A Nearly Optimal All-Pairs Min-Cuts Algorithm in Simple Graphs

Jason Li (Simons-Berkeley)

Joint work with
Debmalya Panigrahi (Duke)
Thatchaphol Saranurak (UMichigan)

FOCS 2021

Introduction

- All-pairs mincut problem: given a graph, find (the value of) the minimum cut separating s and t for all pairs of vertices s, t

Introduction

- All-pairs mincut problem: given a graph, find (the value of) the minimum cut separating s and t for all pairs of vertices s, t
  - Output size $\binom{n}{2}$, so quadratic time is optimal

Introduction

- All-pairs mincut problem: given a graph, find (the value of) the minimum cut separating s and t for all pairs of vertices s, t
  - Output size $\binom{n}{2}$, so quadratic time is optimal
- Gomory-Hu tree: a weighted tree that encodes all pairwise mincuts

- All-pairs mincut problem: given a graph, find (the value of) the minimum cut separating s and t for all pairs of vertices s, t
  - Output size $\binom{n}{2}$, so quadratic time is optimal
- Gomory-Hu tree: a weighted tree that encodes all pairwise mincuts
  - Output size $O(n)$, so linear time is optimal

Introduction

- All-pairs mincut problem: given a graph, find (the value of) the minimum cut separating s and t for all pairs of vertices s, t
  - Output size $\binom{n}{2}$, so quadratic time is optimal
- Gomory-Hu tree: a weighted tree that encodes all pairwise mincuts
  - Output size $O(n)$, so linear time is optimal
  - Given a Gomory-Hu tree, can output all pairs mincuts in optimal time

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n - 1$ calls to s–t mincut

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n - 1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n - 1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n-1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi '08]

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n-1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi '08]
- Recent line of work: *simple* graphs (unweighted, no parallel edges)

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n - 1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi ’08]
- Recent line of work: *simple* graphs (unweighted, no parallel edges)
  - First subcubic algorithm [Abboud, Krauthgamer, Trabelsi ’21]

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n-1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi '08]
- Recent line of work: *simple* graphs (unweighted, no parallel edges)
  - First subcubic algorithm [Abboud, Krauthgamer, Trabelsi '21]
- This work (concurrently [AKT'21]): there is a quadratic time Gomory-Hu tree algorithm for simple graphs

Prior Work and Our Result

- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n-1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi '08]
- Recent line of work: *simple* graphs (unweighted, no parallel edges)
  - First subcubic algorithm [Abboud, Krauthgamer, Trabelsi '21]
- This work (concurrently [AKT'21]): there is a quadratic time Gomory-Hu tree algorithm for simple graphs
  - Implies near-optimal all-pairs mincut (but not Gomory-Hu tree)

Prior Work and Our Result

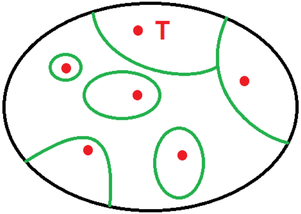
- Gomory and Hu, 1961: Gomory-Hu tree can be solved in $n-1$ calls to s–t mincut
  - All-pairs mincut in linear number of s–t mincut calls
- Best bound known for general graphs, even 60 years later!
- Unweighted graphs $\tilde{O}(mn)$ time [Bhalgat, Hariharan, Kavitha, Panigrahi '08]
- Recent line of work: *simple* graphs (unweighted, no parallel edges)
  - First subcubic algorithm [Abboud, Krauthgamer, Trabelsi '21]
- This work (concurrently [AKT'21]): there is a quadratic time Gomory-Hu tree algorithm for simple graphs
  - Implies near-optimal all-pairs mincut (but not Gomory-Hu tree)
- This talk: focus on all-pairs mincuts only (no GH tree)

Speedup: Main Idea

- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts

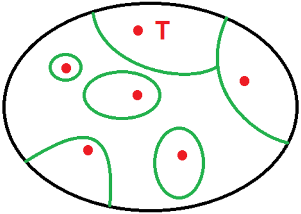
Speedup: Main Idea

- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts
- Isolating Cuts procedure [**L**P'20, AKT'21]: given terminals T, can compute the t–$(T \setminus t)$ mincut for all $t \in T$ in $O(\log |T|)$ many s–t mincut calls



