

School of Engineering and Applied Science (SEAS), Ahmedabad University

**CSE 400: Fundamentals of Probability in Computing**

**Lecture 10 Scribe**

**Group No.:** S1 G17  
**Name:** ARYA PATEL  
**ID:** AU2440250  
**Date:** February 5, 2026

---

## **Lecture 10: Randomized Min-Cut Algorithm**

### **1 Outline**

- Min-Cut Problem
- Why use min-cut?
- What is min-cut?
- Successful min-cut run
- Unsuccessful min-cut run
- Max-Flow Min-Cut Theorem
- Deterministic Min-Cut Algorithm
- Stoer–Wagner Minimum Cut Algorithm
- Pseudocode
- Randomized Min-Cut Algorithm
- Why randomized algorithm?
- Karger’s Randomized Algorithm
- Pseudocode
- Comparison: Deterministic vs Randomized Min-Cut
- Theorem for min-cut set
- Python Simulation

## 2 Min-Cut Problem

### 2.1 Why use min-cut?

We use the min-cut algorithm in various applications to solve problems related to network connectivity, reliability, and optimization.

- **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.
- **Communication Networks:** For understanding the vulnerability of networks to failures, minimum cut can be useful. It helps in building robust and fault-tolerant communication networks.
- **VLSI Design:** In Very Large Scale Integration (VLSI) design, the algorithm is useful for partitioning circuits into smaller components leading to reduced interconnectivity complexity.

### 2.2 What is min-cut?

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

[Minimum Cut] Given a graph  $G = (V, E)$  with  $n$  vertices, the minimum cut (min-cut) problem is to find a cut-set of minimum cardinality in  $G$ .

Min-cut algorithms such as Karger's algorithm are random and can be sensitive to the initial choice of edges.

### 2.3 Edge Contraction

[Edge Contraction] The main operation in the algorithm is edge contraction, which removes an edge  $(u, v)$  from the graph while simultaneously merging vertices  $u$  and  $v$  into a single vertex.

- Vertices  $u$  and  $v$  are merged into one vertex.
- All edges connecting  $u$  and  $v$  are eliminated.
- All other edges are retained.
- The resulting graph may contain parallel edges but no self-loops.

## 3 Successful Min-Cut Run

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

### Graph Contraction Sequence (Successful Run)

Step	Operation	Resulting Vertex Sets
Initial	Original graph	$\{1, 2, 3, 4\}$
1	Contract edge $(2, 3)$	$\{1, \{2, 3\}, 4\}$
2	Contract edge $(\{2, 3\}, 4)$	$\{1, \{2, 3, 4\}\}$

The remaining parallel edges between the two supernodes define the minimum cut.

## 4 Unsuccessful Min-Cut Run

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.

### Graph Contraction Sequence (Unsuccessful Run)

Step	Operation	Resulting Vertex Sets
Initial	Original graph	$\{1, 2, 3, 4\}$
1	Contract critical edge	$\{\{1, 2\}, 3, 4\}$
2	Contract another edge	$\{\{1, 2, 3\}, 4\}$

The final cut does not correspond to the minimum cut of the original graph.

## 5 Max-Flow Min-Cut Theorem

[Max-Flow Min-Cut Theorem] 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.

### Definitions

- **Capacity of a cut:** Sum of capacities of edges oriented from  $X$  to  $Y$ .
- **Minimum cut:** Cut with smallest possible capacity.
- **Maximum flow:** Largest possible flow from source  $S$  to sink  $T$ .

## 6 Deterministic Min-Cut Algorithm

### 6.1 Stoer–Wagner Min-Cut Algorithm

Let  $s$  and  $t$  be two vertices of a graph  $G$ . Let  $G/\{s, t\}$  denote the graph obtained by merging  $s$  and  $t$ .

A minimum cut of  $G$  is the smaller of:

- A minimum  $s$ - $t$  cut of  $G$
- A minimum cut of  $G/\{s, t\}$

### Pseudocode

Algorithm 1: MinimumCutPhase( $G$ ,  $a$ )

$A \leftarrow \{a\}$

while  $A \neq V$  do

    add to  $A$  the most tightly connected vertex

return cut weight

Algorithm 2: MinimumCut( $G$ )

while  $|V| > 1$  do

    choose any  $a$  from  $V$

    MinimumCutPhase( $G$ ,  $a$ )

    if cut-of-the-phase < current minimum then

        store cut

```

    shrink G by merging last two vertices
return minimum cut

```

## 7 Randomized Min-Cut Algorithm

### 7.1 Why Randomized Algorithm?

- Probabilistic guarantee of success
- Fewer iterations
- Efficiency
- Parallelization
- Approximation guarantees
- Avoidance of worst-case instances
- Robustness

### 7.2 Karger's Randomized Algorithm

#### Worked Example (Step-by-Step)

- Start with undirected multigraph  $G = (V, E)$
- Randomly select an edge
- Contract the selected edge
- Repeat until only two vertices remain
- Remaining parallel edges define the cut

#### Pseudocode

```

Algorithm: Recursive-Randomized-Min-Cut( $G, n$ )
if  $n = 2$  then
    return brute-force min-cut
else
    for  $i = 1$  to  $n$  do
         $G' \leftarrow$  apply  $n - n/2$  contractions
         $C' \leftarrow$  Recursive-Randomized-Min-Cut( $G', n/2$ )
        if  $|C'| < |C|$  then
             $C \leftarrow C'$ 
return  $C$ 

```

## 8 Theorem for Min-Cut Set

The algorithm outputs a minimum cut with probability at least

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

**Proof (As Presented)**

At each contraction step, the probability of not contracting a minimum cut edge is:

$$1 - \frac{\lambda}{|E|} \geq 1 - \frac{2}{n}.$$

Thus, the probability that no minimum cut edge is contracted over  $(n - 2)$  steps is:

$$\prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) = \frac{2}{n(n-1)}.$$

**9 Comparison: Deterministic vs Randomized Min-Cut**

Deterministic Min-Cut	Randomized Min-Cut
Exact minimum cut	Approximate with high probability
Higher time complexity	Lower expected time
$O(VE + V^2 \log V)$	$O(V^2)$

**10 Python Simulation**

- Students instructed to open Campuswire post for Lecture 10
- Download the provided `.ipynb` file

---

End of Lecture 10 Scribe