Tight Space-Approximation Tradeoff for the Multi-Pass Streaming Set Cover Problem

Sepehr Assadi

University of Pennsylvania

The Set Cover Problem

- Input: A collection of m sets $S_1, \ldots, S_m$ from a universe $[n]$.

- Goal: Choose a smallest subset C of the sets from $S_1, \ldots, S_m$ such that C covers $[n]$, i.e., $\bigcup_{i \in C} S_i = [n]$.

The Set Cover Problem

- Input: A collection of m sets $S_1, \ldots, S_m$ from a universe $[n]$.
- Goal: Choose a smallest subset C of the sets from $S_1, \ldots, S_m$ such that C covers $[n]$, i.e., $\bigcup_{i \in C} S_i = [n]$.

We use OPT to denote the optimal solution size.

The Set Cover Problem

A classic optimization problem with many applications:

The Set Cover Problem

A classic optimization problem with many applications:

- Information retrieval,
  - e.g., finding a smallest number of documents covering all the topics in a given query.

The Set Cover Problem

A classic optimization problem with many applications:

- Information retrieval,
  - e.g., finding a smallest number of documents covering all the topics in a given query.
- Data mining,
  - e.g., finding a smallest number of features explaining all positive examples, i.e., a "minimal explanation" of a pattern.

The Set Cover Problem

A classic optimization problem with many applications:

- Information retrieval,
  - e.g., finding a smallest number of documents covering all the topics in a given query.
- Data mining,
  - e.g., finding a smallest number of features explaining all positive examples, i.e., a "minimal explanation" of a pattern.
- Web search and advertising,
  - e.g., finding a smallest number of impressions to reach a certain set of users.

The Set Cover Problem

A classic optimization problem with many applications:

- Information retrieval,
  - e.g., finding a smallest number of documents covering all the topics in a given query.
- Data mining,
  - e.g., finding a smallest number of features explaining all positive examples, i.e., a "minimal explanation" of a pattern.
- Web search and advertising,
  - e.g., finding a smallest number of impressions to reach a certain set of users.
- Operation research, machine learning, web host analysis, ...

The Set Cover Problem: Classical Setting

Theoretical aspects:

- One of Karp's original 21 NP-hard problems [Karp, 1972].
- The greedy algorithm that picks the "best" set in each iteration achieves $\ln(n)$ approximation [Johnson, 1974, Slavík, 1997].
- No better approximation factor is possible in polynomial time unless $\mathsf{P} = \mathsf{NP}$ [Lund and Yannakakis, 1994, Feige, 1998, Dinur and Steurer, 2014, Moshkovitz, 2015].

The Set Cover Problem: Classical Setting

Theoretical aspects:

- One of Karp's original 21 NP-hard problems [Karp, 1972].
- The greedy algorithm that picks the "best" set in each iteration achieves $\ln(n)$ approximation [Johnson, 1974, Slavík, 1997].
- No better approximation factor is possible in polynomial time unless $\mathsf{P} = \mathsf{NP}$ [Lund and Yannakakis, 1994, Feige, 1998, Dinur and Steurer, 2014, Moshkovitz, 2015].

In practice,

- The greedy algorithm is highly efficient and surprisingly accurate.
- Returned solution has $< 10\% \cdot \mathsf{OPT}$ sets more than the optimal solution on a typical data set [Grossman and Wool, 1997, Gomes et al., 2006, Cormode et al., 2010].

The Set Cover Problem: Classical Setting

Theoretical aspects:

- One of Karp's original 21 NP-hard problems [Karp, 1972].
- The greedy algorithm that picks the "best" set in each iteration achieves $\ln(n)$ approximation [Johnson, 1974, Slavík, 1997].
- No better approximation factor is possible in polynomial time unless $\mathsf{P} = \mathsf{NP}$ [Lund and Yannakakis, 1994, Feige, 1998, Dinur and Steurer, 2014, Moshkovitz, 2015].

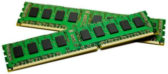
In practice,

- The greedy algorithm is ~~highly efficient~~ and surprisingly accurate.
- Returned solution has $< 10\% \cdot \mathsf{OPT}$ sets more than the optimal solution on a typical data set [Grossman and Wool, 1997, Gomes et al., 2006, Cormode et al., 2010].

as long as the dataset is relatively small!

