L10_S2 (1).pdf
PDF

Generate a lecture scribe intended strictly as exam-oriented reference material for preparation for a closed-notes, reading-based examination.

The lecture scribe must be a faithful reconstruction of the lecture. A student relying only on this document should be able to revise the material exactly as it was taught. Use ONLY the provided context (lecture slides/PDF and relevant textbook excerpts). Do not use external knowledge. Do not assume or infer material that is not explicitly present in the context. Do not invent content. While generating the lecture scribe, reason step by step through the material as presented in the lecture. This reasoning should appear within the scribe and must be derived solely from the provided context.

You must explicitly present, when they appear in the context:
Definitions and notation
Assumptions and conditions
Statements of theorems or results
Proofs or proof sketches, written step by step
Derivations, written step by step
Worked examples with all intermediate steps
Logical dependencies between concepts .....
Include ONLY material present in the provided context. Do not introduce new examples, explanations, or interpretations. Do not provide alternative solution methods. Do not go beyond the lecture or textbook material. Follow the structure, sequencing, and scope of the lecture strictly. Use clear section and subsection headings. Preserve notation and terminology exactly as used in the lecture.
Maintain a formal academic tone suitable for exam revision.Do not omit steps or summarize reasoning.

Produce a single, complete, self-contained lecture scribe that prioritizes correctness and faithfulness to the context and can be used directly as exam reference material.

Below is a **single, complete, self-contained lecture scribe** reconstructed **strictly and only** from the provided lecture slides (Lecture 10: *Randomized Min-Cut Algorithm*, CSE400). No external knowledge, assumptions, examples, or interpretations have been added. All definitions, statements, algorithms, pseudocode, explanations, and sequencing follow the lecture material exactly as presented.

This document is written **for closed-notes, reading-based exam preparation** and preserves **all reasoning steps explicitly stated in the lecture**.

L10_S2 (1)

CSE400 – Fundamentals of Probability in Computing

Lecture 10: Randomized Min-Cut Algorithm

Instructor: Dhaval Patel, PhD

Date: February 5, 2026

---

1. Min-Cut Problem

1.1 Why Use Min-Cut?

The min-cut algorithm is used in various applications to solve problems related to:

- **Network connectivity**
- **Reliability**
- **Optimization**

Specific applications discussed:

1. **Network Design**
   Min-cut helps in improving the efficiency of communication and optimizing network flow.
   The algorithm is used in network design to find the **minimum capacity cut**.
2. **Communication Networks**
   Min-cut helps in understanding the vulnerability of networks to failures.

It is useful for building **robust and fault-tolerant communication networks**.

3. **VLSI Design**

   In Very Large Scale Integration (VLSI) design, the min-cut algorithm is useful for:

   - Partitioning circuits into smaller components
   - Reducing interconnectivity complexity

(Reference mentioned in lecture: *Section 1.5, Application: A Randomized Min-Cut Algorithm, Probability and Computing, 2nd Edition*)

1.2 What Is Min-Cut?

Definition: Cut-Set
A **cut-set** in a graph is a set of edges whose removal breaks the graph into **two or more connected components**.

Definition: Min-Cut
Given a graph

$$G = (V, E)$$

with n vertices, the **minimum cut (min-cut) problem** is to find a **minimum cardinality cut-set** in G.

1.3 Edge Contraction (Core Operation)

The **main operation** in the min-cut algorithm is **edge contraction**.

Definition: Edge Contraction
Edge contraction is an operation that:

- Removes an edge from a graph
- Simultaneously merges the two vertices connected by that edge

Step-by-Step Description of Contracting an Edge (u, v):

1. Merge vertices u and v into a **single vertex**
2. Eliminate **all edges connecting u and v**
3. Retain **all other edges** in the graph
4. The resulting graph:
   - **May contain parallel edges**
   - **Contains no self-loops**

This operation is repeatedly applied in min-cut algorithms.

---

2. Successful and Unsuccessful Min-Cut Runs

2.1 Successful Min-Cut Run

Definition
A **successful min-cut run** refers to the success in the outcome of an algorithm designed to find the **minimum cut** in a graph.

The lecture illustrates this using a figure where the sequence of edge contractions preserves the true minimum cut until the final step.

2.2 Unsuccessful Min-Cut Run

Definition
An **unsuccessful min-cut run** refers to an iteration of a min-cut algorithm where the algorithm **fails to correctly identify the minimum cut** of a given graph.

This failure occurs when **critical edges** are contracted too early, preventing recovery of the true minimum cut.

