Progress Report - 3 : Karger-Stein Algorithm Implementation

Rameez Wasif
Zain Hatim

Implementation Summary

We have successfully implemented the core Karger-Stein algorithm for computing the minimum k-cut in a graph. Key components include:

- Random weighted edge contraction as the basic operation for cut reduction.
- Recursive contraction steps as described in the original Karger-Stein algorithm.
- Multiple independent trials to increase the probability of finding the true minimum cut.
- A brute-force fallback for very small graphs (typically $n \leq 6$) for guaranteed correctness.
- Supernode tracking system to accurately reconstruct final graph partitions after contraction.
- Supporting utilities such as graph loading, visualization, and performance measurement.

Correctness Testing

To verify correctness, we tested our algorithm on both synthetic and structured graphs:

- **Simple triangle + extra node:** Correctly returned a min 2-cut of 3.0.
- **4x4 grid graph:** Output min 2-cut of 4.0, matching expected result.
- **Complete graph (n = 8):** Verified min k-cut $= k - 1$ for unit edge weights.

We also tested edge cases:

- Disconnected graphs (returns infinite cut weight or empty partition).
- Graphs with identical or random edge weights.
- Tiny graphs ($n = 1, 2$) and large graphs ($n > 100$).

A dedicated partition validator was implemented to check if the output forms a proper k-partition.

Complexity & Runtime Analysis

Theoretical Analysis:

- Time Complexity per trial: $O(n^3)$

Identified Bottlenecks:

- Recursive depth grows quickly on large graphs.
- Trials are executed sequentially, leading to slow aggregate runtime.
- Maintaining and updating supernode partitions introduces overhead.

Baseline or Comparative Evaluation

Originally we were going to do it on GLL 19 (an earlier version of this by the same authors) but due to time constraints we chose to do Brute Force.

Compared to Brute Force:

- Brute force requires checking all possible partitions: $O(n^k)$
- Our implementation uses randomized trials and recursive contraction: $O(n^3)$
- We observed exponential speedups for $n > 10$, while maintaining accuracy.

Compared to Basic Karger-Stein:

- Our version includes:
  - Degree-aware edge selection to prioritize important contractions.
  - Nagamochi-Ibaraki sparsification to reduce graph size while preserving cuts.
  - λ_k estimation for more efficient stopping.
- These enhancements improved performance, especially on dense graphs.

Challenges & Solutions

- **High memory usage:** Resolved by sparsifying graphs before trials.
- **Randomness reproducibility:** Introduced seeding support to ensure consistent results across runs.

Enhancements

Algorithm Modifications

- **Nagamochi-Ibaraki Sparsification:** Reduced graph size while preserving key connectivity; led to 2x speedup on graphs with $n > 50$.
- **Duplicate cut filtering:** Canonicalized partitions to avoid repeated recording of the same cut.

New Datasets

- Grid graphs (2D)

Additional Features

- A runtime and trial performance logger for empirical tracking.
- Visualizer to show partitions and highlight crossing edges.

Future Work

- Implement parallel trial execution to utilize multi-core systems.