**Noname manuscript No.**
(will be inserted by the editor)

# Better Streaming Algorithms for the Maximum Coverage Problem

**Andrew McGregor · Hoa T. Vu**

**Abstract** We study the classic NP-Hard problem of finding the maximum $k$-set coverage in the data stream model: given a set system of $m$ sets that are subsets of a universe $\{1, \ldots, n\}$, find the $k$ sets that cover the most number of distinct elements. The problem can be approximated up to a factor $1 - 1/e$ in polynomial time. In the streaming-set model, the sets and their elements are revealed online. The main goal of our work is to design algorithms, with approximation guarantees as close as possible to $1 - 1/e$, that use sublinear space $o(mn)$. Our main results are:

- Two $(1 - 1/e - \epsilon)$ approximation algorithms: One uses $O(\epsilon^{-1})$ passes and $\tilde{O}(\epsilon^{-2}k)$ space whereas the other uses only a single pass but $\tilde{O}(\epsilon^{-2}m)$ space. $\tilde{O}(\cdot)$ suppresses polylog factors.
- We show that any approximation factor better than $(1-(1-1/k)^k) \approx 1-1/e$ in constant passes requires $\Omega(m)$ space for constant $k$ even if the algorithm is allowed unbounded processing time. We also demonstrate a *single-pass*, $(1 - \epsilon)$ approximation algorithm using $\tilde{O}(\epsilon^{-2}m \cdot \min(k, \epsilon^{-1}))$ space.

We also study the maximum $k$-vertex coverage problem in the dynamic graph stream model. In this model, the stream consists of edge insertions and deletions of a graph on $N$ vertices. The goal is to find $k$ vertices that cover the most number of distinct edges.

- We show that any constant approximation in constant passes requires $\Omega(N)$ space for constant $k$ whereas $\tilde{O}(\epsilon^{-2}N)$ space is sufficient for a $(1 - \epsilon)$ approximation and arbitrary $k$ in a single pass.

This work was supported by NSF Awards CCF-0953754, IIS-1251110, CCF-1320719, and a Google Research Award.

Andrew McGregor
University of Massachusetts E-mail: mcgregor@cs.umass.edu

Hoa T. Vu
University of Massachusetts E-mail: hvu@cs.umass.edu

– For regular graphs, we show that $\tilde{O}(\epsilon^{-3}k)$ space is sufficient for a $(1-\epsilon)$ approximation in a single pass. We generalize this to a $(\kappa-\epsilon)$ approximation when the ratio between the minimum and maximum degree is bounded below by $\kappa$.

## 1 Introduction

The *maximum set coverage problem* is a classic NP-Hard problem that has a wide range of applications including facility and sensor allocation [36], information retrieval [7], influence maximization in marketing strategy design [32], and the blog monitoring problem where we want to choose a small number of blogs that cover a wide range of topics [44]. In this problem, we are given a set system of $m$ sets that are subsets of a universe $[n] := \{1, \ldots, n\}$. The goal is to find the $k$ sets whose union covers the largest number of distinct elements. For example, in the application considered by Saha and Getoor [44], the universe corresponds to $n$ topics of interest to a reader, each subset corresponds to a blog that covers some of these topics, and the goal is to maximize the number of topics that the reader learns about if she can only choose $k$ blogs.

It is well-known that the greedy algorithm, which greedily picks the set that covers the most number of uncovered elements, is a $1-1/e$ approximation. Furthermore, unless $P = NP$, this approximation factor is the best possible [27].

The *maximum vertex coverage problem* is a special case of this problem in which the universe corresponds to the edges of a given graph and there is a set corresponding to each node of the graph that contains the edges that are incident to that node. For this problem, algorithms based on linear programming achieve a 3/4 approximation for general graphs [1] and a 8/9 approximation for bipartite graphs [17]. Assuming $P \neq NP$, there does not exist a polynomial-time approximation scheme. Recent work has focused on finding purely combinatorial algorithms for this problem [16].

*Streaming Algorithms.* Unfortunately, for both problems, the aforementioned greedy and linear programming algorithms scale poorly to massive data sets. This has motivated a significant research effort in designing algorithms that could handle large data in modern computation models such as the data stream model and the MapReduce model [12, 37]. In the data stream model, the $k$-set coverage problem and the related set cover problem have received a lot of attention in recent research [9, 11, 20, 26, 29, 49].

Two variants of the data stream model are relevant to our work. In the *streaming-set model* [26, 28, 34, 43, 44, 48], the stream consists of $m$ sets $S_1, \ldots, S_m$ and each $S_i$ is encoded as the list of elements in that set along with a unique ID for the set. For simplicity, we assume that $\mathrm{ID}(S_i) = i$. In the dynamic graph stream model [3–6, 10, 15, 23, 28, 30, 31, 35, 39, 40], relevant to the maximum vertex coverage problem, the stream consists of insertions and deletions of edges of the underlying graph. For a recent survey of research in graph streaming, see [38]. Note that any algorithm for the dynamic graph

stream model can also be used in the streaming-set model; the streaming-set model is simply a special case in which there is no deletion and edges are grouped by endpoint.

### 1.1 Related Work

*Maximum Set Coverage.* Saha and Getoor [44] gave a swap based 1/4 approximation algorithm that uses a single pass and $\tilde{O}(kn)$ space. At any point, their algorithm stores $k$ sets explicitly in the memory as the current solution. When a new set arrives, based on a specific rule, their algorithm either swaps it with the set with the least contribution in the current solution or does nothing and moves on to the next set in the stream. Subsequently, Ausiello et al. [11] gave a slightly different swap based algorithm that also finds a 1/4 approximation using one pass and the same space. Yu and Yuan [49] claimed an $\tilde{O}(n)$ space, single-pass algorithm with an approximation factor around 0.3 based on the aid of computer simulation.

Recently, Badanidiyuru et al. [12] gave a generic single-pass algorithm for maximizing a monotone submodular function on the stream's items subject to the cardinality constraint that at most $k$ objects are selected. Their algorithm guarantees a $1/2 - \epsilon$ approximation. At a high level, based on a rule that is different from [11, 44] and a guess of the optimal value, their algorithm decides if the next item (which is a set in our case) is added to the current solution. The algorithm stops when it reaches the end of the stream or when $k$ items have been added to the solution. In the maximum set coverage problem, the rule requires knowing the coverage of the current solution. As a result, a careful adaptation to the maximum set coverage problem uses $\tilde{O}(\epsilon^{-1}n)$ space. For constant $\epsilon$, this result directly improves upon [11, 44]. Subsequently, Chekuri et al. [21] extended this work to non-monotone submodular function maximization under constraints beyond cardinality.

The set cover problem, which is closely related to the maximum set coverage problem, has been studied in [9, 20, 26, 29, 44]. See [9] for a comprehensive summary of results and discussion.

*Maximum Vertex Coverage.* The streaming maximum vertex coverage problem was studied by Ausiello et al. [11]. They first observed that simply outputting the $k$ vertices with highest degrees is a 1/2 approximation; this can easily be done in the streaming-set model. The main results of their work were $\tilde{O}(kN)$-space algorithms that have better approximation for special types of graph. Their results include a 0.55 approximation for regular graphs and a 0.6075 approximation for regular bipartite graphs. Note that their paper only considered the streaming-set model whereas our results for maximum vertex coverage will consider the more challenging dynamic graph stream model.

### 1.2 Our Contributions

*Maximum $k$-set coverage.* Our main goal is to achieve the $1-1/e$ approximation that is possible in the non-streaming or offline setting.

- We present polynomial time data stream algorithms that achieve a $1-1/e-\epsilon$ approximation for arbitrarily small $\epsilon$. The first algorithm uses one pass and $\tilde{O}(\epsilon^{-2}m)$ space whereas the second algorithm uses $O(\epsilon^{-1})$ passes and $\tilde{O}(\epsilon^{-2}k)$ space. We consider both algorithms to be pass efficient but the second algorithm uses much less space at the cost of using more than one pass. We note that storing the solution itself requires $\Omega(k)$ space. Thus, we consider $\tilde{O}(\epsilon^{-2}k)$ space to be surprisingly space efficient.
- For constant $k$, we show that $\Omega(m)$ space is required by any constant pass (randomized) algorithm to achieve an approximation factor better than $1-(1-1/k)^k$ with probability at least 0.99; this holds even if the algorithm is permitted exponential time. To the best of our knowledge, this is the first non-trivial space lower bound for this problem. However, with exponential time and $\tilde{O}(\epsilon^{-2}m\cdot\min(k,\epsilon^{-1}))$ space we observe that a $1-\epsilon$ approximation is possible in a single pass.

