Faster Cut Sparsification of Weighted Graphs

Joint work with Sebastian Forster

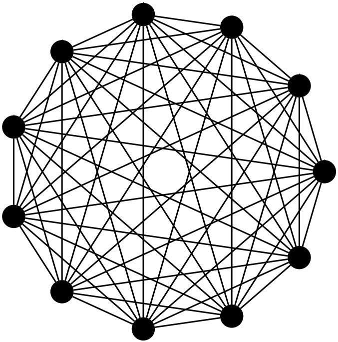
Tijn de Vos

Department of Computer Science
University of Salzburg

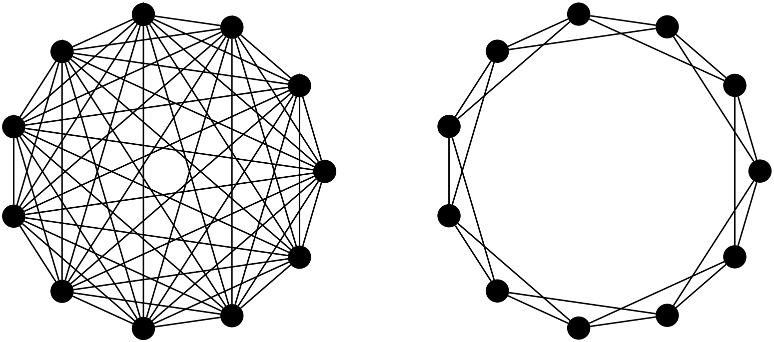
PARIS
LODRON
UNIVERSITÄT
SALZBURG

December 1, 2021

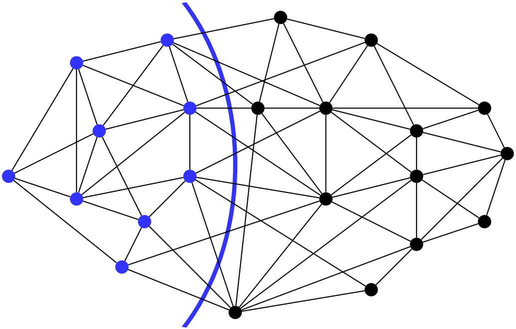
Sparsification



Sparsification

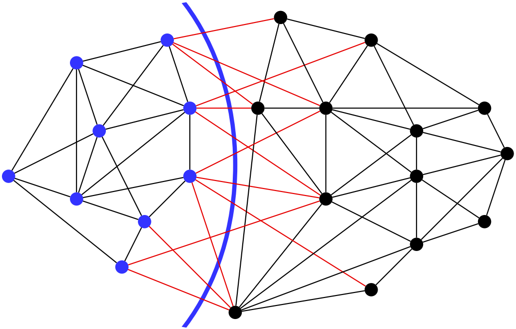


Cuts and Cut Sparsification[1]



[1]Image based on slides by Sebastian Forster

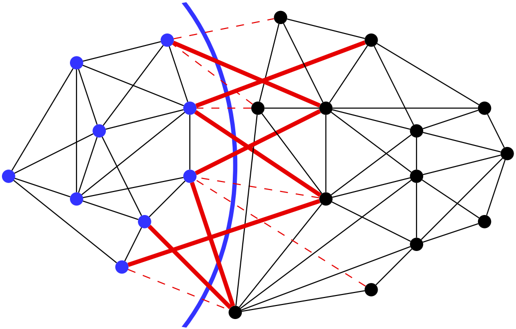
Cuts and Cut Sparsification[1]



Weight of cut: $w_G(C)$

[1] Image based on slides by Sebastian Forster

Cuts and Cut Sparsification[1]



Weight of cut: $w_G(C)$
Weight of sparsified cut $w_H(C)$

[1]Image based on slides by Sebastian Forster

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

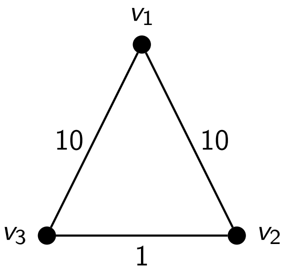
$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$

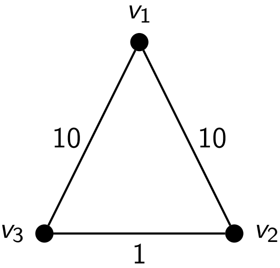


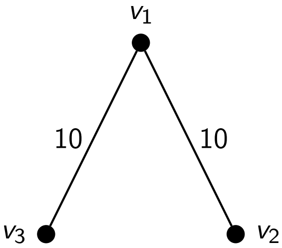
Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$





Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$

- Goal: $|H| = O(n \log n/\epsilon^2)$

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1 - \epsilon)w_G(C) \leq w_H(C) \leq (1 + \epsilon)w_G(C).$$

- Goal: $|H| = O(n \log n/\epsilon^2)$
- Lower bound: $O(n/\epsilon^2)$ [ACK$^+$16]

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$

- Goal: $|H| = O(n \log n/\epsilon^2)$
- Lower bound: $O(n/\epsilon^2)$ [ACK$^+$16]
- Algorithm concerning cuts: $T(m, n) \to T(O(n \log n/\epsilon^2), n)$

Cuts and Cut Sparsification

Definition

A (reweighted) subgraph $H \subseteq G$ is a $(1 \pm \epsilon)$-cut sparsifier for a weighted graph G if for every cut C

$$(1-\epsilon)w_G(C) \leq w_H(C) \leq (1+\epsilon)w_G(C).$$

- Goal: $|H| = O(n \log n/\epsilon^2)$
- Lower bound: $O(n/\epsilon^2)$ [ACK$^+$16]
- Algorithm concerning cuts: $T(m,n) \to T(O(n \log n/\epsilon^2), n)$
- Want size, time, and above property with high probability: $1 - n^{-c}$

Sparsification by Random Sampling

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e / p_e$.

Sparsification by Random Sampling

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e / p_e$.
- Size: $\sum_e p_e$

Sparsification by Random Sampling

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$
- E.g. $p_e = 1/m \implies$ size $\sum_e 1/m = 1$

Sparsification by Random Sampling

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$
- E.g. $p_e = 1/m \implies$ size $\sum_e 1/m = 1$
- Cut sparsifier in expectation

Sparsification by Random Sampling

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$
- E.g. $p_e = 1/m \implies$ size $\sum_e 1/m = 1$
- Cut sparsifier in expectation
- Want with high probability: $1 - n^{-c}$

Connectivity

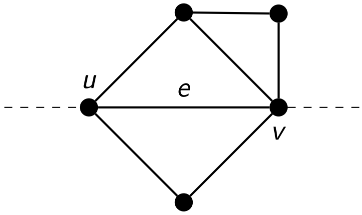
Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v

Connectivity

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v

Connectivity

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v

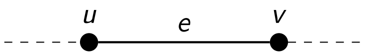


Connectivity $= 1$

Idea: sample $e = (u, v)$ relative to its *connectivity* c_e: the minimum cut separating u and v

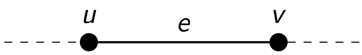
Connectivity $= 1$

Idea: sample $e = (u, v)$ relative to its *connectivity* c_e: the minimum cut separating u and v



Connectivity $= 1$

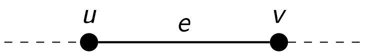
Idea: sample $e = (u, v)$ relative to its *connectivity* c_e: the minimum cut separating u and v



Sample with probability p_e

Connectivity $= 1$

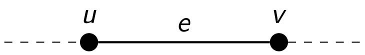
Idea: sample $e = (u, v)$ relative to its *connectivity* c_e: the minimum cut separating u and v



Sample with probability p_e
Success with probability $\leq p_e$

Connectivity $= 1$

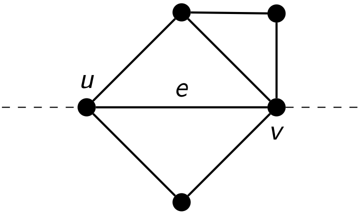
Idea: sample $e = (u, v)$ relative to its *connectivity* c_e: the minimum cut separating u and v



Sample with probability p_e
Success with probability $\leq p_e$
$p_e \geq 1 - n^{-c}$

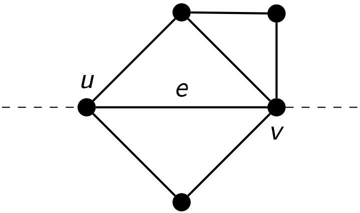
Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v

