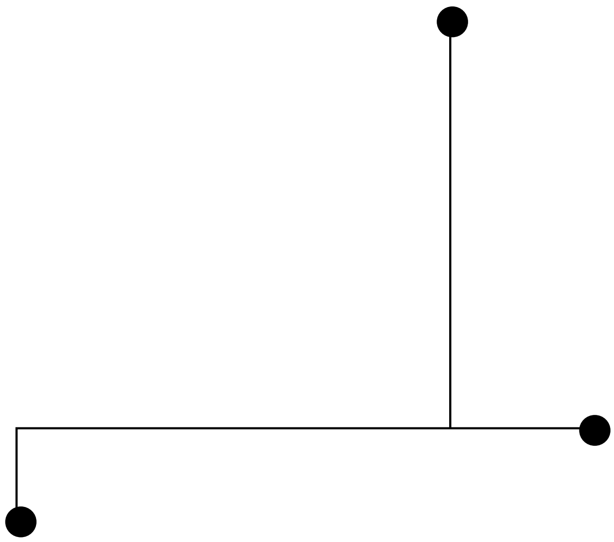
Rectilinear Steiner Trees in the Plane



Martin Zachariasen
University of Copenhagen

Outline

- Structural Properties
  - The Hanan Grid
  - Hwang-topology Full Steiner Trees (FSTs)
  - The Steiner Ratio
- FST Based Exact Algorithms
  - FST Generation
  - FST Concatenation
- Heuristics
- Delay-Optimal Rectilinear Steiner Trees
- Secondary Objectives for RSMTs

The Rectilinear Steiner Tree Problem in the Plane

- Definition: Given a set Z of terminals in the plane find a shortest interconnection of the terminals using the L_1 metric, i.e., only horizontal and vertical lines may be used.

- Complexity: **NP**-hard [Garey & Johnson, 1977].

- Approximation: Has a polynomial-time approximation scheme [Arora, 1996].

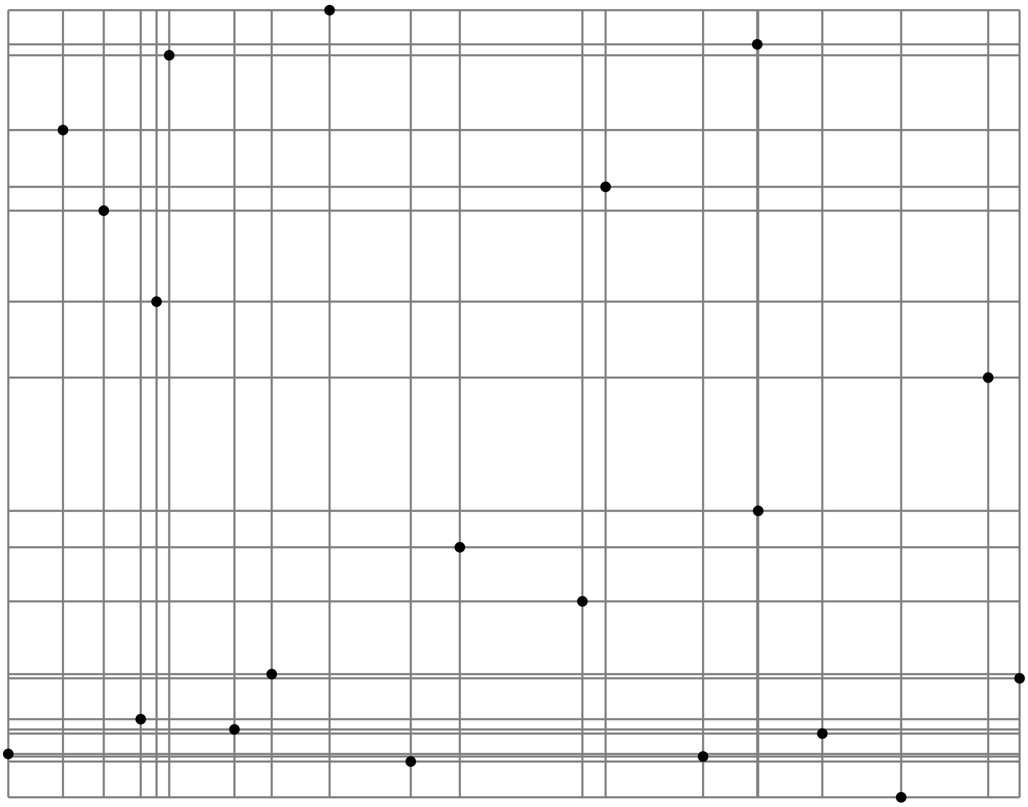
- Exact algorithms:

[Yang & Wing, 1972]	10
[Ganley & Cohoon, 1994]	20
[Salowe & Warme, 1995]	35
[Warme, 1997]	1000
[Warme & Zachariasen, 2000]	10000

