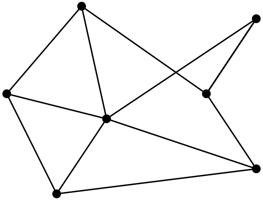
Minimum and Approximate Minimum k-Cuts in Hypergraphs

Chris Bao, Joshua Wang, William Zhao
Mentor: Yuchong Pan

October 12, 2024
MIT PRIMES October Conference

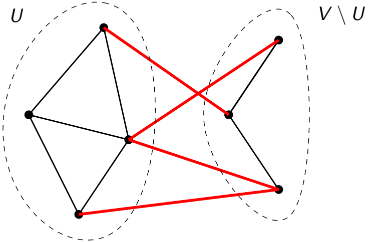
Cuts in Graphs

- Let $G = (V, E)$ be a graph. Throughout, let $|V| = n$ and $|E| = m$.



Cuts in Graphs

- Let $G = (V, E)$ be a graph. Throughout, let $|V| = n$ and $|E| = m$.

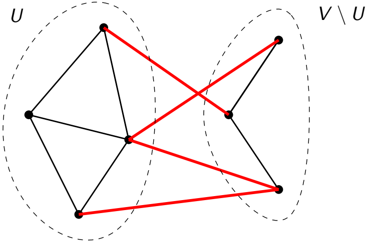


Definition

A cut is a partition of V into two non-empty subsets U and $V \setminus U$.

Cuts in Graphs

- Let $G = (V, E)$ be a graph. Throughout, let $|V| = n$ and $|E| = m$.



Definition

A cut is a partition of V into two non-empty subsets U and $V \setminus U$.

Definition

An edge e crosses a cut U if it has one endpoint in U and the other in $V \setminus U$. The value of a cut is the number of edges that cross it.

Minimum Cuts in Graphs

Definition

The minimum cut problem on a graph G asks us to find a cut of minimum value. Call this value λ.

Minimum Cuts in Graphs

Definition

The minimum cut problem on a graph G asks us to find a cut of minimum value. Call this value λ.

Definition

An α-approximate minimum cut is one that has value at most $\alpha\lambda$.

Minimum Cuts in Graphs

Definition

The minimum cut problem on a graph G asks us to find a cut of minimum value. Call this value λ.

Definition

An α-approximate minimum cut is one that has value at most $\alpha\lambda$.

- Example application: Network reliability analysis

Minimum Cuts in Graphs

Definition

The minimum cut problem on a graph G asks us to find a cut of minimum value. Call this value λ.

Definition

An α-approximate minimum cut is one that has value at most $\alpha\lambda$.

- Example application: Network reliability analysis
- Question: How can we efficiently find minimum and approximate minimum cuts?

Minimum Cuts in Graphs

Definition

The minimum cut problem on a graph G asks us to find a cut of minimum value. Call this value λ.

Definition

An α-approximate minimum cut is one that has value at most $\alpha\lambda$.

- Example application: Network reliability analysis
- Question: How can we efficiently find minimum and approximate minimum cuts?
- Question: How many minimum cuts can be in a graph with n vertices? How many α-approximate minimum cuts?

Karger's algorithm

- The random contraction technique, introduced by Karger (1993), can answer both of these questions.

Karger's algorithm

- The random contraction technique, introduced by Karger (1993), can answer both of these questions.

Definition

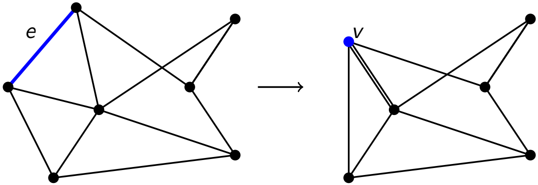
The contraction operation takes a graph G and an edge $e \in E$. It merges the endpoints of e (removing self-loops) to create the graph G/e.

Karger's algorithm

- The random contraction technique, introduced by Karger (1993), can answer both of these questions.

Definition

The contraction operation takes a graph G and an edge $e \in E$. It merges the endpoints of e (removing self-loops) to create the graph G/e.

