Minimum-Cost Bounded-Degree Spanning Tree

Girvar Patidar · Somya Bansal

IIT Delhi

Goal:

Design efficient algorithms for minimum-cost spanning trees under degree bounds.

Includes:

- Existing LP & combinatorial methods
- Our exceedance-based local search ($OPT + 1$ inspired)
- Visual examples and intuition

Problem Overview

- Given:
- ▪ Graph $G = (V, E)$
- ▪ Edge costs c_e
- ▪ Degree bounds b_v
- Goal:
- Find a spanning tree minimizing cost while keeping $\mathrm{degree}(v) \leq b_v$.
- Challenges:
- ▪ NP-hard
- ▪ MST often violates degree bounds
- ▪ Requires approximation/heuristic approaches

Bounded-Degree Example

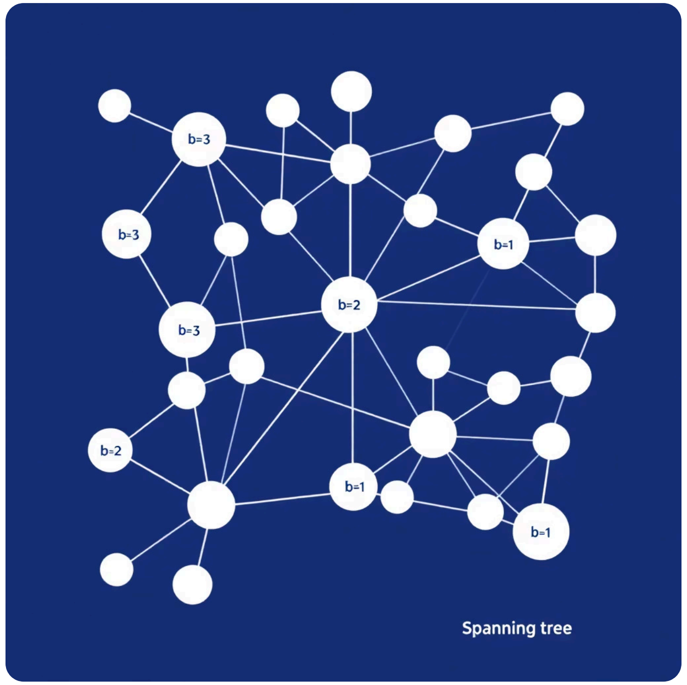
$$e_T(v) = \deg_T(v) - b_v$$

The formula defines the exceedance of a node v in a spanning tree T.

Concept Explanation

- **Exceedance:**
  $e_T(v)$ quantifies how much the degree of node v in the spanning tree T ($\deg_T(v)$) exceeds its specified degree bound (b_v).
- **Goal:**
  The objective is to find a spanning tree that minimizes the maximum exceedance across all nodes, aiming for all nodes to respect their degree bounds as closely as possible.
- **Challenges:**
  Traditional Minimum Spanning Tree (MST) algorithms don't account for degree bounds. This problem is NP-hard, requiring approximation or heuristic approaches to find solutions.

Visual Example



Local Search – Intuition

Local Swap Idea:

1. Add an edge
2. Creates a cycle
3. Remove one edge from cycle to reduce high-degree node

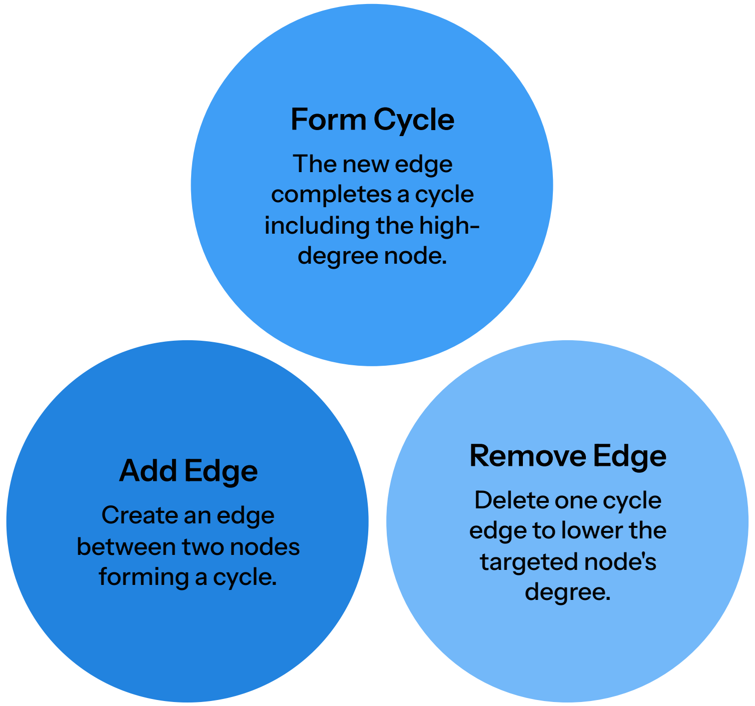
Key Principle:

Only modify edges near worst-degree nodes.

Outcome:

Fast practical improvement even on large graphs.

Visualizing the Local Swap


Form Cycle
The new edge completes a cycle including the high-degree node.
Add Edge
Create an edge between two nodes forming a cycle.
Remove Edge
Delete one cycle edge to lower the targeted node's degree.

Existing Methods

Unweighted Case

Minimize Maximum Degree

The primary objective in the unweighted case is to minimize the highest degree among all nodes in the graph.

NP-Hard Problem

This problem is classified as NP-hard, notably demonstrated by its connection to the Hamiltonian path problem (where degree equals 2).

Approximation

Achieves 2·OPT + log n

Approximation algorithms, specifically local search methods, can achieve results close to the optimal solution.

Simple Swaps

These methods utilize simple iterative swaps to progressively reduce node degrees, finding practical improvements.

LP Formulation Overview

Objective Function:

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

Subject to:

$$\sum_e x_e = |V| - 1$$

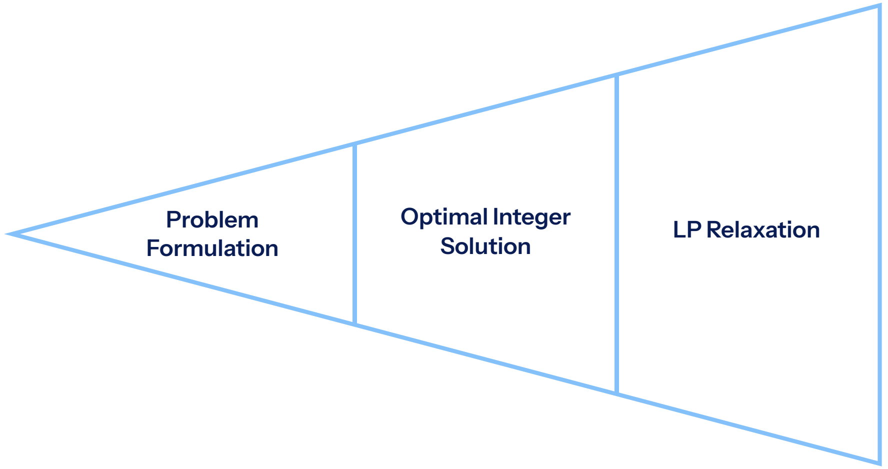
Acyclicity constraints

$$x(\delta(v)) \leq b_v$$

$$x_e \geq 0$$

Purpose:

Provides a lower bound, used for rounding in approximation algorithms.



Deterministic Rounding

Process Steps:

01

Solve LP

Find the optimal solution for the Linear Program, which may contain fractional edge values.

02

Identify Fractional Edges

Pinpoint edges within the solution that have non-integer (fractional) values.

03

Iteratively Prune/Remove/Contract

Apply a series of deterministic rules to systematically convert fractional solutions into integer ones.

