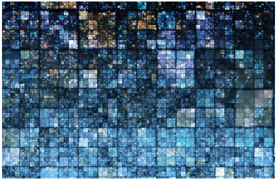
Streaming Algorithms for the Set Cover Problem

Adi Rosén

CNRS

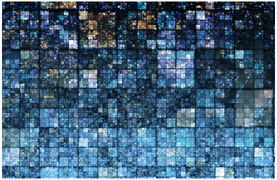
Collège de France, June 2018

Motivation - Big Data



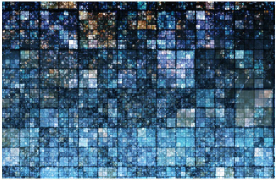
- Huge data sets:
  meteorology, genomics, social networks,...

Motivation - Big Data



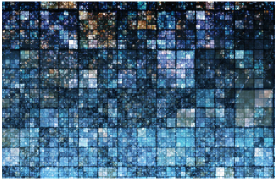
- Huge data sets:
  meteorology, genomics, social networks,...
  - IBM, 2012: 2.5 exabytes (2.5×10^{18}) of data created every day

Motivation - Big Data



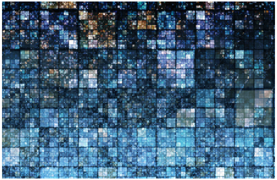
- Huge data sets:
  meteorology, genomics, social networks,...
  - IBM, 2012: 2.5 exabytes (2.5×10^{18}) of data created every day
- World's per-capita capacity to store information doubles every 40 months since the 1987 [Hilbert, López 2011]

Motivation - Big Data



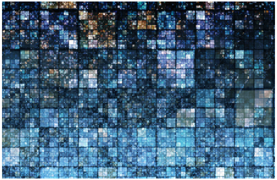
- Huge data sets:
  meteorology, genomics, social networks,...
  - IBM, 2012: 2.5 exabytes (2.5×10^{18}) of data created every day
- World's per-capita capacity to store information doubles every 40 months since the 1987 [Hilbert, López 2011]
- Algorithmic challenges: storage, communication, analysis

Motivation - Big Data



- Huge data sets:
  meteorology, genomics, social networks,...
  - IBM, 2012: 2.5 exabytes (2.5×10^{18}) of data created every day
- World's per-capita capacity to store information doubles every 40 months since the 1987 [Hilbert, López 2011]
- Algorithmic challenges: storage, communication, analysis
- The distributed approach: many servers, massively parallel algorithms
  - Storing the whole available data

Motivation - Big Data



- Huge data sets:
  meteorology, genomics, social networks,...
  - IBM, 2012: 2.5 exabytes (2.5×10^{18}) of data created every day
- World's per-capita capacity to store information doubles every 40 months since the 1987 [Hilbert, López 2011]
- Algorithmic challenges: storage, communication, analysis
- The distributed approach: many servers, massively parallel algorithms
  - Storing the whole available data
- Today: algorithms that store a small fraction of the available data

1. The streaming model
2. The set-cover problem
3. (some of the) Results
4. (some of the) Techniques
   - Single pass, semi-streaming algorithm (unweighted case)
   - Matching lower bound(s)
5. Conclusions and open problems

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream
- Algorithm required to return, at the end of the stream, "good" output

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream
- Algorithm required to return, at the end of the stream, "good" output
- Questions:
  - Interplay between the quality of the output and the memory size.

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream
- Algorithm required to return, at the end of the stream, "good" output
- Questions:
  - Interplay between the quality of the output and the memory size.
  - What is the run-time complexity per input item? In total (amortized) ?

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream
- Algorithm required to return, at the end of the stream, "good" output
- Questions:
  - Interplay between the quality of the output and the memory size.
  - What is the run-time complexity per input item? In total (amortized) ?
  - Interplay between the quality of the output and the number of passes.

Streaming algorithms

- Input presented piece-by-piece as a sequence (aka stream) of items
  - adversarial order
- Algorithms with memory size $\ll$ input size
  - can store only a small fraction of the input
  - memory size typically independent of length of stream
- Algorithm required to return, at the end of the stream, "good" output
- Questions:
  - Interplay between the quality of the output and the memory size.
  - What is the run-time complexity per input item? In total (amortized) ?
  - Interplay between the quality of the output and the number of passes.
- Problems:
  - Selection of k^{th} largest element [Munro, Paterson 1980]
  - Estimating frequency moments [Alon, Matias, Szegedy 1996]
  - Finding heavy hitters [Karp, Papadimitriou, Shenker 2003]
  - Counting distinct elements [Kane, Nelson, Woodruff 2010]
  - Checking balanced parentheses [Magniez, Mathieu, Nayak 2010]

Graph Problems

Graph Problems

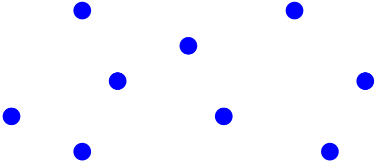
- Input: $G = (V, E)$, $n = |V|$, $m = |E|$

Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random

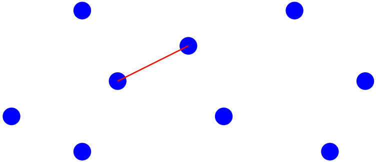
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



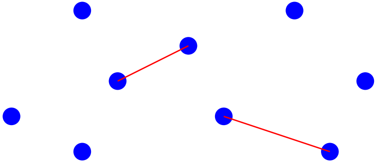
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



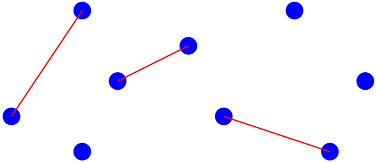
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