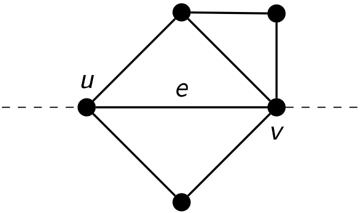


Sample with probability

$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



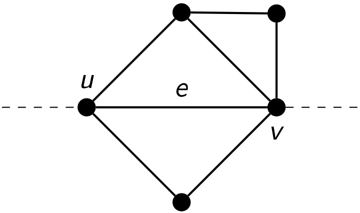
Sample with probability

$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

At least c_e edges e' crossing min cut C with $c_{e'} \leq c_e$

Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



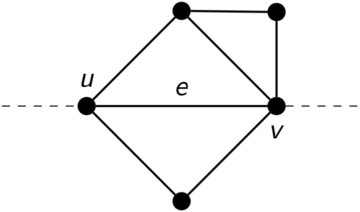
Sample with probability

$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

At least c_e edges e' crossing min cut C with $c_{e'} \leq c_e$, hence $p_{e'} \geq p_e$

Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



Sample with probability

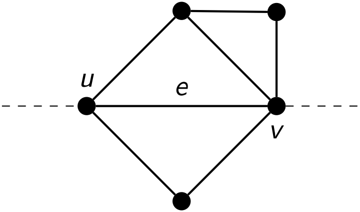
$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

At least c_e edges e' crossing min cut C with $c_{e'} \leq c_e$, hence $p_{e'} \geq p_e$
There is an edge crossing C in H with probability at least

$$1 - \prod_{e' \in C} (1 - p_{e'})$$

Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



Sample with probability

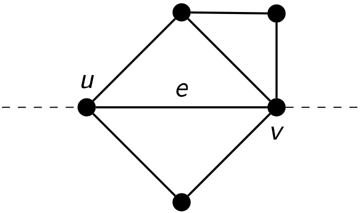
$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

At least c_e edges e' crossing min cut C with $c_{e'} \leq c_e$, hence $p_{e'} \geq p_e$
There is an edge crossing C in H with probability at least

$$1 - \prod_{e' \in C} (1 - p_{e'}) \geq 1 - \left(1 - \frac{c \cdot \log n}{c_e}\right)^{c_e}$$

Connectivity $\gg 1$

Idea: sample $e = (u, v)$ relative to *connectivity* c_e of u and v: the minimum cut separating u and v



Sample with probability

$$p_e = \min\left\{1, \frac{c \cdot \log n}{c_e}\right\}$$

At least c_e edges e' crossing min cut C with $c_{e'} \leq c_e$, hence $p_{e'} \geq p_e$

There is an edge crossing C in H with probability at least

$$1 - \prod_{e' \in C} (1 - p_{e'}) \geq 1 - \left(1 - \frac{c \cdot \log n}{c_e}\right)^{c_e} \geq 1 - e^{-c \cdot \log n} = 1 - n^{-c}$$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\text{Binom}(w_e, p_e)$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\text{Binom}(w_e, p_e)$
- If $r_e > 0$, include e with: $w_e \leftarrow r_e/p_e$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\mathrm{Binom}(w_e, p_e)$
- If $r_e > 0$, include e with: $w_e \leftarrow r_e/p_e$
- Size $\sum_e w_e p_e$:

$$\mathbb{P}[r_e > 0] =$$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\mathrm{Binom}(w_e, p_e)$
- If $r_e > 0$, include e with: $w_e \leftarrow r_e/p_e$
- Size $\sum_e w_e p_e$:

$$\mathbb{P}[r_e > 0] = \sum_{k \geq 1} \mathbb{P}[r_e = k]$$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\mathrm{Binom}(w_e, p_e)$
- If $r_e > 0$, include e with: $w_e \leftarrow r_e/p_e$
- Size $\sum_e w_e p_e$:

$$\mathbb{P}[r_e > 0] = \sum_{k \geq 1} \mathbb{P}[r_e = k] \leq \sum_{k \geq 1} k\mathbb{P}[r_e = k]$$

Sparsification by Random Sampling

Unweighted:

- Include edge e with probability p_e, if sampled: $w_e \leftarrow w_e/p_e$.
- Size: $\sum_e p_e$

Weighted:

- Sample r_e from $\text{Binom}(w_e, p_e)$
- If $r_e > 0$, include e with: $w_e \leftarrow r_e/p_e$
- Size $\sum_e w_e p_e$:

$$\mathbb{P}[r_e > 0] = \sum_{k\geq 1} \mathbb{P}[r_e = k] \leq \sum_{k\geq 1} k\mathbb{P}[r_e = k] = \mathbb{E}[r_e] = w_e p_e$$

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

[FHHP11] Edge Connectivity

For graphs with polynomially bounded integer weights.

