ChatGPT 5.2

L10_S1.pdf
PDF

You are an expert lecture scribe generator

You have to generate a complete lecture scribe intended to serve as exam-oriented reference material.
This scribe must allow a student to fully reconstruct the entire lecture for a closed-notes,
reading-based exam using this document as the sole reference.

MANDATORY CONTEXT AND SCOPE CONSTRAINTS:

Use ONLY the provided lecture slides and permitted textbook material as context.
Treat the provided lecture material as complete, ordered, and mandatory.
Cover the lecture from the FIRST instructional slide to the FINAL instructional slide.
Do NOT stop early.
Do NOT assume logical stopping points.
Every section, subsection, definition, assumption, theorem, proposition, example,
derivation, proof, equation, table, figure, diagram, and stated result appearing
anywhere in the context MUST be included.
If new concepts appear later in the lecture, they MUST also be included.

MANDATORY CONTENT RULES:

Do NOT summarize, compress, or omit material.
Do NOT skip material even if sections appear conceptually independent.
ALL worked examples must be written fully with intermediate steps exactly as presented.
Proofs and derivations must be written step-by-step exactly as in the lecture.
If a slide contains only statements or definitions, reproduce them formally.
If a slide contains no mathematics, reproduce its content without interpretation.
Do NOT introduce new examples, intuition, motivation, commentary, or explanations.
Do NOT add alternative solution methods.
Do NOT add background knowledge.
Do NOT add transitional or conversational text.
Do NOT include anything not directly traceable to the provided context.

STRUCTURE REQUIREMENT:

Use EXACTLY the same structure as the example lecture scribe below.
Do NOT rename sections.
Do NOT reorder sections.
Do NOT add or remove sections.

Example Lecture Scribe (structure only, do not copy content):

Topic Title

Overview
Brief statement describing what the lecture covers.

Definitions and Notation
Formal definitions and symbols introduced in the lecture.

Main Results
Statements of theorems, propositions, or key results, section-wise.
Include figures, graphs, tables, or diagrams ONLY if they explicitly appear in the lecture.
If a figure is referenced in the lecture, include a formal textual description exactly as stated.

Derivations / Proofs
Step-by-step derivations or proofs exactly as presented.

Worked Examples
Example problems solved in class with all intermediate steps.

Summary
Concise recap strictly limited to formulas, results, and definitions already stated.

FINAL INSTRUCTION:

Follow the lecture order strictly from beginning to end.
The output must be complete rather than concise.
This document must be suitable as the sole exam-reference material.

2/2

Topic Title
Randomized Min-Cut Algorithm
(CSE400 - Fundamentals of Probability in Computing, Lecture 10) L10_S1

Overview

This lecture covers:

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 Min-Cut vs Randomized Min-Cut Algorithm

Theorem for min-cut set
Python Simulation
Class Participation L10_S1

Definitions and Notation

Min-Cut Problem

We use the min-cut algorithm in various applications to solve problems related to network connectivity, reliability, and optimization. L10_S1

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. L10_S1

Communication Networks: For understanding the vulnerability of the networks to failures minimum cut can be useful. It helps building robust and fault-tolerant communication networks. L10_S1

VLSI Design: In Very Large Scale Integration (VLSI) design, the algorithm is useful for partitioning circuits into smaller components leading to reduced interconnectivity complexity. L10_S1

Refer to Section 1.5. Application: A Randomized Min-Cut Algorithm in Probability and Computing (2nd Edition). L10_S1

What is min-cut?

A cut-set in a graph is a set of edges whose removal breaks the graph into two or more connected components. L10_S1

Given a graph

$$G = (V, E)$$

with n vertices, the minimum cut – or min-cut – problem is to find a minimum cardinality cut-set in G. L10_S1

Min-cut algorithms, such as Karger's algorithm, are random and can be sensitive to the initial choice of edges. If the algorithm happens to contract critical edges early in the process, it might find a smaller cut. L10_S1

The main operation in the algorithm is edge contraction, which is operation that removes an edge from a graph while simultaneously merging the two vertices. L10_S1

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. The new graph may have parallel edges but no self-loops. L10_S1

(Figures on pages 15–16 illustrate edge contraction, showing merging of vertices and formation of parallel edges without self-loops.) L10_S1

Main Results

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. L10_S1

Figure: Successful Run (page 19 shows successive contractions resulting in the minimum cut). L10_S1

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. L10_S1

Figure: Unsuccessful Run (page 22 shows contractions leading to a non-minimum cut). L10_S1

Max-Flow Min-Cut Theorem

The max-flow min-cut theorem states that:

"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". L10_S1

Capacity of a cut = the sum of the capacity of the edges in the cut that are oriented from a vertex ∈ X to a vertex ∈ Y. L10_S1

Minimum cut = the cut in the network that has the smallest possible capacity. L10_S1