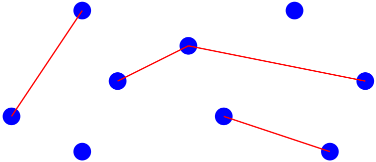
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



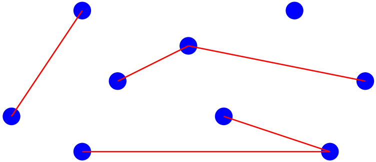
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



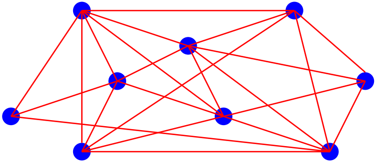
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random



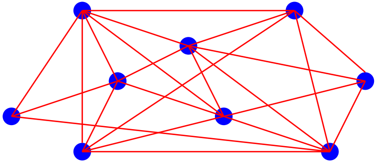
Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random

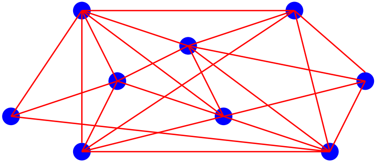


Graph Problems

- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random
- output size often depends on input size

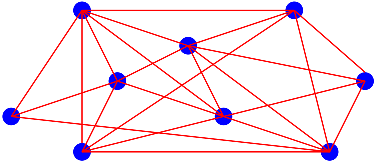


Graph Problems



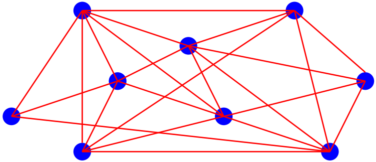
- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random
- output size often depends on input size
- Memory size
  - $o(|G|)$ bits (sublinear)
  - $n \log^{O(1)} m$ bits (a.k.a. semi-streaming)

Graph Problems



- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random
- output size often depends on input size
- Memory size
  - $o(|G|)$ bits (sublinear)
  - $n \log^{O(1)} m$ bits (a.k.a. semi-streaming)
- Problems (semi-streaming setting):

Graph Problems



- Input: $G = (V, E)$, $n = |V|$, $m = |E|$
- Input order:
  - edges vs. nodes
  - adversarial vs. random
- output size often depends on input size
- Memory size
  - $o(|G|)$ bits (sublinear)
  - $n \log^{O(1)} m$ bits (a.k.a. semi-streaming)
- Problems (semi-streaming setting):
  - Distances and diameter
    [Feigenbaum, Kannan, McGregor, Suri, Zhang 2005]
  - Constructing spanners and shortest path trees
    [Feigenbaum, Kannan, McGregor, Suri, Zhang 2008]
  - Maximum matching
    [McGregor 2005; Epstein,Levin,Mestre,Segev 2010; Crouch,Stubbs 2014]
  - Constructing spectral sparsifiers [Ahn, Guha 2009; Kelner, Levin 2013]
  - Maximum Independent Set
    [Halldórsson, Halldórsson, Losievskaja, Szegedy 2010]

The minimum set cover problem

- Input:
  - Universe $\mathcal{U}$ of *items*; $|\mathcal{U}| = n$
  - Collection $\mathcal{S}$ of subsets $S \subseteq \mathcal{I}$; $|\mathcal{S}| = m$.

The minimum set cover problem

- Input:
  - Universe $\mathcal{U}$ of *items*; $|\mathcal{U}| = n$
  - Collection $\mathcal{S}$ of subsets $S \subseteq \mathcal{I}$; $|\mathcal{S}| = m$.
- Output:
  - Subcollection $\mathcal{C} \subseteq \mathcal{S}$ that *covers* $\mathcal{U}$: $\cup_{c \in \mathcal{C}} C = \mathcal{U}$
  - minimize $|\mathcal{C}|$

The minimum set cover problem

- Input:
  - Universe $\mathcal{U}$ of *items*; $|\mathcal{U}| = n$
  - Collection $\mathcal{S}$ of subsets $S \subseteq \mathcal{I}$; $|\mathcal{S}| = m$.
- Output:
  - Subcollection $\mathcal{C} \subseteq \mathcal{S}$ that *covers* $\mathcal{U}$: $\cup_{c \in \mathcal{C}} C = \mathcal{U}$
  - minimize $|\mathcal{C}|$
- NP-hard [Karp 1972]
- Approximable within $1 + \ln n$ [Johnson 1974]
- Not approximable within $(1 - \epsilon) \ln n$ for any $\epsilon > 0$ [Feige 1998]

The minimum set cover problem

- Input:
  - Universe $\mathcal{U}$ of *items*; $|\mathcal{U}| = n$
  - Collection $\mathcal{S}$ of subsets $S \subseteq \mathcal{I}$; $|\mathcal{S}| = m$.
- Output:
  - Subcollection $\mathcal{C} \subseteq \mathcal{S}$ that *covers* $\mathcal{U}$: $\cup_{c \in \mathcal{C}} C = \mathcal{U}$
  - minimize $|\mathcal{C}|$
- NP-hard [Karp 1972]
- Approximable within $1 + \ln n$ [Johnson 1974]
- Not approximable within $(1 - \epsilon) \ln n$ for any $\epsilon > 0$ [Feige 1998]
- *"... a problem whose study has led to the development of fundamental techniques for the entire field of approximation algorithms"* [Vazirani 2001]

Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$

Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$

Minimum set-cover as a (hyper)graph problem

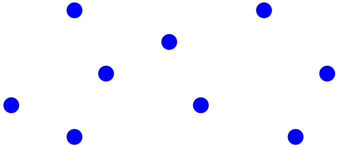
- Input: hypergraph $G = (V, E)$
  - $V =$ set of n nodes
  - $E =$ set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$

Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - $V =$ set of n nodes
  - $E =$ set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)

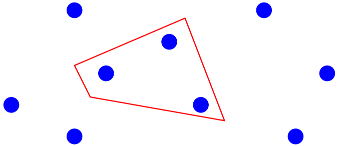
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



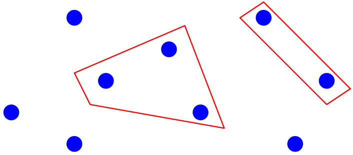
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



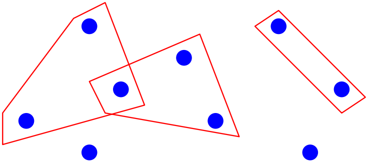
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



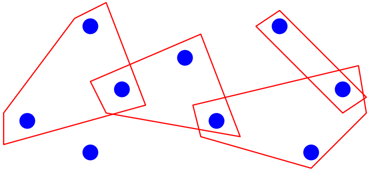
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



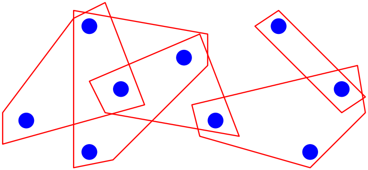
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



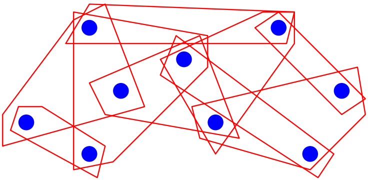
Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



Minimum set-cover as a (hyper)graph problem

- Input: hypergraph $G = (V, E)$
  - V = set of n nodes
  - E = set of m hyperedges $e \subseteq V$
- Output: edge subset $F \subseteq E$ that covers V, i.e., $\bigcup_{e \in F} e = V$
- minimize $|F|$
- Streaming model:
  - G is presented as a stream $e_1, \ldots, e_m$, each e_t given with its nodes.
  - Edge e_t identified by $O(\log m)$ bit $\mathrm{ID}(e_t)$ (e.g., t)



Edge δ-covers

- Generalization of edge (set) cover

Edge δ-covers

- Generalization of edge (set) cover

Definition

For $G = (V, E)$, and $0 < \delta \leq 1$, $F \subseteq E$ is an edge δ-cover of V if $|\{v \in V : \exists f \in F, v \in f\}| \geq \delta \cdot |V|$.

Edge δ-covers

- Generalization of edge (set) cover

Definition

For $G = (V, E)$, and $0 < \delta \leq 1$, $F \subseteq E$ is an edge δ-cover of V if $|\{v \in V : \exists f \in F, v \in f\}| \geq \delta \cdot |V|$.

- edge *cover* = edge 1-cover

Edge δ-covers

- Generalization of edge (set) cover

Definition

For $G = (V, E)$, and $0 < \delta \leq 1$, $F \subseteq E$ is an edge δ-cover of V if $|\{v \in V : \exists f \in F, v \in f\}| \geq \delta \cdot |V|$.

- edge *cover* = edge 1-cover
- Generalization of the set cover problem
  - Given G and δ
  - Find an $F \subseteq E$ that is an edge δ-cover for V, and minimizes $|F|$.

Cover certificates

- Recall: Each edge (set) e is associated with ID(e).

Cover certificates

- Recall: Each edge (set) e is associated with $\mathrm{ID}(e)$.
- "regular" setting:
  - Given $\{\mathrm{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.

Cover certificates

- Recall: Each edge (set) e is associated with $\mathrm{ID}(e)$.
- "regular" setting:
  - Given $\{\mathrm{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.
- Streaming setting:
  - This cannot be checked given only $\{\mathrm{ID}(e) : e \in F\}$.

Cover certificates

- Recall: Each edge (set) e is associated with $\mathrm{ID}(e)$.
- "regular" setting:
  - Given $\{\mathrm{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.
- Streaming setting:
  - This cannot be checked given only $\{\mathrm{ID}(e) : e \in F\}$.
- (some) streaming algorithm output a δ-cover certificate χ: partial function from V to $\mathrm{ID}(E)$ that satisfies

Cover certificates

- Recall: Each edge (set) e is associated with $\mathrm{ID}(e)$.
- "regular" setting:
  - Given $\{\mathrm{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.
- Streaming setting:
  - This cannot be checked given only $\{\mathrm{ID}(e) : e \in F\}$.
- (some) streaming algorithm output a δ-cover certificate χ: partial function from V to $\mathrm{ID}(E)$ that satisfies
  - if $\chi(v) = \mathrm{ID}(e)$, then $v \in e$ (soundness)
  - $|\mathrm{Dom}(\chi)| \geq \delta n$ (δ-coverage)

Cover certificates

- Recall: Each edge (set) e is associated with $\mathrm{ID}(e)$.
- "regular" setting:
  - Given $\{\mathrm{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.
- Streaming setting:
  - This cannot be checked given only $\{\mathrm{ID}(e) : e \in F\}$.
- (some) streaming algorithm output a δ-cover certificate χ: partial function from V to $\mathrm{ID}(E)$ that satisfies
  - if $\chi(v) = \mathrm{ID}(e)$, then $v \in e$ (soundness)
  - $|\mathrm{Dom}(\chi)| \geq \delta n$ (δ-coverage)

$\chi(v)$	7	3	$\perp$	3	4	$\perp$	3	3	7	4
v	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}