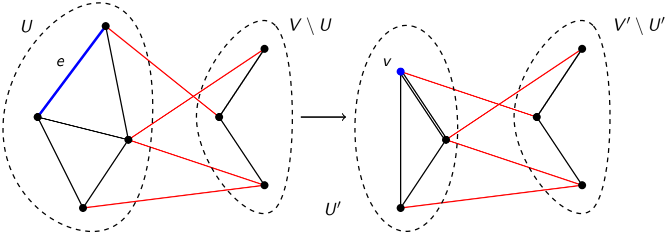


Karger's Algorithm

- For an edge e, a cut $C = (U, V \setminus U)$ will survive the contraction of e if e does not cross C

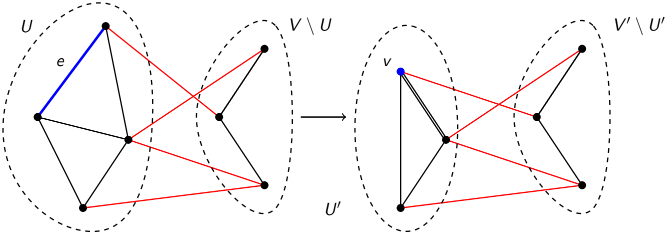
Karger's Algorithm

- For an edge e, a cut $C = (U, V \setminus U)$ will survive the contraction of e if e does not cross C



Karger's Algorithm

- For an edge e, a cut $C = (U, V \setminus U)$ will survive the contraction of e if e does not cross C



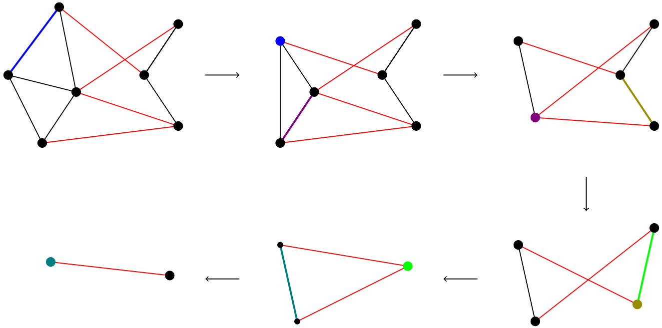
- The smaller a cut C, the more likely C survives the contraction of a randomly selected edge. In particular, a minimum cut is very likely to survive random contraction.

Karger's Algorithm

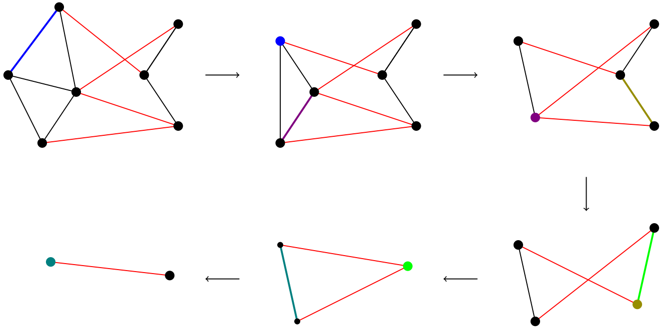
- Karger's Algorithm contracts randomly selected edges until only 2 vertices are left, and then returns the cut between those vertices

Karger's Algorithm

- Karger's Algorithm contracts randomly selected edges until only 2 vertices are left, and then returns the cut between those vertices



- Karger's Algorithm contracts randomly selected edges until only 2 vertices are left, and then returns the cut between those vertices



- The smaller a cut C, the more likely C survives.

Karger's Algorithm

- A given minimum cut C has at least a $\binom{n}{2}^{-1}$ chance of surviving all of the contractions and being returned (Karger 1993)

Karger's Algorithm

- A given minimum cut C has at least a $\binom{n}{2}^{-1}$ chance of surviving all of the contractions and being returned (Karger 1993)

Theorem (Karger 1993)

There are at most $\binom{n}{2} = O(n^2)$ minimum cuts in an n-vertex graph. Furthermore, there are $O(n^{2\alpha})$ α-approximate minimum cuts.

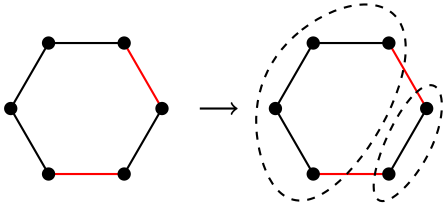
Karger's Algorithm

- A given minimum cut C has at least a $\binom{n}{2}^{-1}$ chance of surviving all of the contractions and being returned (Karger 1993)

Theorem (Karger 1993)

There are at most $\binom{n}{2} = O(n^2)$ minimum cuts in an n-vertex graph. Furthermore, there are $O(n^{2\alpha})$ α-approximate minimum cuts.