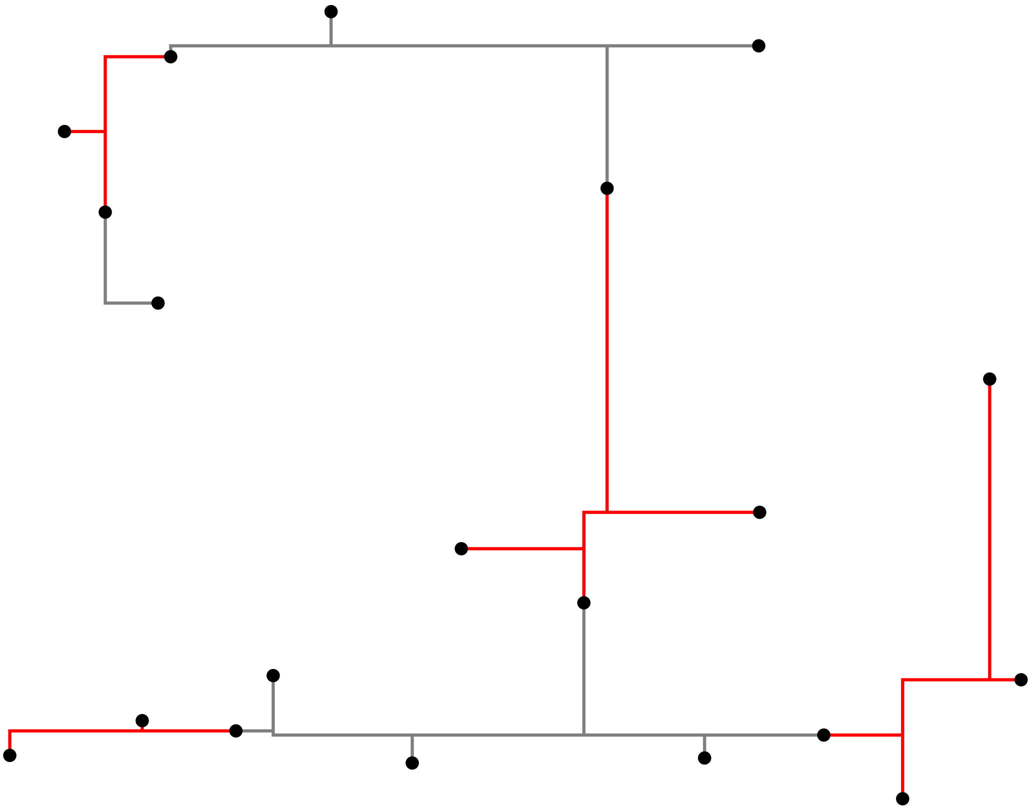
- Applications: VLSI design, printed circuit board layout etc.

The Hanan Grid

- The Steiner points for a rectilinear Steiner minimum tree (RSMT) may be confined to the vertices of the grid graph for Z [Hanan, 1966].



- A *full Steiner tree (FST)* is a Steiner tree in which all terminals are leaves. An RSMT is a union of FSTs.



4-a

Fulsome and Canonical RSMTs

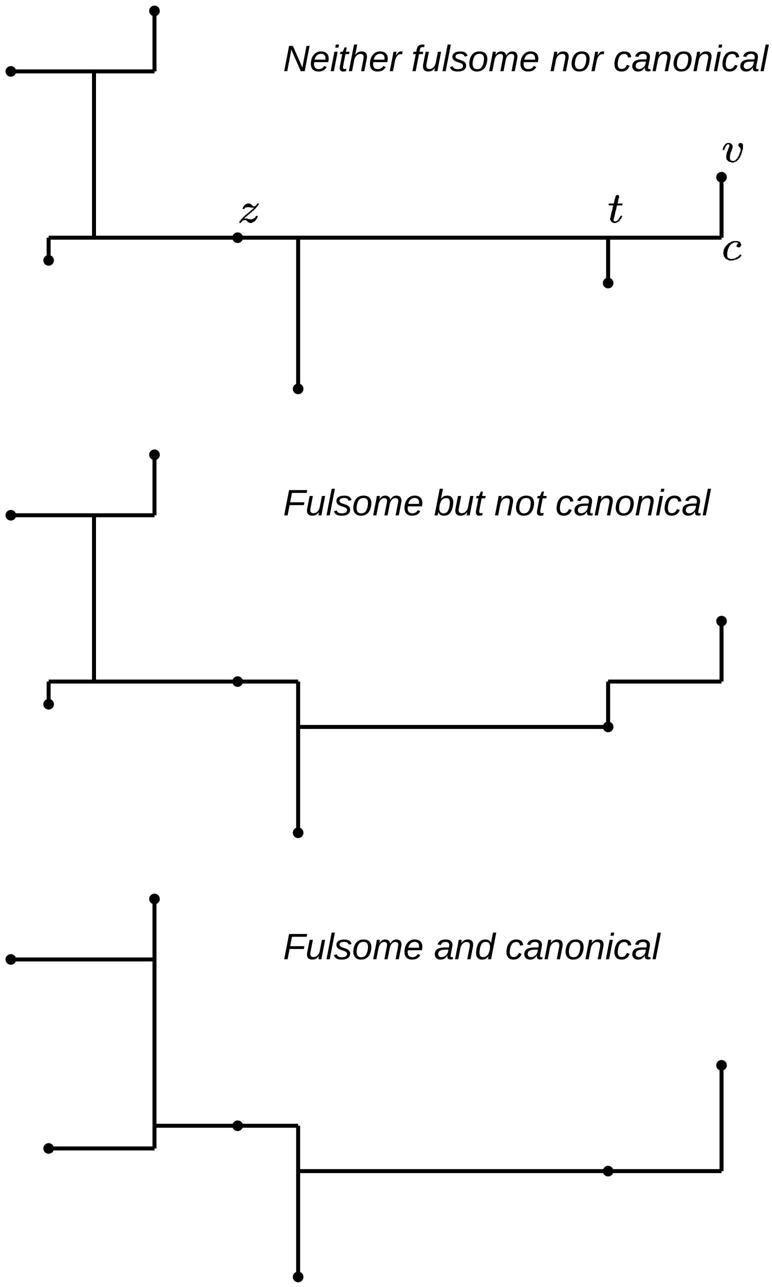
Fulsome RSMT

The number of FSTs is maximized. In particular, no FST can be split into two or more FSTs.

Canonical RSMT

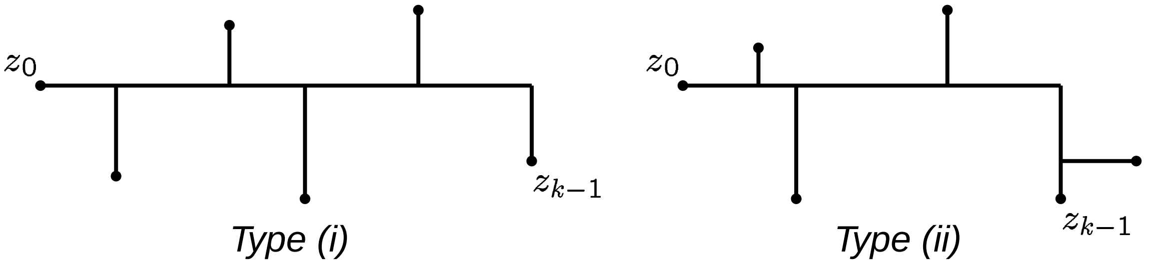
No vertical segment can be moved to the right using sliding and/or flipping operations.

