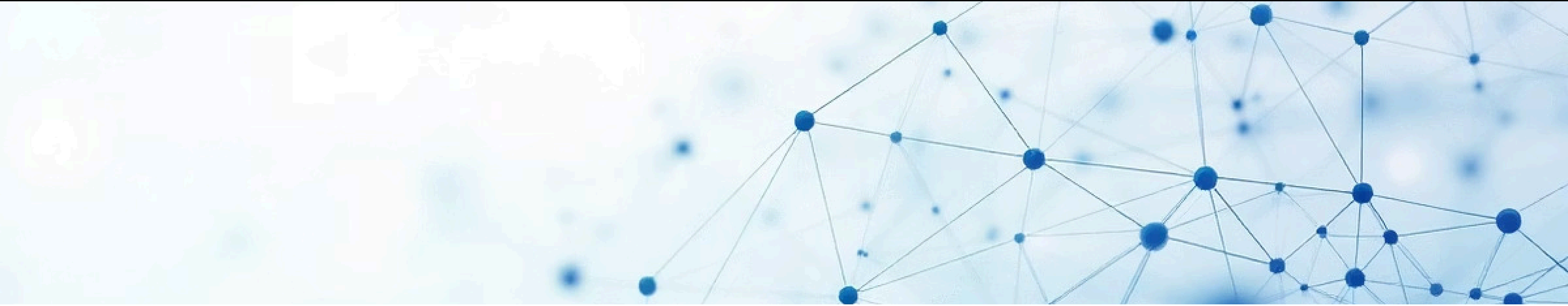


Minimum-Cost Bounded-Degree Spanning Tree

Girvar Patidar | Somya Bansal

Problem Overview

Given

- Graph $G = (V, E)$
- Edge costs c_e
- Degree bounds b_v
- Find a spanning tree minimizing cost while keeping $\mathrm{degree}(v) \leq b_v$.

Research Goals

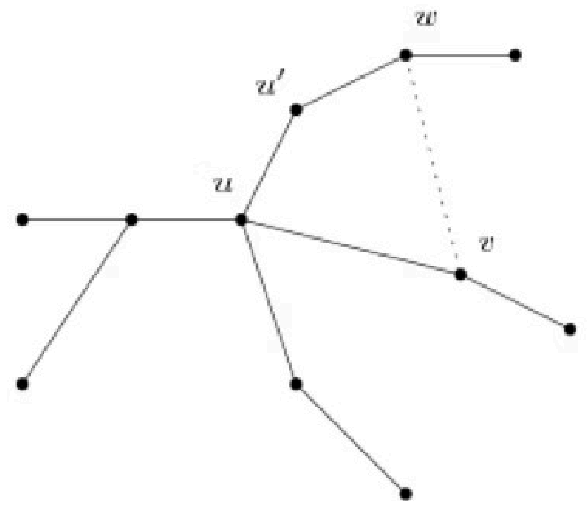
Remove reliance on time-consuming Linear Programming and adapt combinatorial local search techniques.

Local Search Algorithm - Unweighted Graphs

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

Local Move Defination

- Let $d_T(u)$ = degree of node u in current tree *T.*
- Consider all edges (v, w) not in T that create cycle C containing u.
- Select edge where $\max(d_T(v), d_T(w)) \leq d_T(u) - 2$



Move Properties

- Algorithm targets nodes with degree atleast $\max \ d_T - \lceil \log_2 n \rceil$, and terminates if no local move is possible on any such node.
- finds tree T with maximum degree at most $2\mathrm{OPT} + \lceil \log_2 n \rceil$

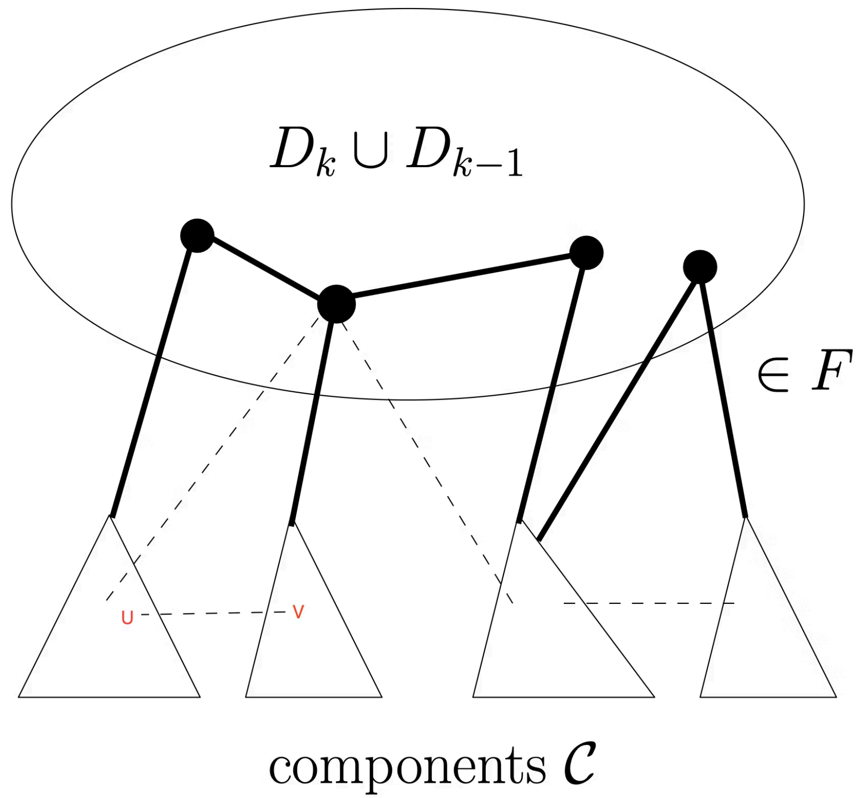
Made with GAMMA

OPT + 1 Local Search Framework

- Let k be the maximum degree in T
- Let D_k = all degree-k vertices in tree T
- Define F as edges incident to $D_k \cup D_{k-1}$
- Let C be components of T formed when F is removed