04

Produce Spanning Tree

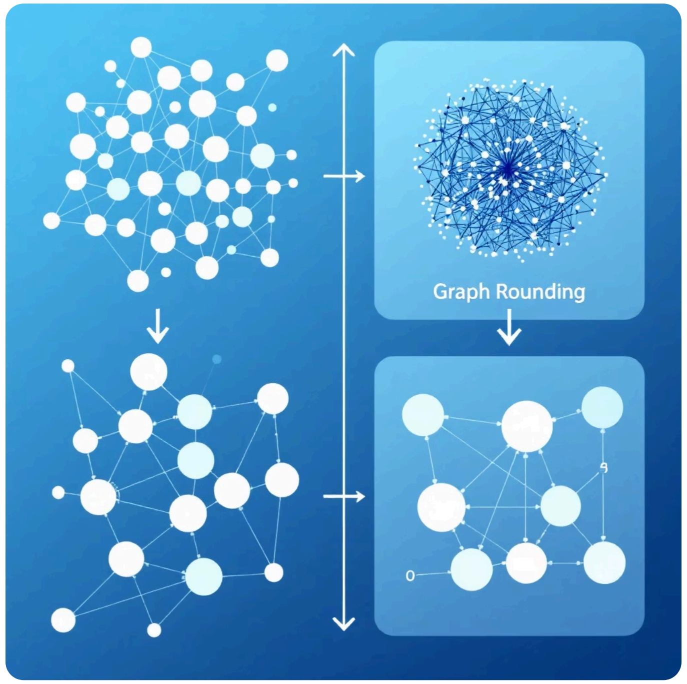
Construct a valid spanning tree using only the integer-valued edges resulting from the rounding process.

Guaranteed Outcome:

A key advantage of this method is a strong guarantee on the degree of each vertex in the final tree:

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

This implies that the degree of any vertex 'v' in the generated spanning tree will not exceed its specified bound 'b_v' by more than two.



Improved +1 Algorithm

01

Initiate LP-based Filtering

Begin a process of repeated filtering based on Linear Program solutions.

02

Identify & Remove Vertices

Locate and remove vertices that already satisfy the degree bound condition (b_v+1).

03

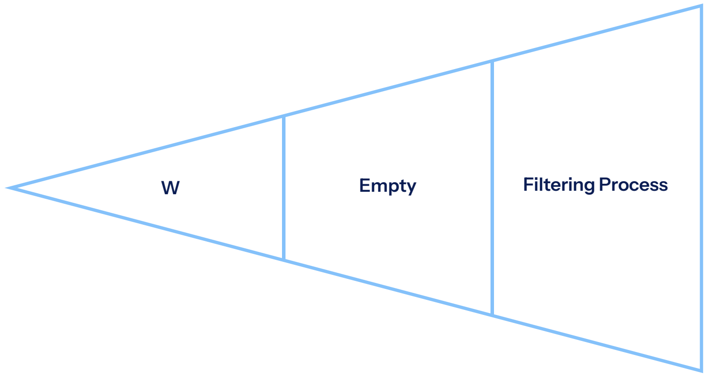
Iteratively Shrink 'W'

Continuously reduce the working set 'W' by applying filtering rules until it becomes empty.

04

Construct Final MST

Formulate the Minimum Spanning Tree (MST) on the graph remaining after the iterative shrinking of 'W'.



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.

Our Contribution: Exceedance-Based Local Search

We define exceedance:

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

Algorithm goal:

Minimize k = max_v e_T(v)

Key advantages:

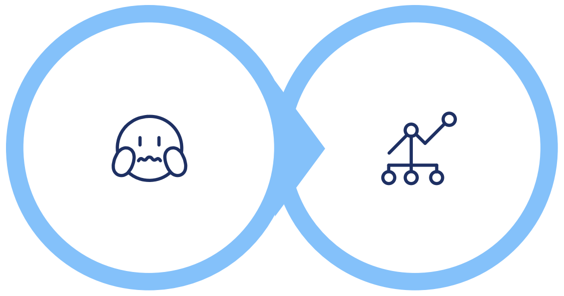
- Works without LP
- Intuitive
- Phase-based like OPT+1 algorithm
- Sensitive to worst-case vertices (D_k)

Example: Exceedance Reduction

Understanding Exceedance Reduction

Let's illustrate how a 'swap' can reduce the exceedance of a node:

- **Initial State:** Consider Node A with a degree of 5 (connected to 5 other nodes) and a predefined bound of 3. Its exceedance, calculated as `deg(A) - bound(A)`, is `5 - 3 = 2`. This indicates an imbalance, often highlighted in red.
- **The Swap:** A 'swap' operation in graph theory typically involves adding a new edge and removing an existing one. The objective here is to reduce the degree of Node A. For instance, we might add an edge that connects two nodes that previously weren't connected to A, and remove an edge that connected to A, effectively rerouting connections.
- **Result:** After the swap, Node A's degree is successfully reduced from 5 to 4. With the same bound of 3, its new exceedance becomes `4 - 3 = 1`. This step reduces the maximum exceedance in the graph, moving towards a more balanced state.