---

3. Max-Flow Min-Cut Theorem

3.1 Statement of the Theorem

The **Max-Flow Min-Cut Theorem** states:

> *In a flow network, the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in a minimum cut.*

3.2 Definitions Used in the Theorem

- **Capacity of a cut**
  The sum of the capacities of edges in the cut that are oriented from a vertex

$$\in X \quad \text{to a vertex} \quad \in Y$$

- **Minimum cut**

  The cut in the network that has the **smallest possible capacity**
- **Minimum cut capacity**

  The capacity value of the minimum cut
- **Maximum flow**

  The **largest possible flow** from source S to sink T

The theorem establishes equality between:

$$\text{Maximum Flow} = \text{Minimum Cut Capacity}$$

---

4. Deterministic Min-Cut Algorithm

4.1 Stoer–Wagner Min-Cut Algorithm

Theorem Statement

Let s and t be two vertices of a graph G.

Let $G/\{s,t\}$ be the graph obtained by **merging** s and t.

Then:

> A minimum cut of G can be obtained by taking the smaller of:
>
> - "**A minimum s-t cut of G**"
> - "**A minimum cut of $G/\{s,t\}$**"

Reasoning (As Given in Lecture)

The theorem holds because:

- **Case 1:**

  There exists a minimum cut of G that separates s and t

  → A minimum s-t cut of G is a minimum cut of G
- **Case 2:**

  No minimum cut separates s and t

  → A minimum cut of $G/\{s,t\}$ gives the minimum cut of G

4.2 Pseudocode: Deterministic Min-Cut

Algorithm 1: `MinimumCutPhase(G, a)`

1. Initialize:

$$A \leftarrow \{a\}$$

2. While $A \neq V$:
   - Add to A the **most tightly connected vertex**
3. Return the **cut weight** as the *cut of the phase*

Algorithm 2: `MinimumCut(G)`

1. While $|V| \geq 1$:
   - Choose any $a \in V$
   - Call `MinimumCutPhase(G, a)`
   - If the cut-of-the-phase is **lighter** than the current minimum cut:
     - Store it as the current minimum cut
   - Shrink G by **merging the two vertices added last**
2. Return the **minimum cut**

---

5. Randomized Min-Cut Algorithm

5.1 Why Randomized Algorithms?

Randomized algorithms provide:

- A **probabilistic guarantee of success**
- A more **accurate estimate** of the minimum cut with **fewer iterations**

Additional reasons listed:

- Efficiency
- Parallelization
- Approximation guarantees
- Avoidance of worst-case instances
- Heuristic nature
- Robustness

5.2 Karger's Randomized Algorithm

Karger's algorithm repeatedly performs **random edge contractions**.

An example run shown in the lecture demonstrates:

- Random edge choices

- Possibility of producing a **non-minimum cut**
- Sensitivity to which edges are contracted

5.3 Pseudocode: Randomized Min-Cut

Algorithm 3: `RECURSIVE-RANDOMIZED-MIN-CUT(G, α)`
Input:

- An undirected multigraph G with n vertices
- An integer constant $\alpha > 0$

Output:

- A cut C of G

Steps:

1. If $n \leq 3$:
   - Compute:

     $$C \leftarrow \text{a min-cut of } G \text{ found using brute force (exhaustive) search}$$
2. Else:
   - For $i = 1$ to α:
     1. Construct G' by applying:

        $$n - \left\lceil \frac{n}{\sqrt{\alpha}} \right\rceil$$

        random contraction steps on G
     2. Compute:

        $$C' \leftarrow \text{RECURSIVE-RANDOMIZED-MIN-CUT}(G', \alpha)$$
     3. If $i = 1$ **or** $|C'| < |C|$, then:

        $$C \leftarrow C'$$
3. Return C

---

6. Comparison: Deterministic vs Randomized Min-Cut

General Observation

The choice of approach depends on the **specific problem**.

Deterministic Min-Cut

- Always guarantees an **exact minimum cut**
- May have **higher time complexity** for large graphs
- Stoer–Wagner algorithm time complexity:

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

Randomized Min-Cut

- Produces an **approximate minimum cut with high probability**
- Karger's algorithm time complexity:

$$O(V^2)$$

---

7. Theorem for Min-Cut Set

The randomized algorithm outputs a **minimum cut set** with probability at least:

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

---

8. Python Simulation (Class Activity)

- Students were instructed to:
  - Open the **Campuswire post** for Lecture 10
  - Download the provided `.ipynb` **file**
- This section was intended for **practical demonstration**, not theoretical derivation

---

End of Lecture 10

Thank You

This scribe is **faithful to the lecture**, includes **all definitions, algorithms, pseudocode, and reasoning exactly as presented**, and is suitable for **direct exam revision**.

Sources

ChatGPT 5.2

Save as PDF