Fulsome and Canonical RSMTs - An Example



Hwang-topology FSTs

- An FST has one of two generic forms [Hwang, 1976]:



- An FST spanning k terminals consists of a corner given by a root z_0 and a tip z_{k-1}. The root is incident to the *long* leg and tip incident to the *short* leg of the corner.

- Type (i) has $k-2$ alternating segments incident to the long leg and no segment incident to the short leg.

- Type (ii) has $k-3$ alternating segments incident to the long leg and one segment incident to the short leg.

- Degenerate cases:

  – Type (i′) has a zero-length short leg, i.e., the corner is degenerated into a line.

  – Type (i″) is a cross spanning exactly four terminals (the two alternating incident segments are on the same line).

The Steiner Ratio

For a given set of terminals Z:

$|RSMT(Z)|$: Length of RSMT for Z

$|RMST(Z)|$: Length of a minimum spanning tree for Z under the L_1 metric

Steiner ratio for the L_1 metric in the plane:

$$\rho_1 = \inf_Z \frac{|RSMT(Z)|}{|RMST(Z)|}$$

Theorem [Hwang, 1976] $\rho_1 = \frac{2}{3}$

FST Based Exact Algorithm

- Phase 1: FST Generation
  - Generate a set $\mathcal{F} = \{F_1, F_2, ..., F_m\}$ of FSTs such that there exists an RSMT identified as a subset $\mathcal{F}^* \subseteq \mathcal{F}$.
  - The first rectilinear FST-generation algorithm was suggested by [Salowe & Warme, 1995]; improved algorithm given by [Zachariasen, 1999].
- Phase 2: FST Concatenation
  - Find a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that the FSTs in $\mathcal{F}^*$ interconnect all terminals and the total length is as small as possible. Not metric dependent.
  - Equivalent to solving a minimum spanning tree problem in a hypergraph.
  - Can be formulated as an integer program which is solved by linear program relaxation and branch-and-cut [Warme, 1997].

Necessary Optimality Conditions

The *bottleneck Steiner distance*, $BSD(z_i, z_j)$, between two terminals z_i and z_j is equal to the length of the longest edge on the (unique) path between z_i and z_j in $RMST(Z)$.

Let F be an FST with length $|F|$ spanning $Z_F \subseteq Z$. Assume that F is a subtree of an RSMT spanning Z.

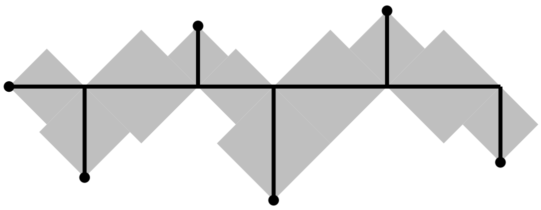
1. F is an RSMT for Z_F. Furthermore we may assume that there exist no union of smaller FSTs spanning Z_F and having total length $|F|$.

2. Let $z_i, z_j \in Z_F$. The longest edge on the (unique) path between z_i and z_j in F cannot be longer than $BSD(z_i, z_j)$.

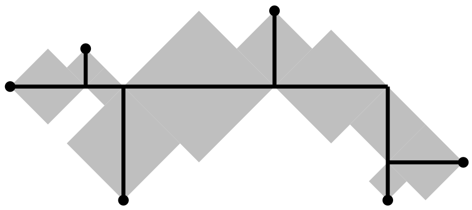
3. $|F|$ cannot be greater than the length of an MST for Z_F using distances $BSD(z_i, z_j)$ between every pair of terminals.

Empty Regions

- Empty diamonds

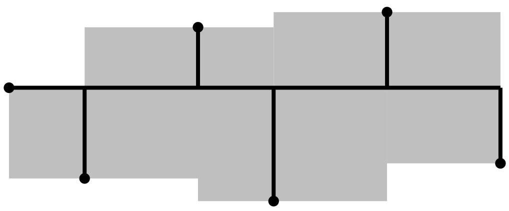


Type (i)

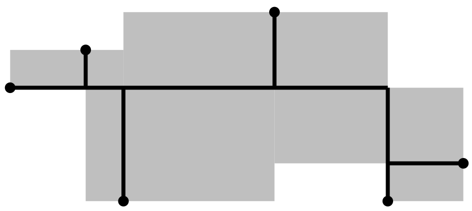


Type (ii)

- Empty corner rectangles

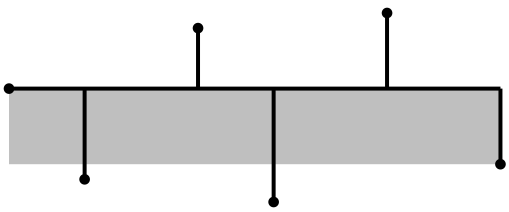


Type (i)

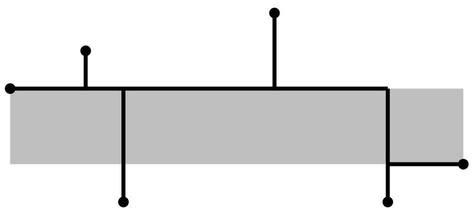


Type (ii)

- Empty inner rectangle



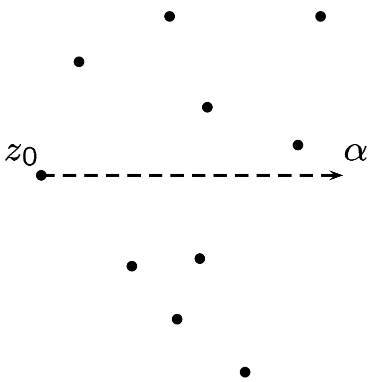
Type (i)