- The bound is tight: take a cycle graph on n vertices



Minimum and Approximate Minimum k-Cuts in Hypergraphs

The Minimum k-Cut Problem

- Our work concerns a twofold generalization of the minimum cut problem: the minimum k-cut problem for hypergraphs.

The Minimum k-Cut Problem

- Our work concerns a twofold generalization of the minimum cut problem: the minimum k-cut problem for hypergraphs.
- First, let's consider k-cuts in graphs.

Definition

A *k-way cut* (or *k-cut* for short) is a partition of the vertices V into k non-empty subsets $U_1, U_2, \ldots, U_k$. An edge crosses a k-cut if its vertices are in different subsets U_i and U_j, and the size of a k-cut is the number of edges crossing it.

The Minimum k-Cut Problem

- Our work concerns a twofold generalization of the minimum cut problem: the minimum k-cut problem for hypergraphs.
- First, let's consider k-cuts in graphs.

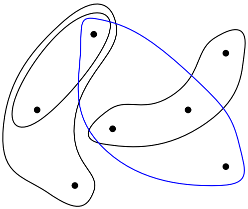
Definition

A k-way cut (or k-cut for short) is a partition of the vertices V into k non-empty subsets $U_1, U_2, \ldots, U_k$. An edge crosses a k-cut if its vertices are in different subsets U_i and U_j, and the size of a k-cut is the number of edges crossing it.

- Karger's algorithm readily generalizes to the k-cut problem - we stop contraction at k vertices remaining instead of two.

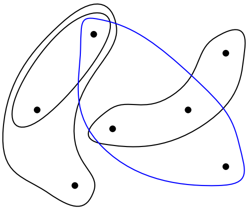
Hypergraphs

- In a graph, edges can connect only two vertices. In a hypergraph, we allow hyperedges to connect multiple vertices (i.e., each hyperedge $e \in E$ is a subset of V, so $E \subseteq 2^V$.)



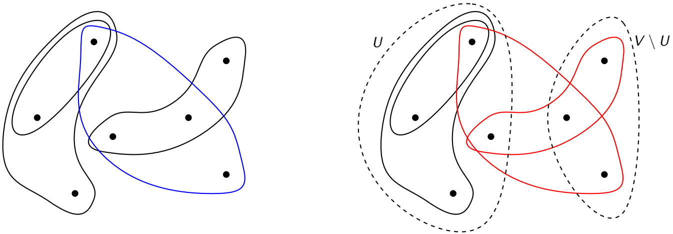
Hypergraphs

- In a graph, edges can connect only two vertices. In a hypergraph, we allow hyperedges to connect multiple vertices (i.e., each hyperedge $e \in E$ is a subset of V, so $E \subseteq 2^V$.)
- The rank of a hyperedge e is the number of vertices associated with it. The rank of a hypergraph is the maximum rank of a hyperedge.



Hypergraphs

- In a graph, edges can connect only two vertices. In a hypergraph, we allow hyperedges to connect multiple vertices (i.e., each hyperedge $e \in E$ is a subset of V, so $E \subseteq 2^V$.)
- The rank of a hyperedge e is the number of vertices associated with it. The rank of a hypergraph is the maximum rank of a hyperedge.



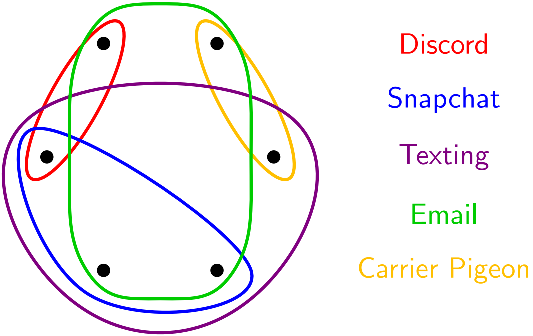
- We generalize the minimum cut and minimum k-cut problems to hypergraphs.

Hypergraphs

- Hypergraphs can model more general networks than graphs, in which groups of nodes influence each other.

Hypergraphs

- Hypergraphs can model more general networks than graphs, in which groups of nodes influence each other.
- Example: Communication platforms



Random Contraction for Hypergraph k-Cut

- We considered a generalization of Karger's algorithm to the minimum k-cut problem for low-rank hypergraphs.

Random Contraction for Hypergraph k-Cut

- We considered a generalization of Karger's algorithm to the minimum k-cut problem for low-rank hypergraphs.

