Fair Cuts:
Motivation, Definition, and Applications

Jason Li (Simons Institute)
Joint with Danupon Nanongkai (MPI),
Debmalya Panigrahi (Duke), and
Thatchaphol Saranurak (UMich)

SODA 2023
January 22, 2023

Minimum Cuts and Uncrossing

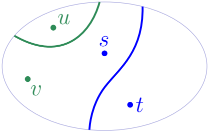
- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t

Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:

Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"



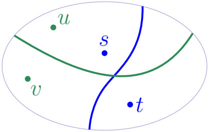
Used in divide-and-conquer algorithms for Gomory-Hu tree (all-pairs mincut): "dividing" on s–t mincut does not destroy u–v mincut

Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not “cross”
  - Can be proved by submodularity of cuts. This talk: flow-based proof.

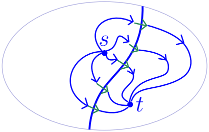
Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:



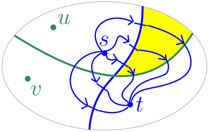
Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:
  - Consider a maximum s–t flow, which saturates the cut edges by max-flow-min-cut theorem



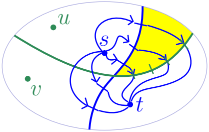
Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:
  - Consider a maximum s–t flow, which saturates the cut edges by max-flow-min-cut theorem



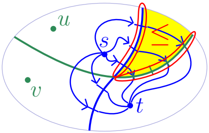
Minimum Cuts and Uncrossing

- *s*–*t* mincut of an (undirected) graph: the smallest set of edges whose removal disconnects *s* and *t*
- Uncrossing property of *s*–*t* mincuts:
  - for any *s*–*t* mincut and vertices u, v, there exists a *u*–*v* mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose *s*–*t* mincut and *u*–*v* mincut cross:
  - Consider a maximum *s*–*t* flow, which saturates the cut edges by max-flow-min-cut theorem



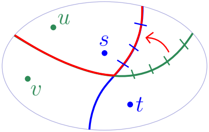
Minimum Cuts and Uncrossing

- *s*–*t* mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of *s*–*t* mincuts:
  - for any *s*–*t* mincut and vertices u, v, there exists a *u*–*v* mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose *s*–*t* mincut and *u*–*v* mincut cross:
  - Consider a maximum *s*–*t* flow, which saturates the cut edges by max-flow-min-cut theorem



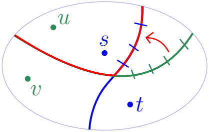
Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:
  - Consider a maximum s–t flow, which saturates the cut edges by max-flow-min-cut theorem



Minimum Cuts and Uncrossing

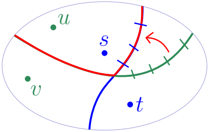
- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:
  - Consider a maximum s–t flow, which saturates the cut edges by max-flow-min-cut theorem



  - Obtained a u–v cut whose size can only be smaller $\implies$ also u–v mincut

Minimum Cuts and Uncrossing

- s–t mincut of an (undirected) graph: the smallest set of edges whose removal disconnects s and t
- Uncrossing property of s–t mincuts:
  - for any s–t mincut and vertices u, v, there exists a u–v mincut that does not "cross"
  - Can be proved by submodularity of cuts. This talk: flow-based proof.
  - Suppose s–t mincut and u–v mincut cross:
  - Consider a maximum s–t flow, which saturates the cut edges by max-flow-min-cut theorem



  - Obtained a u–v cut whose size can only be smaller $\implies$ also u–v mincut
  - We have uncrossed the u–v mincut with the s–t mincut.

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
    - However, algorithm is very complicated
    - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time
  - conceptually much simpler

Approximate Uncrossing?

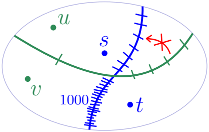
- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time
  - conceptually much simpler
  - can be implemented in parallel

Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time
  - conceptually much simpler
  - can be implemented in parallel
- Can approximate mincuts be uncrossed?

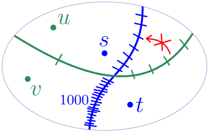
Approximate Uncrossing?

- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time
  - conceptually much simpler
  - can be implemented in parallel
- Can approximate mincuts be uncrossed?
  - Not necessarily...



Approximate Uncrossing?

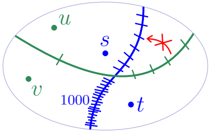
- [Chen-Kyng-Liu-Peng-Gutenberg-Sachdeva'21] s–t mincut can be solved in $m^{1+o(1)}$ time—almost optimal!
  - However, algorithm is very complicated
  - Inherently sequential
- [Sherman'13,Peng'16] $(1+\epsilon)$-approximate s–t mincut in $\tilde{O}(m)$ time
  - conceptually much simpler
  - can be implemented in parallel
- Can approximate mincuts be uncrossed?
  - Not necessarily...