The Set Cover Problem: Big Data Scenario

[Cormode et al., 2010]: A direct implementation of the greedy algorithm scales surprisingly poorly when the data size grows.

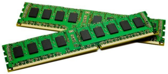


Efficient on main memory

Inefficient on disk

The Set Cover Problem: Big Data Scenario

[Cormode et al., 2010]: A direct implementation of the greedy algorithm scales surprisingly poorly when the data size grows.



Efficient on main memory

Inefficient on disk

One approach: the streaming model for the set cover problem introduced by [Saha and Getoor, 2009].

The Streaming Set Cover Problem

Model:

- Sequential access to the sets:
  - The input sets $S_1, \ldots, S_m$ are presented one by one in a stream.

The Streaming Set Cover Problem

Model:

- Sequential access to the sets:
  - The input sets $S_1, \ldots, S_m$ are presented one by one in a stream.
- Small working memory:
  - The streaming algorithm has a small space to maintain a summary of the input sets.

The Streaming Set Cover Problem

Model:

- Sequential access to the sets:
  - The input sets $S_1, \ldots, S_m$ are presented one by one in a stream.
- Small working memory:
  - The streaming algorithm has a small space to maintain a summary of the input sets.
- Efficiency:
  - The algorithm can make one or few passes over the stream and should output the answer using only the stored summary.

The Streaming Set Cover Problem

Model:

- Sequential access to the sets:
  - The input sets $S_1, \ldots, S_m$ are presented one by one in a stream.
- Small working memory:
  - The streaming algorithm has a small space to maintain a summary of the input sets.
- Efficiency:
  - The algorithm can make one or few passes over the stream and should output the answer using only the stored summary.

Small space:

1. Semi-streaming space, i.e., $\widetilde{O}(n)$.
2. Sub-linear space, i.e., $o(mn)$.

The Streaming Set Cover Problem

Note. We do not restrict the computation time of the algorithms in this model, e.g., allow exponential time computation.

The Streaming Set Cover Problem

Note. We do not restrict the computation time of the algorithms in this model, e.g., allow exponential time computation.

- For theoretical purposes: understanding the space complexity of streaming algorithms in absence of time complexity restrictions.

The Streaming Set Cover Problem

Note. We do not restrict the computation time of the algorithms in this model, e.g., allow exponential time computation.

- For theoretical purposes: understanding the space complexity of streaming algorithms in absence of time complexity restrictions.
- For practical purposes: we rarely need the full power of such exponential time computation anyway.

State of the Art

Many interesting results: [Saha and Getoor, 2009, Cormode et al., 2010, Emek and Rosén, 2014, Demaine et al., 2014, Badanidiyuru et al., 2014, Indyk et al., 2015, Har-Peled et al., 2016, Chakrabarti and Wirth, 2016, Assadi et al., 2016, McGregor and Vu, 2016, Bateni et al., 2016].

State of the Art

Many interesting results: [Saha and Getoor, 2009, Cormode et al., 2010, Emek and Rosén, 2014, Demaine et al., 2014, Badanidiyuru et al., 2014, Indyk et al., 2015, Har-Peled et al., 2016, Chakrabarti and Wirth, 2016, Assadi et al., 2016, McGregor and Vu, 2016, Bateni et al., 2016].

In particular,

- Complete resolution of the complexity of multi-pass semi-streaming algorithms [Chakrabarti and Wirth, 2016].

State of the Art

Many interesting results: [Saha and Getoor, 2009, Cormode et al., 2010, Emek and Rosén, 2014, Demaine et al., 2014, Badanidiyuru et al., 2014, Indyk et al., 2015, Har-Peled et al., 2016, Chakrabarti and Wirth, 2016, Assadi et al., 2016, McGregor and Vu, 2016, Bateni et al., 2016].

In particular,

- Complete resolution of the complexity of multi-pass semi-streaming algorithms [Chakrabarti and Wirth, 2016].
- Complete resolution of the complexity of single-pass sub-linear space streaming algorithms [Assadi et al., 2016].

