Karger's Minimum-Cut Algorithm: Detailed Analysis and Benchmarks

1 Karger Minimum Cut: Model and Algorithm

Graph model. We work with undirected multigraphs $G = (V, E)$ having $n = |V|$ vertices and $m = |E|$ edges; parallel edges are allowed, self-loops are ignored. For any non-empty proper $S \subset V$, the cut value is $\delta(S) = |\{(u, v) \in E : u \in S, v \notin S\}|$. The global minimum cut value is $\lambda(G) = \min_S \delta(S)$. The RAM model with unit-cost $\Theta(\log n)$-bit operations is assumed.

Single-run algorithm. Maintain a disjoint-set union (DSU) with one set per vertex and a component counter $c = n$:

1. While $c > 2$: draw a uniform random edge $(u, v) \in E$. If u and v are in different DSU sets, merge them and decrement c.
2. When exactly two supernodes remain, count edges whose endpoints lie in different DSU sets; return that count.

2 Probabilistic Analysis of Karger

All statements below condition on a fixed target minimum cut $C^\star$ of size $\lambda = \lambda(G)$.

2.1 Edge-incidence bound per step

Lemma 1. *Suppose i supernodes remain and no edge of $C^\star$ has been contracted so far. Then the probability that the next sampled edge lies in $C^\star$ is at most $2/i$.*

Proof. Because no edge of $C^\star$ has been contracted, every cut in the current multigraph corresponds to a cut in the original graph and therefore has value at least λ. In particular, the minimum degree of the current multigraph is at least λ: removing all edges incident to any vertex isolates it and forms a cut of size equal to that degree, which cannot be smaller than λ. With i supernodes, this minimum-degree bound implies at least $i\lambda/2$ edges in total (summing degrees and dividing by two). Exactly λ of those edges belong to $C^\star$. Sampling an edge uniformly therefore yields

$$\Pr[\text{pick edge in } C^\star \mid \text{no previous contraction of } C^\star] \le \frac{\lambda}{i\lambda/2} = \frac{2}{i}.$$

□

2.2 Single-run success probability

Theorem 1. *The probability that a single run returns the fixed minimum cut $C^\star$ is at least $2/(n(n-1))$.*

Proof. Let $p_i = 1-2/i$ be the lower bound on the probability that the contraction step at supernode count i avoids $C^\star$. Starting from $i = n$ and stopping at $i = 2$, the overall survival probability is

$$\prod_{i=n}^{3} p_i = \prod_{i=n}^{3} \left(1 - \frac{2}{i}\right) = \prod_{i=n}^{3} \frac{i-2}{i}.$$

To telescope, rewrite each factor $\frac{i-2}{i}$ as $\frac{i-2}{i-1} \cdot \frac{i-1}{i}$, obtaining

$$\prod_{i=n}^{3} \frac{i-2}{i} = \left(\prod_{i=n}^{3} \frac{i-2}{i-1}\right) \cdot \left(\prod_{i=n}^{3} \frac{i-1}{i}\right).$$

Now expand both products step by step. The first product is

$$\frac{n-2}{n-1} \cdot \frac{n-3}{n-2} \cdot \frac{n-4}{n-3} \cdots \frac{1}{2},$$

whose numerators and denominators cancel in sequence, leaving $\frac{1}{n-1}$. The second product is

$$\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \frac{n-3}{n-2} \cdots \frac{2}{3},$$

which cancels to $\frac{2}{n}$. Multiplying the two surviving terms yields

$$\prod_{i=n}^{3} \frac{i-2}{i} = \frac{1}{n-1} \cdot \frac{2}{n} = \frac{2}{n(n-1)}.$$

Conditional on survival of $C^\star$ through all contractions, the final two supernodes are separated exactly by $C^\star$, so the run outputs it with probability at least $2/(n(n-1))$. □

2.3 Amplification by repetition

Let $p_{\min}(n) = 2/(n(n-1))$ be the guaranteed per-run success probability. Running R independent trials and returning the minimum cut observed yields failure probability

$$\Pr[\text{all fail}] \le (1 - p_{\min}(n))^R \le e^{-p_{\min}(n)R},$$

using $1 - x \le e^{-x}$. Choosing

$$R(n) = \left\lceil \tfrac{1}{2} n(n-1) \ln \tfrac{1}{\delta_{\text{target}}} \right\rceil$$

drives this upper bound down to at most δ_{target} for a user-specified δ_{target}. The constant δ_{target} is fixed in the code; with that fixed value the resulting iteration counts are $R(30) = 23$, $R(60) = 91$, and $R(90) = 206$.