  - Main obstacle to obtaining approximate Gomory-Hu tree (all-pairs mincut) from approximate s–t mincut

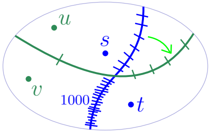
Fair Cuts: Motivation

- Let's look at this example again:



Fair Cuts: Motivation

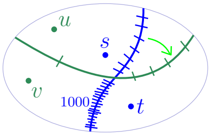
- Let's look at this example again:



- Can locally improve the s–t mincut, i.e., the s–t mincut was locally bad

Fair Cuts: Motivation

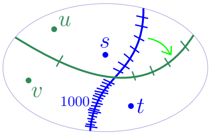
- Let's look at this example again:



- Can locally improve the *s*–*t* mincut, i.e., the *s*–*t* mincut was locally bad
- Fair cuts: *s*–*t* cuts that are nowhere locally bad, i.e., uniformly good

Fair Cuts: Motivation

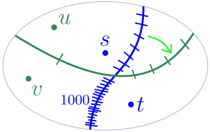
- Let's look at this example again:



- Can locally improve the s–t mincut, i.e., the s–t mincut was locally bad
- Fair cuts: s–t cuts that are nowhere locally bad, i.e., uniformly good
- Ideal theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross

Fair Cuts: Motivation

- Let's look at this example again:



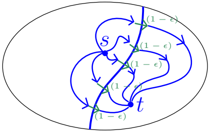
- Can locally improve the s–t mincut, i.e., the s–t mincut was locally bad
- Fair cuts: s–t cuts that are nowhere locally bad, i.e., uniformly good
- Ideal theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross
- To formally define $(1+\epsilon)$-fair, we switch to flow-based perspective again

Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated

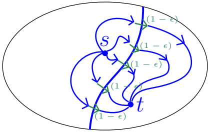
Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity



Fair Cuts: Definition

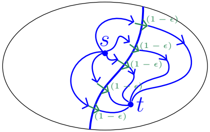
- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity



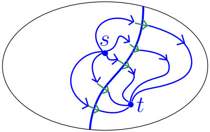
- Theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross

Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity

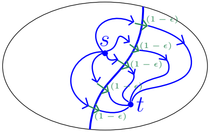


- Theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross
- Proof: consider the nearly saturating flow...

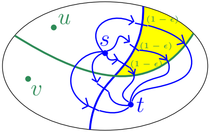


Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity

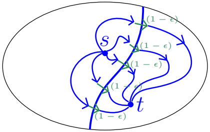


- Theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross
- Proof: consider the nearly saturating flow...

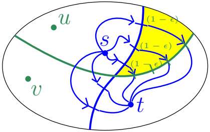


Fair Cuts: Definition

- Definition: an *s*–*t* cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
    - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity

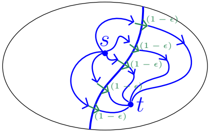


- Theorem: for any $(1+\epsilon)$-fair *s*–*t* cut and vertices u, v, there exists a $(1+\epsilon)$-approximate *u*–*v* cut that does not cross
- Proof: consider the nearly saturating flow...

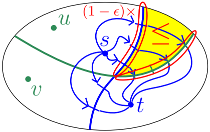


Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity

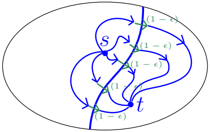


- Theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross
- Proof: consider the nearly saturating flow...

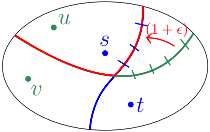


Fair Cuts: Definition

- Definition: an s–t cut is $(1+\epsilon)$-fair if there exists an $s \to t$ flow such that every edge of the cut is nearly saturated
  - The flow along each cut edge in the $s \to t$ direction is at least $\frac{1}{1+\epsilon}$ times edge capacity



- Theorem: for any $(1+\epsilon)$-fair s–t cut and vertices u, v, there exists a $(1+\epsilon)$-approximate u–v cut that does not cross
- Proof: consider the nearly saturating flow...



Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
    - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
    - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
    - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
  - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
  - What about approximate minimum isolating cuts in approximate max-flow time?

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
  - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
  - What about approximate minimum isolating cuts in approximate max-flow time?
    - Naïve approach fails because can't uncross

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
    - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
    - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
    - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
    - What about approximate minimum isolating cuts in approximate max-flow time?
        - Naïve approach fails because can't uncross
        - Fair cuts: approximate minimum isolating cuts in $\tilde{O}(m)$ time

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
  - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
  - What about approximate minimum isolating cuts in approximate max-flow time?
    - Naïve approach fails because can't uncross
    - Fair cuts: approximate minimum isolating cuts in $\tilde{O}(m)$ time
    - Following known reductions: approximate Steiner mincut, approximate Gomory-Hu tree (all-pairs mincut) in $\tilde{O}(m)$ time, also parallel

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
  - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
  - What about approximate minimum isolating cuts in approximate max-flow time?
    - Naïve approach fails because can't uncross
    - Fair cuts: approximate minimum isolating cuts in $\tilde{O}(m)$ time
    - Following known reductions: approximate Steiner mincut, approximate Gomory-Hu tree (all-pairs mincut) in $\tilde{O}(m)$ time, also parallel
- Expander pruning in expander decomposition

Fair Cuts: Applications

- Theorem: can compute $(1+\epsilon)$-approximate fair cuts in $\tilde{O}(m/\epsilon^3)$ time.
  - Follows $(1+\epsilon)$-approximate mincut algorithm [Sherman'13]
- Minimum Isolating Cuts: a useful primitive for graph cut algorithms
  - Given terminals $T \subset V$, the minimum isolating cut at terminal $t \in T$ is the minimum cut separating t from all other terminals.
  - Can compute all minimum isolating cuts (one for each terminal) in about max-flow time
  - What about approximate minimum isolating cuts in approximate max-flow time?
    - Naïve approach fails because can't uncross
    - Fair cuts: approximate minimum isolating cuts in $\tilde{O}(m)$ time
    - Following known reductions: approximate Steiner mincut, approximate Gomory-Hu tree (all-pairs mincut) in $\tilde{O}(m)$ time, also parallel
- Expander pruning in expander decomposition
  - First $\tilde{O}(m)$ time ϕ-expander decomposition algorithm for all values of ϕ