For a slightly worse approximation, a $1/2-\epsilon$ approximation in one pass can be achieved using $\tilde{O}(\epsilon^{-3}k)$ space. This follows by building on the result of Badanidiyuru et al. [12]. However, we provide a simpler algorithm and analysis.

Our approach generalizes to the group cardinality constraint in which there are $\ell$ groups and only $k_i$ sets from group $i$ can be selected. We give a $1/(\ell+1)-\epsilon$ approximation that uses a single pass and $\tilde{O}(\epsilon^{-3}k)$ space where $k=k_1+\ldots+k_\ell$. If $O(\epsilon^{-1}\log k)$ passes are permitted, then we could achieve a $1/2-\epsilon$ approximation by adapting the greedy analysis in [22] to our framework.

Finally, we design a $1/3-\epsilon$ approximation algorithm for the budgeted maximum set coverage problem using one pass and $\tilde{O}(\epsilon^{-1}(n+m))$ space. In this version, each set $S$ has a cost $w_S$ in the interval $[0,L]$. The goal is to find a collection of sets whose total cost does not exceed $L$ that cover the most number of distinct elements. Khuller et al. [33] presented a polynomial time and $1-1/e$ approximation algorithm based on the greedy algorithm and an enumeration technique. Our results are summarized in Figure 1.

Shortly after our original submission, in an independent work, Bateni et al. [14] also presented a polynomial-time, single-pass, $\tilde{O}(\epsilon^{-3}m)$ space algorithm that finds a $1-1/e-\epsilon$ approximation for the maximum $k$-set coverage problem. Furthermore, given unlimited post-processing time, their results also imply a $1-\epsilon$ approximation using a single-pass and $\tilde{O}(\epsilon^{-3}m)$ space. This extension to $1-\epsilon$ approximation is also possible with our approach; see the end of 2.1 for details. We also note that our approach also works in their *edge arrival* model in which the stream reveals the set-element relationships one at a time.

Recently, Assadi proved a space lower bound $\Omega(\epsilon^{-2}m)$ for any $1-\epsilon$ approximation for constant $k$ [8].

*Maximum $k$-vertex coverage.* Compared to the most relevant previous work [11], we study this problem in a more general model, i.e., the dynamic graph stream

| Bound | No. of passes | Space | Approximation | Constraint |
|---|---|---|---|---|
| U | $O\left(\epsilon^{-1}\right)$ | $\tilde{O}(\epsilon^{-2}k)$ | $1-1/e-\epsilon$ | C |
| U | 1 | $\tilde{O}(\epsilon^{-3}k)$ | $1/2-\epsilon$ | C |
| U | 1 | $\tilde{O}(\epsilon^{-2}m)$ | $1-1/e-\epsilon$ | C |
| U | 1 | $\tilde{O}(\epsilon^{-2}m \cdot \min(k, \epsilon^{-1}))$ | $1-\epsilon$ | C |
| L | Constant | $\Omega(mk^{-2})$ | $1-(1-1/k)^k+\epsilon$ | C |
| U | 1 | $\tilde{O}(\epsilon^{-3}k)$ | $1/(\ell+1)-\epsilon$ | G |
| U | $O(\epsilon^{-1}\log k)$ | $\tilde{O}(\epsilon^{-2}k)$ | $1/2-\epsilon$ | G |
| U | 1 | $\tilde{O}(\epsilon^{-1}(n+m))$ | $1/3-\epsilon$ | B |

**Fig. 1** Summary of results for MaxSetCoverage, U: upper bound, L: lower bound, C: cardinality, B: budget, G: group cardinality

| Bound | No. of passes | Space | Approximation |
|---|---|---|---|
| U | 1 | $\tilde{O}(\epsilon^{-2}N)$ | $1-\epsilon$ |
| U | 1 | $\tilde{O}(\epsilon^{-3}k)$ | $\kappa-\epsilon$ |
| L | 1 | $\Omega(N\kappa^3/k)$ | $\kappa+\epsilon$ |

**Fig. 2** Summary of results for MaxVertexCoverage, U: upper bound, L: lower bound, $\kappa$ is ratio of lowest degree to highest degree.

model. We manage to achieve a better approximation and space complexity for general graphs even when comparing to their results for special types of graph. Our results are summarized in Figure 2. In particular, we show that

- $\tilde{O}(\epsilon^{-2}N)$ space is sufficient for a $1-\epsilon$ approximation (or a $3/4-\epsilon$ approximation if restricted to polynomial time) and arbitrary $k$ in a single pass. The algorithms in [11] use $\tilde{O}(kN)$ space and achieve an approximation worse than 0.61 even for special graphs.
- Any constant approximation in constant passes requires $\Omega(N)$ space for constant $k$.
- For regular graphs, we show that $\tilde{O}(\epsilon^{-3}k)$ space is sufficient for $1-\epsilon$ approximation in a single pass. We generalize this to an $\kappa-\epsilon$ approximation when the ratio between the minimum and maximum degree is bounded below by $\kappa$. We also extend this result to hypergraphs.

*Our techniques.* On the algorithmic side, our basic approach is a "guess, subsample, and verify" framework. At a high level, suppose we design a streaming algorithm for approximating the maximum $k$-set coverage that assumes a priori knowledge of a good guess of the optimal coverage. We show that it is a) possible to run same algorithm on a subsampled universe defined by a carefully chosen hash function and b) remove the assumption that a good guess was already known.

If the guess is at least nearly correct, running the algorithm on the subsampled universe results in a small space complexity. However, there are two main challenges. First, an algorithm instance with a wrong guess could use too much space. We simply terminate those instances. The second issue is

more subtle. Because the hash function is not fully independent, we appeal to a special version Chernoff bound. The bound needs not guarantee a good approximation unless the guess is near-correct. To this end, we use the $F_0$ estimation algorithm to verify the coverage of the solutions. Finally, we return the solution with maximum estimate coverage. This framework allows us to restrict the analysis solely to the near-correct guess. The analysis is, therefore, significantly simpler.

Some of our other algorithmic ideas are inspired by previous works. The "thresholding greedy" technique was inspired by [13,20,25]. However, the analysis is different for our problem. Furthermore, to optimize the number of passes, we rely on new observations.

Another algorithmic idea in designing one-pass space-efficient algorithm is to treat the sets differently based on their contributions. During the stream, we immediately add the sets with large contributions to the solution. We store the contribution of each remaining sets explicitly and solve the remaining problem offline. Har-Peled et al. [29] devised a somewhat similar strategy but the details are different.

For the maximum $k$-vertex coverage problem, we show that simply running the streaming cut-sparsifier algorithm is sufficient and optimal up to a polylog factor. The novelty is to treat it as an interesting corner case of a more space-efficient algorithm for near regular graphs, i.e., $\kappa$ is bounded below.

One of the novelties is proving the lower bound via a randomized reduction from the $k$-party set disjointness problem.

*Comparison to the conference publication.* This is an extended and revised version of a preliminary version in ICDT 2017 [41]. In this version, we present the single-pass, $\tilde{O}(\epsilon^{-3}m)$ space and $1-\epsilon$ approximation algorithm that was briefly mentioned in [41]. Furthermore, we provide two new algorithms for the maximum set coverage problem under the group cardinality constraint in Section 2.4.3.

## 2 Algorithms for maximum $k$-set coverage

In this section, we design various algorithms for approximating MaxSetCoverage in the data stream model. Our main algorithmic results in this section are two $1-1/e-\epsilon$ approximation algorithms. The first algorithm uses one pass and $\tilde{O}(\epsilon^{-2}m)$ space whereas the second algorithm uses $O(\epsilon^{-1})$ passes and $\tilde{O}(\epsilon^{-2}k)$ space. We also briefly explore some other trade-offs in a subsequent subsection.

*Notation.* If $\mathcal{A}$ is a collection of sets, then $\mathcal{C}(\mathcal{A})$ denotes the union of these sets.