Theorem (Kogan and Krauthgamer 2014)

In a rank-r hypergraph H with n vertices, there are at most

$$O\left(2^{\alpha r} n^{2\alpha}\right)$$

α*-approximate minimum cuts.*

Random Contraction for Hypergraph k-Cut

- We considered a generalization of Karger's algorithm to the minimum k-cut problem for low-rank hypergraphs.

Theorem (Kogan and Krauthgamer 2014)

In a rank-r hypergraph H with n vertices, there are at most

$$O\left(2^{\alpha r} n^{2\alpha}\right)$$

α-approximate minimum cuts.

Theorem (Bao, Pan, Wang, and Zhao 2024+)

In a rank-r hypergraph H with n vertices, there are at most

$$O\left(k^{\alpha(k-1)r} n^{2\alpha(k-1)}\right)$$

α-approximate minimum k-cuts.

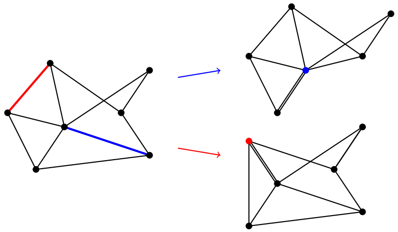
Minimum and Approximate Minimum k-Cuts in Hypergraphs

Karger-Stein Recursive Contraction

- Problem with Karger's Algorithm: Later contractions are likely to destroy a given minimum cut, wasting the earlier contractions.

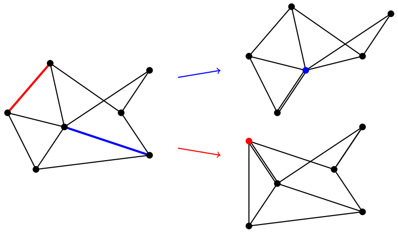
Karger-Stein Recursive Contraction

- Problem with Karger's Algorithm: Later contractions are likely to destroy a given minimum cut, wasting the earlier contractions.
- Karger and Stein (1995) introduced the recursive contraction algorithm, which recursively "branches" by periodically duplicating the graph and running two instances of random contraction



Karger-Stein Recursive Contraction

- Problem with Karger's Algorithm: Later contractions are likely to destroy a given minimum cut, wasting the earlier contractions.
- Karger and Stein (1995) introduced the recursive contraction algorithm, which recursively "branches" by periodically duplicating the graph and running two instances of random contraction



- Branches occur at predetermined times: every time $|V|$ decreases by a factor of $\sqrt{2}$.

Branching Contraction

- Fox et al. (2018) generalized recursive contraction to the hypergraph k-cut case by introducing the branching contraction algorithm.

Branching Contraction

- Fox et al. (2018) generalized recursive contraction to the hypergraph k-cut case by introducing the branching contraction algorithm.
- Here, there is a chance to randomly create a branch every time an edge is selected for contraction.

Branching Contraction

- Fox et al. (2018) generalized recursive contraction to the hypergraph k-cut case by introducing the branching contraction algorithm.
- Here, there is a chance to randomly create a branch every time an edge is selected for contraction.
- Because larger hyperedges are more likely to destroy a minimum k-cut, the probability of branching increases with the size of the contracted hyperedge.

Branching Contraction

- Fox et al. (2018) generalized recursive contraction to the hypergraph k-cut case by introducing the branching contraction algorithm.
- Here, there is a chance to randomly create a branch every time an edge is selected for contraction.
- Because larger hyperedges are more likely to destroy a minimum k-cut, the probability of branching increases with the size of the contracted hyperedge.

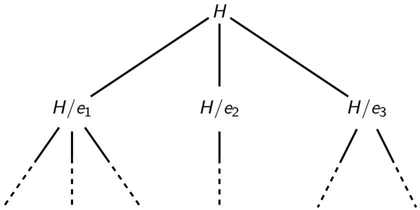
$$P_{\mathrm{Branch}}(e) = 1 - \frac{\binom{n-|e|}{k-1}}{\binom{n}{k-1}}$$

Contraction Tree

- The contracted hypergraphs can be organized into a tree. Although the tree itself is random, we can still analyze its expected properties.

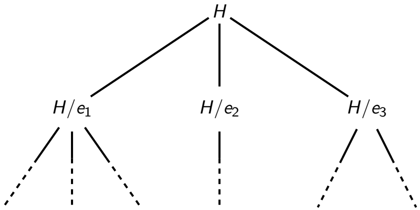
Contraction Tree

- The contracted hypergraphs can be organized into a tree. Although the tree itself is random, we can still analyze its expected properties.