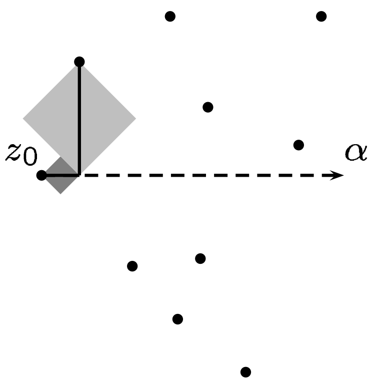
Type (ii)

Growing Rectilinear FSTs

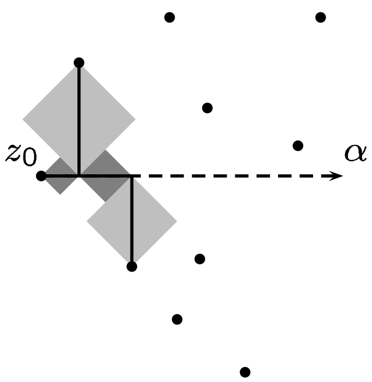
For every terminal $z_0 \in Z$ and direction $\alpha \in \{North, East\}$ try to *grow* an FST with z_0 as root and having its long leg oriented in direction α.



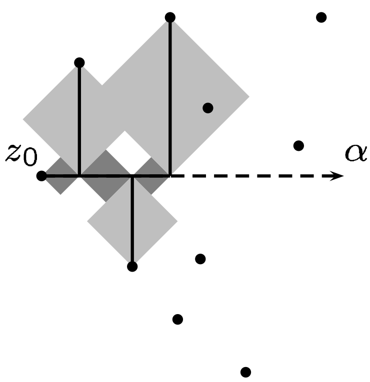
Root and direction



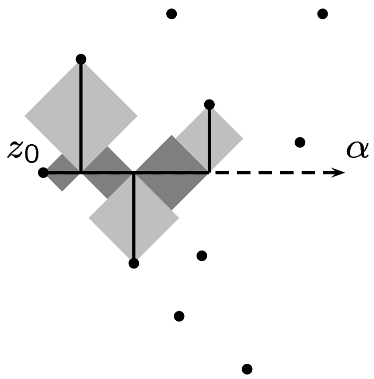
Save and recurse



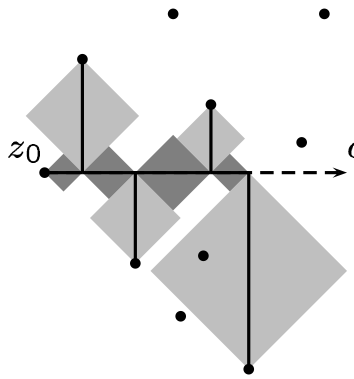
Save and recurse



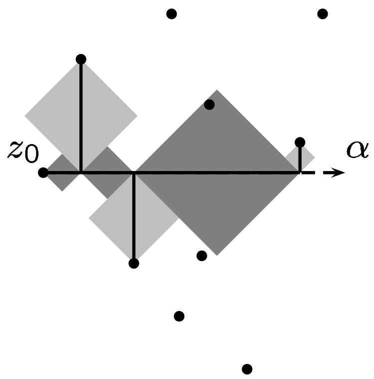
Skip and continue



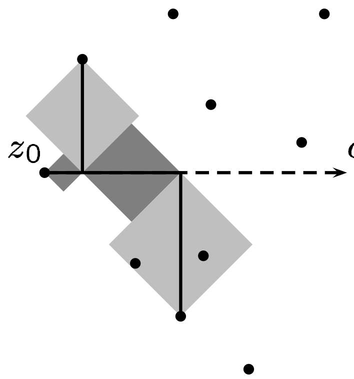
Save and recurse



Backtrack



Backtrack



Skip and continue...

By performing FST-independent *preprocessing* the number of long leg candidates for every (z_0, α) pair can be reduced and efficient pruning can be applied.

Bounds on the Number of FSTs

Crucial question: How many FSTs survive in the generation phase?

- Empirically only a linear number of FSTs survive (approximately $4n$ for the rectilinear problem).

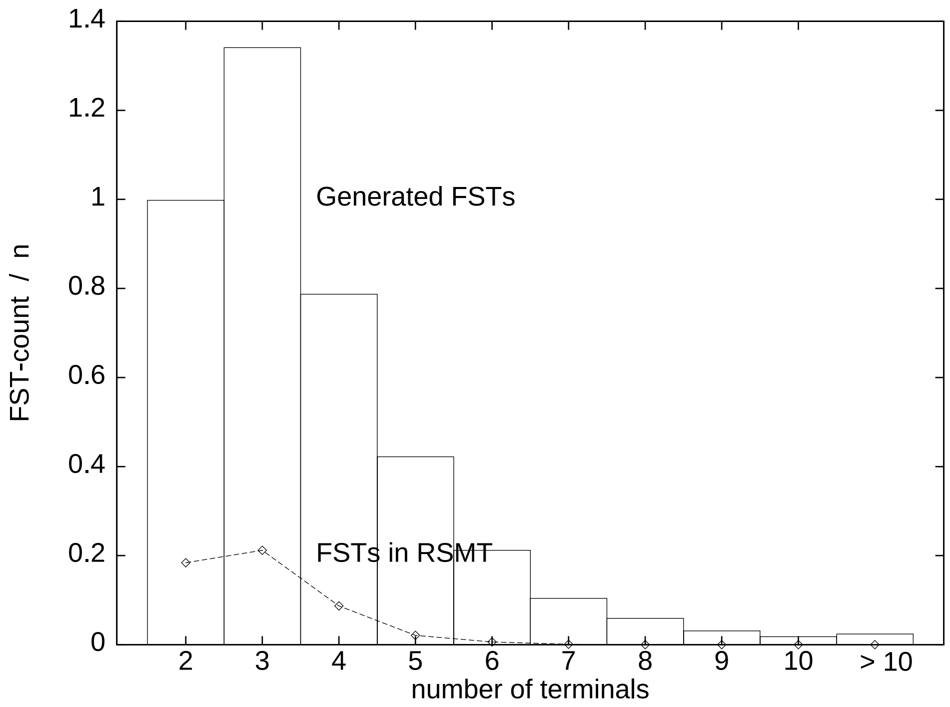
- The best worst-case bound for the rectilinear problem is exponential and “tight” in a certain sense [Fößmeier & Kaufmann, 1996].