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

[FHHP11]	Edge Connectivity
[Kar99]	Minimum Cut

For graphs with polynomially bounded integer weights.

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

[FHHP11]	Edge Connectivity
[Kar99]	Minimum Cut
[BK96]	Strong Connectivity
[SS11]	Conductance
[FHHP11]	Nagamochi-Ibaraki Indices
new	Maximum Spanning Forest Indices

For graphs with polynomially bounded integer weights.

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

[FHHP11]	Edge Connectivity
[Kar99]	Minimum Cut
[BK96]	Strong Connectivity
[SS11]	Conductance
[FHHP11]	NI Indices
new	MSF Indices

For graphs with polynomially bounded integer weights.

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

		Size	Time
[FHHP11]	Edge Connectivity		
[Kar99]	Minimum Cut		
[BK96]	Strong Connectivity	$O(n \log n/\epsilon^2)$	$O(m \log^2 n)$
[SS11]	Conductance		
[FHHP11]	NI Indices	$O(n \log^2 n/\epsilon^2)$	$O(m)$
new	MSF Indices		

For graphs with polynomially bounded integer weights.

Connectivity Estimators λ_e

Sample with $p_e \sim \frac{c\gamma \cdot \log n}{\lambda_e \epsilon^2}$, for some $\lambda_e \leq c_e$

		Size	Time
[FHHP11]	Edge Connectivity		
[Kar99]	Minimum Cut		
[BK96]	Strong Connectivity	$O(n \log n/\epsilon^2)$	$O(m \log^2 n)$
[SS11]	Conductance		
[FHHP11]	NI Indices	$O(n \log^2 n/\epsilon^2)$	$O(m)$
new	MSF Indices	$O(n \log n/\epsilon^2)$	$O(m\alpha(n) \log(m/n))$

For graphs with polynomially bounded integer weights.

Maximum Spanning Forest Indices

Definition

$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.

Maximum Spanning Forest Indices

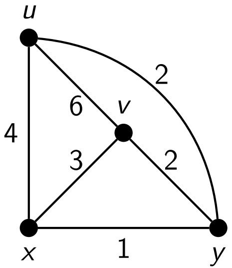
Definition

$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.
MSF index of e, denoted f_e, is the unique index such that $e \in F_{f_e}$.

Maximum Spanning Forest Indices

Definition

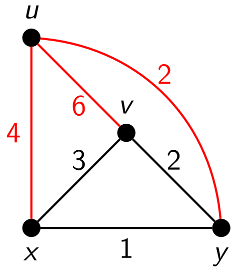
$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.
MSF index of e, denoted f_e, is the unique index such that $e \in F_{f_e}$.



Maximum Spanning Forest Indices

Definition

$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.
MSF index of e, denoted f_e, is the unique index such that $e \in F_{f_e}$.

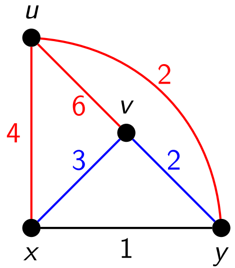


$f_e = 1$

Maximum Spanning Forest Indices

Definition

$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.
MSF index of e, denoted f_e, is the unique index such that $e \in F_{f_e}$.

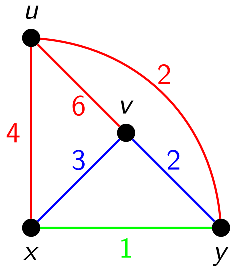


$f_e = 1$
$f_e = 2$

Maximum Spanning Forest Indices

Definition

$\mathcal{F} = \{F_1, \ldots, F_M\}$ is an M-partial maximum spanning forest packing of G if for all $i = 1, \ldots, M$, F_i is a maximum spanning forest in $G \setminus \bigcup_{j=1}^{i-1} F_j$.
MSF index of e, denoted f_e, is the unique index such that $e \in F_{f_e}$.



