Steiner trees

Algorithms and Networks

Universiteit Utrecht

Today

- Steiner trees: what and why?
- NP-completeness
- Approximation algorithms
- Preprocessing

Steiner tree

- ***Given***: connected undirected graph G=(V,E), length for each edge $l(e) \in \mathbf{N}$, set of vertices N: terminals
- ***Question***: find a subtree T of G, such that each vertex of N is on T and the total length of T is as small as possible
  - Steiner tree *spanning N*

Variants

- Points in the plane
- Vertex weights
- Directed graphs

Applications

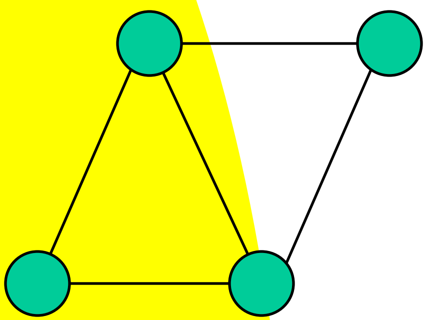
- Wire routing of VLSI
- Customer's bill for renting communication networks in the US
- Other network design and facility location problems

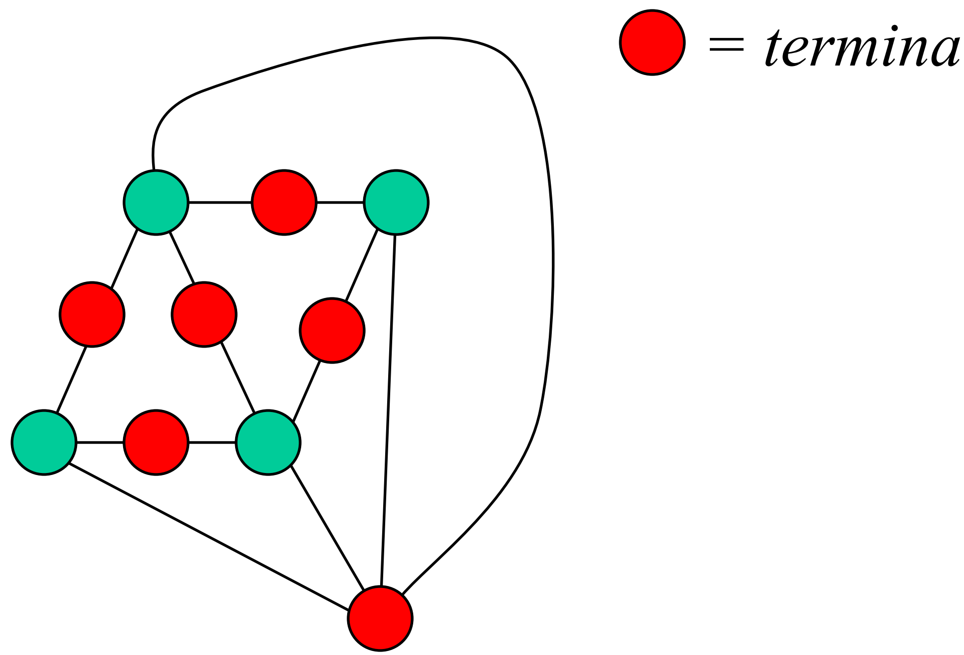
Special cases

- $|N| = 1$: trivial
- $|N| = 2$: shortest path
- $N = V$: minimum spanning tree

NP-completeness

- Decision version is NP-complete
- Vertex cover





Proof of reduction

- Membership of ST in NP: trivial
- Hardness: take instance G=(V,E), *k* of Vertex Cover
- Build G' by subdividing each edge
- Set *N* = set of new vertices
- All edges length 1
- G' has Steiner Tree with |E|+*k* – 1 edges, if and only if G has vertex cover with *k* vertices

Approximation algorithms

- Several different algorithms that guarantee ratio 2 (or, more precise: $2 - 2/n$).
- Shortest paths heuristic
  - Ration $2 - 2/n$ (no proof here)
  - Bases on Prim's minimum spanning tree algorithm

Shortest paths heuristic

- Start with a subtree T consisting of one terminal
- While T does not span all terminals
  - Select a terminal x not in T that is closest to a vertex in T.
  - Add to T the shortest path that connects x with T.

Improving the shortest paths heuristic

- Take the solution T from the heuristic
- Build the subgraph of G, induced by the vertices in T
- Compute a minimum spanning tree of this subgraph
- Repeat
  - Delete non-terminals of degree 1 from this spanning tree
  - Until there are no such non-terminals

Distance networks

- Distance network of G=(V,E) (induced by X)
- Take complete graph with vertex set X
  - Cost of edge $\{v,w\}$ in distance network is length shortest path from v to w in G.
- For set of terminals N, the minimum cost of a Steiner tree in G equals the minimum cost of a Steiner tree in the distance network of G (induced by V).