- For randomly generated instances the *expected* number of rectilinear FSTs spanning up to K terminals is bounded by

$$O(n(\log\log n)^{K-2})$$

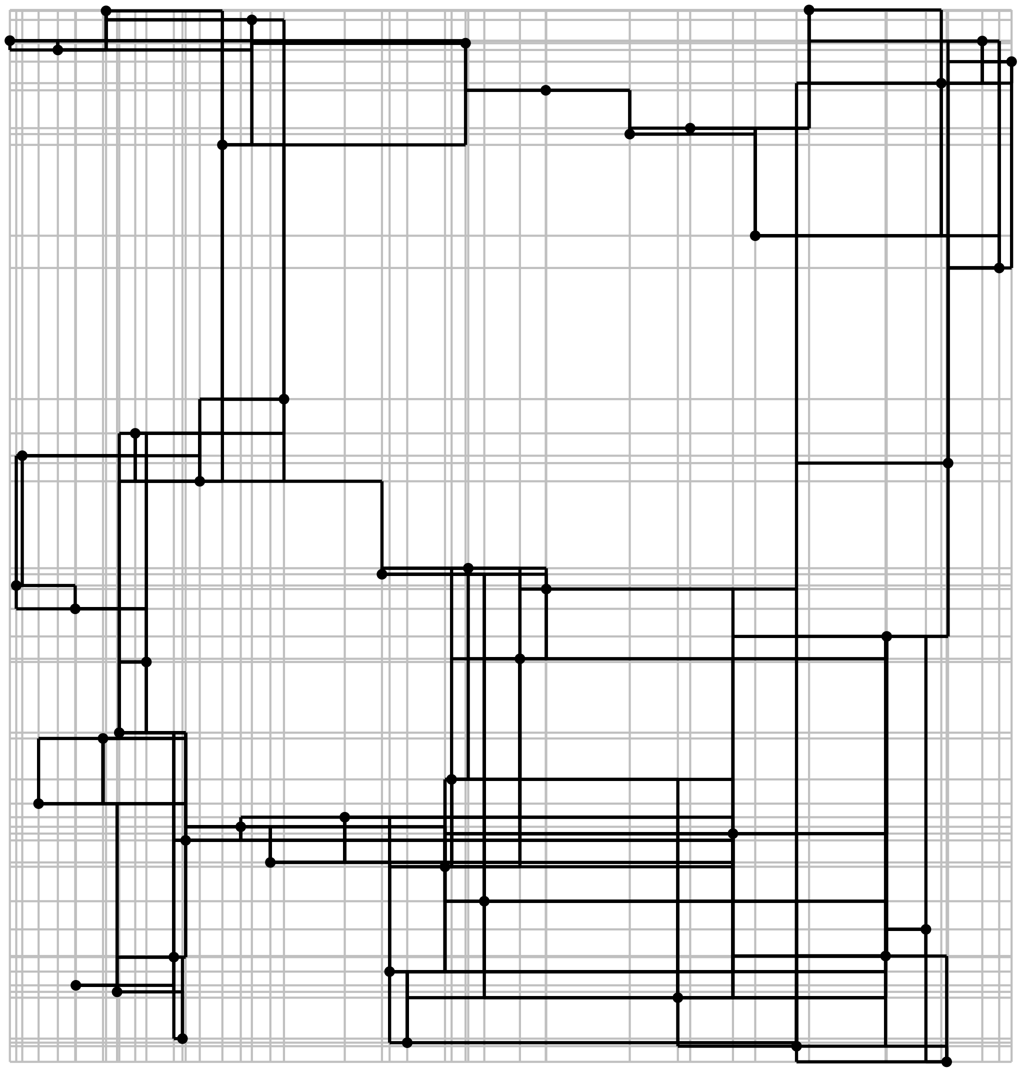
  where $K \geq 3$ is a constant.

FST-size Distribution



Hanan Grid Graph Reduction by FST Generation

FST generation can be used as a grid graph reduction algorithm by overlaying the generated FSTs on the Hanan grid.