### 2.1 $(1-1/e-\epsilon)$ approximation in one pass and $\tilde{O}(\epsilon^{-2}m)$ space

*Approach.* The algorithm adds sets to the current solution if the number of new elements they cover exceeds some threshold. The basic algorithm relies on an

estimate $z$ of the optimal coverage OPT. The threshold for including a new set in the solution is that it covers at least $z/k$ new elements. Unfortunately, this threshold is too high to ensure that we selected sets that achieve the required $1 - 1/e - \epsilon$ approximation and we may want to revisit adding a set, say $S$, that was not added when it first arrived. To facilitate this, we will explicitly store the subset of $S$ that were uncovered when $S$ arrived in a collection of sets $\mathcal{W}$. Because $S$ was not added immediately, we know that this subset is not too large. At the end of the pass, we continue augmenting out current solutions using the collection $\mathcal{W}$.

*Technical Details.* For the time being, we suppose that the algorithm is provided with an estimate $z$ such that $\mathrm{OPT} \leq z \leq 4\,\mathrm{OPT}$. We will later remove this assumption. The algorithm uses $C$ to keep track of the elements that have been covered so far. Upon seeing a new set $S$, the algorithm stores $S \setminus C$ explicitly in $\mathcal{W}$ if $S$ covers few new elements. Otherwise, the algorithm adds $S$ to the solution and updates $C$ immediately. At the end of the stream, if there are fewer than $k$ sets in the solution, we use the greedy approach to find the remaining sets from $\mathcal{W}$ .

The basic algorithm maintains $I \subseteq [m]$, $C \subseteq [n]$ where $I$ corresponds to the ID's of the (at most $k$) sets in the current solution and $C$ is the the union of the corresponding sets. We also maintain a collection of sets $\mathcal{W}$ described above. The algorithm proceeds as follows:

1. Initialize $C = \emptyset$, $I = \emptyset$, $\mathcal{W} = \emptyset$.
2. For each set $S$ in the stream:
   (a) If $|S \setminus C| < z/k$ then $\mathcal{W} \leftarrow \mathcal{W} \cup \{S \setminus C\}$.
   (b) If $|S \setminus C| \geq z/k$ and $|I| < k$, then $I \leftarrow I \cup \{ID(S)\}$ and $C \leftarrow C \cup S$.
3. Post-processing: Greedily add $k - |I|$ sets from $\mathcal{W}$ and update $I$ and $C$ appropriately.

**Lemma 1.** *There exists a single-pass, $O\left(k \log m + mz/k \cdot \log n\right)$-space algorithm that finds a $1 - 1/e$ approximation of* MaxSetCoverage.

*Proof.* We observe that storing the set of covered elements $C$ requires at most $\mathrm{OPT} \log n = O(z \log n)$ bits of space. For each set $S$ such that $S \setminus C$ is stored explicitly in $\mathcal{W}$, we need $O\left(z/k \cdot \log n\right)$ bits of space. Storing $I$ requires $O(k \log m)$ space. Thus, the algorithm uses the space as claimed.

After the algorithm added the $i$th set $S$ to the solution, let $a_i$ be the number of new elements that $S$ covers and $b_i$ be the total number of covered elements so far. Furthermore, for $i > 0$, let $c_i = \mathrm{OPT} - b_i$. Define $a_0 := b_0 := 0$ and $c_0 := \mathrm{OPT}$. At the end of the stream, suppose $|I| = j$. Then, $c_j \leq \mathrm{OPT} - zj/k \leq \mathrm{OPT}(1 - j/k) \leq \mathrm{OPT}(1 - 1/k)^j$. The last inequality holds when $k \geq 2$ and $j$ is a non-negative integer. The case $k = 1$ is trivial since we can simply find the largest set in $\tilde{O}(1)$ space.

Now, we consider the sets that were added in post-processing. We then proceed with the usual inductive argument to show that $c_i \leq (1 - 1/k)^i\,\mathrm{OPT}$

for $i > j$. Before the algorithm added the $(i+1)$th set for $j \leq i \leq k-1$, there must be a set that covers at least $c_i/k$ new elements. Therefore,

$$c_{i+1} = c_i - a_{i+1} \leq c_i(1-1/k) \leq \mathrm{OPT}(1-1/k)^{i+1} \ .$$

The approximation follows since $c_k \leq \mathrm{OPT}(1-1/k)^k \leq 1/e \cdot \mathrm{OPT}$. □

Following the approach outlined in Section 2.3 we may assume $z = O(\epsilon^{-2} k \log m)$ and that $\mathrm{OPT} \leq z \leq 4\,\mathrm{OPT}$.

**Theorem 2.** *There exists a single-pass, $\tilde{O}(\epsilon^{-2}m)$ space algorithm that finds a $1-1/e-\epsilon$ approximation of* MaxSetCoverage *with high probability.*

*Better approximation using more space and unlimited post-processing time.* We observe that a slight modification of the above algorithm can be used to attain a $1-1/(4b)$ approximation for any $b > 1$ if we are permitted unlimited post-processing time and an extra factor of $b$ in the space use. Specifically, we increase the threshold for when to add a set immediately to the solution from $z/k$ to $bz/k$ and then find the optimal collection of $k-|I|$ sets from $\mathcal{W}$ to add in post-processing. It is immediate that this algorithm uses $O(k \log m + mbz/k \cdot \log n)$ space.

Suppose a collection $y$ sets $\mathcal{S}_1$ were added during the stream. These $y$ sets cover

$$|\,\mathcal{C}(\mathcal{S}_1)| \geq y \cdot \frac{bz}{k} \geq \mathrm{OPT} \cdot \frac{yb}{k}$$

elements. On the other hand, the collection of sets $\mathcal{S}_2$ selected in post-processing covers at least $\frac{k-y}{k} \cdot (\mathrm{OPT} - |\,\mathcal{C}(\mathcal{S}_1)|)$ new elements. Then,

$$\begin{aligned}
|\mathcal{C}(\mathcal{S}_1 \cup \mathcal{S}_2)| &\geq |\,\mathcal{C}(\mathcal{S}_1)| + \frac{k-y}{k} \cdot (\mathrm{OPT} - |\,\mathcal{C}(\mathcal{S}_1)|) \\
&= \left(1 - \frac{y}{k}\right) \mathrm{OPT} + \frac{y}{k} \cdot |\,\mathcal{C}(\mathcal{S}_1)| \\
&\geq \left(1 - \frac{y}{k}\right) \mathrm{OPT} + \frac{y}{k}\,\mathrm{OPT} \cdot \frac{yb}{k} \\
&= \mathrm{OPT}\left(1 - \frac{y}{k} + \left(\frac{y}{k}\right)^2 b\right) \\
&\geq \mathrm{OPT}\left(1 - \frac{1}{4b}\right)
\end{aligned}$$

where the last inequality follows by minimizing over $y$. Hence, we obtain a $1-\epsilon$ approximation by setting $b = 4/\epsilon$.

**Theorem 3.** *There exists a single-pass, $\tilde{O}(\epsilon^{-3}m)$ space algorithm that finds a $1-\epsilon$ approximation of* MaxSetCoverage *with high probability.*

### 2.2 $(1 - 1/e - \epsilon)$ approximation in $O(\epsilon^{-1})$ passes and $\tilde{O}(\epsilon^{-2}k)$ space

*Approach.* Our second algorithm is based on the standard greedy approach but instead of adding the set that increases the coverage of the current solution the most at each set, we add a set if the number of new elements covered by this set exceeds a certain threshold. This threshold decreases with each pass in such a way that after only $O(\epsilon^{-1})$ passes, we have a good approximate solution but the resulting algorithm may use too much space. We will fix this by first randomly subsampling each set at different rates and running multiple instantiations of the basic algorithm corresponding to different rates of subsampling.

The basic "decreasing threshold" approach has been used before in different contexts [13, 20, 25]. The novelty of our approach is in implementing this approach such that the resulting algorithm uses small space and a small number of passes. For example, a direct implementation of the approach by Badanidiyuru and Vondrák [13] in the streaming model may require $O(\epsilon^{-1}\log(m/\epsilon))$ passes and $O(n)$ space[1].

*Technical Details.* We will assume that we are given an estimate $z$ of OPT such that $\text{OPT} \leq z \leq 4\,\text{OPT}$. We start by designing a $(1 - 1/e - \epsilon)$ approximation algorithm that uses $\tilde{O}(k + z)$ space and $O(\epsilon^{-1})$ passes. We will subsequently use a sampling approach to reduce the space to $\tilde{O}(\epsilon^{-2}k)$.

