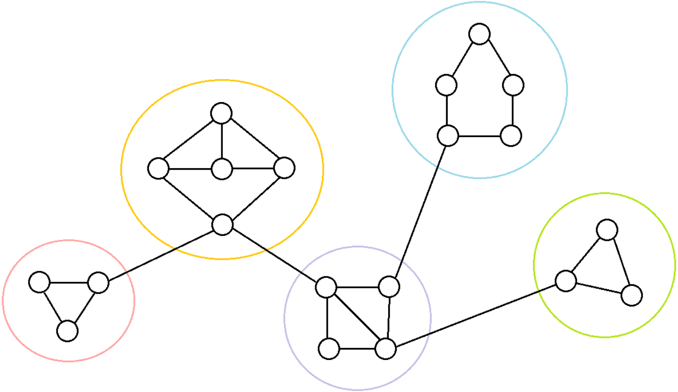


CEPC (Cut-Edge, Pseudo-Component) Theorem

$$
\begin{aligned}
W(G) = & \sum_{i=1}^{n} W(C_i) \\
& + VWW(T) \\
& + \sum_{i=1}^{n} \sum_{s=1}^{k} d_{C_i}(u_i^s) \cdot \left(N - n_{u_i^s}(u_i^s u_j^{s'})\right) \\
& + \sum_{i=1}^{n} \sum_{1 \leq s < t \leq k} d_{C_i}(u_i^s u_i^t) \cdot \left(N - n_{u_i^s}(u_i^s u_j^{s'})\right) \\
& \qquad \cdot \left(N - n_{u_i^t}(u_i^t u_\ell^{t'})\right)
\end{aligned}
$$