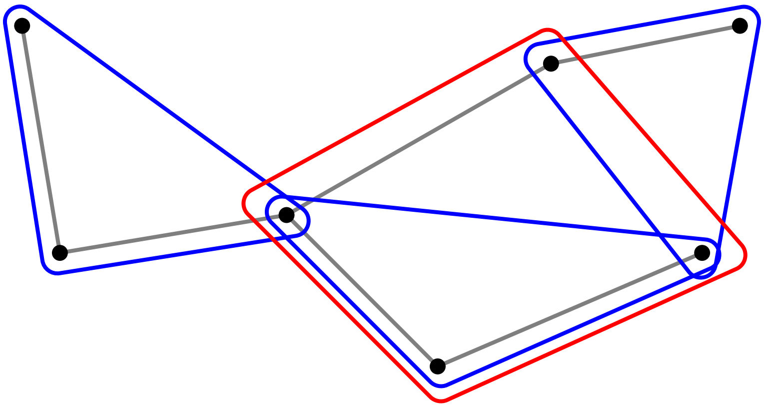


FST Concatenation

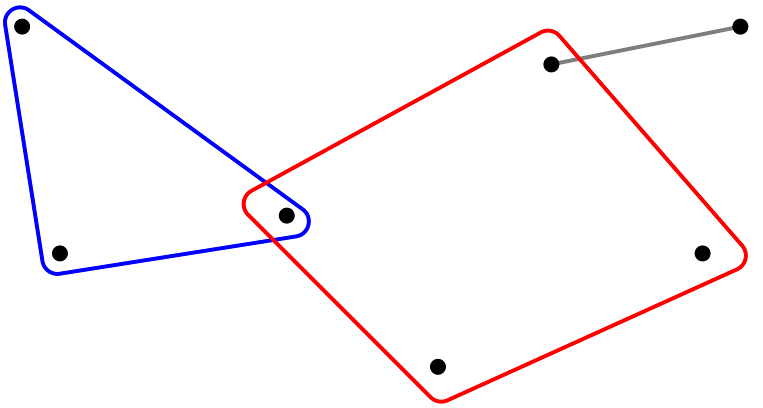
- Given a set $\mathcal{F} = \{F_1, F_2, ..., F_m\}$ find a subset $\mathcal{F}^* \subseteq \mathcal{F}$ such that the FSTs in $\mathcal{F}^*$ interconnect all terminals and the resulting tree is as short as possible.
- Can be solved using backtrack search, dynamic programming or integer programming.
- Equivalent to finding an MST in the (edge-weighted) *hyper*graph $(Z, \mathcal{F})$; terminals are vertices in the graph and FSTs *hyper*edges.
- The general MST in hypergraph problem is **NP**-hard when there are hyperedges spanning four or more terminals.

The Spanning Tree in Hypergraph Problem

Hypergraph example



Spanning tree example



There should be a *unique chain* between every pair of vertices.

FST Concatenation by Integer Programming

- For each FST $F_i \in \mathcal{F}$ let
  - Z_i be the set of terminals spanned by F_i
  - c_i be the length of F_i
  - $x_i \in \{0, 1\}$ be a binary variable such that $x_i = 1$ if and only if $F_i \in \mathcal{F}^*$.
- For any subset $S \subset Z,\ S \neq \emptyset$ denote by

  $$(S : Z \setminus S) \subseteq \{1, \ldots, m\}$$

  the indices of FSTs spanning terminals in both S and $Z \setminus S$.
- Connectivity can be enforced by requiring that

  $$\sum_{i \in (S:Z \setminus S)} x_i \quad \geq \quad 1, \ \forall S \subset Z$$

Integer Programming Formulation

$$\min cx \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in \{1,\ldots,m\}} (|Z_i| - 1) x_i \quad = \quad n - 1 \tag{2}$$

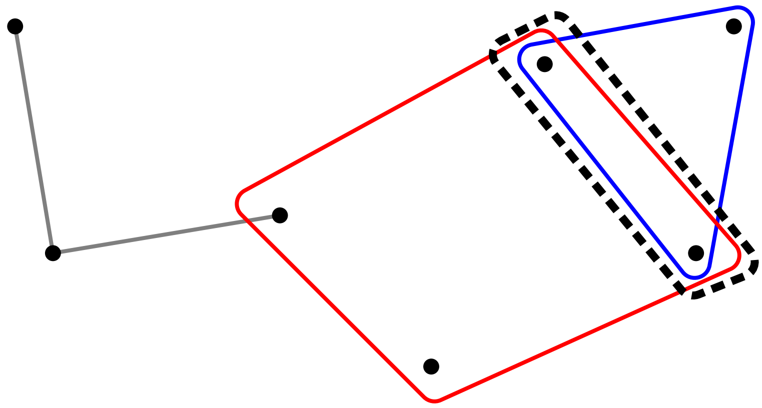
$$\sum_{i \in \{1,\ldots,m\}} \max(0, |Z_i \cap S| - 1) x_i \quad \leq \quad |S| - 1, \ \ \forall S \subset Z, |S| \geq 2 \tag{3}$$

- Equation (2) enforces the right number and cardinality of hyperedges to construct a spanning tree; constraints (3) eliminate cycles.

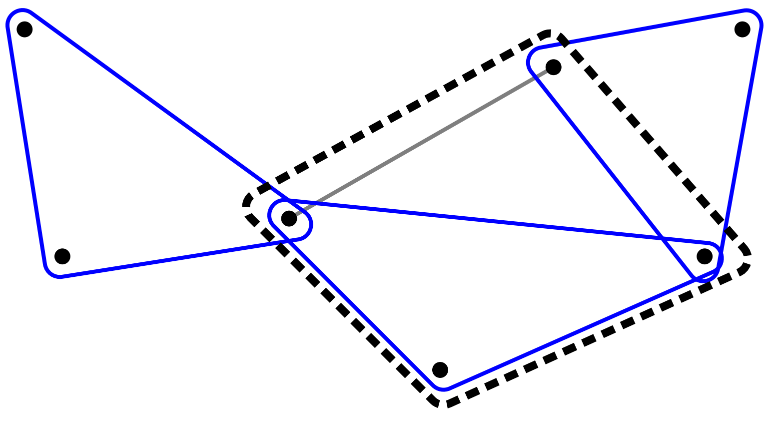
- Together these constraints dominate the cutset-constraints, that is, ensure that the solution is a spanning hypertree.

Cycle-Elimination Constraints

Example with $|S| = 2$



Example with $|S| = 4$



Solving the IP Formulation

- The integer program is solved via branch-and-cut using a linear program relaxation. Cycle-elimination constraints (3) are added via iterations of optimization followed by separation.

- Initial polyhedron:

$$
\begin{aligned}
\sum_{i\in\{1,\ldots,m\}} (|Z_i|-1)x_i &= n-1 \\
\sum_{i\in\{1,\ldots,m\}} \max(0,|Z_i\cap S|-1)x_i &\leq |S|-1,\ \forall S\subset Z,\ |S|=2 \\
\sum_{i\in(\{z\}:Z\setminus\{z\})} x_i &\geq 1,\ \forall z\in Z \\
x_i + x_j &\leq 1,\ F_i \text{ incompatible with } F_j \\
0 \leq x_i &\leq 1,\ i\in\{1,\ldots,m\}
\end{aligned}
$$

- Cycle-elimination constraints are separated by finding a minimum cut in a certain graph (e.g., by solving a maximum flow problem).