State of the Art

Many interesting results: [Saha and Getoor, 2009, Cormode et al., 2010, Emek and Rosén, 2014, Demaine et al., 2014, Badanidiyuru et al., 2014, Indyk et al., 2015, Har-Peled et al., 2016, Chakrabarti and Wirth, 2016, Assadi et al., 2016, McGregor and Vu, 2016, Bateni et al., 2016].

In particular,

- Complete resolution of the complexity of multi-pass semi-streaming algorithms [Chakrabarti and Wirth, 2016].
- Complete resolution of the complexity of single-pass sub-linear space streaming algorithms [Assadi et al., 2016].

Short summary: to ensure efficiency, we need more than $\widetilde{O}(n)$ space and more than one pass!

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

Constant approximation in sub-linear space and constant number of passes!

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

Constant approximation in sub-linear space and constant number of passes!

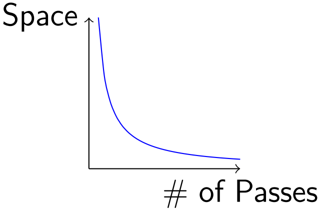
Formally, $O(p)$-Approximation in $\widetilde{O}(m \cdot n^{\Theta(1/p)})$ space and p passes.

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

Constant approximation in sub-linear space and constant number of passes!

Formally, $O(p)$-Approximation in $\widetilde{O}(m \cdot n^{\Theta(1/p)})$ space and p passes.



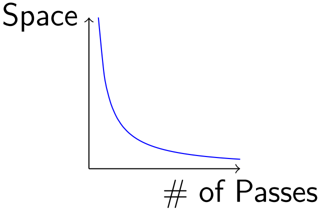
the space-pass tradeoff

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

Constant approximation in sub-linear space and constant number of passes!

Formally, $O(p)$-Approximation in $\widetilde{O}(m \cdot n^{\Theta(1/p)})$ space and p passes.



the space-pass tradeoff

[Har-Peled et al., 2016]:

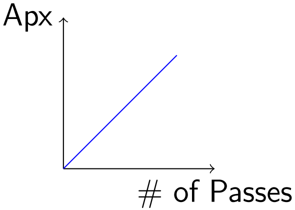
Conjecture. This tradeoff is tight for small approximation factors.

State of the Art

The best known sub-linear space algorithm [Har-Peled et al., 2016]:

Constant approximation in sub-linear space and constant number of passes!

Formally, $O(p)$-Approximation in $\widetilde{O}(m \cdot n^{\Theta(1/p)})$ space and p passes.



the pass-approximation tradeoff

Not a typical pass-approximation tradeoff!

Motivating Questions

- Can we obtain a fixed constant approximation to streaming set cover while improving the space via a small number of passes?

Motivating Questions

- Can we obtain a fixed constant approximation to streaming set cover while improving the space via a small number of passes?
- What is the space-approximation tradeoff for multi-pass streaming algorithms for set cover?

Motivating Questions

- Can we obtain a fixed constant approximation to streaming set cover while improving the space via a small number of passes?

- What is the space-approximation tradeoff for multi-pass streaming algorithms for set cover?
  - We already know an upper bound result:

    α-approximation in $\widetilde{O}(mn^{\Theta(1/\alpha)})$ space [Har-Peled et al., 2016].

Motivating Questions

- Can we obtain a fixed constant approximation to streaming set cover while improving the space via a small number of passes?
  Answer: No!
- What is the space-approximation tradeoff for multi-pass streaming algorithms for set cover?
  - We already know an upper bound result:

    α-approximation in $\widetilde{O}(mn^{\Theta(1/\alpha)})$ space [Har-Peled et al., 2016].

Motivating Questions

- Can we obtain a fixed constant approximation to streaming set cover while improving the space via a small number of passes?
  Answer: No!

- What is the space-approximation tradeoff for multi-pass streaming algorithms for set cover?
  - We already know an upper bound result:

    α-approximation in $\widetilde{O}(mn^{\Theta(1/\alpha)})$ space [Har-Peled et al., 2016].

  Answer: The above space-approximation tradeoff is essentially tight even allowing $\mathrm{polylog}(n)$ passes over the stream!

Our Main Result

Theorem

For $\alpha = o(\log n)$, any p-pass α-approximation algorithm (deterministic or randomized) for the streaming set cover requires $\widetilde{\Omega}\left(\frac{1}{p} \cdot mn^{1/\alpha}\right)$ space, even if the sets are arriving in a random order.

