Min-Cut Algorithms: Deterministic vs. Randomized

L10_S2

PDF

Generate a lecture scribe notes which can be used as exam reference material .Use only the provided context (lecture slides) dont add anything of you own . While writing the scribe reason step by step through all definitions , notation, assumptions , theorem statements , proofs ,derivations and worked examples exactly as given in the lecture slides. Make the logical flow explicit so that a student can follow the reasoning during exam revision . Include proofs, derivations and examples only if they appear in the provided context. Do not introduce new explanations examples or material beyond the lecture scope . Maintain a formal and academic tone and strict faithfulness to the lecture structure

Lecture Scribe Notes: Randomized Min-Cut Algorithm

Course: CSE400 - Fundamentals of Probability in Computing

Lecturer: Dhaval Patel, PhD

Date: February 5, 2026

1. Introduction to the Min-Cut Problem

1.1 Definitions and Terminology

- **Cut-set:** A set of edges in a graph whose removal breaks the graph into two or more connected components.
- **Min-Cut Problem:** Given a graph $G = (V, E)$ with n vertices, the problem is to find a cut-set with minimum cardinality.
- **Edge Contraction:** The fundamental operation used in min-cut algorithms. It involves:
  - Removing an edge (u, v) from the graph.
  - Merging the two vertices u and v into a single vertex.
  - Eliminating all edges connecting u and v (no self-loops).
  - Retaining all other edges, which may result in parallel edges.

1.2 Applications

Min-cut algorithms are utilized for network connectivity, reliability, and optimization:

- **Network Design:** To find the minimum capacity cut and optimize network flow and communication efficiency.
- **Communication Networks:** To understand network vulnerability to failures and build fault-tolerant systems.
- **VLSI Design:** To partition circuits into smaller components, reducing interconnectivity complexity.

2. Max-Flow Min-Cut Theorem

The theorem establishes the relationship between flow and cuts in a network:

- **Statement:** In a flow network, the maximum amount of flow passing from the source (S) to the sink (T) is equal to the total weight of the edges in a minimum cut .
- **Key Components:**
  - **Capacity of a cut:** The sum of capacities of edges oriented from a vertex in set X to a vertex in set Y .
  - **Minimum cut:** The network cut with the smallest possible capacity.
  - **Maximum flow:** The largest possible flow from S to T.

---

3. Deterministic Min-Cut: Stoer-Wagner Algorithm

3.1 Theorem for Deterministic Approach

Let s and t be two vertices of graph G. Let $G/\{s,t\}$ be the graph obtained by merging them. A minimum cut of G is the smaller of:

1. A minimum $s-t$-cut of G (if the min-cut separates s and t).
2. A minimum cut of $G/\{s,t\}$ (if the min-cut does not separate s and t).

3.2 Stoer-Wagner Pseudocode

- **Algorithm 1: MinimumCutPhase(G, a)**
  1. Start with set $A = \{a\}$.

2. While $A \neq V$, add the most tightly connected vertex to A.
3. Return the cut weight of the last phase.

- **Algorithm 2: MinimumCut(G)**
  1. While $|V|[cite_start] \geq 1$:
     - Choose any vertex a from V.
     - Run `MinimumCutPhase(G, a)`.
     - If the "cut-of-the-phase" is lighter than the current minimum cut, store it as the new minimum cut.
     - Shrink G by merging the two vertices added last in the phase.
  2. Return the minimum cut found.

---

4. Randomized Min-Cut: Karger's Algorithm

4.1 Rationale for Randomization

Randomized algorithms provide a probabilistic guarantee of success and can provide accurate estimates with fewer iterations. Benefits include efficiency, parallelization, approximation guarantees, and avoidance of worst-case instances.

4.2 Karger's Algorithm Mechanism and Example

The algorithm is sensitive to the initial choice of edges. If critical edges are contracted early, the algorithm may fail to find the true minimum cut.

Example of an Unsuccessful Run:

- **Input Graph:** Vertices $\{0, 1, 2, 3\}$ with edges a, b, c, d, e.

- **Step 1:** Pick edge 'b', remove it, and fuse corners into vertex $\{0, 2\}$.
- **Step 2:** Pick edge 'd', remove it, and fuse corners into vertex $\{1, 3\}$.
- **Outcome:** The output cut is $\{a, c, e\}$, which is not minimal.
- **Actual Minimal Cuts:** Either $\{b, e\}$ or $\{a, d\}$.

4.3 Karger's Recursive Pseudocode

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

- **Input:** Undirected multigraph G with n vertices; integer constant $\alpha > 0$.
- **Base Case:** If $n \leq \alpha^3$, find min-cut via brute force.
- **Recursive Step:**
  1. For $i = 1$ to α:
     - Create multigraph G' by applying $n - \lceil n/ \quad \alpha \rceil$ random contraction steps.
     - Recursively call the algorithm on G' to get cut C'.
     - If $i = 1$ or $|C'|[cite_start] < |C|$, update $C = C'$.
- **Output:** Cut C.

4.4 Success Probability Theorem

The algorithm outputs a min-cut set with a probability of at least:

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

5. Comparison: Deterministic vs. Randomized

Feature	Deterministic (Stoer-Wagner)	Randomized (Karger's)
Guarantee	Always guarantees an exact minimum cut.	Provides an approximate min-cut with high probability.
Time Complexity	$O(VE + V^2 \log V)$.	$O(V^2)$.
Scalability	May have higher complexity for large graphs.	Generally more efficient for large datasets.

Would you like me to summarize the specific Python simulation steps mentioned for Karger's algorithm?