Contraction Tree

- The contracted hypergraphs can be organized into a tree. Although the tree itself is random, we can still analyze its expected properties.



Theorem (Fox et al. 2018)

The branching contraction algorithm finds a minimum k-cut with high probability in $\tilde{O}(mn^{2k-2})$ expected time

Branching Contraction Time Complexity

- These bounds are tight for graphs. However, $m = O(n^2)$ in a graph, while $m = \Omega(2^n)$ is possible in a hypergraph. A finer examination of the time complexity is needed.

Branching Contraction Time Complexity

- These bounds are tight for graphs. However, $m = O(n^2)$ in a graph, while $m = \Omega(2^n)$ is possible in a hypergraph. A finer examination of the time complexity is needed.
- The time complexity cannot be improved for weighted hypergraphs: take a hypergraph where 2-edges have very large weights. We considered the case of an unweighted hypergraph

Branching Contraction Time Complexity

- These bounds are tight for graphs. However, $m = O(n^2)$ in a graph, while $m = \Omega(2^n)$ is possible in a hypergraph. A finer examination of the time complexity is needed.
- The time complexity cannot be improved for weighted hypergraphs: take a hypergraph where 2-edges have very large weights. We considered the case of an unweighted hypergraph
- Here, there are bounds relating the proportion of small edges to the total number of edges. These can be interpreted as constraints in a linear program.

Branching Contraction Time Complexity

- These bounds are tight for graphs. However, $m = O(n^2)$ in a graph, while $m = \Omega(2^n)$ is possible in a hypergraph. A finer examination of the time complexity is needed.
- The time complexity cannot be improved for weighted hypergraphs: take a hypergraph where 2-edges have very large weights. We considered the case of an unweighted hypergraph
- Here, there are bounds relating the proportion of small edges to the total number of edges. These can be interpreted as constraints in a linear program.

Theorem (Bao, Pan, Wang, and Zhao 2024+)

For an unweighted hypergraph without parallel edges, the branching contraction algorithms works in $\tilde{O}(mn^k + n^{2k})$ expected time.

Minimum and Approximate Minimum k-Cuts in Hypergraphs

Tree Packing

- A forest F is a set of edges that contains no cycles. It is a disjoint union of trees.

Tree Packing

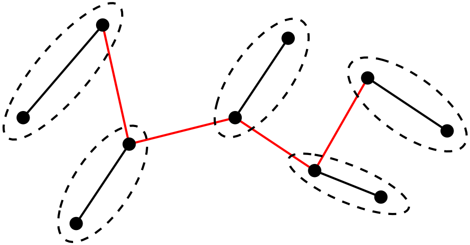
- A forest F is a set of edges that contains no cycles. It is a disjoint union of trees.
- Any k-cut intersects every forest F at least $|F| + k - n$ times.

Tree Packing

- A forest F is a set of edges that contains no cycles. It is a disjoint union of trees.
- Any k-cut intersects every forest F at least $|F| + k - n$ times.
- The converse also holds!

Tree Packing

- A forest F is a set of edges that contains no cycles. It is a disjoint union of trees.
- Any k-cut intersects every forest F at least $|F| + k - n$ times.
- The converse also holds!



$$k = 5$$

$$|F| = 9$$

$$n = 10$$

$$|F| + k - n = 4$$

Fractional k-Cut

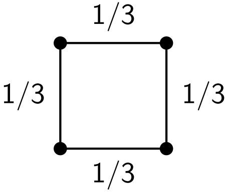
- A k-cut can be viewed as assigning each edge a value of 1 (cut) or 0 (uncut). Instead, let's assign real values in $[0, 1]$ representing how much of an edge is cut.

Fractional k-Cut

- A k-cut can be viewed as assigning each edge a value of 1 (cut) or 0 (uncut). Instead, let's assign real values in $[0, 1]$ representing how much of an edge is cut.
- We will call such a cut a fractional k-cut if its weighted intersection with any forest F is at least $|F| + k - n$.

Fractional k-Cut

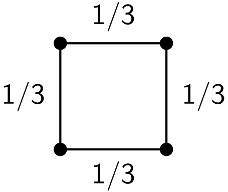
- A k-cut can be viewed as assigning each edge a value of 1 (cut) or 0 (uncut). Instead, let's assign real values in $[0, 1]$ representing how much of an edge is cut.
- We will call such a cut a fractional k-cut if its weighted intersection with any forest F is at least $|F| + k - n$.