Our Main Result

Theorem

For $\alpha = o(\log n)$, any p-pass α-approximation algorithm (deterministic or randomized) for the streaming set cover requires $\widetilde{\Omega}\left(\frac{1}{p} \cdot mn^{1/\alpha}\right)$ space, even if the sets are arriving in a random order.

Remark.

- The lower bound has nothing to do with the NP-hardness of approximating set cover!
- It holds in the regime when $\mathrm{OPT} = O(1)$ in which case set cover admits a trivial poly-time algorithm in the classical setting.

Further Results

- We show that with proper modifications, the algorithm of [Har-Peled et al., 2016] can be implemented in $\widetilde{O}(mn^{1/\alpha})$ space, matching our lower bound up to logarithmic factors.

Further Results

- We show that with proper modifications, the algorithm of [Har-Peled et al., 2016] can be implemented in $\widetilde{O}(mn^{1/\alpha})$ space, matching our lower bound up to logarithmic factors.

- Using similar ideas, we can also prove a tight lower bound for the space complexity of $(1-\varepsilon)$-approximating the streaming maximum coverage problem.

Communication Complexity

We use communication complexity to prove our lower bound.

Communication Complexity

We use communication complexity to prove our lower bound.

Two-player Communication Model:

- Alice gets the sets $S_1, \ldots, S_m$ and Bob gets $T_1, \ldots, T_m$.

$S_1, \ldots, S_m$

$T_1, \ldots, T_m$

Communication Complexity

We use communication complexity to prove our lower bound.

Two-player Communication Model:

- Alice gets the sets $S_1, \ldots, S_m$ and Bob gets $T_1, \ldots, T_m$.
- Their goal is to compute an exact/approximate set cover of their combined input.

$S_1, \ldots, S_m$

$T_1, \ldots, T_m$

Communication Complexity

We use communication complexity to prove our lower bound.

Two-player Communication Model:

- Alice gets the sets $S_1, \ldots, S_m$ and Bob gets $T_1, \ldots, T_m$.
- Their goal is to compute an exact/approximate set cover of their combined input.
- Alice and Bob are allowed to communicate with each other to compute the set cover.

$S_1, \ldots, S_m$ $\qquad$ $T_1, \ldots, T_m$

Communication Complexity

We use communication complexity to prove our lower bound.

Two-player Communication Model:

- Alice gets the sets $S_1, \ldots, S_m$ and Bob gets $T_1, \ldots, T_m$.
- Their goal is to compute an exact/approximate set cover of their combined input.
- Alice and Bob are allowed to communicate with each other to compute the set cover.
- Communication Complexity $\mathrm{CC}(\textsc{SetCover})$: minimum amount of communication needed to solve the problem w.p. $\geq 2/3$.

$S_1, \ldots, S_m$ $T_1, \ldots, T_m$

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

$S_1, \ldots, S_m$

stream s

$T_1, \ldots, T_m$

stream t

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

$\xrightarrow{\mathcal{A}(s)}$

$S_1, \ldots, S_m$

stream s

$T_1, \ldots, T_m$

stream t

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

$\mathcal{A}(s)$ $\longrightarrow$

$\longleftarrow$ $\mathcal{A}(s \circ t)$

$S_1, \ldots, S_m$

stream s

$T_1, \ldots, T_m$

stream t

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

$S_1, \ldots, S_m$

stream s

$$\xrightarrow{\quad\mathcal{A}(s)\quad}$$

$$\xleftarrow{\quad\mathcal{A}(s \circ t)\quad}$$

$$\vdots$$

$$\longrightarrow$$

$T_1, \ldots, T_m$

stream t

Communication Complexity and Streaming

Fact. Any p-pass s-space streaming algorithm $\mathcal{A}$ for set cover implies an $O(p \cdot s)$-communication protocol.

$S_1, \ldots, S_m$

stream s

$$\xrightarrow{\mathcal{A}(s)}$$

$$\xleftarrow{\mathcal{A}(s \circ t)}$$

$$\vdots$$

$$\longrightarrow$$

$T_1, \ldots, T_m$

stream t

Hence, space complexity of p-pass streaming algorithms for the set cover problem $\geq \frac{1}{p} \cdot \mathrm{CC}(\textsc{SetCover})$.

