

Generate a Lecture Scribe as chat from the given PPT "L10_S1_A" using the structure from the PDF "Example_Scribe". Follow the b

L10_S1

📄 Uploaded file not shown

Example_Scribe

PDF PDF

Generate a Lecture Scribe as chat from the given PPT "L10_S1_A" using the structure from the PDF "Example_Scribe".

Follow the below two sets of instructions properly to generate the Lecture Scribe :

Instructions : 1) The lecture scribe is to be generated as per the structure given in Example_Scribe 2) Use the content from PPT L10_S1_A only and DO NOT generate or add new content 3) The lecture scribe should be created as exam reference material 4) The lecture scribe must include proofs, derivations

Instructions for following the example scribe : 1) For List of topics covered, the lecture scribe must cover the list of all topics covered in the PPT 2) For Explanation of Topics Covered, the lecture scribe must cover detailed explanation of topics covered in lecture (pointwise) 3) For List of Definitions and Theorems, the lecture scribe must include Definitions with notations, associated with definitions/assumption with

exact statement of definitions and theorems along with proof of theorem step by step 4) For Important Examples, select examples from the PPT attached (L10_S1_A) in such a way that it covers all the concepts discussed. *Write step by step solution to examples*. INCLUDE EXAMPLES FROM PPT ONLY, DO NOT GENERATE BY YOURSELF 5) For the List of Important Formulas, write the list of all the important formulas from the PPT only and write it in tabular format



Based on the provided presentation "L10_S1.pdf", here is the Lecture Scribe following the structure of "Example_Scribe".

Lecture 10: Randomized Min-Cut Algorithm

1) List of Topics Covered

1. Min-Cut Problem

- Definition of Min-Cut
- Edge Contraction
- Successful vs. Unsuccessful Runs

2. Applications of Min-Cut

- Network Design
- Communication Networks
- VLSI Design

3. Max-Flow Min-Cut Theorem

- Statement of the Theorem

- Relationship between Flow and Cut

4. **Deterministic Min-Cut Algorithm**

- Stoer-Wagner Algorithm
- Logic and Psuedocode

5. **Randomized Min-Cut Algorithm**

- Why use Randomized Algorithms?
- Karger's Randomized Algorithm
- Recursive Randomized Min-Cut

6. **Comparison**

- Deterministic vs. Randomized Approaches
-

2) Explanation of Topics Covered

1) Min-Cut Problem

- **Definition:** The Min-Cut problem involves finding a minimum cardinality cut-set in a graph $G = (V, E)$. A cut-set is a set of edges whose removal disconnects the graph into two or more components.
- **Edge Contraction:** This is the main operation in the algorithm. It involves removing an edge (u, v) and merging the two vertices u and v into a single vertex. All edges connecting u and v are eliminated, while other edges are retained. The resulting graph may contain parallel edges but no self-loops.
- **Algorithm Sensitivity:** Min-cut algorithms, particularly Karger's, are random and sensitive to the initial choice of edges. Contracting critical edges early can lead to finding a smaller cut, while incorrect choices may lead to an unsuccessful run.

2) Applications of Min-Cut The algorithm is used to solve problems regarding network connectivity, reliability, and

optimization.

- **Network Design:** Used to find the minimum capacity cut, improving communication efficiency and optimizing flow.
- **Communication Networks:** Helps understand network vulnerability to failures, aiding in building robust, fault-tolerant systems.
- **VLSI Design:** Useful for partitioning circuits into smaller components to reduce interconnectivity complexity.

3) Max-Flow Min-Cut Theorem

- This theorem connects flow networks to cuts. It states that in a flow network, the maximum amount of flow passing from a source to a sink is equal to the total weight of the edges in a minimum cut.

4) Deterministic Min-Cut Algorithm (Stoer-Wagner)

- This approach guarantees an exact minimum cut.
- The algorithm operates by finding a cut in the current graph in a "natural way," merging vertices, and storing the "cut-of-the-phase" if it is lighter than the current minimum.
- While exact, it may have higher time complexity for large graphs ($O(VE + V^2 \log V)$).