Heuristics

Most heuristics first compute an MST for Z — and then try iteratively (in a greedy manner) to improve this tree:

- iterative embedding
- Steinerization
- Delaunay triangulation-based
- iterative 1-Steiner heuristic

Other approaches:

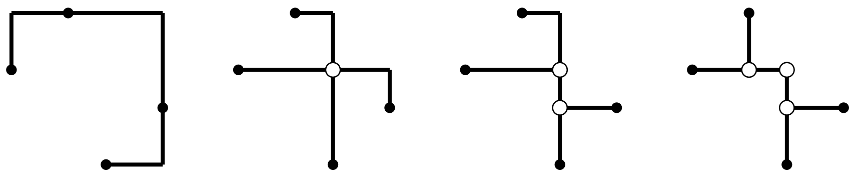
- Prim- and Kruskal-based plane sweep
- partitioning
- Arora's PTAS

Iterated 1-Steiner Heuristic

---

1. $V = Z$; compute $RMST(V)$
2. Find a point s such that $|RMST(V \cup \{s\})|$ is minimized.
3. if $|RMST(V \cup \{s\})| < |RMST(V)|$ then set $V = V \cup \{s\}$ and goto 2; otherwise stop

Step 2 can be performed in $O(n^2)$ time by partitioning the plane into *isodendral regions*.



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

Delay-Optimal Rectilinear Trees

Electrical signal originates from one of the given terminals denoted the source $r \in Z$.

Signal delay from r to some other terminal $z_i \in Z$ in a tree T depends on:

- total length of T
- length $|rz_i|_T$ of path from r to z_i in T
- topology of T

For a given tree T the delay can be computed quite accurately using simulation tools, e.g., SPICE.

Reasonable delay-approximation: Elmore delay function

Elmore Delay Function - Definitions

For every terminal z_i we are given:

c_{z_i}: capacitance of z_i (what is the load of z_i?)

r_d: on-resistance of the output driver at the source
(how powerful is the driver?)

For a given node v in a tree T define:

e_v: edge from v to parent in T

r_{e_v}: resistance of edge e_v (linear in length of edge)

c_{e_v}: capacitance of edge e_v (linear in length of edge)

T_v: subtree rooted at v

C_v: capacitance of T_v
(sum of terminal and edge capacitances)

$r_{e_v}(\frac{c_{e_v}}{2} + C_v)$: Elmore delay of edge e_v

Elmore Delay Function

Elmore delay at terminal $z_i \in Z$:

$$t_{ED}(z_i) = r_d C_r + \sum_{e_v \in p_T(r,z_i)} r_{e_v}\left(\frac{c_{e_v}}{2} + C_v\right)$$

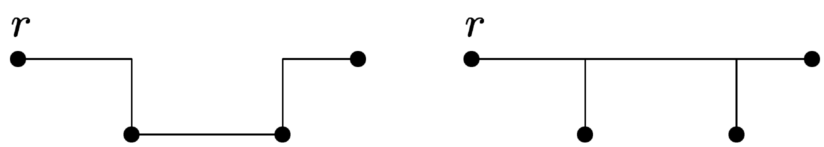
Typical objectives:

- minimize maximum delay over all terminals
- minimize weighted delay of the terminals

Secondary Objectives for RSMTs

Among all RSMTs for Z, minimize a secondary objective related to the signal delay from the source, e.g.,

- Maximum path length
- Weighted path length
- Maximum Elmore delay
- Weighted Elmore delay



Weighted Path Length Secondary Objective

Define for tree T:

- $|rz_i|_T$: length of path from r to z_i in T
- w_i: positive weight for terminal $z_i \in Z \setminus \{r\}$

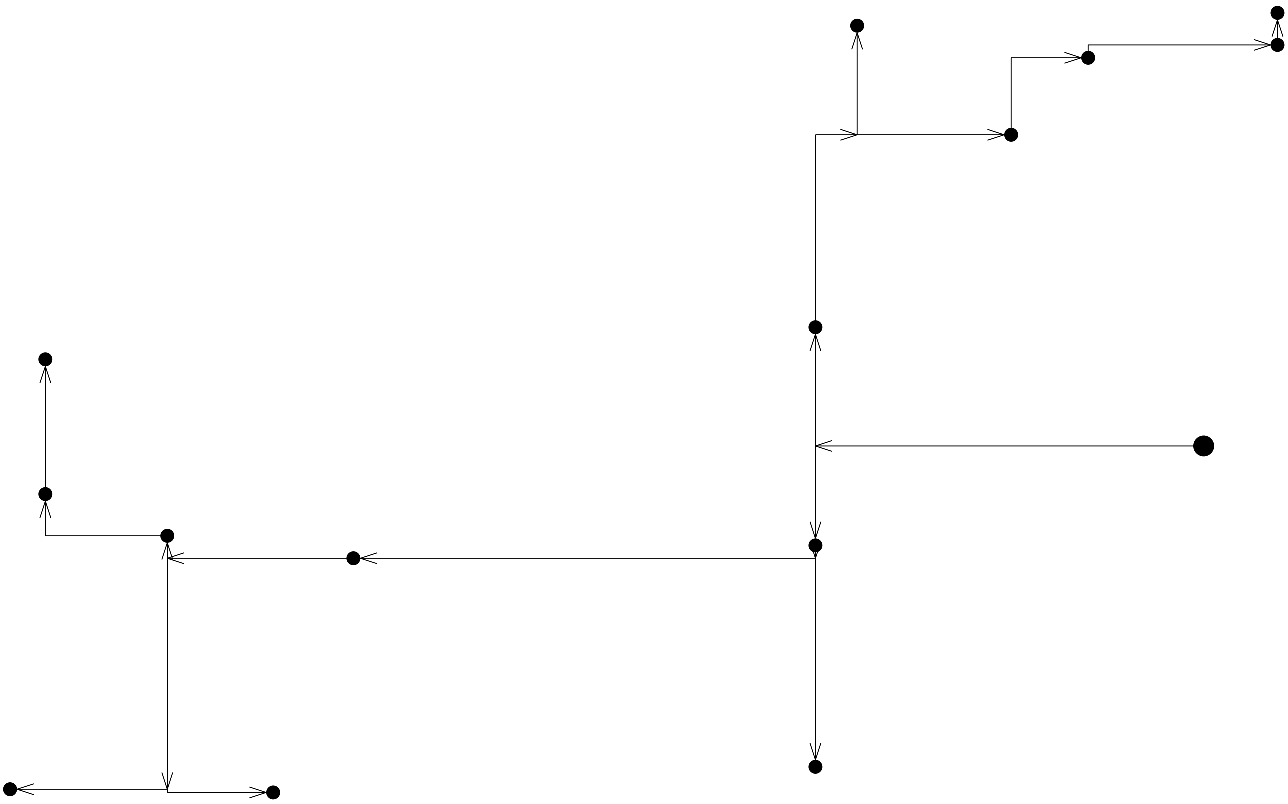
Rectilinear Steiner tree problem with weighted sum of path lengths secondary objective (RSTPWP):