As with the previous algorithm, the basic algorithm in this section also maintains $I \subseteq [m]$, $C \subseteq [n]$ where $I$ corresponds to the ID's of the (at most $k$) sets in the current solution and $C$ is the the union of the corresponding sets. The algorithm proceeds as follows:

1. Initialize $C = \emptyset$ and $I = \emptyset$
2. For $j = 1$ to $1 + \lceil \log_\alpha(4e) \rceil$ where $\alpha = 1 + \epsilon$:
   (a) Make a pass over the stream. For each set $S$ in the stream:
      i. If $|I| < k$ and $|S \setminus C| \geq \frac{z}{k(1+\epsilon)^{j-1}}$ then
      $I \leftarrow I \cup \{ID(S)\}$ and $C \leftarrow C \cup S$ .

**Lemma 4.** *There exists an $O(\epsilon^{-1})$-pass, $O(k \log m + z \log n)$-space algorithm that finds a $1 - 1/e - \epsilon$ approximation of* MaxSetCoverage.

To analyze the algorithm, we introduce some notation. After the $i$th set was picked, let $a_i$ be the number of new elements covered by this set and let $b_i$ be the total number of covered elements so far. Furthermore, let $c_i = \text{OPT} - b_i$. We define $a_0 := 0$ and $b_0 := 0$.

**Lemma 5.** *Suppose the algorithm picks $k'$ sets. For $0 \leq i \leq k' - 1$, $a_{i+1} \geq c_i/(\alpha k)$.*

*Proof.* Suppose the algorithm added the $(i+1)$th set $S$ during the $j$th pass. Consider the set of covered elements $C$ just before the algorithm added the set $S$.

[1] Note that their work addressed the more general problem of maximizing sub-modular functions.

We first consider the case where $j = 1$. Then, the algorithm only adds $S$ if

$$|S \setminus C| \geq z/k \geq \mathrm{OPT}\,/k \geq c_i/k \geq c_i/(\alpha k)\ .$$

Now, we consider the case where $j > 1$. Note that just before the algorithm added $S$, there must exist a set $S'$ (which could be $S$) that had not been already added where $|S' \setminus C| \geq c_i/k$. This follows because the optimal collection of $k$ sets covers at least $c_i$ elements that are currently uncovered and hence one of these sets must cover at least $c_i/k$ new elements. But since $S'$ had not already been added, we know that $S'$ was not added during the first $j-1$ passes and thus, $|S' \setminus C| < z/(k\alpha^{j-2})$. Therefore,

$$z/(k\alpha^{j-2}) > |S' \setminus C| \geq c_i/k$$

and in particular, $z/(k\alpha^{j-1}) > c_i/(k\alpha)$. Since the algorithm picked $S$, we have

$$a_{i+1} = |S \setminus C| \geq z/(k\alpha^{j-1}) \geq c_i/(k\alpha)$$

as required. □

*Proof of Lemma 4.* It is immediate that the number of passes is $O(\epsilon^{-1})$. The algorithm needs to store the sets $I$ and $C$. Since $|C| \leq z$, the total space is $O(k \log m + z \log n)$.

To argue about the approximation factor, we first prove by induction that we always have $c_i \leq \left(1 - \frac{1}{\alpha k}\right)^i \mathrm{OPT}$ for $i \leq k'$. Trivially, $c_0 \leq \left(1 - \frac{1}{\alpha k}\right)^0 \mathrm{OPT}$. Suppose $c_i \leq (1 - \frac{1}{\alpha k})^i \,\mathrm{OPT}$. Then, according to Lemma 5, $a_{i+1} \geq c_i/(\alpha k)$. Thus,

$$c_{i+1} = c_i - a_{i+1} \leq c_i - \frac{c_i}{\alpha k} = c_i \left(1 - \frac{1}{\alpha k}\right) \leq \mathrm{OPT} \left(1 - \frac{1}{\alpha k}\right)^{i+1}\ .$$

Suppose the final solution contains $k$ sets. Then

$$c_k \leq \left(1 - \frac{1}{\alpha k}\right)^k \mathrm{OPT} \leq e^{-1/\alpha}\,\mathrm{OPT} \leq (1/e + \epsilon)\,\mathrm{OPT}\,.$$

As a result, the final solution covers $b_k = \mathrm{OPT} - c_k \geq (1 - 1/e - \epsilon)\,\mathrm{OPT}$ elements.

Suppose the collection of sets $\mathcal{S}$ chosen by the algorithm contains fewer than $k$ sets. We define $\tilde{S} := S \setminus \mathcal{C}(\mathcal{S})$ to be the set of elements in $S$ that are not covered by the final solution. For each set $S$ in the optimal solution $\mathcal{O}$, if $S$ is unpicked, then $|\tilde{S}| \leq z/(4ek)$. Therefore,

$$\begin{aligned}
\mathrm{OPT} &= \left| \bigcup_{S \in \mathcal{O}} (S \cap \mathcal{C}(\mathcal{S})) \right| + \left| \bigcup_{S \in \mathcal{O} \setminus \mathcal{S}} \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \sum_{S \in \mathcal{O} \setminus \mathcal{S}} \left|\tilde{S}\right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{z}{4e} \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{\mathrm{OPT}}{e}\ .
\end{aligned}$$

Hence, $|\mathcal{C}(\mathcal{S})| \geq (1 - 1/e)\,\mathrm{OPT}$. □

Following the approach outlined in Section 2.3 we may assume $z = O(\epsilon^{-2}k \log m)$ and that $\mathrm{OPT} \leq z \leq 4\,\mathrm{OPT}$.

**Theorem 6.** *There exists an $O(\epsilon^{-1})$-pass, $\tilde{O}(\epsilon^{-2}k)$ space algorithm that finds a $1 - 1/e - \epsilon$ approximation of* MaxSetCoverage *with high probability.*

### 2.3 Removing Assumptions via Guessing, Sampling, and Sketching

In this section, we address the fact that in the previous two sections we assumed a priori knowledge of a constant approximation of the maximum number of elements that could be covered and that this optimum was of size $O(\epsilon^{-2}k \log m)$.

Addressing both issues are interrelated and are based on a subsampling approach. The basic idea is to run the above algorithms on a new instance formed by removing occurrences of certain elements in $[n]$ from all the input sets. The goal is to reduce the maximum coverage to $\min(n, O(\epsilon^{-2}k \log m))$ while ensuring that a good approximation in the subsampled instance corresponds to a good approximation in the original instance. In the rest of this section we will assume that $k = o(\epsilon^2 n/\log m)$ since otherwise this bound is trivial.

In this section, we will need to use the following Chernoff bound for limited independent random variables.

**Theorem 7** (Schmidt et al. [45])**.** *Let $X_1, \ldots, X_n$ be boolean random variables. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathrm{E}[X]$. Suppose $\mu \leq n/2$. If $X_i$ are $\lceil \gamma\mu \rceil$-wise independent, then*

$$\Pr[|X - \mu| \geq \gamma\mu] \leq \exp\left(-\lfloor \min(\gamma, \gamma^2) \cdot \mu/3 \rfloor\right) .$$

*Subsampling.* Assume we know a value $v$ that satisfies $\mathrm{OPT}/2 \leq v \leq \mathrm{OPT}$. Let $c$ be some sufficiently large constant and set $\lambda = c\epsilon^{-2}k \log m$. Let $h : [n] \to \{0, 1\}$ be drawn from a family of $2\lambda$-wise independent hash functions where

$$p := \Pr[h(e) = 1] = \lambda/v.$$

The space to store $h$ is $\tilde{O}(\epsilon^{-2}k)$. For any set $S$ that is a subset of $[n]$, we define

$$S' := \{e \in S : h(e) = 1\}.$$

The next lemma and its corollary will allow us to argue that approximating the maximum coverage among the elements $\{e \in [n] : h(e) = 1\}$ gives only a slightly weaker approximation of the maximum coverage among the original set of elements.

**Lemma 8.** *With high probability[2], for all collections of $k$ sets $S_1, \ldots, S_k$ in the stream, $|S'_1 \cup \ldots \cup S'_k| = |S_1 \cup \ldots \cup S_k|p \pm \epsilon vp$ .*

[2] We consider $1 - 1/\mathrm{poly}(m)$ or $1 - 1/\mathrm{poly}(n)$ as high probability.