2.4 Time complexity

One contraction run examines $O(m)$ edges and performs DSU operations in $O(\alpha(n))$ amortized time, giving $O(m\alpha(n))$ time and $O(n)$ space. Repetition multiplies time by $R(n)$ without changing asymptotic space.

3 Implementation Snapshot and Baseline

The C++ source `src/main.cpp` stores both an edge list and an adjacency matrix per sampled graph. Karger is implemented exactly as analyzed, with DSU using path compression and union by size, and independent RNG states `seed` = `seed_base`+`rep`. The code always performs the predetermined $R(n)$ repetitions above and returns the minimum cut found. Stoer–Wagner is included only as an exact baseline oracle against which Karger is timed and scored; no probabilistic analysis is needed for it.

4 Graph Families

- `two_cluster_gnp` (dist_id = 0): planted bisection with even n; $p_{\text{in}} = 0.3$, $p_{\text{out}} = 0.05$; regenerated until connected.
- `gnp_sparse` (dist_id = 1): Erdős–Rényi with $p = \min(1, 6.0/n)$; regenerated until connected.
- `gnp_dense` (dist_id = 2): Erdős–Rényi with $p = 0.2$; regenerated until connected.
- `adversarial_barbell` (dist_id = 3): two cliques of sizes $\lfloor n/2 \rfloor$, $\lceil n/2 \rceil$ joined by one bridge edge; $\lambda(G) = 1$.
- `cubic_regular` (dist_id = 4): simple 3-regular graph from the configuration model with rejection until simplicity and connectivity hold.

5 Benchmark Methodology

Configuration. The driver `scripts/run_benchmarks.py` evaluates all combinations of Karger and Stoer–Wagner across the five distributions above, sizes $n \in \{30, 60, 90\}$, `reps` = 5, and `graphs_per_rep` = 20. Seeds are deterministic functions of $(\text{algo}, \text{dist}, n)$.

Protocol. For each graph, Stoer–Wagner runs once (not timed) to supply λ_{truth}. The chosen algorithm is timed. Per repetition the binary outputs `<total_time_ns> <sum_sq_error>` with `sum_sq_error` = $\sum(\lambda_{\text{hat}} - \lambda_{\text{truth}})^2$. The plotting script aggregates mean time per graph and mean squared error (MSE) and writes figures to `results/plots/`.

6 Benchmark Results (plots)

Each distribution has one figure with time-per-graph and MSE subplots:

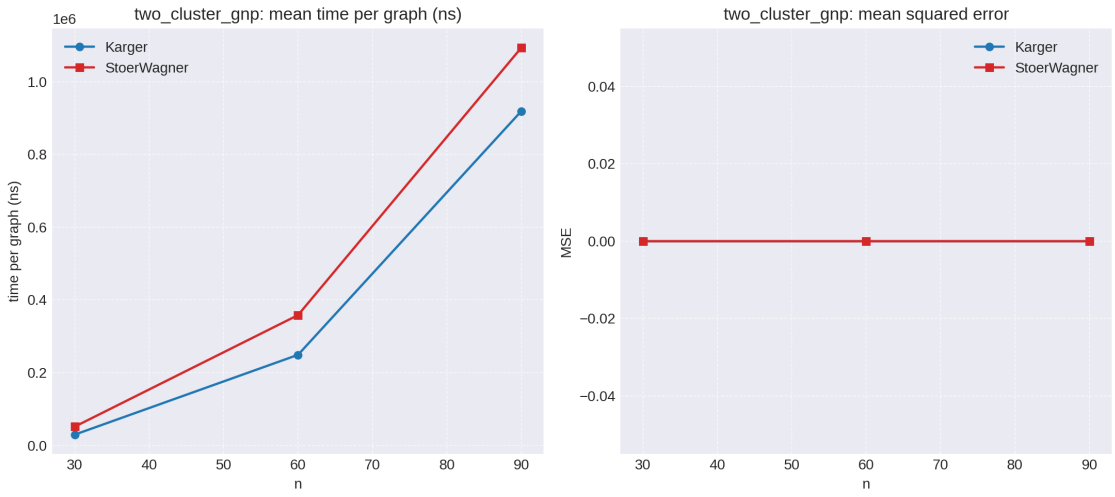

Figure 1: `two_cluster_gnp`: runtime and MSE.