Cover certificates

- Recall: Each edge (set) e is associated with ID(e).
- "regular" setting:
  - Given $\{\text{ID}(e) : e \in F\}$ easy to check for given $v \in V$ if v covered by F.
- Streaming setting:
  - This cannot be checked given only $\{\text{ID}(e) : e \in F\}$.
- (some) streaming algorithm output a δ-cover certificate χ: partial function from V to ID(E) that satisfies
  - if $\chi(v) = \text{ID}(e)$, then $v \in e$ (soundness)
  - $|\mathrm{Dom}(\chi)| \geq \delta n$ (δ-coverage)
  - objective: minimize $|\mathrm{Im}(\chi)|$

$\chi(v)$	7	3	$\perp$	3	4	$\perp$	3	3	7	4
v	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}

Single pass [Emek, Rosén 2014]

Theorem

There is a semi-streaming algorithm that on an input hypergraph $G = (V, E)$ *uses* $O(n \log n)$ *space, and for every* $0 \leq \epsilon < 1$ *produces a* $(1 - \epsilon)$*-cover certificate* χ_ϵ *for* G *such that*

$$|\mathrm{Im}(\chi_\epsilon)| = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\}\right) \cdot |OPT| \ ,$$

where OPT *is the optimal edge cover for* G.

Single pass [Emek, Rosén 2014]

Theorem

There is a semi-streaming algorithm that on an input hypergraph $G = (V, E)$ *uses* $O(n \log n)$ *space, and for every* $0 \leq \epsilon < 1$ *produces a* $(1 - \epsilon)$*-cover certificate* χ_ϵ *for* G *such that*

$$|\mathrm{Im}(\chi_\epsilon)| = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\}\right) \cdot |OPT| \ ,$$

where OPT *is the optimal edge cover for* G*.*

- this statement assuming $m = n^{O(1)}$

Single pass [Emek, Rosén 2014]

Theorem

There is a semi-streaming algorithm that on an input hypergraph $G = (V, E)$ uses $O(n \log n)$ space, and for every $0 \leq \epsilon < 1$ produces a $(1-\epsilon)$-cover certificate χ_ϵ for G such that

$$|\mathrm{Im}(\chi_\epsilon)| = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\}\right) \cdot |OPT| \ ,$$

where OPT is the optimal edge cover for G.

- this statement assuming $m = n^{O(1)}$
- extends to the weighted case: benefit for nodes; costs for sets

Single pass [Emek, Rosén 2014]

Theorem

There is a semi-streaming algorithm that on an input hypergraph $G = (V, E)$ uses $O(n \log n)$ space, and for every $0 \leq \epsilon < 1$ produces a $(1 - \epsilon)$-cover certificate χ_ϵ for G such that

$$|\mathrm{Im}(\chi_\epsilon)| = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\}\right) \cdot |OPT| \ ,$$

where OPT is the optimal edge cover for G.

- this statement assuming $m = n^{O(1)}$
- extends to the weighted case: benefit for nodes; costs for sets
- run-time per edge $e_t \in E$ is $O(|e_t| \log |e_t|)$

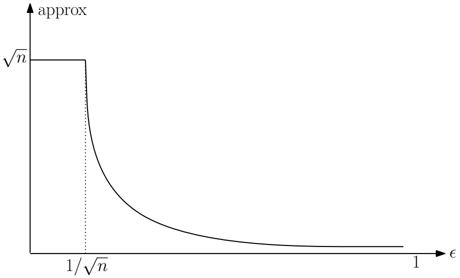
Single pass [Emek, Rosén 2014]

Theorem

There is a semi-streaming algorithm that on an input hypergraph $G = (V, E)$ uses $O(n \log n)$ space, and for every $0 \leq \epsilon < 1$ produces a $(1-\epsilon)$-cover certificate χ_ϵ for G such that

$$|\mathrm{Im}(\chi_\epsilon)| = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\}\right) \cdot |OPT| \ ,$$

where OPT is the optimal edge cover for G.



Single pass [Emek, Rosén 2014]

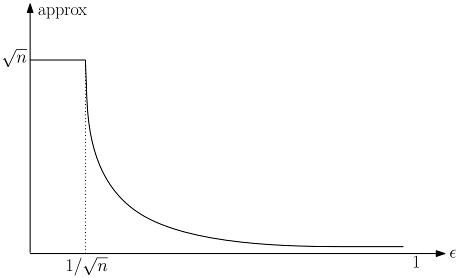
Theorem

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$*, and for every* $\epsilon \geq 1/\sqrt{n}$*, guarantees to output a* $(1-\epsilon)$*-cover certificate* χ *with* $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho_\epsilon \cdot |\mathtt{Opt}|$*, then* $\rho_\epsilon = \Omega(1/\epsilon)$.

Single pass [Emek, Rosén 2014]

Theorem

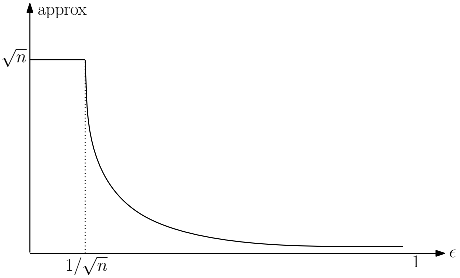
If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and for every $\epsilon \geq 1/\sqrt{n}$, guarantees to output a $(1-\epsilon)$-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho_\epsilon \cdot |\mathtt{Opt}|$, then $\rho_\epsilon = \Omega(1/\epsilon)$.