*Proof.* Fix any collection of $k$ sets $S_1, \ldots, S_k$. Let $D = |S_1 \cup \ldots \cup S_k|$ and $D' = |S'_1 \cup \ldots \cup S'_k|$. We first observe that since $k = o(\epsilon^2 n/\log m)$, we may assume that $\lambda = o(n)$.

$$\mu := \mathrm{E}\left[D'\right] = pD \leq p\,\mathrm{OPT} < 2pv = 2\lambda \leq n/2.$$

Appealing to the Chernoff bound with limited independence (Theorem 7) with the binary variables $X_i = 1$ if and only if $i \in S_1 \cup \ldots \cup S_k$ and $h(i) = 1$, i.e, $D' = \sum_{i=1}^{n} X_i$, we have

$$\Pr\left[|D' - \mu| \geq \epsilon vp\right] = \Pr\left[|D' - \mu| \geq \gamma Dp\right] \leq \exp\left(-\lfloor \min(\gamma, \gamma^2) \cdot \mu/3 \rfloor\right)$$

where $\gamma = \epsilon v/D$ since the hash function is $\lceil \gamma\mu \rceil = \lceil \epsilon vp \rceil$-wise independent. But note that

$$\begin{aligned}\exp\left(-\lfloor \min(\gamma, \gamma^2) \cdot \frac{\mu}{3} \rfloor\right) &= \exp\left(-\lfloor \min(1, \gamma) \cdot \frac{\epsilon vp}{3} \rfloor\right) \\ &\leq \exp\left(-\left\lfloor \frac{1}{2} \cdot \frac{ck \log m}{3} \right\rfloor\right) \leq \frac{1}{m^{10k}}\end{aligned}$$

where we use the fact that $\gamma = \epsilon v/D \geq \epsilon/2$ because $D \leq \mathrm{OPT} \leq 2v$. The lemma follows by taking the union bound over all $\binom{m}{k}$ collections of $k$ sets. □

In particular, the following corollary establishes that a $1/t$ approximation when restricted to elements in $\{e \in [n] : h(e) = 1\}$ yields a $(1/t - 2\epsilon)$ approximation and at most $p\,\mathrm{OPT}(1+\epsilon) = O(\epsilon^{-2} k \log m)$ of these elements can be covered by $k$ sets.

**Corollary 9.** *Let* $\mathrm{OPT}'$ *be optimal number of elements that can be covered from* $\{e \in [n] : h(e) = 1\}$*. Then,*

$$p\,\mathrm{OPT}(1+\epsilon) \geq \mathrm{OPT}' \geq p\,\mathrm{OPT}(1-\epsilon)$$

*Furthermore if* $U_1, \ldots, U_k$ *satisfies* $|U'_1 \cup \ldots \cup U'_k| \geq p\,\mathrm{OPT}(1-\epsilon)/t$ *for* $t \geq 1$ *then*

$$|U_1 \cup \ldots \cup U_k| \geq \mathrm{OPT}(1/t - 2\epsilon)\ .$$

*Proof.* The fact that $\mathrm{OPT}' \geq p\,\mathrm{OPT}(1-\epsilon)$ follows by applying Lemma 8 to the optimal solution. According to Lemma 8, for all collections of $k$ sets $U_1, \ldots, U_k$, we have

$$|U'_1 \cup \ldots \cup U'_k| = |U_1 \cup \ldots \cup U_k|p \pm \epsilon vp \leq p\,\mathrm{OPT}(1+\epsilon)$$

which implies the first inequality.

Now, suppose $|U'_1 \cup \ldots \cup U'_k| \geq p\,\mathrm{OPT}(1-\epsilon)/t$. Since $|U'_1 \cup \ldots \cup U'_k| - \epsilon vp \leq |U_1 \cup \ldots \cup U_k|p$, we deduce that $|U_1 \cup \ldots \cup U_k| \geq \mathrm{OPT}(1-\epsilon)/t - \epsilon v \geq \mathrm{OPT}(1/t - 2\epsilon)$. □

Hence, since we know $v$ such that $\mathrm{OPT}/2 \leq v \leq \mathrm{OPT}$, then we know that

$$(1-\epsilon)\lambda \leq \mathrm{OPT}' \leq 2(1+\epsilon)\lambda \tag{1}$$

with high probability according to Corollary 9. Then, by setting $z = 2(1+\epsilon)\lambda$, we ensure that $\mathrm{OPT}' \leq z \leq 4\,\mathrm{OPT}'$.

*Guessing $v$ and $F_0$ Sketching* We still need to address how to compute $v$ such that $\text{OPT}/2 \leq v \leq \text{OPT}$. The natural approach is to make $\lceil \log_2 n \rceil$ guesses for $v$ corresponding to $1, 2, 4, 8 \ldots$ since one of these will be correct.[3] We then perform multiple parallel instantiations of the algorithm corresponding to each guess. This increases the space by a factor of $O(\log n)$.

But how do we determine which instantiation corresponds to the correct guess? The most expedient way to deal with this question is to sidestep the issue as follows. Instantiations corresponding to guesses that are too small may find it is possible to cover $\omega(\epsilon^{-2} k \log m)$ elements so we will terminate any instantiation as soon as it covers more than $O(\epsilon^{-2} k \log m)$ elements. Note that by Corollary 9 and Equation 1, we will not terminate the instantiation corresponding to the correct guess.

Among the instantiations that are not terminated we simply return the best solution. To find the best solution we want to estimate $|\cup_{i \in I} S_i|$, i.e., the coverage of the corresponding sets *before* the subsampling. To compute this estimate in small space we can use the *$F_0$-sketching* technique. For the purposes of our application, we can summarize the required result as follows:

**Theorem 10** (Cormode et al. [24]). *There exists an $\tilde{O}(\epsilon^{-2} \log \delta^{-1})$-space algorithm that, given a set $S \subseteq [n]$, can construct a data structure $\mathcal{M}(S)$, called an $F_0$* sketch *of $S$, that has the property that the number of distinct elements in a collection of sets $S_1, S_2, \ldots, S_r$ can be approximated up to a $1 + \epsilon$ factor with probability at least $1 - \delta$ given the collection of $F_0$ sketches $\mathcal{M}(S_1), \mathcal{M}(S_2), \ldots, \mathcal{M}(S_r)$.*

For the algorithms in the previous section, we can maintain a sketch $\mathcal{M}(C)$ of the set of covered elements in $\tilde{O}(\epsilon^{-2} \log \delta^{-1})$ space and from this can estimate the desired coverage. We set $\delta \leftarrow \Theta(1/n \cdot \log n)$ so that coverages of all non-terminated instances are estimated up to a factor $(1 + \epsilon)$ with high probability.

## 2.4 Other Algorithmic Results

In this final subsection, we briefly review some other algorithmic results for MaxSetCoverage, either with different trade-offs or for a "budgeted" version of the problem.

### *2.4.1 $(1 - \epsilon)$ approximation in one pass and $\tilde{O}(\epsilon^{-2} mk)$ space*

In the previous subsection, we gave a single-pass $1 - 1/e - \epsilon$ approximation using $\tilde{O}(\epsilon^{-2} m)$ space. Here we observe that if we are permitted $\tilde{O}(\epsilon^{-2} mk)$ space and unlimited post-processing time then a $1 - \epsilon$ approximation can be achieved directly from the $F_0$ sketches.

[3] The number of guesses can be reduced to $\lceil \log_2 k \rceil$ if the size of the largest set is known since this gives a $k$ approximation of OPT. The size of the large set can be computed in one additional pass if necessary.

In particular, in one pass we construct the $F_0$ sketches of all $m$ sets, $\mathcal{M}(S_1), \ldots, \mathcal{M}(S_m)$ where the failure probability of the sketches is set to $\delta = 1/(nm^k)$. Thus, at the end of the stream, one can $1+\epsilon$ approximate the coverage $|S_{i_1} \cup \ldots \cup S_{i_k}|$ for each collection of $k$ sets $S_{i_1}, \ldots, S_{i_k}$ with probability at least $1 - 1/(nm^k)$. Since there are at most $\binom{m}{k} \leq m^k$ collections of $k$ sets, appealing to the union bound, we guarantee that the coverages of all of the collections of $k$ sets are preserved up to a $1+\epsilon$ factor with probability at least $1 - 1/n$. The space to store the sketches is $\tilde{O}(\epsilon^{-2}mk)$.