A fractional 2-cut smaller than the minimum 2-cut

Fractional k-Cut

- A k-cut can be viewed as assigning each edge a value of 1 (cut) or 0 (uncut). Instead, let's assign real values in $[0, 1]$ representing how much of an edge is cut.
- We will call such a cut a fractional k-cut if its weighted intersection with any forest F is at least $|F| + k - n$.



A fractional 2-cut smaller than the minimum 2-cut

In graphs, the minimum k-cut has value at most $2(1 - 1/n)$ times the minimum fractional k-cut. Furthermore, the minimum fractional k-cut is easier to compute.

Approximating Minimum k-cut Using Tree Packing

- Quanrud (2019) begins by computing a $(1+\varepsilon)$-approximate minimum fractional k-cut, using linear programming and a technique called Multiplicative Weight Update.

Approximating Minimum k-cut Using Tree Packing

- Quanrud (2019) begins by computing a $(1+\varepsilon)$-approximate minimum fractional k-cut, using linear programming and a technique called Multiplicative Weight Update.
- The fractional k-cut can be "rounded" to a $2(1+\varepsilon)$-approximate minimum k-cut.

Approximating Minimum k-cut Using Tree Packing

- Quanrud (2019) begins by computing a $(1+\varepsilon)$-approximate minimum fractional k-cut, using linear programming and a technique called Multiplicative Weight Update.
- The fractional k-cut can be "rounded" to a $2(1+\varepsilon)$-approximate minimum k-cut.
- Take every edge with weight at least $1/2$. If this does not form a k-cut, then greedily complete with cuts from the minimum spanning tree (or forest).

Hypertrees

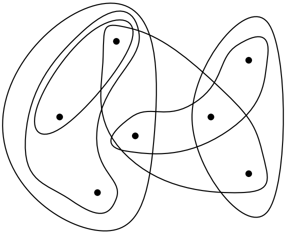
Definition

A hyperforest is a set of hyperedges F such that any subset $X \subseteq F$ contains at least $|X| + 1$ vertices.

Hypertrees

Definition

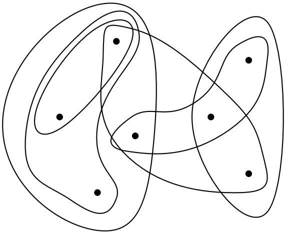
A hyperforest is a set of hyperedges F such that any subset $X \subseteq F$ contains at least $|X| + 1$ vertices.



Hypertrees

Definition

A hyperforest is a set of hyperedges F such that any subset $X \subseteq F$ contains at least $|X| + 1$ vertices.



Theorem (Bao, Pan, Wang, and Zhao 2024+)

A minimum hypertree can be computed in $O(mn^2)$ time.

Hypertree Packing

- With the concept of a hyperforest, we can generalize the tree packing method to hypergraphs.

Hypertree Packing

- With the concept of a hyperforest, we can generalize the tree packing method to hypergraphs.
- We use the MWU method and greedy rounding to compute a $r(1+\varepsilon)$ approximation to the minimum k-cut problem in hypergraphs.

Hypertree Packing

- With the concept of a hyperforest, we can generalize the tree packing method to hypergraphs.
- We use the MWU method and greedy rounding to compute a $r(1+\varepsilon)$ approximation to the minimum k-cut problem in hypergraphs.

Theorem (Bao, Pan, Wang, and Zhao 2024+)

There exists a $r(1+\varepsilon)$-approximation to minimum k-cut in

$$O(mn\log^2 n/\varepsilon^2 + mn^2)$$

time for hypergraphs, where r is the rank of the hypergraph.

Acknowledgements

- We thank our mentor, Yuchong Pan, for his guidance and support.
- We thank PRIMES-USA and the organizers for this opportunity.

References

- D. Karger and C. Stein. *A new approach to the minimum cut problem.* J. ACM **43** (1996).
- D. Kogan and R. Krauthgamer. *Sketching cuts in graphs and hypergraphs.* ITCS '15 **6** (2015).
- M. Baïou and F. Barahona. *On some algorithmic aspects of hypergraphic matroids.* Discrete Math. **346** (2023)
- K. Fox, D. Panigrahi, and F. Zhang. *Minimum cut and minimum k-cut in hypergraphs via branching contractions.* SIAM (2019)
- K. Quanrud. *Fast and Deterministic Approximations for k-Cut.* LIPIcs **145** (2019).