Weighted Min-Cut: A Cross-Paradigm Algorithm

Sagnik Mukhopadhyay

University of Sheffield, UK
Weighted min-cut and our results
The (global) mincut problem

Remove edges to disconnect the graph (minimize total weight)

“This problem] plays an important role in the design of communication networks. If a few links are cut …”

LEDA (http://www.algorithmic-solutions.info/)
The (global) mincut problem

• Cut:
  • Set of edges whose removal disconnects $G$.

• Min-cut:
  • Cut with minimum total edge weight.

Goal: Find a min-cut in a weighted graph.
State of the arts: near-linear time

<table>
<thead>
<tr>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karger SODA’93</td>
</tr>
<tr>
<td>$O(n^2 \log^2 n)$</td>
</tr>
<tr>
<td>Karger, Stein STOC’93</td>
</tr>
<tr>
<td>$O(n^2 \log^2 n)$</td>
</tr>
<tr>
<td>Karger STOC’96</td>
</tr>
<tr>
<td>$O(m \log^3 n)$</td>
</tr>
</tbody>
</table>

Assume $m \geq n \log^6 n$ for simplicity

Problem:

Complicated dynamic programming & data structures

Hard to adapt to new computational models

No efficient algorithms in distributed, streaming, query complexity

n=#nodes, m=#edges
Models of computation

**Cut Query**: Allowed to query arbitrary cuts of $G$. Charged once per query.

**Dynamic semi-streaming**: $O(n \text{ polylog } n)$ bits of internal memory. Charged once per pass.

**Sequential**: Standard unit-cost RAM model.

**Parallel PRAM**: Concurrent read exclusive write. Complexity is (work, depth) of computation.

**Distributed CONGEST**: Bandwidth restricted ($O(\log n)$ bits per round). Charged once per round.
Previous work

- **Parallel PRAM**
  - [GG SPAA’18] $O(m \log^4 n)$ work, $O(\log^3 n)$ depth

- **Sequential model**
  - [Karger STOC’96] $O(m \log^3 n)$ Rand

- **Dynamic semi-streaming model**
  - [KT STOC’15] $O(m \log^{12} n)$ Det
  - [HRW SODA’17] $O(m \log^2 n)$ Det
  - [GNT, SODA’20] $O(m \log n)$ Rand

- **Cut-query model**
  - [RSW ITCS’19] $\tilde{O}(n)$

- **Simple graphs**
  - [ACK STOC’19, RSW ITCS’19, AD SOSA’21] 2 passes
  - [GNT SODA’20] $\tilde{O}(n^{0.9} + D^{0.2} n^{0.8})$

- **Distributed CONGEST**
  - [DHNS STOC’19] $\tilde{O}(n^{0.998} + D^{0.003} n^{0.997})$

- **Weighted min-cut**

- **Simple graphs!**

- **Weighted min-cut**

- **Simple graphs!**
Our results

Sequential model

Dynamic semi-streaming model

Distributed CONGEST

Weighted min-cut

Parallel PRAM

Cut-query model

[RSW ITCS’19] $\tilde{O}(n)$

[GG SPAA’18] $O(m \log^4 n)$ work, $O(\log^3 n)$ depth

[Karger STOC’96] $O(m \log^3 n)$

[KT STOC’15] $O(m \log^{12} n)$

[HRW SODA’17] $O(m \log^2 n)$

[GNT, SODA’20] $O(m \log n)$

[ACK STOC’19, RSW ITCS’19, AD SOSA’21] 2 passes

[DHNS STOC’19] $\tilde{O}(n^{0.998} + D^{0.003}n^{0.997})$

[GNT SODA’20] $\tilde{O}(n^{0.9} + D^{0.2}n^{0.8})$
Our results

- **Parallel PRAM**
  - Weighted: \( O(m \log n) \) work, \( O(\log^3 n) \) depth
  - Sequential model
  - Weighted: \( O(m \log n) \) work, \( O(\log^3 n) \) depth

- **Dynamic semi-streaming model**
  - Weighted: \( \tilde{O}(n) \)

- **Cut-query model**
  - Weighted: \( \tilde{O}(n) \)

- **Distributed CONGEST**
  - Weighted: \( \tilde{O}(\sqrt{n} + D) \)

- **Weighted min-cut**
  - \( O(m \log n) \) work, \( O(\log^3 n) \) depth

- **SPAA 2021**

- **STOC 2020**
  - Weighted: \( \tilde{O}(n) \)
  - Dynamic semi-streaming model

