9.1

Minimum Cost Spanning Tree (MCST) problem

Katta G. Murty, IOE 612 Lecture slides 9

Originally studied for designing min cost connecting grid (in distribution, transportation, communication applications) to connect a set of cities. Under arc lengths > 0 min cost connecting network will be a spanning tree (ST). Earliest algo. (Boruvka's) dates back to 1926!

Undirected, connected $G = (\mathcal{N}, \mathcal{A}, c)$. To find an unconstrained MCST in G. c arbitrary.

Constrained MCST problems (typical constraints involve degree constraints at nodes) are usually NP-hard.

One approach: Select a node 1 say. Find a shortest path tree rooted at 1. $\forall i \neq 1$ the path from 1 to i in this tree is the shortest. So accept this ST as the solution.

Theorem 1: If T_0 is an MCST in G, every in-tree edge must be a min cost edge in its fundamental cutset.

Theorem 2: If T_0 is an MCST in G, every out-of-tree edge must be a max cost edge in its fundamental cycle.

Theorem 3: (Converse of Theorem 2): Any ST in G satisfying *"every out-of-tree edge is a max cost edge in its fundamental cycle"* is an MCST.

Theorem 4: F is a forest & $(X; \bar{X})$ is a cut s. th. $F \cap (X; \bar{X}) = \emptyset$. $(p; q)$ is a min cost edge in $(X; \bar{X})$.

For the constrained MCST problem: *"Among all STs containing F as a subset, find a min cost one"* there exists an opt. sol. that also contains edge $(p; q)$.

Corollary 1: $(p; q)$ is a min cost edge in some cut in G. Then there exists an MCST containing $(p; q)$ as an in-tree arc.

Corollary 2: F is a forest satisfying: "there exists an MCST containing all edges in F". $(p; q)$ is a min cost edge in a cut $(X; \bar{X})$ s. th. $F \cap (X; \bar{X}) = \emptyset$. Then there exists an MCST containing all edges in $F \cup (p; q)$.

Theorem 5: F is a forest & $\mathcal{C}$ a simple cycle in G. $(r; s)$ is a max cost edge among those in $\mathcal{C} \backslash F$.

For the constrained MCST problem: *"Among all STs containing F as a subset, find a min cost one"* there exists an opt. sol. not containing $(r; s)$

All efficient algos. are of the "*build up*" type, & can be interpreted as **Greedy methods**. Begin with forest containing isolated nodes. In each step ≥ 1 edges added connecting forest components.

Prim's algo.: 1957 Prim's paper. But method appeared in 1930 Jarnil's paper. Dijkstra (1959) developed data structures to bring complexity down to $O(n^2)$. Best algo. for **dense networks**.

Forest will always be one tree + remaining isolated nodes. Tree grows by one arc per step. Tree nodes are **permanently labeled nodes**. **Temp. labels** on out-of-tree nodes j of form:

$(\emptyset, \infty)$:	if no edge joining j to an in-tree node so far.
(p_j, d_j):	if j adjacent to at least one in-tree node. Here $d_j = \min\{c_{ij} : i$ in-tree and $(i;j) \in \mathcal{A}\}$. p_j is an i that attains this minimum.

Initialization: Permanently label 1 with $\emptyset$. Temp. label all j s. th. $(1, j) \in \mathcal{A}$ with $(1, c_{ij})$. Temp. label all other nodes with $(\emptyset, \infty)$.

General Step: Find temp. label with smallest distance index, suppose it is (p_r, d_r) on node r. Perm. label r with PI p_r (i.e., add r and edge (p_r, r) to tree).

If tree spanning TERMINATE.

Otherwise, $\forall$ out-of-tree nodes j with temp. label (p_j, d_j)

if $c_{rj} < d_j$ change label on j to (r, c_{rj})

otherwise leave label on j unchanged.

Go to next step.

Proof of correctness, and complexity.

Other methods suitable for sparse networks only.

Kruskal's Method

Initialization: Initial forest $(\{1\}, \emptyset), \ldots, (\{n\}, \emptyset)$. Order edges in increasing order of cost & begin examining them in this order.

General step: Let edge to be examined be $(i; j)$.

If i, j belong to same component of forest at this stage, discard this edge, go to next step.

If i, j belong to different components of forest at this stage, include $(i; j)$ in forest, merging the two components into one tree.

TERMINATE if there is only component in forest. Otherwise go to next step.

Proof of correctness & complexity.

Boruvka's algo.

If all edge costs are not distinct, adopt a tie breaker rule for the minimum in every pair of costs. For example, number edges as $e_1, \ldots, e_m$. If $c_r = c_s$ assume least cost edge in pair $\{e_r, e_s\}$ to be e_t where $t = \min\{r, s\}$.

Initialization: Start with $(\{1\}, \emptyset), \ldots, (\{n\}, \emptyset)$.

General step: Let forest be $F_1 = (\mathcal{N}_1, \mathcal{A}_1), \ldots, F_\ell = (\mathcal{N}_\ell, \mathcal{A}_\ell)$

$\forall h = 1$ to ℓ find a least cost edge in cut $(\mathcal{N}_h, \mathcal{N} \backslash \mathcal{N}_h)$, add all these edges to the forest.

Repeat until forest becomes a spanning tree.