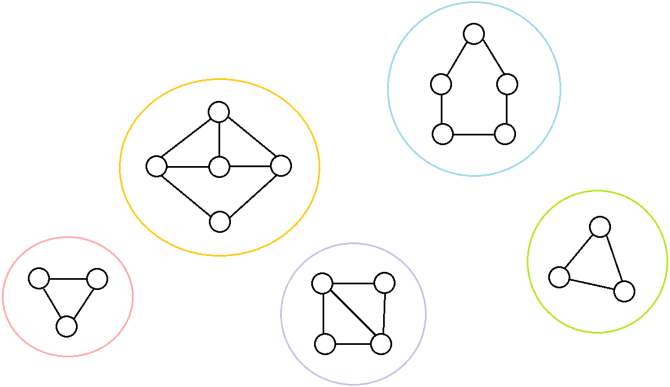
Term 1: $\sum_{i=1}^{n} W(C_i)$

The contribution from paths within the same pseudo-component



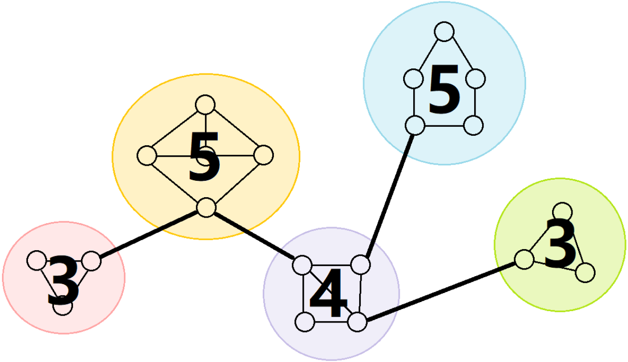
Term 1: $\sum_{i=1}^{n} W(C_i)$

The contribution from paths within the same pseudo-component



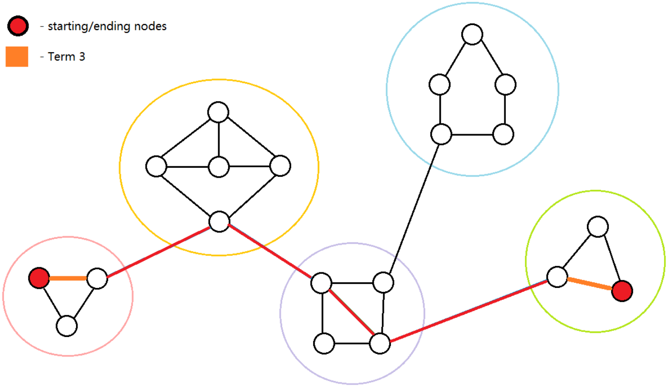
Term 2: $VWW(T)$

The contribution from the cut-edges to paths between pseudo-components



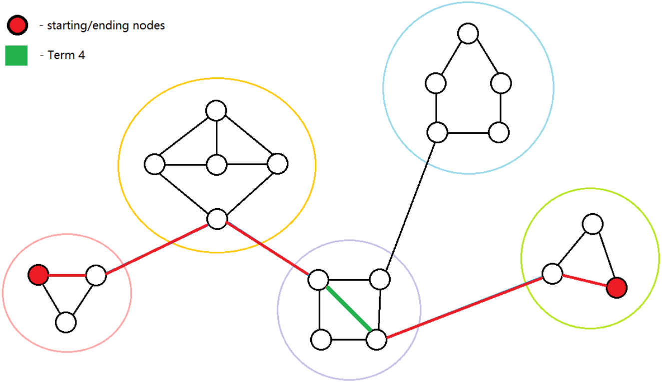
Term 3: $\sum_{i=1}^{n} \sum_{s=1}^{k} d_{C_i}(u_i^s) \cdot \left(N - n_{u_i^s}(u_i^s u_j^{s'})\right)$

Contribution from edges in start/end p.c.'s for paths between p.c.'s



Term 4: $\sum_{i=1}^{n} \sum_{1 \leq s < t \leq k} d_{C_i}(u_i^s u_i^t) \cdot \left(N - n_{u_i^s}(u_i^s u_j^{s'})\right) \cdot \left(N - n_{u_i^t}(u_i^t u_\ell^{t'})\right)$

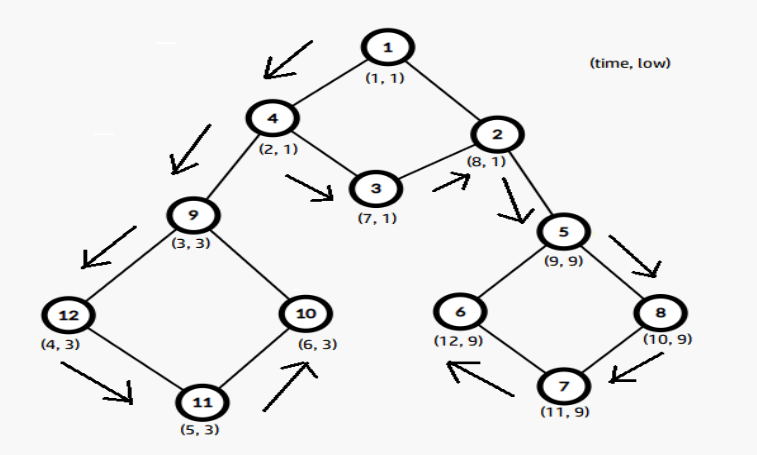
Contribution from edges in intermediate p.c.'s for paths between p.c.'s



CEPC Algorithm Code

1. Determine the pseudo-components and weighted tree structure of a given graph
    - Done by DFS (Depth-First Search)
    - Store the cut-edges and cut-vertices of a pseudo-component
2. Calculate the Wiener Index using the CEPC formula
    - DFS the weighted tree
    - calculate each term of the CEPC formula

