Directed k-MTM Multicast Implementation

Approximation Algorithms in the Telephone Model

PRESENTED BY

Hamza Abdullah

Overview

- Problem & Motivation
- Algorithm
- Implementation
- Results
- Demo
- Conclusion

Problem & Motivation

- Objective: Inform any k≤t of t terminals in a directed network under the telephone model
  - Each informed node may call one neighbor per synchronous round
- Why it matters:
  - Database replication consistency
  - Sensor-network firmware updates
  - Vector-clock synchronization delay
- Core challenge:
  - NP-hard combinatorial problem
  - Matching constraint (one call/node/round) limits throughput

Algorithm

- Greedy Packing
  - Extract ρ=sq_root[k⌉ vertex-disjoint subtrees of depth ≤D*, each covering ρ terminals.
- Partition Matroid Cover
  - Cover the remaining terminals
  - Half-approx greedy
  - Continuous-greedy
  - Lazy-greedy speed-up
- Stitch & Simulate
  - Combine subtrees + cover edges into one tree, then simulate telephone-model rounds to count broadcast time.

Implementation

- graph_loader.py – generates ER and clique graphs, picks ttt and kkk ratios
- greedy_packing.py – extracts the depth-D* subtrees
- pmcover.py, pmcover_continuous.py, pmcover_lazy.py – three variants of matroid-constrained set-cover
- complete.py – stitches subtrees and cover edges into a single multicast tree
- simulator.py – runs the telephone-model broadcast rounds on the final tree

Enhancements:

- Lazy-greedy speed-up matches half-approx coverage in 5× less time
- Demo CLI (demo/demo.py) for one-command end-to-end runs

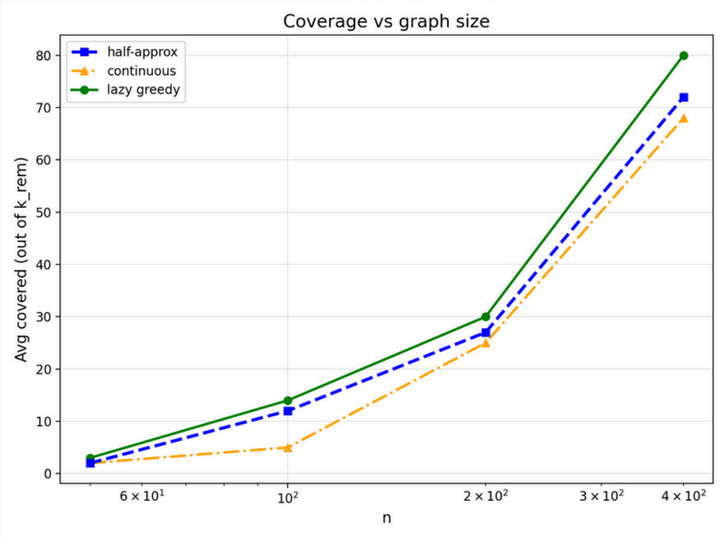
Results

Coverage vs. Runtime (ER graph: n=500, p=0.02, k=0.6t, D*=3)

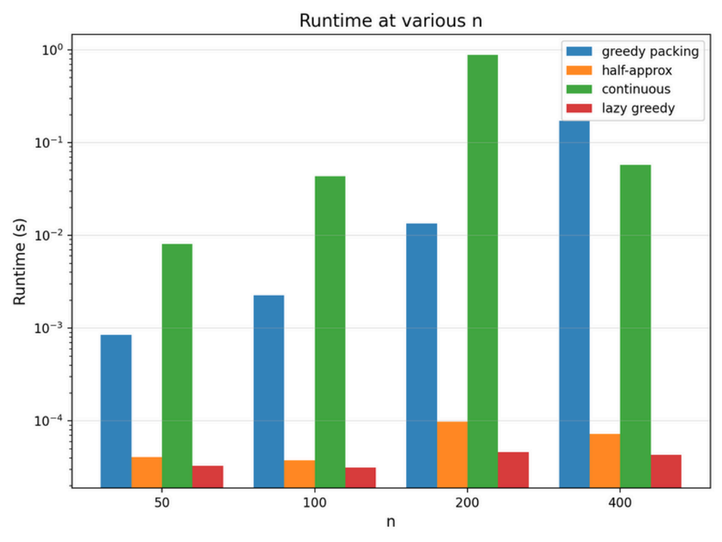
- Half-approx: covers ~45 % of the remaining k in ~0.02 s
- Continuous-greedy: covers ~65 % in ~0.80 s
- Lazy-greedy: matches half-approx coverage (~45 %) in ~0.03 s

Broadcast Rounds

- Full pipeline (greedy + PMCoverlazy) informs k in ~5 rounds
- Greedy-only packing stalls after ~2 rounds



coverage_comparison.png



runtime_comparison.png

Demo

Conclution

- We delivered a full Python implementation of the directed k-MTM algorithm, matching theoretical guarantees in practice.
- Empirical benchmarks show the trade-off between coverage quality (continuous-greedy) and runtime (lazy-greedy).
- A user-friendly CLI and CI pipeline ensure reproducibility and ease of exploration.

Thank you

PRESENTED BY

Hamza Abdullah