Communication Complexity of Set Cover

Fix an $\alpha \ll \log n$. Define:

$\textsc{SetCover}$: the two-player communication problem of finding an α-approximation to the set cover problem.

Theorem

$$CC(\textsc{SetCover}) = \widetilde{\Omega}(m \cdot n^{1/\alpha})$$

Communication Complexity of Set Cover

Fix an $\alpha \ll \log n$. Define:

$\textsc{SetCover}$: the two-player communication problem of finding an α-approximation to the set cover problem.

Theorem

$$CC(\textsc{SetCover}) = \widetilde{\Omega}(m \cdot n^{1/\alpha})$$

We create a distribution $\mathcal{D} := \frac{1}{2} \cdot \mathcal{D}^{\mathsf{Y}} + \frac{1}{2} \cdot \mathcal{D}^{\mathsf{N}}$ whereby:

Communication Complexity of Set Cover

Fix an $\alpha \ll \log n$. Define:

$\textsc{SetCover}$: the two-player communication problem of finding an α-approximation to the set cover problem.

Theorem

$$CC(\textsc{SetCover}) = \widetilde{\Omega}(m \cdot n^{1/\alpha})$$

We create a distribution $\mathcal{D} := \frac{1}{2} \cdot \mathcal{D}^{\mathsf{Y}} + \frac{1}{2} \cdot \mathcal{D}^{\mathsf{N}}$ whereby:

1. Every instance sampled from $\mathcal{D}^{\mathsf{Y}}$ (Yes instance), has OPT $= 2$.

Communication Complexity of Set Cover

Fix an $\alpha \ll \log n$. Define:

$\textsc{SetCover}$: the two-player communication problem of finding an α-approximation to the set cover problem.

Theorem

$$CC(\textsc{SetCover}) = \widetilde{\Omega}(m \cdot n^{1/\alpha})$$

We create a distribution $\mathcal{D} := \frac{1}{2} \cdot \mathcal{D}^{\mathsf{Y}} + \frac{1}{2} \cdot \mathcal{D}^{\mathsf{N}}$ whereby:

1. Every instance sampled from $\mathcal{D}^{\mathsf{Y}}$ (Yes instance), has $\mathsf{OPT} = 2$.
2. Each instance sampled from $\mathcal{D}^{\mathsf{N}}$ (No instance), has $\mathsf{OPT} > 2\alpha$ w.p. $1 - o(1)$.

Communication Complexity of Set Cover

Fix an $\alpha \ll \log n$. Define:

$\textsc{SetCover}$: the two-player communication problem of finding an α-approximation to the set cover problem.

Theorem

$$CC(\textsc{SetCover}) = \widetilde{\Omega}(m \cdot n^{1/\alpha})$$

We create a distribution $\mathcal{D} := \frac{1}{2} \cdot \mathcal{D}^{\mathsf{Y}} + \frac{1}{2} \cdot \mathcal{D}^{\mathsf{N}}$ whereby:

1. Every instance sampled from $\mathcal{D}^{\mathsf{Y}}$ (Yes instance), has $\mathsf{OPT} = 2$.
2. Each instance sampled from $\mathcal{D}^{\mathsf{N}}$ (No instance), has $\mathsf{OPT} > 2\alpha$ w.p. $1 - o(1)$.
3. Distinguishing between Yes and No instances requires $\widetilde{\Omega}(mn^{1/\alpha})$ communication.

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:
   - Each Z_i is a random set of size $\approx n - n^{(1-1/\alpha)}$ chosen from $[n]$.
   - Think of creating Z_i by (essentially) removing each element from $[n]$ w.p. $\approx 1/n^{1/\alpha}$.

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:
   - Each Z_i is a random set of size $\approx n - n^{(1-1/\alpha)}$ chosen from $[n]$.
   - Think of creating Z_i by (essentially) removing each element from $[n]$ w.p. $\approx 1/n^{1/\alpha}$.