5) Randomized Min-Cut Algorithm (Karger's)

- **Why Randomized?:** Randomized algorithms offer a probabilistic guarantee of success and can provide accurate estimates with fewer iterations. They are often efficient, robust, and parallelizable.
- **Process:** The algorithm randomly picks edges to contract. It continues this process until only two

vertices remain. The edges connecting these two vertices represent the cut.

- **Recursive Approach:** To improve probability, a recursive version (Recursive-Randomized-Min-Cut) is used where the graph is contracted to a certain size, and the process is repeated.

6) Comparison

- **Deterministic:** Always finds the exact minimum cut but is computationally expensive ($O(VE + V^2 \log V)$).
 - **Randomized:** Provides an approximate minimum cut with high probability and has a time complexity of $O(V^2)$. The choice of approach depends on the specific problem constraints.
-

3) List of Definitions and Theorems

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

3) Definition (Edge Contraction) In contracting an edge (u, v) , we merge the two vertices u and v into one vertex, eliminate all edges connecting u and v , and retain all other edges in the graph.

4) Theorem (Max-Flow Min-Cut) "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".

- **Maximum Flow:** The largest possible flow from source S to sink T .
- **Minimum Cut:** The cut in the network that has the smallest possible capacity.

5) Theorem (Stoer-Wagner Logic) 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 and a minimum cut of $G/\{s, t\}$.

- **Proof/Reasoning:** The theorem holds since either there is a minimum cut of G that separates s and t (in which case a minimum $s - t$ -cut is the minimum cut), or there is none (in which case a minimum cut of $G/\{s, t\}$ is the solution).

6) Theorem (Min-Cut Set Probability) The randomized algorithm outputs a min-cut set with probability at least:

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

(Note: Derived from slide visual notation).

4) Important Examples

Example 1: Successful Min-Cut Run

- **Context:** A graph undergoes edge contraction steps to find the minimum cut.
- **Process:**
 1. The algorithm randomly selects edges to contract.
 2. Edges are removed, and vertices are merged.
 3. In a successful run, the algorithm avoids contracting the specific edges that form the true

minimum cut.

4. **Result:** The final two vertices are connected by the minimum number of edges possible for that graph. The visual example in the slides shows a graph reduced to two super-nodes connected by 2 edges, correctly identifying the min-cut of size 2.

Example 2: Unsuccessful Min-Cut Run

- **Context:** An iteration where the random choice of edges fails to find the optimal cut.
- **Process:**
 1. The algorithm selects an edge that belongs to the actual minimum cut set to contract.
 2. This "critical edge" is removed and the vertices merged, making it impossible to output the true minimum cut later.
 3. **Result:** The final cut found has a higher cardinality (e.g., 3 edges) than the true minimum cut (e.g., 2 edges). This highlights the sensitivity to the initial choice of edges.

Example 3: Karger's Randomized Algorithm Execution

- **Given:** A graph with numbered vertices (0, 1, 2, 3) and edges (a, b, c, d, e).
- **Step 1:** Pick edge 'b' (connecting 0 and 2). Remove it and fuse corners 0 and 2 into one vertex $\{0, 2\}$.
- **Step 2:** The graph now has nodes $\{0, 2\}$, 1, and 3.
- **Step 3:** Pick edge 'd' (connecting 1 and 3). Remove it and fuse corners 1 and 3 into one vertex $\{1, 3\}$.
- **Final State:** Two vertices remain: $\{0, 2\}$ and $\{1, 3\}$.
- **Output Cut:** The edges connecting these two super-nodes are $\{a, c, e\}$.

- **Observation:** The output cut size is 3 (edges a, c, e). However, the minimal cut for this graph is actually size 2 (either $\{b, e\}$ or $\{a, d\}$).
- **Conclusion:** This specific example demonstrates a run where the output is *not* the minimal cut, illustrating the probabilistic nature of the algorithm.

5) List of Important Formulas

Formula / Algorithm	Description
Stoer-Wagner Complexity	$O(V \cdot E + V^2 \log V)$
Karger's Complexity	$O(V^2)$
Karger's Success Probability	$\geq \frac{2}{n^2 - n}$
Recursive Step Threshold	If $n \leq \alpha^3$, use brute force
Contraction Steps	Apply $n - \lceil \frac{n}{\alpha} \rceil$ contraction steps

