Routing Topology Algorithms

Mustafa Ozdal
Introduction

• How to connect nets with multiple terminals?
• Net topologies needed before point-to-point routing between terminals.

• Several objectives:
  – Minimum wirelength
  – Best timing
  – Routability
Example

(receiver)

(driver)
Example – Star topology (suboptimal)

Connect each receiver to the driver independently.
Example – Min Wirelength Topology

- : receiver
- : driver
Outline

• Definitions and basic algorithms
  – Minimum Spanning Trees (MST)
  – Steiner Trees
  – Rectilinear Steiner Trees

• Wirelength vs timing tradeoff
Minimum Spanning Tree (MST)

• Consider a connected graph $G = (V, E)$
  – $V$: terminals
  – $E$: potential connections between terminals
  – $w(e)$: wirelength of edge $e$

• MST: The set of edges $E_T$ such that:
  – $E_T$ is a subset of $E$
  – The graph $T = (V, E_T)$ is connected
  – The total edge weight of $E_T$ is minimum
MST Example

• 5 vertices
• 10 edges
• Weight of edge $e$ is the Manhattan distance of $e$
• What is the MST?
MST Example

- The edge set $E_T$:
  - $W(E_T)$ is minimum
  - $T = (V, E_T)$ is still connected

- Note: $T = (V, E_T)$ must contain $n$ vertices and $n-1$ edges.
Kruskal’s MST Algorithm

1. Initialize T as empty set

2. Define a disjoint set corresponding to each vertex in V.

3. Sort edges in non-decreasing order of weights

4. For each e = (v1, v2) in the sorted edge list
   1. If v1 and v2 belong to different sets
      1. Add e to T
      2. Merge the sets corresponding to v1 and v2

5. Return T
Kruskal’s MST Algorithm - Example

- Initially, each vertex is a disjoint set (different color)
- We will process edges from shortest to longest
Kruskal’s MST Algorithm – Example

Step 1

• Start from the shortest edge

• The vertices connected are in different sets

• Add the edge to MST

• Merge the vertices
Kruskal’s MST Algorithm - Example

Step 2

- Process the next shortest edge.
- The vertices connected are in different sets.
- Add the edge to MST.
- Merge the vertices.
Kruskal’s MST Algorithm – Example

Step 3

• Process the next shortest edge.
• The vertices connected are in different sets.
• Add the edge to MST.
• Merge the vertices.
Kruskal’s MST Algorithm – Example

Step 4

- Process the next shortest edge.
- The vertices connected are in different sets.
- Add the edge to MST.
- Merge the vertices.
Kruskal’s MST Algorithm – Example
Step 5

• Process the next shortest edge.
• The vertices connected are in the same set
• Skip the edge
Kruskal’s MST Algorithm – Example

Step 6

- Process the next shortest edge.
- The vertices connected are in different sets.
- Add the edge to MST.
- Merge the vertices.
Kruskal’s MST Algorithm – Example

Step 7

- All vertices are connected
- MST edges are highlighted
Prim’s MST Algorithm

1. Initialize $V_T$ and MST to be empty set

2. Pick a root vertex $v$ in $V$ (e.g. driver terminal)

3. Add $v$ to $V_T$

4. While $V_T$ is not equal to $V$
   1. Find edge $e = (v_1, v_2)$ such that
      1. $v_1$ is in $V_T$
      2. $v_2$ is NOT in $V_T$
      3. weight of $e$ is minimum
   2. Add $e$ to MST
   3. Add $v_2$ to $V_T$

5. Return MST
Prim’s MST Algorithm - Example

- Initially, $V_T$ contains the root vertex (e.g. driver)
Prim’s MST Algorithm - Example

- Pick the shortest edge between $V_T$ and $V - V_T$
- Add that edge to MST
- Expand $V_T$
Prim’s MST Algorithm - Example

• Pick the shortest edge between $V_T$ and $V - V_T$
• Add that edge to MST
• Expand $V_T$
Prim’s MST Algorithm - Example

- Pick the shortest edge between $V_T$ and $V - V_T$
- Add that edge to MST
- Expand $V_T$
Prim’s MST Algorithm - Example

- Pick the shortest edge between \( V_T \) and \( V - V_T \)
- Add that edge to MST
- Expand \( V_T \)
Prim’s MST Algorithm - Example

• Pick the shortest edge between $V_T$ and $V - V_T$

• Add that edge to MST

• Expand $V_T$
Prim's MST Algorithm - Example

- Pick the shortest edge between \( V_T \) and \( V - V_T \)
- Add that edge to MST
- Expand \( V_T \)
Prim’s MST Algorithm - Example

- All vertices are included in $V_T$
- MST edges are highlighted
Find the min-cost edge set that connects a given vertex set.

Possible to solve it optimally in $O(E \log E)$ time
  - Kruskal’s algorithm
  - Prim’s algorithm

In general Prim’s algorithm is better to control timing tradeoffs because we expand a wavefront from the driver.
Steiner Trees

• Similar to MSTs, but:
  – Extra intermediate vertices can be added to reduce wirelength.

MST

Steiner tree

Steiner point
Rectilinear Steiner Trees

- Steiner trees of which edges are all Manhattan
  - i.e. The routing of the slanted edges are all pre-determined

Steiner tree

Rectilinear Steiner tree
Steiner Tree Algorithms

• Steiner tree problem is NP-complete
  – Most likely there’s no polynomial time optimal algorithm
  – Note: MST problem can be solved optimally in $O(E \log E)$

• Many Steiner tree heuristics
  – Iteratively add Steiner points to an MST
  – Route each edge of MST allowing Steiner points be created in the process.
  – Exponential time algorithms: Based on ILP, SAT, SMT solvers
  – A popular and practical algorithm: FLUTE

FLUTE for Steiner Tree Generation

Example

- “Press” terminals from each side
FLUTE for Steiner Tree Generation

Example

- Press from left
- From the leftmost terminal to the second leftmost one.
FLUTE for Steiner Tree Generation

Example

- Create a Steiner edge corresponding to pressing edge
- Create a Steiner point at the new location

It is proven that pressing maintains optimality of Steiner tree.
FLUTE for Steiner Tree Generation

Example

- Press from top
FLUTE for Steiner Tree Generation
Example

- Press from top
FLUTE for Steiner Tree Generation

Example

- Press from right
FLUTE for Steiner Tree Generation

Example

- Press from bottom
FLUTE for Steiner Tree Generation

Example

- Problem is reduced to the rectangle at the center.
- How to connect these 4 Steiner (orange) points?
FLUTE for Steiner Tree Generation

Example

- FLUTE pre-computes all Steiner tree solutions for such rectangles up to 10 terminals.

- After pressing, if there are less than 10 nodes, returns the solution from database.
FLUTE for Steiner Tree Generation

• The solutions for simple problems are stored in a database.

• After pressing operations, if the problem is turned into one of those in the database, the best solution in the database is returned.

• If not in the database, use reduction heuristics.

Canonical solutions stored in database
FLUTE for Steiner Tree Generation

Example

• Solution inside the center rectangle is chosen from the database.
FLUTE for Steiner Tree Generation

Example

• Final rectilinear Steiner tree
Cost Metrics

• Many tradeoffs to consider for routing topologies

• Topology with best wirelength can have poor timing

• Topology with best wirelength and timing may not be routable
Example

Wirelength/Timing Tradeoff

: receiver

: driver
Example
Min Wirelength

: receiver
: driver
Example
Min Wirelength

Large delay to receiver

: receiver
: driver
Example

Better Timing – Worse Wirelength