Time and Space Complexity Analysis of the Karger-Stein Algorithm

Rameez Wasif
Zain Hatim

Overall Time Complexity Analysis

Recursive Contraction

- Each level reduces the graph size by $\sqrt{2}$.
- Number of levels: $\mathcal{O}(\log n)$.
- Each level makes 2 recursive calls.
- Total recursive calls: $\mathcal{O}(2^{\log n}) = \mathcal{O}(n)$.

Per Trial Complexity

- Edge selection: $\mathcal{O}(m)$.
- Contraction: $\mathcal{O}(m)$.
- Cut weight calculation: $\mathcal{O}(k^2 n^2)$.
- Total per trial: $\mathcal{O}(k^2 n^2)$.

Number of Trials

- Total trials required: $\mathcal{O}(n^2 \log n)$.
- Each trial has $\mathcal{O}(n)$ recursive calls.
- Each recursive call does $\mathcal{O}(k^2 n^2)$ work.

Final Complexity

$$\mathcal{O}(n^2 \log n) \times \mathcal{O}(n) \times \mathcal{O}(k^2 n^2) = \mathcal{O}(k^2 n^5 \log n)$$

Note: This is a conservative upper bound. Actual complexity is better due to:

- Graph size reduces in recursive calls.
- Early termination in many trials.
- Sparsification significantly reduces m.

More Accurate Complexity (from the paper)

- Per trial: $\mathcal{O}(n^2 k^{-2} \log^3 n)$
- Trials: $\mathcal{O}(n^2 k^{-2} \log n)$

$$\text{Total: } \mathcal{O}(n^4 k^{-4} \log^4 n)$$

Our implementation matches this theoretical complexity, with small additional overhead from:

- Supernode tracking
- Multiple cut discovery
- Visualization and logging

Space Complexity

- Graph representation:
  - $\mathcal{O}(n + m)$ for sparse adjacency list.
  - $\mathcal{O}(n^2)$ for dense graphs.
- Recursive stack:
  - $\mathcal{O}(\log n)$ levels.
  - $\mathcal{O}(n)$ per level $\Rightarrow \mathcal{O}(n \log n)$.
- Supernode tracking:
  - Worst case: $\mathcal{O}(n^2)$.

Total space complexity: $\mathcal{O}(n^2)$.

Optimization Impact

- **Nagamochi-Ibaraki Sparsification:** Reduces m to $\mathcal{O}(nk)$, improving total complexity to $\mathcal{O}(n^3 k^{-2} \log^4 n)$.
- **Degree-Aware Edge Selection:** Reduces number of contractions; improves constant factors.
- **Early Termination:** Cuts down on trials after finding optimal results; further speed-up.

Conclusion: The implementation closely matches the theoretical runtime from the original paper, with practical improvements for real-world performance.