Single pass [Emek, Rosén 2014]

Theorem

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and for every $\epsilon \geq 1/\sqrt{n}$, guarantees to output a $(1-\epsilon)$-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho_\epsilon \cdot |\mathtt{Opt}|$, then $\rho_\epsilon = \Omega(1/\epsilon)$.



Single pass [Emek, Rosén 2014]

Theorem

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and for an $\epsilon \geq 1/\sqrt{n}$ guarantees to output a $(1-\epsilon)$-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho_\epsilon \cdot |\mathtt{Opt}|$, then $\rho_\epsilon = \Omega(1/\epsilon)$.

Theorem

Fix some constant real $\alpha > 0$.
If a randomized streaming algorithm uses memory of size $o(n^{1+\alpha})$, and for an $\epsilon \geq n^{-1/2+\alpha}$ guarantees to output a $(1-\epsilon)$-cover F with $\mathbb{E}[|F|] = \rho_\epsilon \cdot |\mathtt{Opt}|$, then $\rho_\epsilon = \Omega(\frac{\log\log n}{\log n} \cdot \frac{1}{\epsilon})$.

Single Pass, sublinear space [Assadi, Khanna, Li 2016]

Theorem

For any $\alpha = o(\sqrt{n}/\log n)$, and $m = poly(n)$, any randomized single-pass streaming algorithm that α-approximates the set cover problem with probability at least $2/3$ requires $\Omega(mn/\alpha)$ bits of space.

Single Pass, sublinear space [Assadi, Khanna, Li 2016]

Theorem

For any $\alpha = o(\sqrt{n}/\log n)$*, and* $m = poly(n)$*, any randomized single-pass streaming algorithm that* α*-approximates the set cover problem with probability at least* $2/3$ *requires* $\Omega(mn/\alpha)$ *bits of space.*

Theorem

For any $\alpha = o(\sqrt{n/\log n})$*, and* $m = poly(n)$*, any randomized single-pass streaming algorithm that* α*-approximates the size of the optimal set cover with probability at least* 0.9 *requires* $\Omega(mn/\alpha^2)$ *bits of space.*

Single Pass, sublinear space [Assadi, Khanna, Li 2016]

Theorem

For any $\alpha = o(\sqrt{n}/\log n)$, and $m = poly(n)$, any randomized single-pass streaming algorithm that α-approximates the set cover problem with probability at least $2/3$ requires $\Omega(mn/\alpha)$ bits of space.

Theorem

For any $\alpha = o(\sqrt{n/\log n})$, and $m = poly(n)$, any randomized single-pass streaming algorithm that α-approximates the size of the optimal set cover with probability at least 0.9 requires $\Omega(mn/\alpha^2)$ bits of space.

- Matching deterministic upper bound of set cover
- Matching randomized upper bound for estimating the size

Single Pass, sublinear space [Assadi, Khanna, Li 2016]

Theorem

For any $\alpha = o(\sqrt{n}/\log n)$, and $m = poly(n)$, any randomized single-pass streaming algorithm that α-approximates the set cover problem with probability at least $2/3$ requires $\Omega(mn/\alpha)$ bits of space.

Theorem

For any $\alpha = o(\sqrt{n/\log n})$, and $m = poly(n)$, any randomized single-pass streaming algorithm that α-approximates the size of the optimal set cover with probability at least 0.9 requires $\Omega(mn/\alpha^2)$ bits of space.

- Matching deterministic upper bound of set cover
- Matching randomized upper bound for estimating the size
- These results only for 1-covers

Multiple Passes [Chakrabarti, Wirth 2016]

Theorem

For every $p \geq 1$, there is a p-pass semi-streaming deterministic algorithm for weighted $(1-\epsilon)$ set-cover that returns a cover certificate that approximates the 1-cover up to $O(p \cdot \min\{n^{1/(p+1)}, \epsilon^{-1/p}\})$.

Theorem

Let $c > 0$ be a constant. If A is a randomized p-pass streaming algorithm for $(1-\epsilon)$ set cover, $0 <$ epsilon $< 1/2$, that for all large enough n and m, returns an α-approximation, $\alpha < \frac{1}{8c(p+1)^2} \cdot \min\{n^{1/(p+1)}, \epsilon^{-1/p}\}$, then A uses $\Omega(n^c/p^3)$ space.

Multiple Passes [Chakrabarti, Wirth 2016]

Theorem

For every $p \geq 1$, there is a p-pass semi-streaming deterministic algorithm for weighted $(1-\epsilon)$ set-cover that returns a cover certificate that approximates the 1-cover up to $O(p \cdot \min\{n^{1/(p+1)}, \epsilon^{-1/p}\})$.

Theorem

Let $c > 0$ be a constant. If A is a randomized p-pass streaming algorithm for $(1-\epsilon)$ set cover, $0 <$ epsilon $< 1/2$, that for all large enough n and m, returns an α-approximation, $\alpha < \frac{1}{8c(p+1)^2} \cdot \min\{n^{1/(p+1)}, \epsilon^{-1/p}\}$, then A uses $\Omega(n^c/p^3)$ space.

- lower bound on decision problem

Multiple Passes, sublinear space [Har-Peleg, Indyk, Mahabadi, Vakilian 2016 ; Assadi 2017]

Theorem (HIMV)

For every $p \geq 1$, there is a p-pass randomized algorithm for the set-cover problem that uses $\tilde{O}(mn^{1/p})$ space, and with high probability returns an $O(p)$ approximation.

Multiple Passes, sublinear space [Har-Peleg, Indyk, Mahabadi, Vakilian 2016 ; Assadi 2017]

Theorem (HIMV)

For every $p \geq 1$, there is a p-pass randomized algorithm for the set-cover problem that uses $\tilde{O}(mn^{1/p})$ space, and with high probability returns an $O(p)$ approximation.

