Paper Selection Proposal

Rameez Wasif (08479)
Zain Hatim (08343)

Paper Details

- Title: The Karger-Stein Algorithm Is Optimal for k-Cut
- Authors: Anupam Gupta, Euiwoong Lee, Jason Li
- Conference: STOC (Symposium on Theory of Computing), 2020
- DOI/Link: `https://doi.org/10.1145/3357713.3384285`

Summary

This paper focuses on the k-Cut problem: given a weighted graph and an integer k, the goal is to remove a set of edges with the smallest possible total weight such that the graph breaks into at least k connected components. The problem generalizes the standard min-cut (where k = 2).

The authors show that the Karger-Stein algorithm, which uses random edge contractions, is actually optimal for solving the k-Cut problem for any fixed k. They prove that it outputs a correct solution with high probability and runs in $\tilde{O}(n^k)$ time, which matches known lower bounds up to logarithmic factors. The paper also includes a detailed analysis of how the graph structure evolves during the contractions and gives new bounds on the number of minimum k-cuts.

Justification

This paper is a great fit for our project because it solves a well-known open question using an elegant and relatively simple approach. The Karger-Stein algorithm is a randomized algorithm that is easy to understand and implement, but the paper also offers deep theoretical insights.

We also plan to compare this method with a well-known deterministic algorithm by Chekuri, Quanrud, and Xu (SODA 2018), which solves the k-Cut problem using tree packings and dynamic programming in $O(mn^{2k-3})$ time. This makes the project more meaningful, as we can compare a randomized contraction-based method with a deterministic one that uses a very different strategy.

Implementation Feasibility

The Karger-Stein algorithm is based on a simple idea: repeatedly contract random edges until only k nodes remain. The paper gives a clear description of how to extend this idea to solve the general k-Cut problem efficiently.

Why it's feasible:

- The algorithm is well-explained and supported by pseudocode in the paper

- We can test it on both real-world and synthetic graphs using tools like NetworkX or SNAP datasets.
- We'll evaluate how often the algorithm finds the correct solution, how fast it runs, and how it scales with larger graphs.
- Since it's probabilistic, we'll also study how the success rate changes with multiple runs.

We'll use the Chekuri et al. algorithm as a benchmark to see how our implementation compares in terms of performance and simplicity.

Team Responsibilities

- Zain & Rameez – Reading & Theory: Read the paper in detail & understand how the algorithm works.
- Zain & Rameez – Coding: Implement the Karger-Stein algorithm
- Zain & Rameez – Testing & Analysis: Run experiments and compare with the deterministic benchmark.
- Member 4 – Presentation: Write a report, maintain the GitHub repository and create the final presentation.