$f_e = 1$
$f_e = 2$
$f_e = 3$

MSF Indices and Connectivity

Claim

The connectivity of e is at least $f_e \cdot w_e$

MSF Indices and Connectivity

Claim

The connectivity of e is at least $f_e \cdot w_e$

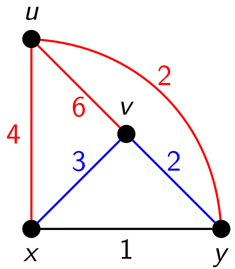
Proof. Denote $e = (u, v)$. For $i = 1, \ldots, f_e$, there is a path in F_i from u to v with each edge of weight at least w_e. $\square$

MSF Indices and Connectivity

Claim

The connectivity of e is at least $f_e \cdot w_e$

Proof. Denote $e = (u, v)$. For $i = 1, \ldots, f_e$, there is a path in F_i from u to v with each edge of weight at least w_e. $\square$



$f_e = 1$
$f_e = 2$
$f_e = 3$

Computing M-Partial MSF Indices

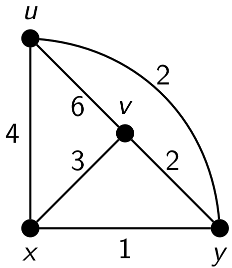
- Peeling off M forests is too slow: takes $O(Mm)$ time

Computing M-Partial MSF Indices

- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest

Computing M-Partial MSF Indices

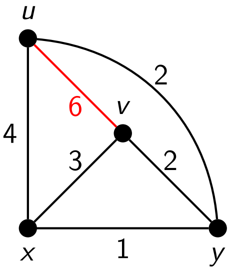
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

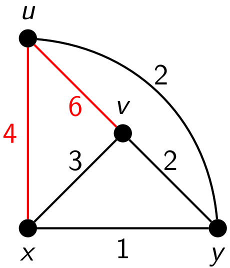
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

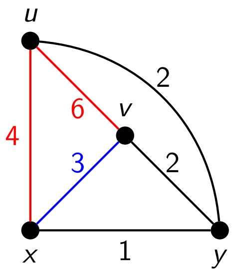
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

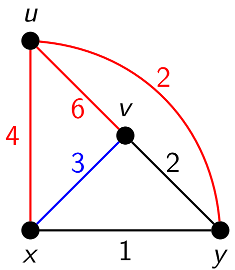
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

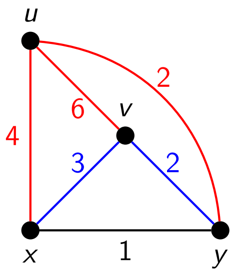
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

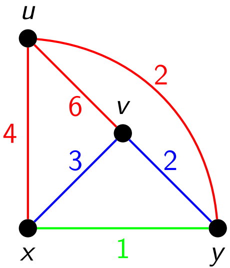
- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest



1. Sorted edges:
   uv, ux, xv, uy, vy, xy

Computing M-Partial MSF Indices

- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest
- Takes time $O(m\alpha(n)\log M)$

Computing M-Partial MSF Indices

- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest
- Takes time $O(m\alpha(n) \log M)$
  1. Radix sort in $O(m)$ time

Computing M-Partial MSF Indices

- Peeling off M forests is too slow: takes $O(Mm)$ time
- Instead:
  1. Sort edges according to weight
  2. Put each edge in first available forest
- Takes time $O(m\alpha(n)\log M)$
  1. Radix sort in $O(m)$ time
  2.
     - Pay $\log M$ for binary search for first available forest
     - Pay $\alpha(n)$ for maintaining Union-Find data structures

Naive Sparsification[2]

1. Compute MSF indices upto $M = n$ in time $O(m\alpha(n)\log n)$.
2. Sample with $p_e \sim \frac{c\gamma \cdot \log n}{f_e \cdot w_e \epsilon^2}$.

[2] Companion report

Naive Sparsification[2]

1. Compute MSF indices upto $M = n$ in time $O(m\alpha(n)\log n)$.
2. Sample with $p_e \sim \frac{c\gamma \cdot \log n}{f_e \cdot w_e \epsilon^2}$.

Results in size

$$\sum_e w_e p_e = \frac{c \cdot \log n}{\epsilon^2} \sum_e 1/f_e$$

[2] Companion report

Naive Sparsification[2]

1. Compute MSF indices upto $M = n$ in time $O(m\alpha(n)\log n)$.
2. Sample with $p_e \sim \frac{c\gamma \cdot \log n}{f_e \cdot w_e \epsilon^2}$.

