Habib University
Algorithms And Design Analysis

Team-3 (L5)

KARGER STEIN FOR K-CUTS

PRESENTED BY RAMEEZ WASIF & ZAIN HATIM

GOAL:

Find an optimal randomized algorithm for the minimum k-cut problem in general graphs.

INPUT:

Given a graph and integer k, remove a minimum-weight set of edges to split the graph into k components.

- Undirected weighted graph $G=(V, E, w)$
- Integer $k \geq 2$

OUTPUT:

Minimum k-cut

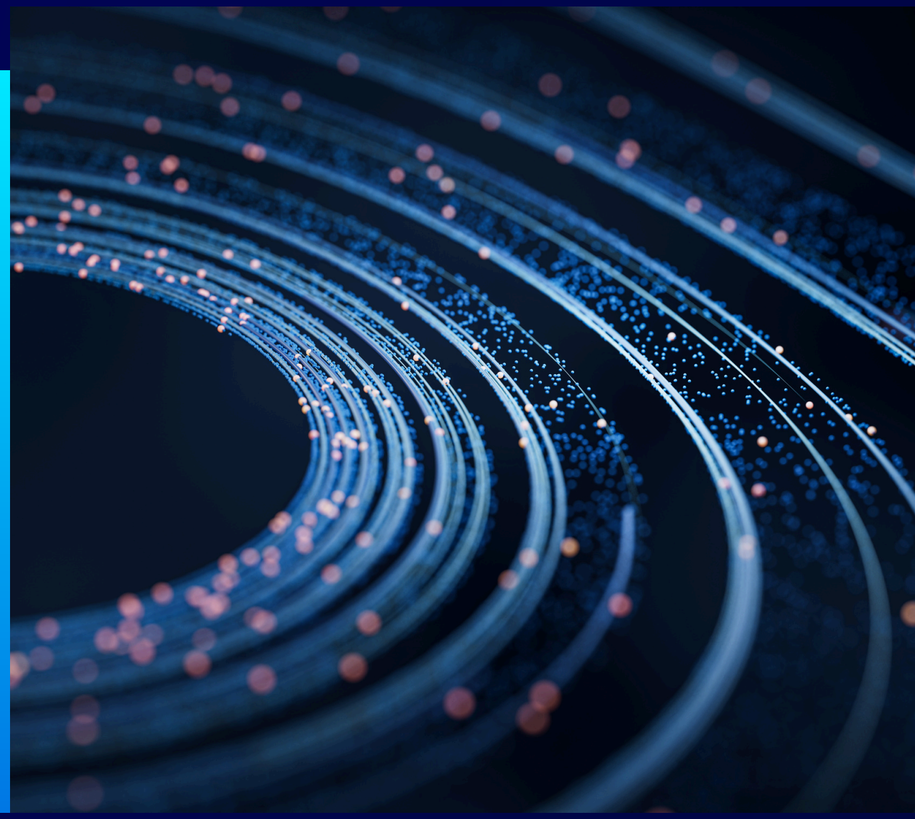
ALGORITHM OVERVIEW

Why Karger-Stein?

- Karger-Stein improves this to near-optimal randomized time.
- Combines random contraction with recursion for better success probability.

Core Ideas:

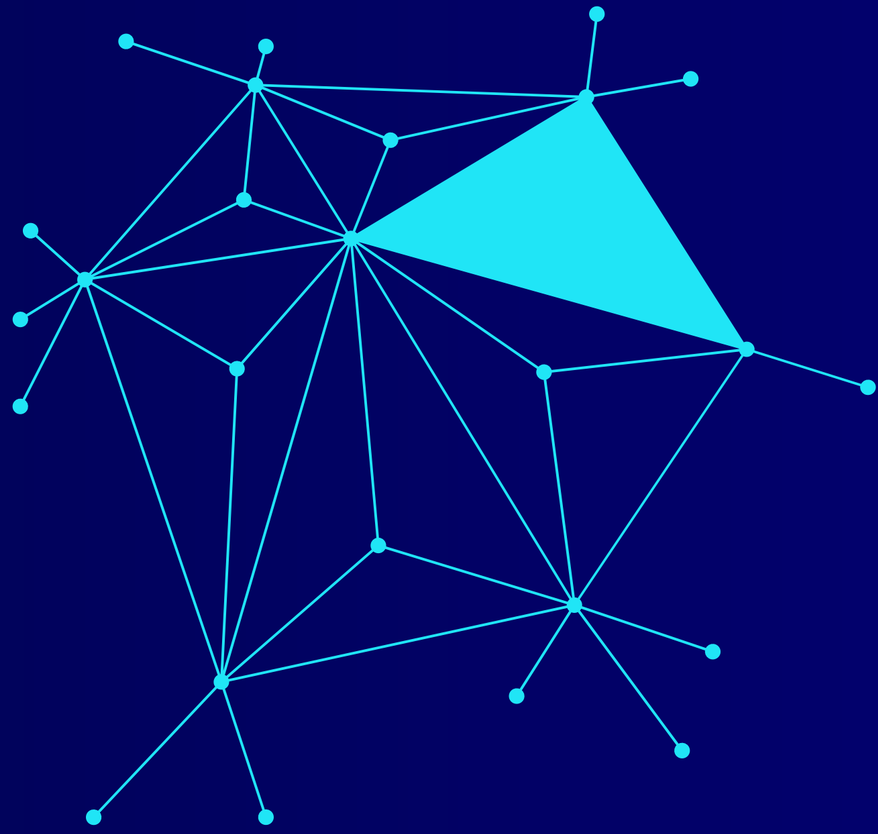
- Contract edges without removing the minimum k-cut
- Apply recursively to maintain correctness
- Repeat trials to boost probability of finding the correct cut



RECURSIVE

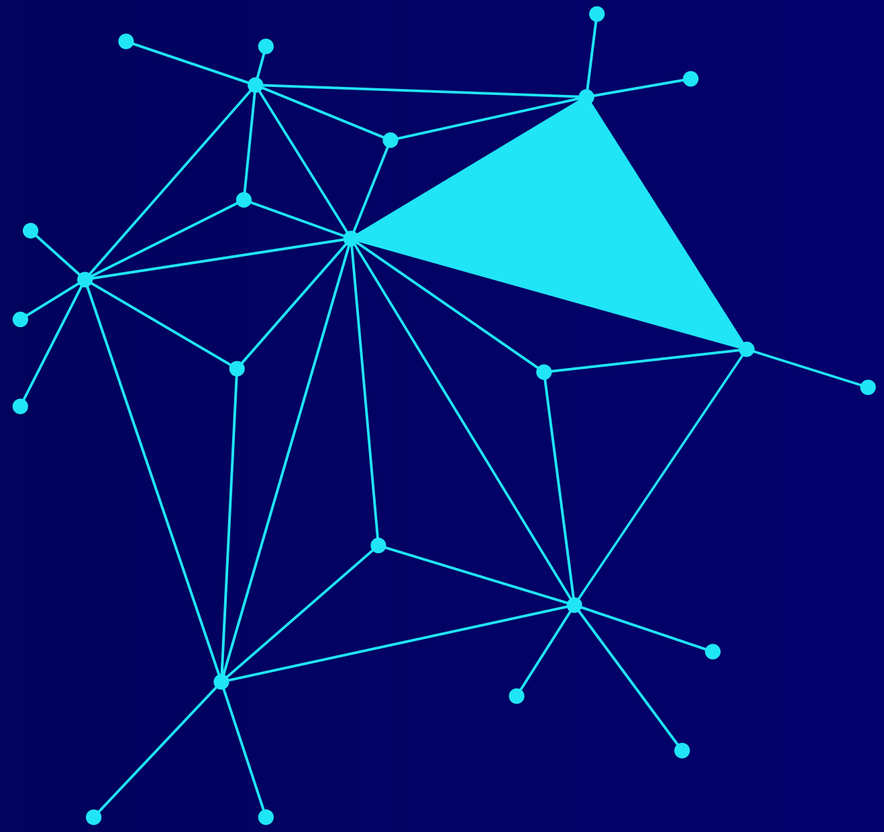
To recursively reduce the graph size and select the best result from independent trials.