**Theorem 11.** *There exists a single-pass, $\tilde{O}(\epsilon^{-2}mk)$-space algorithm that finds a $1-\epsilon$ approximation of* MaxSetCoverage *with high probability .*

*2.4.2 $(1/2-\epsilon)$ approximation in one pass and $\tilde{O}(\epsilon^{-3}k)$ space*

We next observe that it is possible to achieve a $1/2-\epsilon$ approximation using a single pass and $\tilde{O}(\epsilon^{-3}k)$ space. Consider the following simple single-pass algorithm that uses an estimate $z$ of OPT such that $\text{OPT} \leq z \leq (1+\epsilon)\,\text{OPT}$. As with previous algorithms, the basic algorithm in this section also maintains $I \subseteq [m]$, $C \subseteq [n]$ where $I$ corresponds to the ID's of the (at most $k$) sets in the current solution and $C$ is the the union of the corresponding sets. The algorithm proceeds as follows:

1. Initialize $C = \emptyset$ and $I = \emptyset$.
2. For each set $S$ in the stream:
   (a) If $|S \setminus C| \geq z/(2k)$ and $|I| < k$ then $I \leftarrow I \cup \{ID(S)\}$ and $C \leftarrow C \cup S$.

The described algorithm is a $1/2-\epsilon$ approximation. To see this, if the solution consists of $k$ sets, then the final solution obviously covers at least $z/2 \geq \text{OPT}/2$ elements. Now we consider the case in which the collection of sets $\mathcal{S}$ chosen by the algorithm contains fewer than $k$ sets. We define $\tilde{S} := S \setminus \mathcal{C}(\mathcal{S})$ to be the set of elements in $S$ that are not covered by the final solution. For each set $S$ in the optimal solution $\mathcal{O}$, if $S$ is unpicked, then $|\tilde{S}| \leq z/(2k)$. Therefore,

$$\begin{aligned}
\text{OPT} &= \left| \bigcup_{S \in \mathcal{O}} (S \cap \mathcal{C}(\mathcal{S})) \right| + \left| \bigcup_{S \in \mathcal{O} \setminus \mathcal{S}} \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \sum_{S \in \mathcal{O} \setminus \mathcal{S}} \left| \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{z}{2} \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{\text{OPT}(1+\epsilon)}{2}
\end{aligned}$$

and thus $|\mathcal{C}(\mathcal{S})| \geq \frac{1-\epsilon}{2}\,\text{OPT}$.

We note that the above algorithm uses $O(k \log m + z \log n)$ space but we can use an argument similar to that used in Section 2.3 to reduce this to $\tilde{O}(\epsilon^{-3}k)$.

The only difference is since we need $z$ such that $\mathrm{OPT}' \leq z \leq (1+\epsilon)\,\mathrm{OPT}'$ we will guess $v$ in powers of $1+\epsilon/4$ and set $\lambda = 16c\epsilon^{-2}k\log m$. Then Eq. 1 becomes $(1-\epsilon/4)\lambda \leq \mathrm{OPT}' \leq (1+\epsilon/4)^2\lambda$ and hence $z = (1+\epsilon/4)^2\lambda$ is a sufficiently good estimate.

**Theorem 12.** *There exists a single-pass, $\tilde{O}(\epsilon^{-3}k)$ space algorithm that finds a $1/2-\epsilon$ approximation of* MaxSetCoverage *with high probability.*

### *2.4.3 Group Cardinality Constraint Maximum Coverage*

In this version, we consider a version of the problem where each set belongs to a group amongst $\ell$ groups $G_1, G_2, \ldots, G_\ell$ and we are allowed to pick at most $k_1$ sets from group $G_1$, $k_2$ sets from group $G_2$, and so on. A $1-1/e$ approximation is possible for the offline version of this problem via linear programming [2, 47]. Furthermore, Chekuri and Kumar showed that the offline greedy algorithm is guaranteed to return a $1/2$ approximation [22].

*Single-pass algorithm.* We first observe that by simply applying the previous $1/2-\epsilon$ approximation algorithm for each group and returning the best solution, we obtain a $(1/2-\epsilon)/\ell$ approximation. The main idea for the improved algorithm is to set a threshold for when to add a set that depends on the group to which this set belongs.

We now present an algorithm that returns a $1/(\ell+1)-\epsilon$ approximation. The basic algorithm maintains the sets $I_i$ for each $i = 1, 2, \ldots, \ell$ where $I_i$ corresponds to the IDs of the sets from group $G_i$ that are in the current solution. Similar to previous algorithms, $C$ is used to keep track of the current coverage. Finally, the algorithm also uses an estimate $z$ of OPT such that $\mathrm{OPT} \leq z \leq (1+\epsilon)\,\mathrm{OPT}$. The detailed algorithm proceeds as follows:

1. Initialize $C = \emptyset$ and $I_i = \emptyset$ for each $i = 1, \ldots, \ell$.
2. For each set $S \in G_i$ in the stream:
   (a) If $|S \setminus C| \geq z/((\ell+1)k_i)$ and $|I_i| < k_i$ then
   $I_i \leftarrow I_i \cup \{ID(S)\}$ and $C \leftarrow C \cup S$ .

If there exists a group $G_i$ in which $k_i$ sets are selected, then it is clear that the solution covers at least $z/(\ell+1)$ elements. On the other hand, suppose that for all groups $G_i$, fewer than $k_i$ sets are selected. As before, $\mathcal{S}$ and $\mathcal{C}(\mathcal{S})$ are the collection of sets in the solution and their union respectively. Again, we define $\tilde{S} := S \setminus \mathcal{C}(\mathcal{S})$. Furthermore, let $\mathcal{O}_i$ denote the sets in $G_i$ that are

also in the optimal solution, i.e., $\mathcal{O}_i = \mathcal{O} \cap G_i$. We have

$$\begin{aligned}
\text{OPT} &= \left| \bigcup_{S \in \mathcal{O}} (S \cap \mathcal{C}(\mathcal{S})) \right| + \left| \bigcup_{S \in \mathcal{O} \setminus \mathcal{S}} \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \sum_{i=1}^{\ell} \sum_{S \in \mathcal{O}_i \setminus \mathcal{S}} \left| \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \sum_{i=1}^{\ell} \sum_{S \in \mathcal{O}_i \setminus \mathcal{S}} \frac{z}{(\ell+1)k_i} \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{\ell \cdot z}{\ell+1} \ .
\end{aligned}$$

Therefore,

$$|\mathcal{C}(\mathcal{S})| \geq \text{OPT} - \frac{\ell \cdot z}{\ell+1} \geq \left( \frac{1}{\ell+1} - \epsilon \right) \text{OPT} \ .$$

Let $k = k_1 + k_2 + \ldots + k_\ell$. The above algorithm uses $O(k \log m + z \log n)$ space but we can use an argument similar to that used in Sections 2.3 and 2.4.2 to reduce this to $\tilde{O}(\epsilon^{-3}k)$. We summarize the result as the following theorem.

**Theorem 13.** *There exists a single-pass, $\tilde{O}(\epsilon^{-3}k)$ space algorithm that finds a $1/(\ell+1)$ approximation of group cardinality constraint* MaxSetCoverage *with high probability.*

*Multiple-pass algorithm.* Next, we demonstrate a $1/2 - \epsilon$ approximation that uses $O(\epsilon^{-1} \log k)$ passes. The idea is similar to the algorithm in Section 2.2 where we pick a set if its contribution is above a threshold. We decrease the threshold by a factor $(1+\epsilon)$ after each pass. The main difference is to not pick a set if that violates the group constraint. Here, we assume that $\text{OPT} \leq z \leq 2\,\text{OPT}$. The detailed algorithm proceeds as follows:

1. Initialize $C \leftarrow \emptyset$ and $I_i = \emptyset$ for each $i = 1, \ldots, \ell$.
2. For $j = 1$ to $\lceil \log_\alpha(4k) \rceil$ where $\alpha = 1 + \epsilon$
   (a) Make a pass over the stream. For each set $S \in G_i$:
      i. If $|S \setminus C| \geq z/\alpha^j$ and $|I_i| < k_i$, then
         $I_i \leftarrow I_i \cup \{ID(S)\}$ and $C \leftarrow C \cup S$ .