Initial

Reduced

Algorithmic Framework: Local Search

Steps:

01

Compute k = max exceedance

02

Identify D_k (worst) and D_{k−1} (near-worst)

03

Search for improving swap

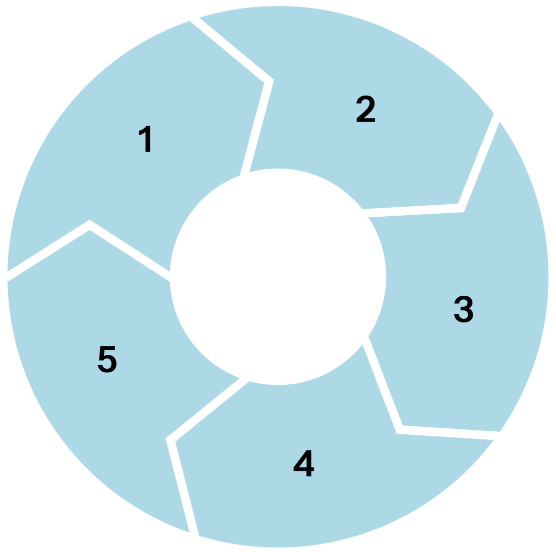
04

If swap reduces k → apply it

05

Stop when no improving move exists

Iterative Process:



1 Compute k

Max exceedance

2 Identify D_k

Worst & near-worst

3 Search

For improving swap

4 Apply Swap

If k is reduced

5 Stop

No improving move

Focus: Only swaps affecting D_k are considered for optimization.

Lower Bound Guarantee

Logical Flow:

1

Condition Met

No improving move exists at level k.

2

Implication

Any spanning tree must have max exceedance $\geq k-1$.

3

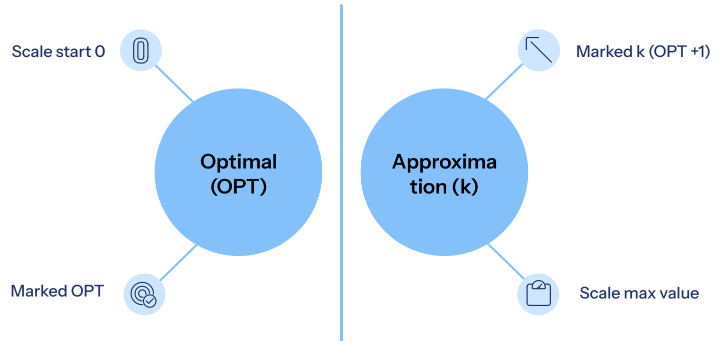
Deduction

This implies our final k is close to the optimal solution.

Theoretical Guarantee & Visual:

$$k \leq OPT + 1$$

This critical inequality is a **theoretical guarantee** of the algorithm's performance.



This means we achieve a **+1 approximation** on the maximum exceedance value, ensuring our solution is always within one unit of the optimal solution.

Comparison with Existing Methods

LP Rounding

- Cost-focused
- Needs LP solver
- +1 or +2 degree violation

Classical Local Search

- Degree minimization
- No cost consideration

Our Method

- Minimizes exceedance
- Simple
- OPT+1 guarantee
- Purely combinatorial

Conclusion

Key Achievements

Visual, Intuitive Metric

Developed an exceedance-based metric that is both visual and easy to understand.

Effective Local Search

Implemented a local search approach that significantly reduces worst violations.

OPT+1 Approximation

Achieved a strong theoretical guarantee with an OPT+1 approximation.

Practical Algorithm

Created an algorithm that is practical, efficient, and easy to implement.

Future Work

→ **Weighted-Case Integration**

Incorporate weighted scenarios to enhance applicability in diverse situations.

→ **Faster Swap Selection**

Optimize swap selection processes for improved algorithm performance.

→ **Hybrid LP + Local Search**

Explore combined Linear Programming and local search methodologies for superior results.