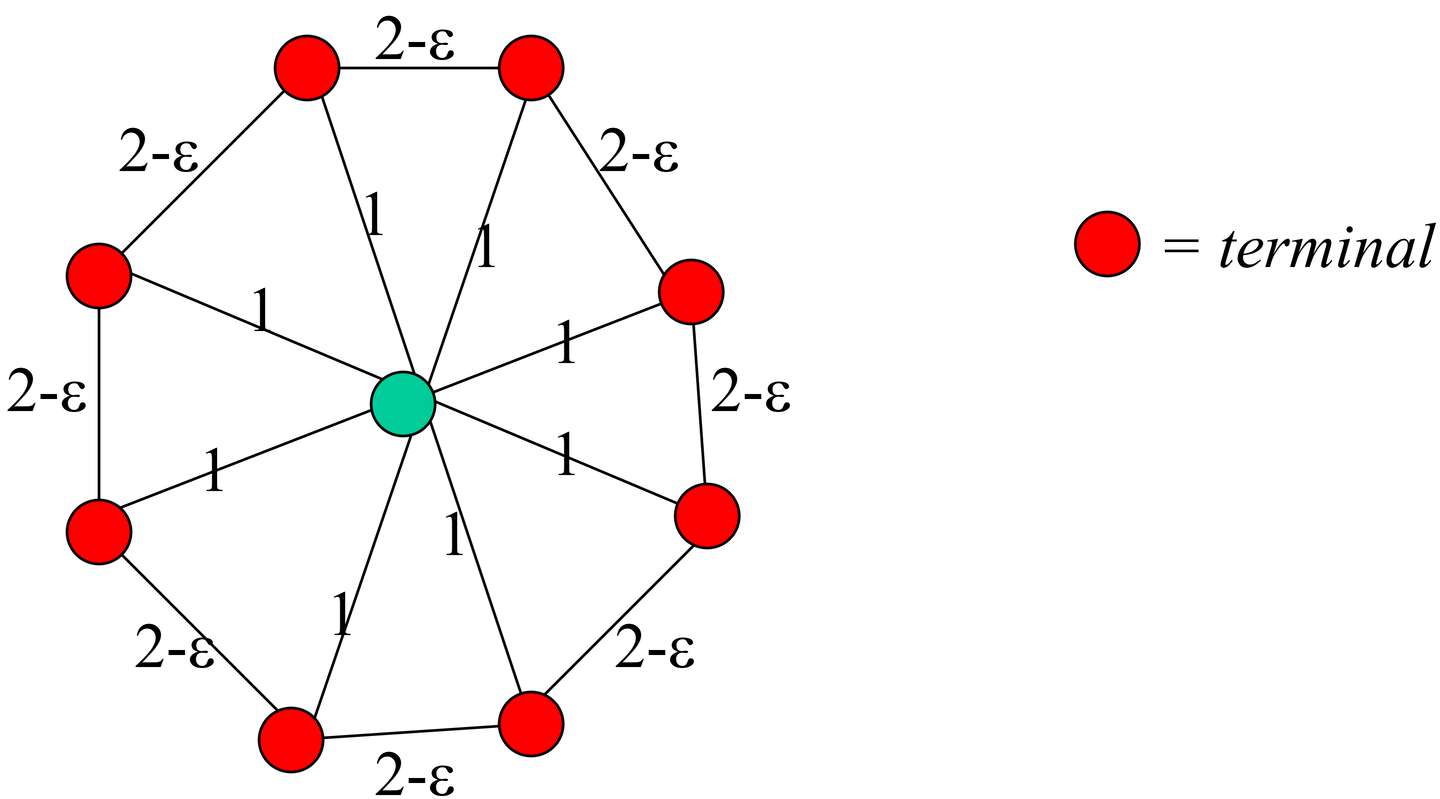
Distance network heuristic

- Construct the distance network $D_G(N)$ (induced by N)
- Determine a minimum spanning tree of $D_G(N)$
- Replace each edge in the minimum spanning tree by a corresponding shortest path.
  - Let T_D be the corresponding subgraph of G
  - It can be done such that T_D is a tree
- Make the subgraph of G induced by the vertices in T_D
- Compute a minimum spanning tree of this subgraph
- Remove non-terminals of degree 1 from this spanning tree, until there are no such non-terminals.

Distance network heuristic has ratio 2

- Look at optimal Steiner tree T*
- Take closed walk L around T* visiting each edge twice
- See this as a collection of paths between successive terminals
- Delete the longest of these, and we get a walk L'; cost of L' $\leq$ cost(T*) * (2 – 2/*r*)
- cost(T_D) $\leq$ cost(L').
  – L' is a spanning tree in $D_G(N)$
- Final network has cost at most cost(T_D).