Recall that $k = k_1 + \ldots + k_\ell$. Suppose the algorithm picks $k$ sets $S_1, S_2, \ldots, S_k$ in that order. Consider an optimal solution $\mathcal{O} = \{O_1, \ldots, O_k\}$ and a bijection $\pi : [k] \to [k]$ that satisfies the following:

- If $\pi(i) = j$, then $S_i$ and $O_j$ belong to the same group.
- If $S_i$ is $O_j$, then $\pi(i) = j$.

When $S_i \in G_t$ was picked in the $j$th iteration for $j > 1$, by the second property of $\pi$, we deduce that $O_{\pi(i)}$ had not been picked in the $(j-1)$th iteration. Furthermore, the first property of $\pi$ ensures that since we picked $S_i$

in the $j$th iteration, we know that $O_{\pi(i)}$ was available to pick in the $(j-1)$th iteration; however, its contribution was smaller than $z/(k\alpha^{j-1})$. Let $\hat{S}_i := S_i \setminus (S_1 \cup \ldots S_{i-1})$ and $\tilde{O}_i := O_i \setminus \mathcal{C}(\mathcal{S})$.

Therefore,

$$\left|\tilde{O}_{\pi(i)}\right| < \frac{z}{\alpha^{j-1}} = \alpha \cdot \frac{z}{\alpha^j} \leq \alpha \left|\hat{S}_i\right| \ .$$

In the case $j = 1$, obviously, $\left|\hat{S}_i\right| \geq z/\alpha \geq \left|\tilde{O}_{\pi(i)}\right| \cdot 1/\alpha$. We deduce that

$$|\mathcal{C}(\mathcal{S})| = \sum_{i=1}^{k} \left|\hat{S}_i\right| \geq \frac{1}{\alpha} \sum_{1}^{k} \left|\tilde{O}_{\pi(i)}\right| \geq \frac{1}{\alpha} \left(\mathrm{OPT} - |\mathcal{C}(\mathcal{S})|\right) \ .$$

Therefore, $|\mathcal{C}(\mathcal{S})| \geq \mathrm{OPT}/(2+\epsilon)$.

Suppose the algorithm picks fewer than $k$ sets. We can then proceed with our usual argument

$$\mathrm{OPT} = \left|\bigcup_{S \in \mathcal{O}} (S \cap \mathcal{C}(\mathcal{S}))\ \right| + \left|\bigcup_{S \in \mathcal{O} \setminus \mathcal{S}} \tilde{S}\ \right| \leq |\mathcal{C}(\mathcal{S})| + \sum_{S \in \mathcal{O} \setminus \mathcal{S}} \left|\tilde{S}\right| \leq |\mathcal{C}(\mathcal{S})| + \frac{z}{4}$$
$$\leq |\mathcal{C}(\mathcal{S})| + \frac{\mathrm{OPT}}{2}$$

which in turn implies that $|\mathcal{C}(\mathcal{S})| \geq \mathrm{OPT}/2$. By repeating the subsampling argument in Section 2.3, we have the following.

**Theorem 14.** *There exists a $\tilde{O}(\epsilon^{-2}k)$ space algorithm that finds a $1/2 - \epsilon$ approximation of group cardinality constraint* MaxSetCoverage *in $O(\epsilon^{-1} \log k)$ passes with high probability .*

### *2.4.4 Budgeted Maximum Coverage*

In this variation, each set $S$ has a cost $w_S \in [0, L]$. The problem asks to find the collection of sets whose total cost is at most $L$ that covers the most number of distinct elements. For $I \subseteq [n]$, we use $w(I)$ to denote $\sum_{i \in I} w_{S_i}$.

We present the algorithm assuming knowledge of an estimate $z$ such that $\mathrm{OPT} \leq z \leq (1+\epsilon)\,\mathrm{OPT}$; this assumption can be removed by running the algorithm for guesses $1, (1+\epsilon), (1+\epsilon)^2, \ldots$ for $z$ and returning the best solution found. The basic algorithm maintains $I \subseteq [m]$, $C \subseteq [n]$ where $I$ corresponds to the ID's of the (at most $k$) sets in the current solution and $C$ is the the union of the corresponding sets. The algorithm proceeds as follows:

1. Initialize $C = \emptyset$ and $I = \emptyset$
2. For each set $S$ in the stream:
   (a) If $|S \setminus C| \geq \frac{2z}{3} \cdot \frac{w_S}{L}$ then:
      i. If $w(I) + w_S > L$: Terminate and return:

$$I \leftarrow \begin{cases} I & \text{if } |C| \geq |S| \\ \{ID(S)\} & \text{if } |C| < |S| \end{cases}$$

ii. $I \leftarrow I \cup \{ID(S)\}$ and $C \leftarrow C \cup S$.

**Lemma 15.** *If the clause in line 2ai is never satisfied, then the algorithm returns a $1/3 - \epsilon$ approximation.*

*Proof.* Suppose the collection of sets chosen by the algorithm is $\mathcal{S}$. As before, we define $\tilde{S} := S \setminus \mathcal{C}(\mathcal{S})$ to be the set of elements in $S$ that are not covered by the final solution. For each set $S$ in the optimal solution $\mathcal{O}$, if $S$ is unpicked, then $|\tilde{S}| \leq 2z/3 \cdot w_S/L$. Therefore,

$$\begin{aligned}
\text{OPT} &= \left| \bigcup_{S \in \mathcal{O}} (S \cap \mathcal{C}(\mathcal{S})) \right| + \left| \bigcup_{S \in \mathcal{O} \setminus \mathcal{S}} \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \sum_{S \in \mathcal{O} \setminus \mathcal{S}} \left| \tilde{S} \right| \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{2z}{3} \\
&\leq |\mathcal{C}(\mathcal{S})| + \frac{2\,\text{OPT}(1+\epsilon)}{3} ,
\end{aligned}$$

and thus $|\mathcal{C}(\mathcal{S})| \geq \frac{1-2\epsilon}{3}\,\text{OPT}$. □

**Lemma 16.** *If the clause in line 2ai is satisfied at some point, then the algorithm returns a $1/3$ approximation.*

*Proof.* Suppose the clause is satisfied when the set $S$ is being considered. Then

$$|S \setminus C| + |C| \geq \frac{2z}{3} \cdot \frac{w_S + w(I)}{L} \geq \frac{2z}{3}$$

where we used the fact that $w_S + w(I) > L$. The claim then follows immediately. □

The algorithm needs to store the IDs of the sets in the solution as well as the current coverage $C$. Therefore, it uses $\tilde{O}(m+n)$ space.

**Theorem 17.** *There exists a single-pass, $\tilde{O}(\epsilon^{-1}(m+n))$-space algorithm that finds a $1/3 - \epsilon$ approximation of budgeted* MaxSetCoverage.

## 3 Algorithms for Maximum $k$-Vertex Coverage

In this section, we present algorithms for the maximum $k$-vertex coverage problem. We present our results in terms of hypergraphs for full generality. The generalization to hypergraphs can also be thought of as a natural "hitting set" variant of maximum coverage, i.e., the stream consists of a sequence of sets and we want to pick $k$ elements in such a way to maximize the number of sets that include a picked element.

*Notation.* Given a hypergraph $G$ and a subset of nodes $S$, we define $\mathcal{C}_G(S)$ to be the number of edges that contain at least one node in $S$. Recall that the maximum $k$-vertex coverage problem is to approximate the maximum value of $\mathcal{C}_G(S)$ over all sets $S$ containing $k$ nodes. We use $E_G$ and $V_G$ to denote the set of edges and nodes of the hypergraph $G$ respectively.