- If the graph has ≤ 6 nodes (threshold value that we set), return as it is.
- Repeatedly contract edges with probability proportional to their weight until only until you reach $t = \lceil n / \sqrt{2} \rceil$ nodes
- Create two independent copies of this contracted graph.
- Apply the contraction process recursively and independently on both copies.
- Return the cut with the smaller total weight from the two recursive branches.



RECURSIVE EXAMPLE

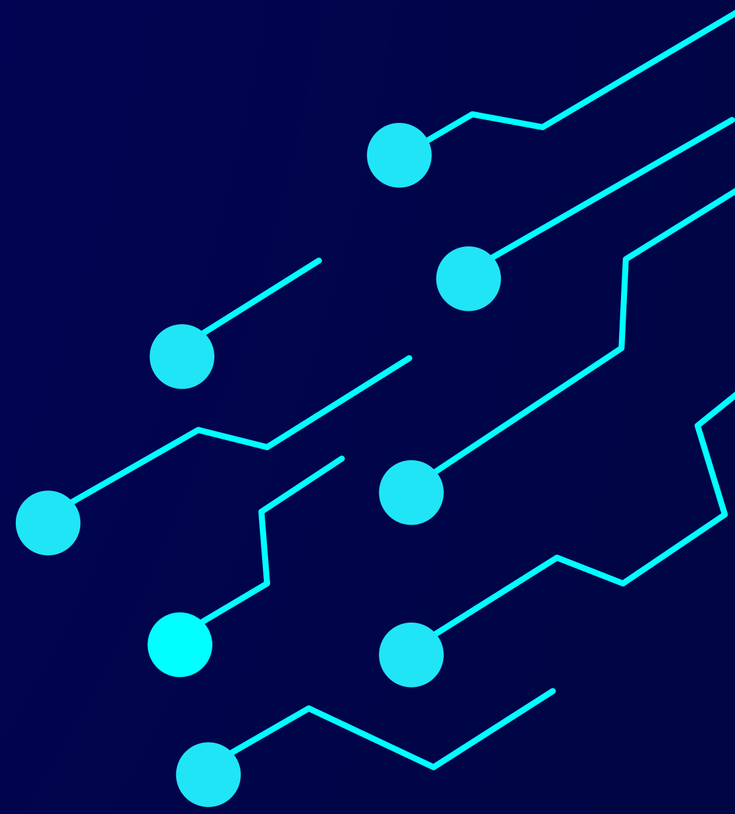
- Start with 8 nodes
- Contract to 6 nodes → fork into Graph A and B
- A → further contracts to A1, A2 → returns min(A1, A2)
- B → further contracts to B1, B2 → returns min(B1, B2)
- Final answer = min(min(A1, A2), min(B1, B2))



SPARSIFICATION

Reduces graph size (edges) while preserving minimum k-cut properties

- Calculate maximum connectivity value (maximum number of edge-disjoint paths that exist between any pair of vertices)
- Sort edges by weight (descending)
- Build sparsified graph with max_connectivity * n edges
- Only add edges that don't create cycles
- Verify k-cut preservation
- If not preserved, go back to original



TIME COMPLEXITY BFREAKDOWN

Recurrence contraction only

- Recurrence Relation: $T(n) = 2T(n / \sqrt{2}) + O(n^2)$
- Solves to: $T(n) = O(n^2 \cdot \log n)$ per trial
- Success Probability per Trial $\approx 1 / n^k$
- To Succeed with High Probability Run : $O(n^k \cdot \log n)$ trials
- Total Time Complexity (Without Optimization): $O(n^k \cdot \log n) \times O(n^2 \cdot \log n) = O(n^{k+2} \cdot \log^2 n)$

OMPTIMIZING WITH SPARSIFICATION

Edge Reduction (Nagamochi-Ibaraki Sparsification)

- Sparsified Edge Count : $O(\lambda_k \cdot n)$ instead of $O(n^2)$
- Contraction Becomes Cheaper : $O(n \cdot \lambda_k \cdot \log n)$

FINAL TIME COMPLEXITY

Still need $O(n^k \cdot \log n)$ trials

- Our final time complexity comes out to be O(n^2 (logn)^2)

THANK YOU