- Approximation factor degrades with passes (space improves)

Multiple Passes, sublinear space [Har-Peleg, Indyk, Mahabadi, Vakilian 2016 ; Assadi 2017]

Theorem (HIMV)

For every $p \geq 1$, there is a p-pass randomized algorithm for the set-cover problem that uses $\tilde{O}(mn^{1/p})$ space, and with high probability returns an $O(p)$ approximation.

Theorem (A)

For every $p \geq 1$, $\alpha = o(\log n / \log\log n)$, any algorithm that makes p passes, and returns with constant probability an α approximation, uses $\tilde{\Omega}(mn^{1/\alpha}/p)$ space.

Multiple Passes, sublinear space [Har-Peleg, Indyk, Mahabadi, Vakilian 2016 ; Assadi 2017]

Theorem (HIMV)

For every $p \geq 1$, there is a p-pass randomized algorithm for the set-cover problem that uses $\tilde{O}(mn^{1/p})$ space, and with high probability returns an $O(p)$ approximation.

Theorem (A)

For every $p \geq 1$, $\alpha = o(\log n / \log\log n)$, any algorithm that makes p passes, and returns with constant probability an α approximation, uses $\tilde{\Omega}(mn^{1/\alpha}/p)$ space.

- Lower bound applies to estimating the size.

1. The streaming model
2. The set-cover problem
3. (some of the) Results
4. (some of the) Techniques
   - Single pass, semi-streaming algorithm (unweighted case)
   - Matching lower bound(s)
5. Conclusions and open problems

Single Pass ϵ-oblivious algorithm [Emek, Rosén 2014]

- Hypergraph $G = (V, E = \{e_1, \ldots, e_m\})$

Single Pass ϵ-oblivious algorithm [Emek, Rosén 2014]

- Hypergraph $G = (V, E = \{e_1, \ldots, e_m\})$
- streaming stage: $\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow$ data structure $\mathcal{D}$

Single Pass ϵ-oblivious algorithm [Emek, Rosén 2014]

- Hypergraph $G = (V, E = \{e_1, \ldots, e_m\})$
- streaming stage: $\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow$ data structure $\mathcal{D}$
  - space used $O(n \log n)$
  - run-time per edge $e_t \in E$ is $O(|e_t| \log |e_t|)$

Single Pass ϵ-oblivious algorithm [Emek, Rosén 2014]

- Hypergraph $G = (V, E = \{e_1, \ldots, e_m\})$
- streaming stage: $\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow$ data structure $\mathcal{D}$
  - space used $O(n \log n)$
  - run-time per edge $e_t \in E$ is $O(|e_t| \log |e_t|)$
- output stage: $\texttt{Alg}_{\text{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1 - \epsilon)$-cover certificate χ_ϵ

Single Pass ϵ-oblivious algorithm [Emek, Rosén 2014]

- Hypergraph $G = (V, E = \{e_1, \dots, e_m\})$
- streaming stage: $\texttt{Alg}_{\text{streaming}}(e_1, \dots, e_m) \longrightarrow$ data structure $\mathcal{D}$
  - space used $O(n \log n)$
  - run-time per edge $e_t \in E$ is $O(|e_t| \log |e_t|)$
- output stage: $\texttt{Alg}_{\text{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ
  - no additional memory
  - running time $O(n \log n)$.

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$

$\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v
- Initially: $\mathrm{eid}(v) \leftarrow \perp$, $\mathrm{qlt}(v) \leftarrow 0$ for every $v \in V$

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v
- Initially: $\mathrm{eid}(v) \leftarrow \perp$, $\mathrm{qlt}(v) \leftarrow 0$ for every $v \in V$

Definition

$X \subseteq e_t$ is good at time t if $\lceil \lg |X| \rceil > \mathrm{qlt}(v)$ for every $v \in X$.

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v
- Initially: $\mathrm{eid}(v) \leftarrow \perp$, $\mathrm{qlt}(v) \leftarrow 0$ for every $v \in V$

Definition

$X \subseteq e_t$ is good at time t if $\lceil \lg |X| \rceil > \mathrm{qlt}(v)$ for every $v \in X$.

Intuitively: the larger X is, "better" is the coverage by $X \subseteq e_t$

$\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v
- Initially: $\mathrm{eid}(v) \leftarrow \perp$, $\mathrm{qlt}(v) \leftarrow 0$ for every $v \in V$

Definition

$X \subseteq e_t$ is good at time t if $\lceil \lg |X| \rceil > \mathrm{qlt}(v)$ for every $v \in X$.

Intuitively: the larger X is, "better" is the coverage by $X \subseteq e_t$

- Update rule: upon arrival of edge e_t
  - $X^* \leftarrow$ largest good subset of e_t

$\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- Maintains 2 variables for each $v \in V$
  - $\mathrm{eid}(v) = \mathrm{ID}(e)$ for some $e \in E$ s.t. $v \in e$
  - $\mathrm{qlt}(v) =$ integer capturing the quality of e in covering v
- Initially: $\mathrm{eid}(v) \leftarrow \perp$, $\mathrm{qlt}(v) \leftarrow 0$ for every $v \in V$

Definition

$X \subseteq e_t$ is good at time t if $\lceil \lg |X| \rceil > \mathrm{qlt}(v)$ for every $v \in X$.

