Lecture Scribe: Randomized Min-Cut Algorithm

Course: CSE400 – Fundamentals of Probability in Computing
Lecture: 10
Instructor: Dhaval Patel, PhD
(Prepared strictly from the provided PDF)

1. Min-Cut Problem

1.1 Why Use Min-Cut?

Min-cut algorithms are used to solve problems related to:

- **Network connectivity**
- **Network reliability**
- **Network optimization**

The goal is to identify **critical edges** whose removal disconnects the network with **minimum cost or size**.

Applications of Min-Cut

1. **Network Design**
   - Used to improve communication efficiency.
   - Helps find the **minimum capacity cut** in a network.
2. **Communication Networks**
   - Helps analyze **network vulnerability to failures**.
   - Supports the design of **fault-tolerant and robust networks**.
3. **VLSI Design**
   - Used for **partitioning circuits** into smaller components.
   - Reduces **interconnectivity complexity**.

2. What is Min-Cut?

2.1 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.

2.2 Definition: Minimum Cut

Given a graph

$$G = (V, E)$$

with n vertices,

- The **minimum cut (min-cut)** problem is to find a **cut-set with minimum cardinality**.

2.3 Important Observation

- Min-cut algorithms such as **Karger's algorithm** are **randomized**.
- Their result depends on **which edges are selected early**.
- If **critical edges** are contracted early, the algorithm may **fail to find the true minimum cut**.

3. Edge Contraction

3.1 Definition: Edge Contraction

The **main operation** used in min-cut algorithms is **edge contraction**.

3.2 Step-by-Step Edge Contraction Process

For an edge (u, v):

1. Vertices u and v are merged into a **single vertex**.
2. All edges between u and v are removed.
3. All remaining edges are retained.
4. The resulting graph:
    - May contain **parallel edges**
    - Contains **no self-loops**

4. Min-Cut Runs

4.1 Successful Min-Cut Run

A **successful min-cut run** refers to:

- An execution of the algorithm that **correctly identifies the minimum cut** of the graph.

(Shown graphically in the lecture slides.)

4.2 Unsuccessful Min-Cut Run

An **unsuccessful min-cut run** refers to:

- An execution where the algorithm **fails to identify the minimum cut**.
- This typically happens when **important edges are contracted too early**.

(Shown graphically in the lecture slides.)

5. Max-Flow Min-Cut Theorem

5.1 Theorem Statement

> 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.

5.2 Definitions Used in the Theorem

1. **Capacity of a Cut**
   - The sum of capacities of edges oriented from:
     - vertex $\in X$ to vertex $\in Y$
2. **Minimum Cut**
   - The cut with the **smallest capacity**
3. **Minimum Cut Capacity**
   - The capacity value of the minimum cut
4. **Maximum Flow**
   - The largest possible flow from **source** S to **sink** T

6. Deterministic Min-Cut Algorithm

6.1 Stoer–Wagner Min-Cut Algorithm

Theorem Used

Let s and t be two vertices of graph G.

Let

$$G/\{s,t\}$$

be the graph obtained by **merging** s **and** t.

A minimum cut of G is obtained by taking the **smaller** of:

1. A **minimum** s–t **cut** of G
2. A **minimum cut of** $G/\{s,t\}$

Logical Reasoning (As Given in Lecture)

There are two cases:

1. **Case 1:**
   A minimum cut of G separates s and t
   $\rightarrow$ Then the **minimum** s–t **cut** is the minimum cut of G

2. **Case 2:**
   No minimum cut separates s and t
   $\rightarrow$ Then the **minimum cut of** $G/\{s,t\}$ is valid

Hence, the theorem holds.

6.2 Pseudocode

Algorithm 1: MinimumCutPhase(G, a)

Step-by-Step

1. Initialize:
$$A \leftarrow \{a\}$$
2. While $A \neq V$:
   - Add to A the **most tightly connected vertex**
3. Return:
   - The **cut weight**, called the *cut of the phase*

Algorithm 2: MinimumCut(G)

Step-by-Step

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

7. Randomized Min-Cut Algorithm

7.1 Why Randomized Algorithms?

Randomized algorithms provide:

- **Probabilistic guarantees of success**
- **Accurate estimates** with fewer iterations

Advantages Mentioned

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

7.2 Karger's Randomized Algorithm

- A **randomized algorithm** for finding the minimum cut
- Uses **random edge contraction**
- Success probability increases with repeated runs

(Pseudocode shown in lecture slides.)

8. Deterministic vs Randomized Min-Cut

Deterministic Min-Cut

- Always guarantees **exact minimum cut**
- Higher time complexity for large graphs
- **Stoer–Wagner complexity**:

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

Randomized Min-Cut

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

$$O(V^2)$$

9. Theorem for Randomized Min-Cut

Theorem Statement

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

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

where n is the number of vertices.

10. Python Simulation (Lecture Activity)

- Students are instructed to:
  - Open the **Campuswire post for Lecture 10**
  - Download the provided `.ipynb` **file**
  - Run the simulation to observe randomized min-cut behavior

End of Lecture

Thank You