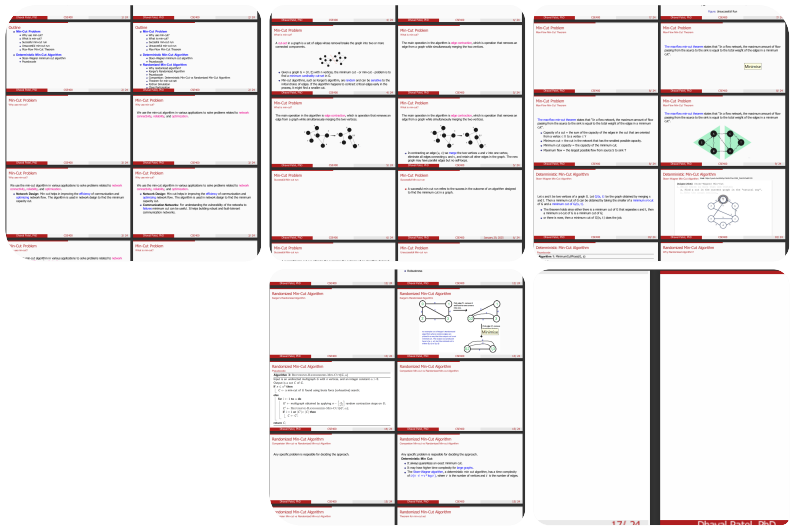
L10_Lecture Scribe



Generate a lecture scribe suitable for exam preparation. While writing the scribe, explain the definitions and assumptions explicitly, give step-by-step reasoning for definitions, assumptions, proofs and the example step-by-step solutions from the provided context only. All the reasoning should be based solely on the provided context. Make the logical flow explicit so that a student can follow the reasoning during revision. Do not introduce material that is not present in the context.

Below is a **complete structured lecture scribe for exam preparation** built **only from the provided lecture slides** on the **Randomized Min-Cut Algorithm (CSE400 – Lecture 10)**.

I explicitly explain:

- every definition
- every assumption
- every algorithmic step
- successful vs unsuccessful runs
- deterministic vs randomized reasoning
- theorem statements and their meaning

All logic follows **only what appears in the slides** — no extra theory added.

📘 CSE400 – Lecture 10 Scribe

Randomized Min-Cut Algorithm

---

1. Min-Cut Problem — Motivation

Why use min-cut?

The lecture states:

> Min-cut is used in applications related to **network connectivity, reliability, and optimization**.

We now explain each application step-by-step as given:

(a) Network Design

• Goal: improve **efficiency of communication**
• Also helps in **optimizing network flow**

Reasoning:

→ A minimum cut finds the smallest set of edges whose removal disconnects the network
→ This identifies the **minimum capacity cut**
→ Used to design networks with efficient flow

(b) Communication Networks

• Used to study **vulnerability to failures**

Reasoning:

→ If a small cut disconnects the network easily
→ the network is fragile

Thus:

→ Min-cut helps build **robust and fault-tolerant networks**

(c) VLSI Design

• Used to partition circuits into smaller components

Reasoning:

→ Min-cut minimizes **interconnectivity complexity**
→ reduces wiring and layout complexity

---

2. What is a Cut-Set?

Definition (from slide):

> A cut-set in a graph is a set of edges whose removal breaks the graph into two or more connected components.

Logical meaning:

1. Start with a connected graph
2. Remove some edges
3. If the graph becomes disconnected
4. That edge set is a **cut-set**

---

3. Min-Cut Problem

Given:

> A graph $G = (V, E)$ with n vertices

Goal:

> Find a **minimum cardinality cut-set**

Meaning:

• Among all possible cut-sets
• choose the one with the **smallest number of edges**

That is the **minimum cut**.

Important observation from lecture:

> Min-cut algorithms like Karger's are random and sensitive to early edge choices.

Reasoning:

• If the algorithm contracts a **critical edge early**
• the true minimum cut may be destroyed
• resulting in a larger cut

Thus:

→ randomness can cause success or failure

---

4. Edge Contraction — Core Operation

The main operation used in randomized min-cut is:

Edge Contraction

Definition:

> Remove an edge (u, v) and merge vertices u and v into one vertex.