Intuitively: the larger X is, "better" is the coverage by $X \subseteq e_t$

- Update rule: upon arrival of edge e_t
  - $X^* \leftarrow$ largest good subset of e_t
  - For every $v \in X^*$:
    $\mathrm{eid}(v) \leftarrow \mathrm{ID}(e_t) \quad ; \quad \mathrm{qlt}(v) \leftarrow \lceil \lg |X^*| \rceil$

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- $e_t = \{v_1, \ldots, v_{12}\}$

$\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- $e_t = \{v_1, \ldots, v_{12}\}$

$\mathrm{eid}_t(v)$	$\perp$	$\perp$	3	1	9	8	6	7	6	24	19	26
$\mathrm{qlt}_t(v)$	0	0	1	2	2	3	5	5	5	7	8	8
$v \in e_t$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}

$\texttt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- $e_t = \{v_1, \ldots, v_{12}\}$

$\mathrm{eid}_t(v)$	$\perp$	$\perp$	3	1	9	8	6	7	6	24	19	26
$\mathrm{qlt}_t(v)$	0	0	1	2	2	3	5	5	5	7	8	8
$v \in e_t$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}

$$\underbrace{v_1 \; v_2 \; v_3 \; v_4 \; v_5}_{X^*}$$

$\mathtt{Alg}_{\text{streaming}}(e_1, \ldots, e_m) \longrightarrow \mathcal{D}$

- $e_t = \{v_1, \ldots, v_{12}\}$

$\mathrm{eid}_{t+1}(v)$	t	t	t	t	t	8	6	7	6	24	19	26
$\mathrm{qlt}_{t+1}(v)$	3	3	3	3	3	3	5	5	5	7	8	8
$v \in e_t$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}

$\underbrace{v_1 \; v_2 \; v_3 \; v_4 \; v_5}_{X^*}$

$\mathtt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- Notation:

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- Notation:
  - $\mathrm{qlt}_\infty(v) = \mathrm{qlt}(v)$ upon termination
  - $\mathrm{eid}_\infty(v) = \mathrm{eid}(v)$ upon termination

$\mathtt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- Notation:
  - $\mathrm{qlt}_\infty(v) = \mathrm{qlt}(v)$ upon termination
  - $\mathrm{eid}_\infty(v) = \mathrm{eid}(v)$ upon termination
  - $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
    i.e., all edges that give high-quality $(> r)$ coverage to at least one node

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \le \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- Notation:
  - $\mathrm{qlt}_\infty(v) = \mathrm{qlt}(v)$ upon termination
  - $\mathrm{eid}_\infty(v) = \mathrm{eid}(v)$ upon termination
  - $S(r) = \{e \in E \mid \exists v \in V$ s.t. $\mathrm{eid}_\infty(v) = \mathrm{ID}(e)$ and $\mathrm{qlt}_\infty(v) > r\}$
    i.e., all edges that give high-quality ($> r$) coverage to at least one node
  - $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \le r\}$
    i.e., all nodes that have poor ($\le r$) quality for their coverage

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \le \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
  - i.e., all edges that give high-quality ($> r$) coverage to at least one node
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \le r\}$
  - i.e., all nodes that have poor ($\le r$) quality for their coverage

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \le \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
  - i.e., all edges that give high-quality ($> r$) coverage to at least one node
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \le r\}$
  - i.e., all nodes that have poor ($\le r$) quality for their coverage

Given $0 \le \epsilon < 1$:

$\mathtt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
  - i.e., all edges that give high-quality ($> r$) coverage to at least one node
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \leq r\}$
  - i.e., all nodes that have poor ($\leq r$) quality for their coverage

Given $0 \leq \epsilon < 1$:

- If $\epsilon \geq 1/\sqrt{n}$
  - Pick largest integer r^* s.t. $|I(r^*)| \leq \epsilon \cdot n$
  - Return $\chi : V \to \mathrm{eid}(E)$ that maps every $v \in V - I(r^*)$ to $\mathrm{eid}_\infty(v)$.

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \le \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
  - i.e., all edges that give high-quality $(> r)$ coverage to at least one node
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \le r\}$
  - i.e., all nodes that have poor $(\le r)$ quality for their coverage

Given $0 \le \epsilon < 1$:

- If $\epsilon \ge 1/\sqrt{n}$
  - Pick largest integer r^* s.t. $|I(r^*)| \le \epsilon \cdot n$
  - Return $\chi : V \to \mathrm{eid}(E)$ that maps every $v \in V - I(r^*)$ to $\mathrm{eid}_\infty(v)$.
- If $\epsilon < 1/\sqrt{n}$
  - Return $\chi : V \to \mathrm{eid}(E)$ that maps every $v \in V$ to $\mathrm{eid}_\infty(v)$.

$\texttt{Alg}_{\mathsf{RAM}}(\mathcal{D}, 0 \leq \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \leq r\}$

Lemma

For every $r \in \mathbb{Z}_{\geq 0}$, $|I(r)| < 2^{r+1} \cdot |\texttt{Opt}|$.

Lemma

For every $r \in \mathbb{Z}_{\geq 0}$, $|S(r)| < n/2^{r-1}$.

$\texttt{Alg}_{\text{RAM}}(\mathcal{D}, 0 \le \epsilon < 1) \longrightarrow (1-\epsilon)$-cover certificate χ_ϵ

- $S(r) = \{e \in E \mid \exists v \in V \text{ s.t. } \mathrm{eid}_\infty(v) = \mathrm{ID}(e) \text{ and } \mathrm{qlt}_\infty(v) > r\}$
- $I(r) = \{v \in V \mid \mathrm{qlt}_\infty(v) \le r\}$

Lemma

For every $r \in \mathbb{Z}_{\ge 0}$, $|I(r)| < 2^{r+1} \cdot |\texttt{Opt}|$.