Weighted Tree Structure



Tables

CEPC Algorithm works best with more pseudo-components

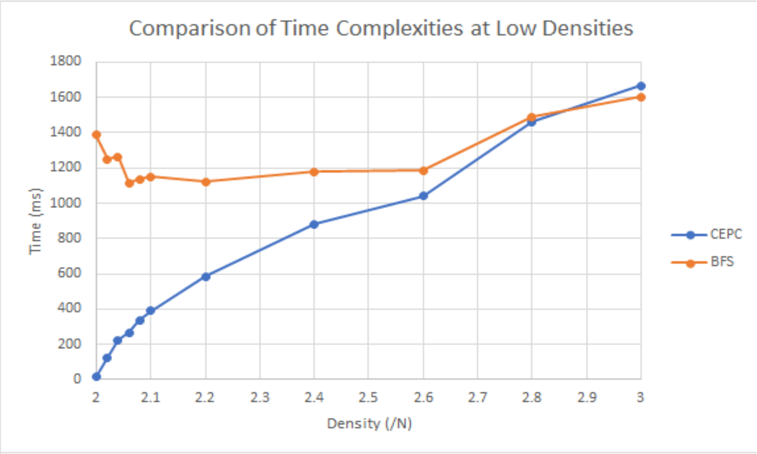
Table: CEPC Algorithm

Density (/N)	Time (ms)
2	16
2.02	125
2.04	225
2.06	264
2.08	335
2.1	388
2.2	584
2.4	880
2.6	1039
2.8	1462
3	1666
4	1928
5	1918
10	2472
15	3623
20	4153

Table: BFS Algorithm

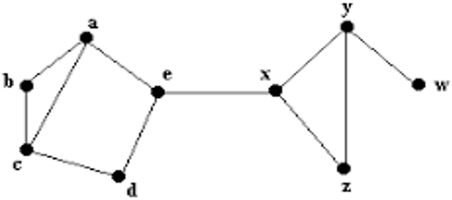
Density (/N)	Time (ms)
2	1387
2.02	1250
2.04	1264
2.06	1115
2.08	1136
2.1	1151
2.2	1123
2.4	1179
2.6	1184
2.8	1490
3	1604
4	1704
5	1624
10	2085
15	2980
20	3417

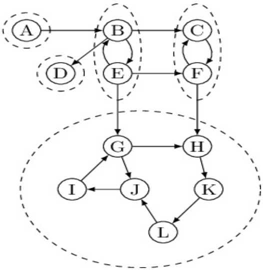
Graphs


Comparison of Time Complexities at Low Densities
Time (ms)
0
200
400
600
800
1000
1200
1400
1600
1800
2
2.1
2.2
2.3
2.4
2.5
2.6
2.7
2.8
2.9
3
Density (/N)
CEPC
BFS

Summary

- **Generalized trees to graphs with many cut-edges**
  - Consider pseudo-components as "nodes"
- Applied characteristics and existing algorithms for trees to analyze the Wiener Index of a general graph

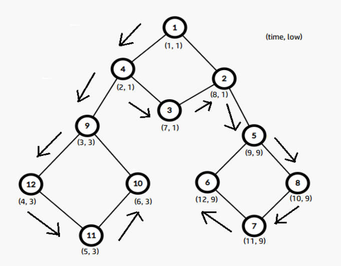


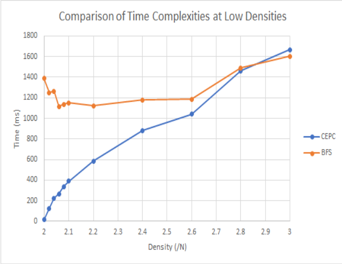


Summary

CEPC Algorithm:

- determined pseudo-components and weighted tree structure
- calculated Wiener Index using the characteristics of trees
  - $W(C_i)$ using the BFS algorithm
  - $VWW(T)$ for weighted trees as part of the CEPC Formula





Acknowledgements

My sincerest thanks to the following individuals:

- Hua Wang, PhD., Professor of Mathematics, Georgia Southern University
- Kevin Xiao, Westwood High School, Austin, TX
- Albert Jiang, Jasper High School, Dallas, TX
- My family and friends for their continued support

Thank you to everyone for listening!