2. We create S_i and T_i such that $S_i \cup T_i = Z_i$:

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:
   - Each Z_i is a random set of size $\approx n - n^{(1-1/\alpha)}$ chosen from $[n]$.
   - Think of creating Z_i by (essentially) removing each element from $[n]$ w.p. $\approx 1/n^{1/\alpha}$.
2. We create S_i and T_i such that $S_i \cup T_i = Z_i$:
   - Each element $e \in Z_i$ goes to $\begin{cases} S_i & \text{w.p. } 1/3 \\ T_i & \text{w.p. } 1/3 \\ \text{both } S_i \text{ and } T_i & \text{o.w.} \end{cases}$.

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:
   - Each Z_i is a random set of size $\approx n - n^{(1-1/\alpha)}$ chosen from $[n]$.
   - Think of creating Z_i by (essentially) removing each element from $[n]$ w.p. $\approx 1/n^{1/\alpha}$.

2. We create S_i and T_i such that $S_i \cup T_i = Z_i$:
   - Each element $e \in Z_i$ goes to $\begin{cases} S_i & \text{w.p. } 1/3 \\ T_i & \text{w.p. } 1/3 \\ \text{both } S_i \text{ and } T_i & \text{o.w.} \end{cases}$.

This creates a No instance.

A Hard Input Distribution for SetCover

We construct Alice and Bob's input sets as follows:

1. Create m sets $Z_1, \ldots, Z_m$:
   - Each Z_i is a random set of size $\approx n - n^{(1-1/\alpha)}$ chosen from $[n]$.
   - Think of creating Z_i by (essentially) removing each element from $[n]$ w.p. $\approx 1/n^{1/\alpha}$.
2. We create S_i and T_i such that $S_i \cup T_i = Z_i$:
   - Each element $e \in Z_i$ goes to $\begin{cases} S_i & \text{w.p. } 1/3 \\ T_i & \text{w.p. } 1/3 \\ \text{both } S_i \text{ and } T_i & \text{o.w.} \end{cases}$.

To create a Yes instance, we choose $i^\star \in [m]$ uniformly at random and let $Z_i = [n]$.

A Hard Input Distribution for SetCover

OPT in Yes instances?

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances?

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. (Informal) Optimal solution either picks both S_i and T_i or neither of them.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. (Informal) Optimal solution either picks both S_i and T_i or neither of them.

- $S_i \cup T_i = Z_i$, hence covering everything except for $n^{1-1/\alpha}$ elements.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. (Informal) Optimal solution either picks both S_i and T_i or neither of them.

- $S_i \cup T_i = Z_i$, hence covering everything except for $n^{1-1/\alpha}$ elements.
- $S_i \cup T_j$ covers $\approx 8n/9$ elements as S_i and T_j are two independent random sets of size $\approx 2n/3$.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. W.p. $1 - o(1)$, no α-subsets of $Z_1, \ldots, Z_m$ can cover $[n]$.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. W.p. $1 - o(1)$, no α-subsets of $Z_1, \ldots, Z_m$ can cover $[n]$.

- The probability that a fixed element $e \in [n]$ is not covered by a fixed α-subset is: $\approx \left(1/n^{1/\alpha}\right)^{\alpha} \approx \frac{1}{n}$

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. W.p. $1 - o(1)$, no α-subsets of $Z_1, \ldots, Z_m$ can cover $[n]$.

- The probability that a fixed element $e \in [n]$ is not covered by a fixed α-subset is: $\approx \left(1/n^{1/\alpha}\right)^{\alpha} \approx \frac{1}{n}$
- The expected number of uncovered elements by any fixed α-subset is then ≈ 1.

A Hard Input Distribution for SetCover

OPT in Yes instances? 2; pick $S_{i^\star}$ and $T_{i^\star}$ as $S_{i^\star} \cup T_{i^\star} = Z_{i^\star} = [n]$.

OPT in No instances? $> 2\alpha$ w.p. $1 - o(1)$.

Claim. W.p. $1 - o(1)$, no α-subsets of $Z_1, \ldots, Z_m$ can cover $[n]$.