Example where bound is met



Upgrading heuristic

- $W = \varnothing$; w = maxint; $D = D_G(N)$
- **repeat**
  - Identify set of three terminals $A=\{a,b,c\}$ such that $w=$ cost(T_D(N)) – cost($T_{D'}$(N')) – cost(T_G(A)) is as large as possible
    - D' (N') is obtained from D (N) by contracting A to one vertex
    - T_D(N) denotes min spanning tree of D
    - T_G(A) denotes min steiner tree in G with terminals A
  - **if** (w = = 0) **then** apply distance network heuristic with terminal set $W \cup N$; stop
  - **else** add to W the non-terminal of degree 3 in T_G(A); D=D'

On upgrading heuristic

- Correctness: when no non-terminal vertex of degree 3 in $T_G(A)$, then w=0
  - $T_D(N)$ can be constructed using edges of the other two
- Ratio: 11/7
- Other method:
  - +- 1.55 (Robins, Zelikowsky, 2000)

Small number of terminals

- Suppose $|N| = r$ is small.
- Compute distance network $D_G(V)$
- There is a minimum cost Steiner tree in $D_G(V)$ that contains at most $r - 2$ non-terminals.
  - Any Steiner tree has one that is not longer without non-terminal vertices of degree 1 and 2
  - A tree with r leaves and internal vertices of degree at least 3 has at most $r - 2$ internal vertices
- Polynomial time algorithm for Steiner tree when we have O(1) terminals.

Solving O(1) terminals

- Polynomial time algorithm for Steiner tree when we have O(1) terminals:
  - Enumerate all sets W of at most $r - 2$ non-terminals
  - For each W, find a minimum spanning tree in the distance network of $N \cup W$
  - Take the best over all these solutions
- Takes polynomial time for fixed r.
- Heuristics to do this more clever?

Simple preprocessing

- Steiner tree can be solved separately on each biconnected component
- Non terminals of degree at most 2:
  - Reduce graph:
    - Delete non-terminal of degree 1
    - Connect neighbors of non-terminals of degree 2
      - Edge length is sum of lengths of 2 edges
- Long edges can be deleted
  - If $l(v,w) > \mathrm{d}(v,w)$ then delete edge $\{v,w\}$.

Bottleneck Steiner distance

- Path between v and w can be seen as number of successive *elementary paths*
  - Pieces ending/starting at v, w or terminal
- Steiner distance of path: length of largest elementary path
- Bottleneck Steiner distance: minimum Steiner distance over all paths between v and w
- Can be computed with modification of shortest paths algorithm

Reducing non-terminals

- Consider non-terminal z. Consider network B(z), with vertex set N[z] and lengths the bottleneck Steiner distances.
- Write B(z)[W] for subnetwork of B(z) induced by W.
- **Lemma**. If for every subset W of N(z) of size at least 3, the cost of the minimum spanning tree of B(z)[W] is at most the cost of the edges $\{z,w\}$ over all $w \in$ W, then

  z has degree at most 2 in at least one minimum cost Steiner Tree

Use of lemma

- Remove z
- For pairs of neighbors v, w of z
  - If $\{v,w\} \in E$, set length of edge to minimum of cost(v,w), cost(v,z)+cost(z,w)
  - Otherwise, add edge $\{v,w\}$ with cost cost(v,z)+cost(z,w)

Long edges

- If $d(v,w) < cost(v,w)$, then edge $\{v,w\}$ can be removed.
- If $d(v,w) = cost(v,w)$, and there is a shortest path not via edge $\{v,w\}$, then edge can be removed.

Paths with one terminal

Suppose $\{v,w\}$ is an edge, and there is a terminal z with $\mathrm{cost}(v,w) > \max(d(v,z),d(w,z))$ then $\{v,w\}$ can be removed.

- A Steiner tree with $\{v,w\}$ can be improved: how can we repair a Steiner tree – $\{v,w\}$?

PTm-test (paths with many terminals)

- Let b(v,w) the bottleneck Steiner distance from v to w.
- If cost(v,w) > b(v,w) then edge $\{v,w\}$ can be removed.
- Proof.
  - Consider Steiner tree T1 with such edge $\{v,w\}$.
  - Look at T1 – $\{v,w\}$. Splits in two trees T2 and T3.
  - Consider bottleneck shortest path from v to w.
  - Take elementary path P0 with one edge in T2 and T3.
  - Length of P0 at most b(v,w).
  - T1 – $\{v,w\}$+ P0 has length less than T1 and spans all terminals
  - Take subgraph of P0 that spans all terminals and is a tree

Polynomial solvable cases

- When bounded treewidth
- E.g., for series parallel graphs
  - Compute for part with terminals *s* and *t*
    - Minimum cost subtree in part spanning *s* and *t*
    - Minimum cost subtree in part spanning *s*, but not *t*
    - Minimum cost subtree in part spanning *t*, but not *s*
    - Minimum cost subtree in part spanning neither *s* and *t*
    - Minimum cost of two subtrees, one spanning *s* and one spanning *t*
- Strongly chordal graphs with unit costs