Minimum-Cost Bounded-Degree Spanning Tree

Presented by Girvar Patidar & Somya Bansal

Made with GAMMA

The Problem: Finding Optimal Spanning Trees

Objective

Given a weighted undirected connected graph, find a minimum-cost spanning tree where each node's degree is within a specified bound.

Research Goals

Remove reliance on time-consuming Linear Programming and adapt combinatorial local search techniques.

Made with GAMMA

Local Search for Unweighted Graphs

Minimum-Degree Spanning Tree Problem

Find a spanning tree T that minimizes the maximum degree of any vertex v in T.

Computational Complexity

- **NP-hard:** The problem is computationally intensive.
- **Hamiltonian Path:** A spanning tree with max degree 2 is a Hamiltonian path, which is NP-complete to decide.

Exact solutions are infeasible, driving the need for approximation algorithms and heuristics.

Made with GAMMA

Designing the Local Search Algorithm

A polynomial-time local search algorithm finds a tree T with a maximum degree at most $2 \cdot \mathrm{OPT} + \lceil \log_2 n \rceil$.

Local Move Definition

Identify edges not in T that create a cycle. Select an edge (v, w) where $\max(d_T(v), d_T(w)) \leq d_T(u) - 2$.

Move Properties

After a move, $d_{T'}(u) = d_T(u) - 1$. The algorithm targets high-degree nodes and terminates when no further moves are possible.

Made with GAMMA

OPT + 1 Local Search Framework

This framework operates in phases and subphases to reduce node degrees.

Phase-Based Execution

Algorithm runs in phases, each removing all degree-k nodes. Each phase has subphases targeting single degree-k nodes.

Subphase Actions

Identify reducible degree-(k-1) nodes, mark them, and update graph components. This enables local moves to reduce node degrees.

Made with GAMMA

Linear Programming Formulation

We use an LP formulation for a graph $G = (V, E)$ with edge costs $c_e \geq 0$ and degree bounds $b_v \geq 1$ for constrained vertices $W \subseteq V$.

$$\text{Minimize: } \sum_{e \in E} c_e x_e$$

Subject to:

- $\sum_{e \in E} x_e = |V| - 1$ (spanning tree)
- $\sum_{e \in E(S)} x_e \leq |S| - 1, \forall S \subseteq V, |S| \geq 2$ (acyclicity)
- $\sum_{e \in \delta(v)} x_e \leq b_v, \forall v \in W$ (degree bounds)
- $x_e \geq 0, \forall e \in E$ (non-negativity)

Made with GAMMA

LP-Based Deterministic Rounding

An iterative process to derive a spanning tree from the LP relaxation.

Solve LP Relaxation

Obtain a basic optimal solution x and identify its support $E(x) = \{e : x_e > 0\}$.

Iterative Reduction

Repeatedly remove zero-edges, add single incident edges to solution, and remove vertices from W under specific conditions.

This returns a spanning tree F with cost $\leq$ LP optimal value, and $\deg_F(v) \leq b_v + 2$ for each $v \in W$.

Made with GAMMA

Refinement to +1 Violation Bound

An improved algorithm guarantees a tighter bound on degree violations.

1

Improved Algorithm

Repeatedly solve LP relaxation and remove vertices v from W that are guaranteed to have $\leq b_v + 1$ support edges.

2

Resulting Tree

When W is empty, compute an ordinary minimum-cost spanning tree. This yields $\deg_F(v) \leq b_v + 1$.

Made with GAMMA

Flow-Based Constraint Reduction

Motivation

Replace the exponential number of cycle constraints with a polynomial-sized flow formulation.

Extended Variable Set

- Original edge variables: x_e
- Flow variables: $f^a_{uv} \geq 0$ for each arc (u, v) and commodity $a \in V \setminus \{r\}$.

Flow Constraints

Capacity constraints:
$\forall \{u, v\} \in E, \forall a \in V \setminus \{r\} : f^a_{uv} \leq x_{uv}, f^a_{vu} \leq x_{uv}.$

Per-commodity supply/demand ensures flow from source *a* to sink *r*.

Made with GAMMA

Feedback Vertex Set (FVS)

- A set $F \subseteq V$ is an FVS if removing F makes the graph acyclic (every cycle intersects F).
- Goal: find F of minimum total cost $c(F) = \sum_{v \in F} c(v)$.
- Problem is NP-hard, so we use approximation algorithms.

Made with GAMMA

Local Ratio for FVS: How It Works

- **Compute α:** $\alpha = \min_{v \in V} c(v)/(d(v)-1)$, where $d(v)$ is the degree of v (assume $d(v) \geq 2$).
- **Reduce costs:** For every vertex set: $c'(v) = c(v) - \alpha(d(v)-1)$. (By choice of α, at least one v has $c'(v) = 0$.)
- **Select & remove:** Let $F_0 = \{v \in V : c'(v) = 0\}$. Add F_0 to the solution and remove them from the graph: $G' \leftarrow G \setminus F_0$.
- **Recurse:** Run the same procedure on $(G', c'|_{V(G')})$ to get F'.
- **Combine:** The final FVS is $F = F_0 \cup F'$.

Made with GAMMA

Bounded-Degree Spanning Tree

- Each vertex v has a degree bound b_v.
- In any spanning tree T, define the exceedance: $e_T(v) = d_T(v) - b_v$. (How much the degree of v exceeds its allowed limit.)
- The quality of a tree is measured by its maximum exceedance: $\max_{v \in V} e_T(v)$.
- Goal: Find a spanning tree T that minimizes this maximum exceedance: $\min_T \max_v e_T(v)$.

Made with GAMMA

Local Search Framework

- For the current tree T, compute the current worst exceedance: $k = \max_v e_T(v)$.
- Define the critical vertex sets: $D_k = \{v : e_T(v) = k\}$, $D_{k-1} = \{v : e_T(v) = k - 1\}$. These are the vertices closest to violating the bounds the most.
- The algorithm searches for a local move (edge swap) that:
  - reduces the degree of at least one vertex in D_k,
  - while maintaining a valid spanning tree.

Made with GAMMA

Lower Bound on the Optimal Exceedance

- Suppose at level k no local improving move exists.
- Then every spanning tree T' must have: $\max_v e_{T'}(v) \geq k - 1$.
- Meaning: Even the optimal spanning tree cannot reduce the worst exceedance below $k - 1$.
- Therefore: $k \leq \mathrm{OPT}_{\mathrm{exc}} + 1$.
- This gives the algorithm a $+1$ approximation guarantee.

Made with GAMMA