- The probability that a fixed element $e \in [n]$ is not covered by a fixed α-subset is: $\approx \left(1/n^{1/\alpha}\right)^{\alpha} \approx \frac{1}{n}$
- The expected number of uncovered elements by any fixed α-subset is then ≈ 1.
- Use some concentration result + union bound to finalize the claim.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. For a fixed $i \in [m]$, detecting whether $Z_i = [n]$ or $Z_i = [n] \setminus (n^{1-1/\alpha}$ random elements $)$, requires $\Omega(n^{1/\alpha})$ communication.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. For a fixed $i \in [m]$, detecting whether $Z_i = [n]$ or $Z_i = [n] \setminus (n^{1-1/\alpha}$ random elements $)$, requires $\Omega(n^{1/\alpha})$ communication.

- Intuitively, to "catch" any of the missing elements, Alice and Bob need to communicate $\Omega(n^{1/\alpha})$ elements.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. For a fixed $i \in [m]$, detecting whether $Z_i = [n]$ or $Z_i = [n] \setminus (n^{1-1/\alpha}$ random elements $)$, requires $\Omega(n^{1/\alpha})$ communication.

- Intuitively, to "catch" any of the missing elements, Alice and Bob need to communicate $\Omega(n^{1/\alpha})$ elements.
- Can be formalized using a reduction from the set disjointness problem.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. $\mathrm{CC}(\textsc{SetCover}) \geq m \times$ communication complexity of distinguishing $Z_i = [n]$ and $|Z_i| = n - n^{1-1/\alpha}$.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. $\text{CC}(\textsc{SetCover}) \geq m \times$ communication complexity of distinguishing $Z_i = [n]$ and $|Z_i| = n - n^{1-1/\alpha}$.

- The input consists of m pairs (S_i, T_i) and the index $i^\star$ is unknown to Alice and Bob.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. $\text{CC}(\textsc{SetCover}) \geq m \times$ communication complexity of distinguishing $Z_i = [n]$ and $|Z_i| = n - n^{1-1/\alpha}$.

- The input consists of m pairs (S_i, T_i) and the index $i^\star$ is unknown to Alice and Bob.
- They need to check each pair S_i and T_i separately.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. $\mathrm{CC}(\textsc{SetCover}) \geq m \times$ communication complexity of distinguishing $Z_i = [n]$ and $|Z_i| = n - n^{1-1/\alpha}$.

- The input consists of m pairs (S_i, T_i) and the index $i^\star$ is unknown to Alice and Bob.
- They need to check each pair S_i and T_i separately.
- Can be formalized using information complexity and a direct sum-style argument.

The Lower Bound for Set Cover on $\mathcal{D}$

Why distinguishing between Yes and No instances is hard?

Claim. $\mathrm{CC}(\textsc{SetCover}) \geq m \times$ communication complexity of distinguishing $Z_i = [n]$ and $|Z_i| = n - n^{1-1/\alpha}$.

- The input consists of m pairs (S_i, T_i) and the index $i^\star$ is unknown to Alice and Bob.
- They need to check each pair S_i and T_i separately.
- Can be formalized using information complexity and a direct sum-style argument.
- There are some subtle technical challenges in applying this idea!

The Lower Bound for SetCover: Wrapup

Distinguishing between Yes and No instances of $\mathcal{D}$ requires

$$m \cdot \widetilde{\Omega}(n^{1/\alpha}) = \widetilde{\Omega}(mn^{1/\alpha})$$

bits of communication.

The Lower Bound for SetCover: Wrapup

Distinguishing between Yes and No instances of $\mathcal{D}$ requires

$$m \cdot \widetilde{\Omega}(n^{1/\alpha}) = \widetilde{\Omega}(mn^{1/\alpha})$$

bits of communication.

This implies a lower bound of $\widetilde{\Omega}\left(\frac{1}{p} \cdot mn^{1/\alpha}\right)$ on the space complexity of p-pass α-approximation streaming algorithm for set cover over adversarialy ordered streams.

The Lower Bound for SetCover: Wrapup

Distinguishing between Yes and No instances of $\mathcal{D}$ requires

$$m \cdot \widetilde{\Omega}(n^{1/\alpha}) = \widetilde{\Omega}(mn^{1/\alpha})$$

bits of communication.

This implies a lower bound of $\widetilde{\Omega}\left(\frac{1}{p} \cdot mn^{1/\alpha}\right)$ on the space complexity of p-pass α-approximation streaming algorithm for set cover over adversarialy ordered streams.

Some additional steps are required to extend this lower bound to random order streams.