Lemma

For every $r \in \mathbb{Z}_{\ge 0}$, $|S(r)| < n/2^{r-1}$.

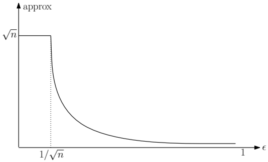
Idea of proof of approximation factor:

- $S(r^*) < n/2^{r^*-1} \le \frac{1}{\epsilon} \cdot |I(r^*)| \cdot \frac{1}{2^{r^*-1}} < \frac{4}{\epsilon} \cdot |\texttt{Opt}|$.

Lower Bound - single pass, semi-streaming

Theorem (1-coverage)

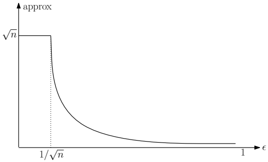
If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and guarantees to output a (1)-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho \cdot |\mathtt{Opt}|$, then $\rho = \Omega(\sqrt{n})$.



Lower Bound - single pass, semi-streaming

Theorem (1-coverage)

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and guarantees to output a (1)-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho \cdot |\mathtt{Opt}|$, then $\rho = \Omega(\sqrt{n})$.

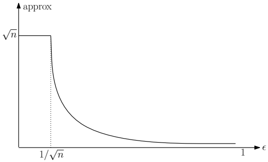


Distribution $\mathcal{G}$ over n-node hypergraphs (based on affine planes)

Lower Bound - single pass, semi-streaming

Theorem (1-coverage)

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and guarantees to output a (1)-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho \cdot |\mathtt{Opt}|$, then $\rho = \Omega(\sqrt{n})$.



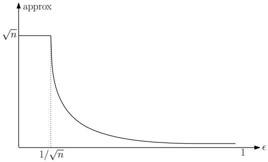
Distribution $\mathcal{G}$ over n-node hypergraphs (based on affine planes)

- $\mathtt{Opt}(G) = O(1)$ for every $G \in \mathcal{G}$

Lower Bound - single pass, semi-streaming

Theorem (1-coverage)

If a randomized streaming algorithm uses memory of size $o(n^{3/2})$, and guarantees to output a (1)-cover certificate χ with $\mathbb{E}[|\mathrm{Im}(\chi)|] = \rho \cdot |\mathtt{Opt}|$, then $\rho = \Omega(\sqrt{n})$.



Distribution $\mathcal{G}$ over n-node hypergraphs (based on affine planes)

- $\mathtt{Opt}(G) = O(1)$ for every $G \in \mathcal{G}$
- Every deterministic streaming algorithm with memory $o(n^{3/2})$ that outputs 1-cover certificate χ has $\mathbb{E}_{\mathcal{G}}[|\mathrm{Im}(\chi)|] = \Omega(\sqrt{n})$.

1. The streaming model
2. The set-cover problem
3. (some of the) Results
4. (some of the) Techniques
   - Single pass, semi-streaming algorithm (unweighted case)
   - Matching lower bound(s)
5. Conclusions and open problems

Conclusions

Our results [Emek, Rosén 2014]:

Subsequent work:

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1-\epsilon)$ cover certificates.

Subsequent work:

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1-\epsilon)$ cover certificates.
- Almost tight results on approximation factor in 1-pass semi-streaming of $(1-\epsilon)$ covers.

Subsequent work:

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1 - \epsilon)$ cover certificates.
- Almost tight results on approximation factor in 1-pass semi-streaming of $(1 - \epsilon)$ covers.
- Producing in a streaming setting a data structure, then, for given ϵ, extracting output.

Subsequent work:

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1-\epsilon)$ cover certificates.
- Almost tight results on approximation factor in 1-pass semi-streaming of $(1-\epsilon)$ covers.
- Producing in a streaming setting a data structure, then, for given ϵ, extracting output.

Subsequent work:

- Tight tradeoffs in single pass between sub-linear ($o(mn)$) space and approximation of 1-covers [Assadi, Khanna, Li 2016].

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1-\epsilon)$ cover certificates.
- Almost tight results on approximation factor in 1-pass semi-streaming of $(1-\epsilon)$ covers.
- Producing in a streaming setting a data structure, then, for given ϵ, extracting output.

Subsequent work:

- Tight tradeoffs in single pass between sub-linear ($o(mn)$) space and approximation of 1-covers [Assadi, Khanna, Li 2016].
- (Almost) tight tradeoffs between number of passes and approximation in semi-streaming $(1-\epsilon)$-covers [Charkrabarti, Wirth 2016]

Conclusions

Our results [Emek, Rosén 2014]:

- Tight results on approximation factor in 1-pass semi-streaming ($\tilde{O}(n)$ space) of $(1-\epsilon)$ cover certificates.
- Almost tight results on approximation factor in 1-pass semi-streaming of $(1-\epsilon)$ covers.
- Producing in a streaming setting a data structure, then, for given ϵ, extracting output.

Subsequent work:

- Tight tradeoffs in single pass between sub-linear ($o(mn)$) space and approximation of 1-covers [Assadi, Khanna, Li 2016].
- (Almost) tight tradeoffs between number of passes and approximation in semi-streaming $(1-\epsilon)$-covers [Charkrabarti, Wirth 2016]
- (Almost) tight tradeoffs between number of passes and approximation in sub-linear ($o(mn)$) space 1-cover. [Har-Peleg, Indyk, Mahabadi, Vakilian 2016 ; Assadi 2017]

Open Problems

- Extend results for sublinear space to $(1-\epsilon)$ covers.

Open Problems

- Can we approximate the optimal $(1 - \epsilon)$ cover ?

Thank you

Thank You