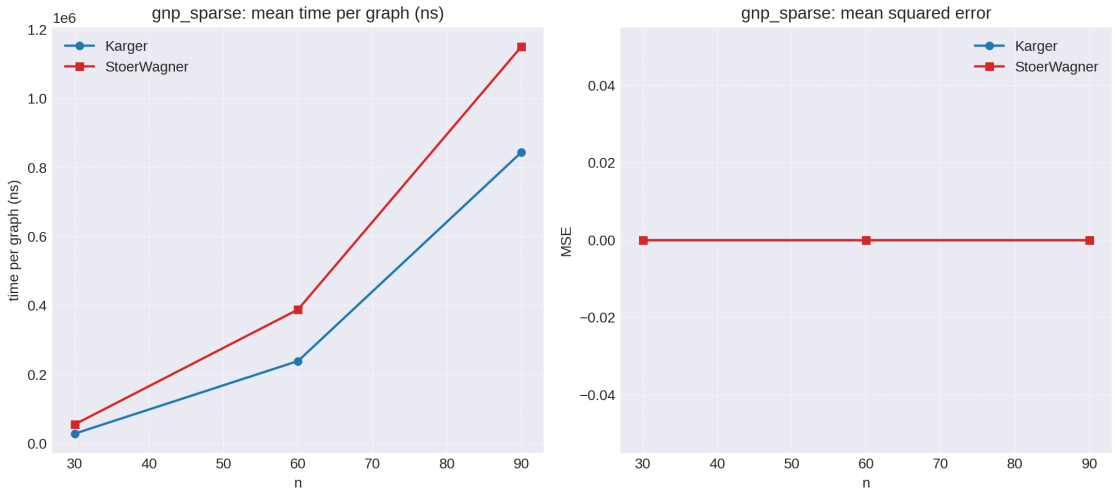

Figure 2: `gnp_sparse`: runtime and MSE.

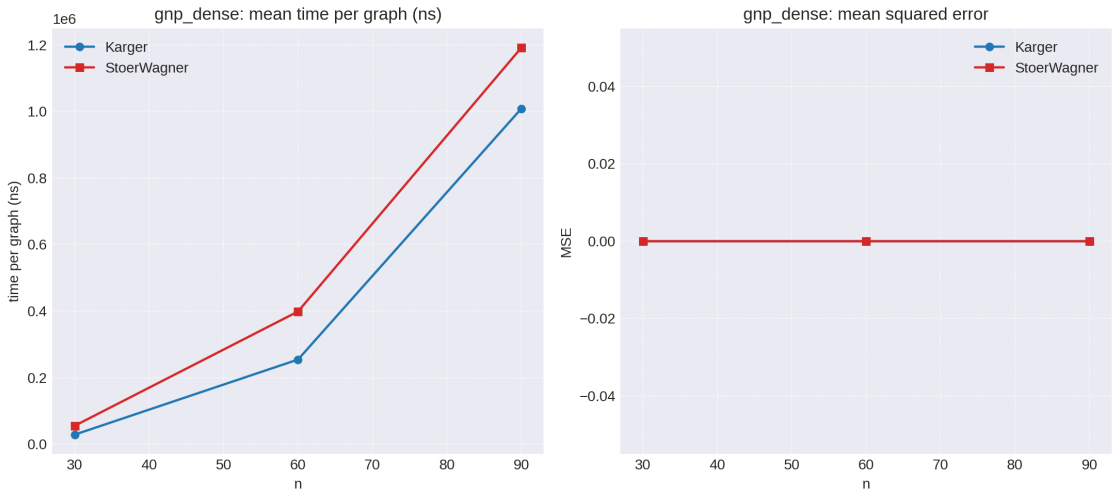

Figure 3: `gnp_dense`: runtime and MSE.