- **STOC 2020**
  - Weighted: \( \log n \) passes
  - [ACK STOC’19, RSW ITCS’19, AD SOSA’21] 2 passes

- **STOC 2021**
  - [DHNS STOC’19] \( \tilde{O}(n^{0.998} + D^{0.003}n^{0.997}) \)
  - [GNT, SODA’20] \( \tilde{O}(n^{0.9} + D^{0.2} \cdot n^{0.8}) \)

- **STOC 2021**
  - Weighted: \( \tilde{O}(\sqrt{n} + D) \)
Today: Cut-query model

A cut $(S_1, \bar{S}_1)$
Value of the cut $(S_1, \bar{S}_1)$

A cut $(S_2, \bar{S}_2)$
Value of the cut $(S_2, \bar{S}_2)$
Today: Cut-query model

A cut \((S_1, \overline{S_1})\)
Value of the cut \((S_1, \overline{S_1})\)

A cut \((S_2, \overline{S_2})\)
Value of the cut \((S_2, \overline{S_2})\)

Graph cuts \(\Leftrightarrow\) Submodular function

Cut-query \(\Leftrightarrow\) Submodular function minimization via query
Our result

Common barrier: How to overcome?

Remark:

• Assuming simple graph makes life a lot easier!

• None of the improvements on simple graphs follow Karger’s framework.

What is so hard about Karger’s framework?
2-respecting cut

- Main subroutine of Karger’s algorithm.

- Spanning tree:
  - A tree $T$ with edges from $E(G)$ that spans all vertices.

- 2-respecting cut:
  - Cut $C$ with at most 2 edges from $T$. 
2-respecting cut

- Spanning tree:
  - A tree $T$ with edges from $E(G)$ that spans all vertices.

- 2-respecting cut:
  - Cut $C$ with at most 2 edges from $T$. 

Not 2-respecting
2-respecting cut

- Spanning tree:
  - A tree $T$ with edges from $E(G)$ that spans all vertices.

- 2-respecting cut:
  - Cut $C$ with at most 2 edges from $T$.

**Goal:** Find a 2–respecting min-cut in a weighted graph given a spanning tree $T$.

**Theorem (Karger).** Efficient algorithm for solving 2-respecting min-cut implies efficient algorithm for solving min-cut.
Our result

New improved algorithm for min 2-resp cut problem.

One (schematic) algorithm that works across models
A closer look...

Simple and efficient algorithm for \textbf{min 2-resp cut for all models}

Edges of $T$

Edges of $T$

Cut($e_i, e_j$)
A closer look...

- $n^2$ many entries to look up.
- Goal: Minimize the number of look-ups to find a minimum value.
A quick detour: Matrix min-entry puzzle
Matrix min-entry problem

Puzzle time!

Input: I will write down an $n \times n$ matrix M (you don’t see M but know $n$).

Promise: **Monotonicity** - column minimums never move up as we go right

Example:

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Goal: Find the minimum entry in M.

Query: You can ask for one entry at a time. **How many entries do you need?**
Solution for matrix min-entry

$O(n \log n)$ queries by divide and conquer.
1. Find $\min$ in the middle column
   • by asking for everything in that column.
2. The recurse on both sides.
3. Search area decreases by half.

Puzzle time!

In fact, there is an $O(n)$-query solution [SMAWK]. But it’s not useful for us (too sequential)
Matrix min-entry to Global min-cut
2-respecting min-cut: Easy to find spanning tree $T$ that 2-contains the min-cut.

**Cost matrix:** Guess two tree-edges crossing the cuts Entries in $n \times n$ matrix

There are $O(n^2)$ candidate cuts

Cut-query model: Can find 1 entry by 1 query.
2-respecting min-cut: Easy to find spanning tree $T$ that 2-
contains the min-cut.

**Cost matrix:** Guess two tree-edges crossing the cuts
Entries in $n \times n$ matrix

There are $O(n^2)$ candidate cuts

Too expensive to compute all matrix entries

1 stream-pass can compute only $n$ entries.
Cute trick: Assume we know two paths in $T$ where the best candidates are in

Lemma: The corresponding cost matrix is monotone like in the puzzle!
Key idea: Spanning tree with 2 paths

Bipartite path problem

Cost(e_i, e_j), e_i ∈ L, e_j ∈ R
Cost matrix revisited

Question: Why is the cost matrix monotone?
Structure of cost matrix

Add +3

\[ e'_2 \]
\[ \vdots \]
\[ e'_4 \]
Structure of cost matrix

Add +3

Add +4
Structure of cost matrix

- Add
- Add +10

- Technical
Monotonicity of cost matrix

These operations result in a **monotone matrix**.