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

This fully resolves the space-approximation tradeoff for multi-pass streaming algorithms.

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

This fully resolves the space-approximation tradeoff for multi-pass streaming algorithms.

Open question: How many passes do we need to obtain the optimal space complexity for α-approximation?

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

This fully resolves the space-approximation tradeoff for multi-pass streaming algorithms.

Open question: How many passes do we need to obtain the optimal space complexity for α-approximation?

- Best known upper bound is $O(\alpha)$ passes [Har-Peled et al., 2016].

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

This fully resolves the space-approximation tradeoff for multi-pass streaming algorithms.

Open question: How many passes do we need to obtain the optimal space complexity for α-approximation?

- Best known upper bound is $O(\alpha)$ passes [Har-Peled et al., 2016].
- We know it cannot be one pass [Assadi et al., 2016].

Summary

For the multi-pass streaming set cover problem:

$\Theta(mn^{1/\alpha})$ space is both sufficient and necessary for obtaining an α-approximation.

This fully resolves the space-approximation tradeoff for multi-pass streaming algorithms.

Open question: How many passes do we need to obtain the optimal space complexity for α-approximation?

- Best known upper bound is $O(\alpha)$ passes [Har-Peled et al., 2016].
- We know it cannot be one pass [Assadi et al., 2016].
- Conjectured by [Har-Peled et al., 2016] that $\Theta(\alpha)$ is tight.

Assadi, S., Khanna, S., and Li, Y. (2016).
Tight bounds for single-pass streaming complexity of the set cover problem.
In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 698–711.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. (2014).
Streaming submodular maximization: massive data summarization on the fly.
In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 671–680.

Bateni, M., Esfandiari, H., and Mirrokni, V. S. (2016).
Almost optimal streaming algorithms for coverage problems.
CoRR, abs/1610.08096.

Chakrabarti, A. and Wirth, A. (2016).
Incidence geometries and the pass complexity of semi-streaming set cover.
In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1365–1373.

Cormode, G., Karloff, H. J., and Wirth, A. (2010).
Set cover algorithms for very large datasets.
In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 479–488.

Demaine, E. D., Indyk, P., Mahabadi, S., and Vakilian, A. (2014).
On streaming and communication complexity of the set cover problem.

In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 484–498.

Dinur, I. and Steurer, D. (2014).
Analytical approach to parallel repetition.
In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633.

Emek, Y. and Rosén, A. (2014).
Semi-streaming set cover - (extended abstract).
In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 453–464.

Feige, U. (1998).
A threshold of ln n for approximating set cover.
J. ACM, 45(4):634–652.

Gomes, F. C., de Meneses, C. N., Pardalos, P. M., and Viana, G. V. R. (2006).
Experimental analysis of approximation algorithms for the vertex cover and set covering problems.
Computers & OR, 33(12):3520–3534.

Grossman, T. and Wool, A. (1997).
Computational experience with approximation algorithms for the set covering problem.
European Journal of Operational Research, 101(1):81–92.

Har-Peled, S., Indyk, P., Mahabadi, S., and Vakilian, A. (2016).
Towards tight bounds for the streaming set cover problem.
In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 371–383.

Indyk, P., Mahabadi, S., and Vakilian, A. (2015).
Towards tight bounds for the streaming set cover problem.

CoRR, abs/1509.00118.

Johnson, D. S. (1974).
Approximation algorithms for combinatorial problems.
J. Comput. Syst. Sci., 9(3):256–278.

Karp, R. M. (1972).
Reducibility among combinatorial problems.
In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103.

Lund, C. and Yannakakis, M. (1994).
On the hardness of approximating minimization problems.
J. ACM, 41(5):960–981.

McGregor, A. and Vu, H. T. (2016).
Better streaming algorithms for the maximum coverage problem.
CoRR, abs/1610.06199. To appear in ICDT (2017).

Moshkovitz, D. (2015).
The projection games conjecture and the np-hardness of ln n-approximating set-cover.
Theory of Computing, 11:221–235.

Saha, B. and Getoor, L. (2009).
On maximum coverage in the streaming model & application to multi-topic blog-watch.
In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 697–708.

Slavík, P. (1997).
A tight analysis of the greedy algorithm for set cover.
J. Algorithms, 25(2):237–254.