Approach

- Identify degree $(k-1)$ vertices for local moves
- Mark nodes as reducible
- Remove marked nodes from D_{k-1} and update F and C



Linear Program Algorithm - Weighted Graph

Given $G = (V, E)$, costs $c_e \geq 0$ for all $e \in E$, set $W \subseteq V$, Integer bounds $b_v \geq 1$ for all $v \in W$

LP Formulation Overview -

Objective Function:

$$\text{Minimize} \sum_e c_e x_e$$

Subject to:

$$\sum_e x_e = |V| - 1 \quad (spanning\ tree\ constraint)$$

$$\sum_{e \in E(S)} x_e \leq |\ S\ | - 1,\ \forall S \subseteq V,\ |\ S\ | \geq 2 \quad (acyclicity)$$

$$x(\delta(v)) \leq b_v \quad (degree\ bounds)$$

$$x_e \geq 0\ (non - negativity)$$

Flow-Based Constraint Reduction

Motivation

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

Idea

- Supply exactly 1 unit of flow from each node (except $root$) to the $root$
- Apply Conservation of flow (from each node(v)) at each node (u) of the graph
- Flow value (f) should not exceed the capacity of the edge (x_e)

$$\sum_{w:(v,w)\in A} f_{wv} - \sum_{w:(w,v)\in A} f_{vw} = \begin{cases} 1 & \text{if } v = a \text{ (source)} \\ -1 & \text{if } v = r \text{ (sink/root)} \\ 0 & \text{otherwise} \end{cases}$$

Deterministic Rounding

- **Solve LP**

  Work with support $E(x) = \{e : x_e > 0\}$

  If vertex v has exactly one incident edge in $E(x)$: $(u, v) \in E(x)$

  - Add to solution $(u, v) \to F$
  - Remove v and decrement b_u if $u \in W$

  Else if $v \in W$ has ≤ 3 incident edges in $E(x)$: remove v from W

Guaranteed Outcome:

The algorithm ensures that degree of any vertex 'v' in the final spanning tree will not exceed its specified bound 'b_v' by more than two.

$$deg(v) \leq b_v + 2$$

Improved +1 Algorithm

- **Any basic feasible solution (when $W \neq \emptyset$) contains some $v \in W$ with at most $b_v + 1$ edges in $E(x)$.**

1. Repeatedly solve LP relaxation
2. Remove vertex $v \in W$ guaranteed by lemma (if support $\leq b_v + 1$ edges)
3. When W becomes empty, compute ordinary minimum-cost spanning tree on remaining support

Guaranteed Outcome: This improved algorithm ensures a tighter bound on vertex degrees.

$$deg(v) \leq b_v + 1$$

The degree of any vertex 'v' in the final spanning tree will not exceed its specified bound 'b_v' by more than one.

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.

Approach:

- **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 $F = F_0 \cup F'$.

Our Contribution:

Local Search - Bounded Spanning Tree

Problem :

Let $G = (V, E)$ be a connected undirected graph, and for each vertex $v \in V$ let $b_v \geq 1$ be an integer upper bound on its degree

We define exceedance by :

$$e_T(v) = deg_T(v) - b_v$$

Algorithm goal:

$$\text{Minimize } max_v \ (e_T(v))$$

Algorithmic Framework: Local Search

→ Idea is similar to the algorithm for minimising the maximum degree of spanning Tree T

Iterative Process Flow:

1

Compute k

Max exceedance

2

Identify D_k, D_{k-1} , F and C

for decreasing the exceedance k

3

Mark D_{k-1} nodes reducible for Local moves and Update F, C

Used to decrease the exceedance of D_k

4

Stop if no local move is possible

It is guaranteed that $k <= OPT_{exc} + 1$

Proof Sketch - $OPT \geq k$ - 1

$$S = D_k \cup D_{k-1}$$

$$\sum_{v \in S} d_T(v) \geq | F |$$

$$\sum_{v \in S} e_T(v) \geq | F | - \sum_{v \in S} b_v$$

$$| F | \geq \sum b_v + k|D_k| + (k-1)|D_{k-1}| - (| S | - 1)$$

$$OPT_{exc} \geq k - 1$$

- any spanning tree must add ⩾ |F| edges touching S
- Subtract degree bounds
- Nodes in S contribute degrees $b_v + k$ or $b_v + k - 1$, which counts edges inside S twice and edges leaving S once.
- Since edges inside S form a forest (⩽ $| S | - 1$), the remaining counted edges give a **lower bound on** $| F |$.

Thank you