Minimum cut capacity = the capacity of the minimum cut. L10_S1

Maximum flow = the largest possible flow from source S to sink T. L10_S1

(Figure on page 26 shows a flow network with source S, sink T, intermediate vertices, and capacities illustrating the theorem.) L10_S1

Deterministic Min-Cut Algorithm

Stoer–Wagner Min-Cut Algorithm
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 $G/\{s,t\}$. L10_S1

The theorem holds since either:

• There is a minimum cut of G that separates s and t, then a minimum s-t-cut of G is a minimum cut of G ;

or

• There is none, then a minimum cut of $G/\{s,t\}$ does the job. L10_S1

(Page 28 contains a diagram illustrating Stoer–Wagner Min-Cut and a link reference.) L10_S1

Pseudocode

Algorithm 1: MinimumCutPhase(G, a)

A ← {a};
while A ≠ V do
  add to A the most tightly connected vertex;
return the cut weight as the "cut of the phase"; L10_S1

Algorithm 2: MinimumCut(G)

while |V| ≥ 1 do

```
  choose any a from V;
  MinimumCutPhase(G, a);
  if the cut-of-the-phase is lighter than the current minimum cut then
    store the cut-of-the-phase as the current minimum cut;
  shrink G by merging the two vertices added last;
return the minimum cut;
```
L10_S1

Randomized Min-Cut Algorithm

Why Randomized Algorithm?
Randomized algorithms provide a probabilistic guarantee of success. It provides a more accurate estimate of the minimum cut with fewer iterations. L10_S1

Efficiency
Parallelization
Approximation Guarantees
Avoidance of Worst-Case Instances
Heuristic Nature
Robustness L10_S1

Karger's Randomized Algorithm

(Page 34 contains an illustrated example of Karger's Randomized Algorithm.) L10_S1

The figure shows:

Pick edge 'b', remove it and fuse its two corners into one.
Pick edge 'd', remove it and fuse its two corners into one.

An example run of Karger's Randomized algorithm where random edges are picked in a way that the output cut is not minimal cut. The output cut produced here is {a, c, e}, but the minimal cut is either {b, e} or {a, d}. L10_S1

Pseudocode

Algorithm 3: Recursive-Randomized-Min-Cut(G, α)

Input is an undirected multigraph G with n vertices, and an integer constant $\alpha > 0$.
Output is a cut C of G.

```
if n ≤ α³ then
  C ← a min-cut of G found using brute force (exhaustive) search;
else
  for i ← 1 to α do
    G′ ← multigraph obtained by applying n − ⌈n/√α⌉ random contraction steps on G;
    C′ ← Recursive-Randomized-Min-Cut(G′, α);
    if i = 1 or |C′| < |C| then
      C ← C′;
return C;
```
L10_S1

Comparison: Deterministic Min-Cut vs Randomized Min-Cut Algorithm

Any specific problem is responsible for deciding the approach. L10_S1

Deterministic Min Cut:

It always guarantees an exact minimum cut.
It may have higher time complexity for large graphs.
The Stoer-Wagner algorithm, a deterministic min cut algorithm, has a time complexity of

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

where V is the number of vertices and E is the number of edges. L10_S1

Randomized Min Cut:

It provides an approximate minimum cut with high probability.
The Karger's algorithm, a randomized algorithm used for finding the minimum cut in a graph, has a time complexity of

$$O(V^2),$$

where V is the number of vertices in the graph. L10_S1

Theorem for Min-Cut Set

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

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

L10_S1

Python Simulation

- Open Campuswire Post regarding today's lecture (L10)
- Download the .ipynb file pasted in the comments L10_S1

Derivations / Proofs

The theorem for Stoer–Wagner Min-Cut holds since:

Either there is a minimum cut of G that separates s and t, then a minimum s-t-cut of G is a minimum cut of G;

or there is none, then a minimum cut of G/{s, t} does the job. L10_S1

Worked Examples

Karger's Randomized Algorithm Example (page 34):

Pick edge 'b', remove it and fuse its two corners into one.
Pick edge 'd', remove it and fuse its two corners into one.

Output cut produced: {a, c, e}.
Minimal cut: either {b, e} or {a, d}. L10_S1

Summary

Cut-set: set of edges whose removal disconnects graph.

Min-cut: minimum cardinality cut-set.

Edge contraction: merge two vertices, remove connecting edge, allow parallel edges, no self-loops.

Max-Flow Min-Cut Theorem: maximum flow equals minimum cut capacity.

Stoer–Wagner algorithm: deterministic, time complexity $O(V \cdot E + V^2 \log V)$.

Karger's algorithm: randomized, time complexity $O(V^2)$.

Sources y of outputting min-cut ≥ $\frac{2}{n(n-1)}$. L10_S1

[redacted]