Speedup: Main Idea

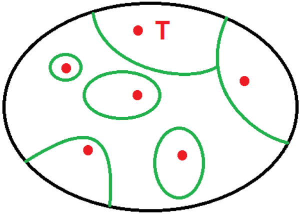
- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts
- Isolating Cuts procedure [**L**P'20, AKT'21]: given terminals T, can compute the t–$(T \setminus t)$ mincut for all $t \in T$ in $O(\log |T|)$ many s–t mincut calls



- When does isolating cuts capture the true s–t mincuts?

Speedup: Main Idea

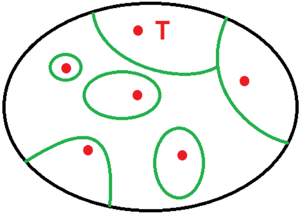
- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts
- Isolating Cuts procedure [**L**P'20, AKT'21]: given terminals T, can compute the t–$(T \setminus t)$ mincut for all $t \in T$ in $O(\log |T|)$ many s–t mincut calls



- When does isolating cuts capture the true s–t mincuts?
  - when the s–t mincut is also a s–$(T \setminus s)$ cut

Speedup: Main Idea

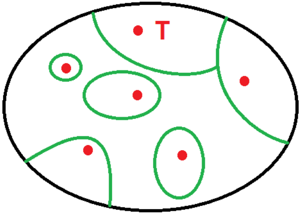
- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts
- Isolating Cuts procedure [**L**P'20, AKT'21]: given terminals T, can compute the t–$(T \setminus t)$ mincut for all $t \in T$ in $O(\log |T|)$ many s–t mincut calls



- When does isolating cuts capture the true s–t mincuts?
  - when the s–t mincut is also a s–$(T \setminus s)$ cut
  - when the only terminal on s's side is s itself

Speedup: Main Idea

- Main Idea: spend a *single* s–t mincut time to compute *many* s–t mincuts
- Isolating Cuts procedure [**L**P'20, AKT'21]: given terminals T, can compute the t–$(T \setminus t)$ mincut for all $t \in T$ in $O(\log |T|)$ many s–t mincut calls



- When does isolating cuts capture the true s–t mincuts?
  - when the s–t mincut is also a s–$(T \setminus s)$ cut
  - when the only terminal on s's side is s itself
  - in other words, s–t mincut should be "unbalanced"

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side
- Recover s–t mincut for all s, t satisfying the above by repeating $O(k^2 \log n)$ times $\implies$ total running time $\tilde{O}(k^2)$ many s–t mincut calls.

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side
- Recover s–t mincut for all s, t satisfying the above by repeating $O(k^2 \log n)$ times $\implies$ total running time $\tilde{O}(k^2)$ many s–t mincut calls.
- Formally, we use notion of a *well-linked* set

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side
- Recover s–t mincut for all s, t satisfying the above by repeating $O(k^2 \log n)$ times $\implies$ total running time $\tilde{O}(k^2)$ many s–t mincut calls.
- Formally, we use notion of a *well-linked* set
  - If a set of terminals is ϕ-well-linked, then for any terminals s, t, the s–t mincut has at most $1/\phi$ terminals on either s's or t's side

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side
- Recover s–t mincut for all s, t satisfying the above by repeating $O(k^2 \log n)$ times $\implies$ total running time $\tilde{O}(k^2)$ many s–t mincut calls.
- Formally, we use notion of a *well-linked* set
  - If a set of terminals is ϕ-well-linked, then for any terminals s, t, the s–t mincut has at most $1/\phi$ terminals on either s's or t's side
  - Think: expander, except with terminals

Enforcing Unbalanced

- Suppose we have the following promise: the s–t mincut has at most k terminals on s's side
- Idea: *randomly sample* terminals at rate $1/k$; with probability $\Omega(1/k^2)$, sample s and t and no other terminals on s's side
- Recover s–t mincut for all s, t satisfying the above by repeating $O(k^2 \log n)$ times $\implies$ total running time $\tilde{O}(k^2)$ many s–t mincut calls.
- Formally, we use notion of a *well-linked* set
  - If a set of terminals is ϕ-well-linked, then for any terminals s, t, the s–t mincut has at most $1/\phi$ terminals on either s's or t's side
  - Think: expander, except with terminals
- Main technical tool: well-linked decomposition (only for simple graphs!)