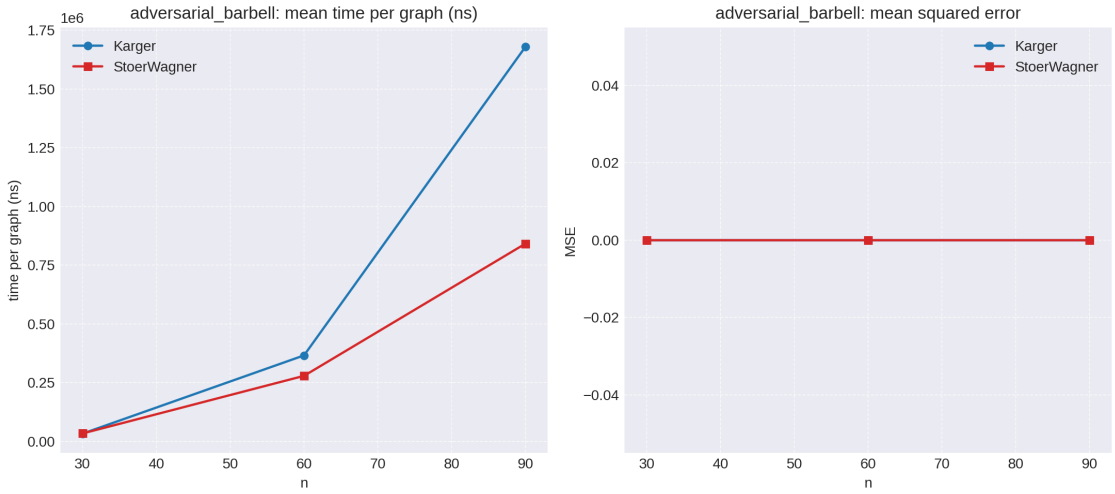

Figure 4: `adversarial_barbell`: runtime and MSE.

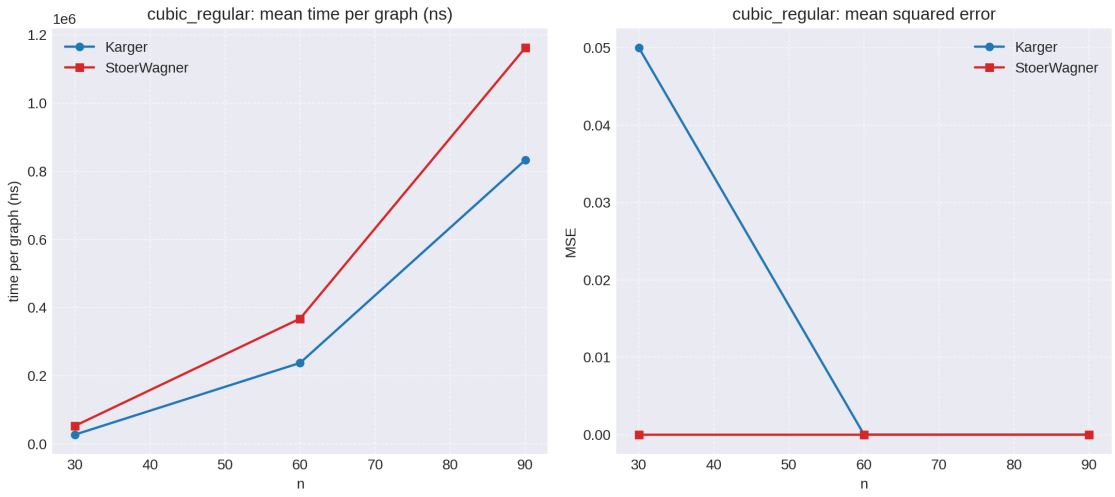

Figure 5: `cubic_regular`: runtime and MSE.

Plot interpretation. Karger is consistently faster than Stoer–Wagner on the random Erdős–Rényi families, with the time curves reflecting roughly quadratic growth in n due to repeated contractions. On the adversarial barbell, the deterministic baseline overtakes Karger for larger n because Karger still pays for all repetitions despite the single bridge. MSE is zero everywhere except for occasional misses on `cubic_regular` at $n = 30$, highlighting that the fixed repetition counts suffice for these inputs but remain probabilistic.

7 Real-World Context and Practicality

Applications.

- Network reliability: edge connectivity equals the fewest links whose failure disconnects the network, guiding robustness targets.
- VLSI partitioning: low-cut partitions reduce inter-block wiring, improving routability and timing slack.
- Image segmentation: graph cuts separate foreground and background in pixel or superpixel graphs.

Why Karger is rarely used in production.

- Success probability of a single run scales as $2/(n(n-1))$; large graphs require many repetitions to gain confidence, raising runtime.
- Outputs are probabilistic and non-certifying, conflicting with reliability demands in networking and hardware design.
- Deterministic (e.g., Stoer–Wagner) or stronger randomized algorithms offer predictable behavior and avoid reruns, so practitioners favor them.