Results in size

$$\sum_e w_e p_e = \frac{c \cdot \log n}{\epsilon^2} \sum_e 1/f_e \leq \frac{cn \log n}{\epsilon^2} \sum_{i=1}^{m/n} 1/i$$

[2] Companion report

Naive Sparsification[2]

1. Compute MSF indices upto $M = n$ in time $O(m\alpha(n)\log n)$.
2. Sample with $p_e \sim \frac{c\gamma \cdot \log n}{f_e \cdot w_e \epsilon^2}$.

Results in size

$$\sum_e w_e p_e = \frac{c \cdot \log n}{\epsilon^2} \sum_e 1/f_e \leq \frac{cn \log n}{\epsilon^2} \sum_{i=1}^{m/n} 1/i \leq \frac{cn \cdot \log n}{\epsilon^2} \log(m/n)$$

[2] Companion report

Naive Sparsification[2]

1. Compute MSF indices upto $M = n$ in time $O(m\alpha(n)\log n)$.
2. Sample with $p_e \sim \frac{c\gamma \cdot \log n}{f_e \cdot w_e \epsilon^2}$.

Results in size

$$\sum_e w_e p_e = \frac{c \cdot \log n}{\epsilon^2} \sum_e 1/f_e \leq \frac{cn \log n}{\epsilon^2} \sum_{i=1}^{m/n} 1/i \leq \frac{cn \cdot \log n}{\epsilon^2} \log(m/n)$$

Goal is time $O(m\alpha(n)\log(m/n))$ and size $O(n \log n/\epsilon^2)$.

[2] Companion report

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$
    4. Compute k_i-partial MSF packing, add those edges to F_i

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$
    4. Compute k_i-partial MSF packing, add those edges to F_i
4. Sample edges $e \in F_j$ with $p_e \sim 1/(2^j w_e)$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$
    4. Compute k_i-partial MSF packing, add those edges to F_i
4. Sample edges $e \in F_j$ with $p_e \sim 1/(2^j w_e)$

Time: $O(m\alpha(n)\log(m/n))$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$
    4. Compute k_i-partial MSF packing, add those edges to F_i
4. Sample edges $e \in F_j$ with $p_e \sim 1/(2^j w_e)$

Time: $O(m\alpha(n)\log(m/n))$
Size: $O(n\log n/\epsilon^2 \log(m/(n\log(n)/\epsilon^2)))$

[3] Based on [FHHP11] for unweighted graphs

Fancy Sparsification[3]

1. $\rho \leftarrow \Theta(\log n/\epsilon^2)$
2. Compute ρ-partial MSF packing, add those edges to F_0
3. For $i = 0$ to i_{end}:
    1. Sample remaining edges with probability $1/2$
    2. If sampled $w_e \leftarrow 2w_e$
    3. $k_i \leftarrow \rho \cdot 2^{i+1}$
    4. Compute k_i-partial MSF packing, add those edges to F_i
4. Sample edges $e \in F_j$ with $p_e \sim 1/(2^j w_e)$

Time: $O(m\alpha(n)\log(m/n))$
Size: $O(n \log n/\epsilon^2 \log(m/(n \log(n)/\epsilon^2))) \to O(n \log n/\epsilon^2)$

[3] Based on [FHHP11] for unweighted graphs

Conclusion

Theorem

$G = (V, E)$ polynomial weighted, $M > 0$. There exists an algorithm that computes an M-partial MSF packing in $O(m\alpha(n) \log M)$ time.

Conclusion

Theorem

$G = (V, E)$ polynomial weighted, $M > 0$. There exists an algorithm that computes an M-partial MSF packing in $O(m\alpha(n) \log M)$ time.

Theorem

$G = (V, E)$ weighted, $\epsilon > 0$. There exists an algorithm that computes a $(1 \pm \epsilon)$-cut sparsifier for G with high probability, in time $O(m\alpha(n) \log(m/n))$ and with size is $O\left(n \log n/\epsilon^2\right)$.

References

[ACK+16] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319, 2016.

[BK96] András A Benczúr and David R Karger. Approximating st minimum cuts in $\tilde{O}(n^2)$ time. In *Proc. of the Symposium on Theory of Computing (STOC)*, pages 47–55, 1996.

[FHHP11] Wai Shing Fung, Ramesh Hariharan, Nicholas J A Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proc. of the Symposium on Theory of Computing (STOC)*, pages 71–80, New York, NY, USA, 2011.

[Kar99] David R Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999.

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.