Step-by-step:

1. Pick an edge (u, v)
2. Merge u and v into a single vertex
3. Remove all edges between u and v
4. Keep all other edges connected to the merged vertex
5. Parallel edges may appear
6. No self-loops remain

Purpose:

Each contraction:

→ reduces number of vertices
→ preserves some cuts probabilistically

---

5. Successful vs Unsuccessful Min-Cut Runs

✅ Successful Min-Cut Run

Definition:

> A run where the algorithm correctly finds the minimum cut.

Meaning:

- No critical edges were contracted early
- The final two super-nodes represent the true min-cut

❌ Unsuccessful Min-Cut Run

Definition:

> A run where the algorithm fails to identify the true minimum cut.

Reasoning:

- Important min-cut edges were contracted
- so the cut disappears
- final cut is larger than minimum

---

6. Max-Flow Min-Cut Theorem

Statement (from slide):

> In a flow network, the maximum flow from source to sink equals the total weight of edges in a minimum cut.

Definitions explained:

- **Capacity of a cut** = sum of capacities of edges crossing the cut
- **Minimum cut** = cut with smallest capacity
- **Maximum flow** = largest possible flow from source S to sink T

Logical relation:

Max possible flow you can push
= smallest bottleneck separating S and T

Thus:

$$\text{Max Flow} = \text{Min Cut Capacity}$$

---

7. Deterministic Min-Cut Algorithm (Stoer–Wagner)

Idea:

Let s and t be two vertices.

Form graph $G/\{s,t\}$ by merging s and t.

Then:

> The minimum cut of G is the smaller of:
> - **"a minimum s-t cut in G"**
> - **"a minimum cut in the merged graph"**

Reasoning:

• Either the min cut separates s and t
• or it doesn't

If it does → it's an s-t cut
If not → merging doesn't destroy it

Thus one of them must be the true minimum.

Pseudocode Structure (from slide)

Algorithm 1: MinimumCutPhase(G, a)

1. Start with A = {a}
2. Repeatedly add the most tightly connected vertex to A
3. When all vertices added → the cut weight of this phase is returned

Algorithm 2: MinimumCut(G)

1. Repeatedly run MinimumCutPhase
2. Track smallest cut seen
3. Merge last two vertices of each phase
4. Continue until one vertex remains
5. Return smallest cut found

Time Complexity (from slide):

$$O(V \cdot E + V^2 \log V)$$

---

8. Why Randomized Min-Cut?

The lecture states:

Randomized algorithms provide:

• probabilistic guarantee of success
• fewer iterations
• efficiency
• parallelization
• robustness
• avoidance of worst-case instances

---

9. Karger's Randomized Min-Cut Algorithm

Core idea:

Repeatedly apply **random edge contraction** until only two vertices remain.

Remaining parallel edges = cut size.

Why random?

Each contraction:

→ preserves the minimum cut with some probability

→ repeated trials increase chance of success

---

10. Recursive Randomized Algorithm (from pseudocode)

Input:

• Undirected multigraph G

• Integer α > 0

Output:

• A cut C

Step-by-step:

If n ≤ 3:

→ compute min-cut using brute force

Else:

Repeat α times:

1. Perform about $n - \sqrt{n/2}$ random contractions
2. Obtain smaller graph G′
3. Recursively compute min-cut C′ of G′
4. Keep the smallest cut among all trials

Return the smallest cut found.

---

11. Deterministic vs Randomized — Comparison

Deterministic Min-Cut:

✔ Always exact

❗ Higher time complexity for large graphs

Stoer-Wagner complexity:

$$O(VE + V^2 \log V)$$

Randomized Min-Cut:

- ✔ Approximate with high probability
- ✔ Faster for large graphs

Karger's algorithm complexity:

$$O(V^2)$$

---

12. Success Probability Theorem

From slide:

> The algorithm outputs a minimum cut with probability at least:

$$\frac{2}{n(n-1)}$$

Meaning:

• For one run, probability is small

• But repeating many times increases success greatly

Thus:

→ randomized approach works by repetition

✅ Final Logical Flow Summary

1. Min-cut identifies weakest edge separation
2. Used in networks, reliability, VLSI
3. Cut-set disconnects graph
4. Min-cut = smallest such set
5. Randomized algorithm contracts edges

6. Success depends on avoiding critical edges
7. Deterministic guarantees correctness but slower
8. Randomized is faster with probability
9. Repetition ensures high success

If you'd like, I can now:

✅ Convert this into **LaTeX exam-ready notes**
✅ Add **algorithm boxes formatted cleanly**
✅ Create **one-page revision sheet**
✅ Add **probability intuition for Karger's success**

Just tell me 👍