Well-linked Decomposition

- Given a parameter d and terminal set T of vertices of degree $\geq d$, we can partition T into $\tilde{O}(n/d)$ groups, each of which is ϕ-well-linked for some $\phi = 1/n^{o(1)}$. Algorithm takes $m^{1+o(1)}$ time.

Well-linked Decomposition

- Given a parameter d and terminal set T of vertices of degree $\geq d$, we can partition T into $\tilde{O}(n/d)$ groups, each of which is ϕ-well-linked for some $\phi = 1/n^{o(1)}$. Algorithm takes $m^{1+o(1)}$ time.
- For each partition, compute all-pairs mincut given the unbalanced guarantee.

Well-linked Decomposition

- Given a parameter d and terminal set T of vertices of degree $\geq d$, we can partition T into $\tilde{O}(n/d)$ groups, each of which is ϕ-well-linked for some $\phi = 1/n^{o(1)}$. Algorithm takes $m^{1+o(1)}$ time.
- For each partition, compute all-pairs mincut given the unbalanced guarantee.
  - We reduce the number of edges to $O(nd)$ by Nagamochi-Ibaraki sparsification; we only need to look at s–t mincuts of size at most, say, $2d$

Well-linked Decomposition

- Given a parameter d and terminal set T of vertices of degree $\geq d$, we can partition T into $\tilde{O}(n/d)$ groups, each of which is ϕ-well-linked for some $\phi = 1/n^{o(1)}$. Algorithm takes $m^{1+o(1)}$ time.
- For each partition, compute all-pairs mincut given the unbalanced guarantee.
    - We reduce the number of edges to $O(nd)$ by Nagamochi-Ibaraki sparsification; we only need to look at s–t mincuts of size at most, say, $2d$
    - *Assuming linear-time s–t mincut*, total running time is $\tilde{O}(n/d) \cdot \widehat{O}(nd) = \widehat{O}(n^2)$.

Well-linked Decomposition

- Given a parameter d and terminal set T of vertices of degree $\geq d$, we can partition T into $\tilde{O}(n/d)$ groups, each of which is ϕ-well-linked for some $\phi = 1/n^{o(1)}$. Algorithm takes $m^{1+o(1)}$ time.
- For each partition, compute all-pairs mincut given the unbalanced guarantee.
  - We reduce the number of edges to $O(nd)$ by Nagamochi-Ibaraki sparsification; we only need to look at s–t mincuts of size at most, say, $2d$
  - *Assuming linear-time s–t mincut*, total running time is $\tilde{O}(n/d) \cdot \widehat{O}(nd) = \widehat{O}(n^2)$.
- For terminals in different partitions, need to look further into Gomory-Hu tree structure (not in this talk)

Conclusion

- Near-optimal all-pairs mincut algorithm for simple graphs (not optimal for Gomory-Hu tree)

Conclusion

- Near-optimal all-pairs mincut algorithm for simple graphs (not optimal for Gomory-Hu tree)
- New technical idea: well-linked decomposition

Conclusion

- Near-optimal all-pairs mincut algorithm for simple graphs (not optimal for Gomory-Hu tree)
- New technical idea: well-linked decomposition
- Open: faster Gomory-Hu tree algorithms?

Conclusion

- Near-optimal all-pairs mincut algorithm for simple graphs (not optimal for Gomory-Hu tree)
- New technical idea: well-linked decomposition
- Open: faster Gomory-Hu tree algorithms?
    - Some exciting progress in submission!