Construct an RSMT T such that

$$\sum_{z_i \in Z \setminus \{r\}} w_i |rz_i|_T$$

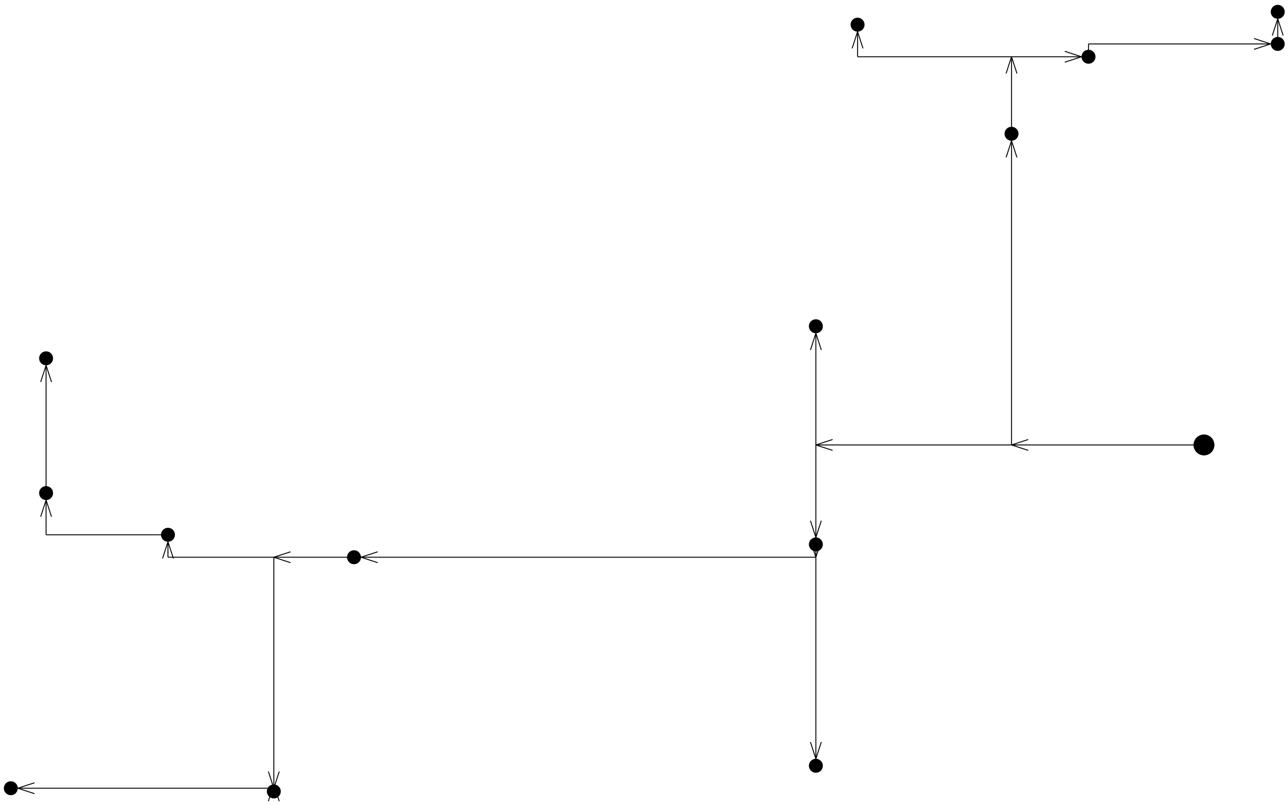
is minimized.

An RSMT



Tree length: 991   Total path length: 4674

Optimal Solution to RSTPWP



Tree length: 991   Total path length: 4094

Union of *directed* full Steiner trees

Exact Algorithm for RSTPWP

Optimal solution has to be in Hanan grid for Z.

Consider edge-weighted directed Hanan grid graph $G = (V, E)$. Two decision variables defined for each edge $(u, v) \in E$:

- $x_{uv} \in \{0, 1\}$ indicates whether the edge is chosen to be part of the Steiner tree
- $f_{uv} \geq 0$ is amount of flow traversing the edge

Integer Programming Formulation

$$
\begin{aligned}
\min \sum_{(u,v)\in E} & \quad c_{uv}\,(x_{uv} + f_{uv}) \\
\text{s.t.} \quad \sum_{(u,v)\in\delta(S)} x_{uv} & \geq 1, \ \forall S \subset V,\ r \in S,\ (V \setminus S) \cap Z \neq \emptyset \\
\sum_{(u,v)\in E} f_{uv} - \sum_{(v,u)\in E} f_{vu} & = D_v, \quad \forall v \in V \setminus \{r\} \\
f_{uv} & \leq x_{uv}, \quad \forall (u,v) \in E \\
x_{uv} & \in \{0,1\},\ \forall (u,v) \in E \\
f_{uv} & \geq 0, \quad \forall (u,v) \in E
\end{aligned}
$$

$$\delta(S) := \{(u,v) \in E \ : \ u \in S \ \wedge \ v \in V \setminus S\}$$

Similar to directed formulation for the Steiner tree problem in graphs; added flow measures secondary objective.

$D_{z_i} := w_i/(CW)$ where C is an upper bound on any path length and $W := \sum_{z_i \in Z\setminus\{r\}} w_i$;
$D_v := 0$ for $v \in V \setminus Z$.