• Queries needed to solve bipartite path problem = $O(n \log n)$. 
Cute trick: Assume we know two paths in $T$ where the best candidates are in

Lemma: The corresponding cost matrix is \textit{monotone} like in the puzzle!

Decompose $T$ into paths using \textit{path decomposition} (e.g. heavy-light, bough/layering decomposition) & the simulate matrix min-entry algorithm for \textit{some} pairs of paths

Leftover details: Picking a few pairs

Heavy-light decomposition
Choosing a few pairs from decomposition
Picking up few pairs of path

**Lemma (Not quite true):** Each path can be paired with **only one path** such that the minimum 2-respect cut belongs to one such pair.

\[
\text{Cut}(e_3, e_4) = \deg(3) + \deg(4) - 2 \times \text{wt}(3,4)
\]

**Star graph**

\[
\text{Cut}(e_3, e_4) < \deg(3), \deg(4)
\]

\[
\text{wt}(3,4) > \frac{\deg(3)}{2}, \frac{\deg(4)}{2}
\]
Picking up few pairs of path

Lemma (Not quite true): Each path can be paired with **only one path** such that the minimum 2-respect cut belongs to one such pair.

Star graph

\[
\text{Cut}(e_3, e_4) < \deg(3), \deg(4) \quad \text{wt}(3,4) > \frac{\deg(3)}{2}, \frac{\deg(4)}{2}
\]

\( e_3 \) can only pair with \( e_4 \).
Picking up few pairs of path

Lemma (Not quite true): Each path can be paired with only one path such that the minimum 2-respect cut belongs to one such pair.

Lemma (Almost true): Each path can be paired with only one root-to-leaf path such that the minimum 2-respect cut belongs to one such pair.

log $n$ many paths from the tree decomposition
All results follow from one schematic algorithm (with different implementation details)

1. Decompose $T$ into paths using bough/layering decomposition.

2. Each edge $e$ in $A$ picks path $B$ that is interesting to it.

3. Simulate the matrix min-entry algorithm on the cost matrix of every such pair $(A, B)$ after contracting useless edges.

(The case where two candidates are in the same path is easy)
Cut-query simulation

Query simulation:

- Cut-query
  - Trivial
  - $O(n)$ cut queries in 1 pass
- Streaming
  - $O(n)$ cut queries in 1 pass
- Sequential
  - Range operation DS

More on range DS
Summary

• Spanning tree packing. Cut sparsifier with random neighbor sampling

• 2-respecting min-cut is the main bottle-neck of min-cut.

• Spanning tree: Path Can be solved quickly using properties of
  • Monotone matrix

• Spanning tree: Star-graph Can be solved quickly using edge pairing

• Putting it all together: General spanning tree Tree decomposition and almost correct lemma

Skipped this! Karger framework
Open problems
Open problems

• Graph problems in 2-party communication setting.

• Min-cut in dynamic setting.

• Directed min-cut and vertex connectivity.

• Other graph problems admitting cross-paradigm algorithm (?)
Thank you.
Simulating a cut-query
Range operation data-structure

• Takes $m$ points in a 2-d plane.

• Preprocessing: $O(m)$. # elements x depth

• Query: An axis-aligned rectangle $R$.
  • Count points in $R$: $O(n^c)$.
  • Sample point from $R$: $O(n^c)$.
    • Amortized.

[ Gawrychowski-Mozes-Weimann, 2020 ]
Range operation on spanning tree

- $\mathcal{C}(4^1, 10^1)$: 1 range `count` query.

[Gawrychowski-Mozes-Weimann, 2020]
Range operation on spanning tree

- $C(4^4, V - 4^4)$: 2 range count queries.

[Gawrychowski-Mozes-Weimann, 2020]
Range operation on spanning tree

Matrix min-entry requires $O(m) + \tilde{O}(n^{1+\varepsilon})$ time.

Minimum 2-resp cut requires $O(m) + \tilde{O}(n^{1+\varepsilon})$ time.

Minimum cut requires $O(m \log n) + \tilde{O}(n^{1+\varepsilon})$ time.

Pre processing

(A little more work)

Range operation

log $n$ many tree packing

[Gawrychowski-Mozes-Weimann, 2020]