The size of a cut $(S, V \setminus S)$ in a hypergraph $G$, denoted as $\delta_G(S)$, is defined as the number of hyperedges that contain at least one node in both $S$ and $V \setminus S$. In the case that $G$ is weighted, $\delta_G(S)$ denotes the total weight of the cut. A core idea to our approach is to use *hypergraph sparsification*:

**Definition 18** ($\epsilon$-sparsifier)**.** *Given a hypergraph $G = (V, E)$, we say that a weighted subgraph $H = (V, E')$ is an $\epsilon$-sparsifier for $G$ if for all $S \subseteq V$, $\delta_G(S) \approx_\epsilon \delta_H(S)$.*

Any graph on $N$ nodes has an $\epsilon$-sparsifier with only $\tilde{O}(\epsilon^{-2}N)$ edges [46]. Similarly, any hypergraph in which the maximum size of the hyperedges is bounded by $d$ (rank $d$ hypergraphs) has an $\epsilon$-sparsifier with only $\tilde{O}(\epsilon^{-2}dN)$ edges. Furthermore, an $\epsilon$-sparsifier can be constructed in the dynamic graph stream model using one pass and $\tilde{O}(\epsilon^{-2}dN)$ space [28,30].

First, we show that it is possible to approximate all the coverages by constructing a sparsifier of a slightly modified graph. In particular, we construct the sparsifier $H$ of the graph $G'$ with an extra node $v$, i.e., $V_{G'} = V_G \cup \{v\}$, and for every hyperedge $e \in E_G$, we put the hyperedge $e \cup \{v\}$ in $E_{G'}$. It is easy to see that for all $S$ that is a subset of $V_G$, we have $\mathcal{C}_G(S) = \delta_{G'}(S)$. Therefore, it is immediate that we could $1 + \epsilon$ approximate all the coverages in $G$ by constructing the sparsifer of $G'$.

**Theorem 19.** *There exists a single-pass, $\tilde{O}(\epsilon^{-2}dN)$-space algorithm that finds a $1 - \epsilon$ approximation of* MaxVertexCoverage *of rank $d$ hypergraphs with high probability.*

The above theorem assumes unbounded post-processing time. If $k$ is constant, the post-processing will be polynomial. For larger $k$, if we still require polynomial running time then, after constructing the $\epsilon$-sparsifier $H$, we could either use the $(1-(1-1/d)^d)$ approximation algorithm via linear programming [1] or the folklore $(1 - 1/e)$ approximation greedy algorithm.

### 3.1 Algorithm for Near-Regular Hypergraphs

In this subsection, we show that it is possible to reduce the space used to $\tilde{O}(\epsilon^{-3}dk)$ in the case of hypergraphs that are regular or nearly regular. Define $\kappa \leq 1$ to be the ratio between the smallest degree and the largest degree; for a regular hypergraph $\kappa = 1$. We show that a $(\kappa - \epsilon)$ approximation is possible using $\tilde{O}(\epsilon^{-3}dk)$ space for rank $d$ hypergraphs. This also implies a $(1 - \epsilon)$ approximation for regular hypergraphs.

**Theorem 20.** *There exists a single-pass, $\tilde{O}(\epsilon^{-3}dk)$-space algorithm that finds a $(\kappa - \epsilon)$ approximation of* MaxVertexCoverage *of hypergraphs of rank $d$ with high probability .*

*Proof.* Suppose we uniformly sample a set $S$ of $k$ nodes. Let $L_S(y) = \max(0, |y \cap S| - 1)$. Then the coverage of $S$ satisfies

$$\mathcal{C}_G(S) = \sum_{y \in E_G} I[S \cap y \neq \emptyset] = \sum_{y \in E_G} (|S \cap y| - L_S(y)) \geq kt_1 - \sum_{y \in E_G} L_S(y) \ .$$

where the last inequality follows since every node in $S$ covers at least $t_1$ hyperedges.

Let $\xi_y(j)$ denote the event that $j$ nodes in the hyperedge $y$ are in $S$ and let $|y|$ denote the number of nodes in $y$. We have

$$\mathrm{E}\left[L_S(y)\right] = \sum_{j=1}^{|y|} (j-1) \Pr\left[\xi_y(j)\right] = \left( \sum_{j=0}^{|y|} j \Pr\left[\xi_y(j)\right] \right) - 1 + \Pr\left[\xi_y(0)\right] \ .$$

The sum $\sum_{j=0}^{|y|} j \Pr\left[\xi_y(j)\right]$ is the expected value of the hypergeometric distribution and therefore it evaluates to $|y|k/N$. Furthermore,

$$\begin{aligned} \Pr\left[\xi_y(0)\right] = \prod_{i=0}^{k-1} \left(1 - \frac{|y|}{N-i}\right) &\leq \left(1 - \frac{|y|}{N}\right)^k \\ &\leq \exp\left(-\frac{k|y|}{N}\right) \leq 1 - \frac{k|y|}{N} + \frac{1}{2}\left(\frac{k|y|}{N}\right)^2 . \end{aligned}$$

The last inequality follows from taking the first three terms of the Taylor's expansion. Hence,

$$\mathrm{E}\left[L_S(y)\right] \leq \frac{k|y|}{N} - 1 + 1 - \frac{k|y|}{N} + \frac{1}{2}\left(\frac{k|y|}{N}\right)^2 = \frac{1}{2}\left(\frac{k|y|}{N}\right)^2 .$$

Hence, if $N \geq 4kd/\epsilon$, then

$$\begin{aligned} \sum_{y \in E_G} \mathrm{E}\left[L_S(y)\right] \leq \frac{1}{2} \sum_{y \in E_G} \left(\frac{k|y|}{N}\right)^2 \leq \frac{1}{2} d \left(\frac{k}{N}\right)^2 \sum_{y \in E_G} |y| &\leq \frac{1}{2} d \left(\frac{k}{N}\right)^2 N t_2 \\ &\leq \frac{1}{8} \epsilon k t_2 \ . \end{aligned}$$

By an application of Markov's inequality,

$$\Pr\left[ \sum_{y \in E_G} L_S(y) \geq \epsilon k t_2 \right] \leq 1/8 \ .$$

Thus, if we sample $O(\log N)$ sets of $k$ nodes in parallel, with high probability, there is a sample set $S$ of $k$ nodes satisfying $\sum_{y \in E_G} L_S(y) \leq \epsilon k t_2$ which implies that $\mathcal{C}_G(S) \geq kt_1 - \epsilon k t_2 \geq (\kappa - \epsilon)\,\mathrm{OPT}$. If $N \leq 4kd/\epsilon$, we simply construct the sparsifier of $G'$ as described above to achieve a $1 - \epsilon$ approximation. □

## 4 Lower Bounds

In this section, we prove space lower bounds for data stream algorithms that approximate MaxSetCoverage or MaxVertexCoverage. In particular, these imply that improving over an $(1 - 1/e)$ approximation of MaxSetCoverage with constant passes and constant $k$ requires $\Omega(m)$ space. Recall that, still assuming $k$ is constant, we designed a constant-pass algorithm that returned a $(1 - 1/e - \epsilon)$ approximation using $\tilde{O}(\epsilon^{-2}k)$ space. For constant $k$, we also show that improving over a $\kappa$ approximation (where $\kappa$ is the ratio between the lowest degree and the highest degree) for MaxVertexCoverage requires $\Omega(N\kappa^3)$ space. Our algorithm returned a $\kappa - \epsilon$ approximation using $\tilde{O}(\epsilon^{-3}k)$ space.

*Approach.* We prove both bounds by a reduction from $r$-player set-disjointness in communication complexity. In this problem, there are $r$ players where the $i$th player has a set $S_i \subseteq [u]$. It is promised that exactly one of the following two cases happens.

- Case 1 (NO instance): All the sets are pairwise disjoint.
- Case 2 (YES instance): There is a unique element $e \in [u]$ such that $e \in S_i$ for all $i \in [r]$.

The goal of the communication problem is the $r$th player answers whether the input is a YES instance or a NO instance correctly with probability at least 0.9. We shall denote this problem by $\mathsf{DISJ}_r(u)$.

The communication complexity of the above problem in $p$-round, one-way model (where each round consists of player 1 sending a message to player 2, then player 2 sending a message to player 3 and so on) is $\Omega(u/r)$ [19] even if the players may use public randomness. This implies that in any randomized communication protocol, the maximum message sent by a player contains $\Omega(u/(pr^2))$ bits. Without loss of generality, we could assume that $|S_1 \cup S_2 \cup \ldots \cup S_r| \geq u/4$ via a padding argument.

**Theorem 21.** *Assuming $n = \Omega(\epsilon^{-2}k \log m)$, any constant-pass algorithm that finds a $(1+\epsilon)(1-(1-1/k)^k)$ approximation of* MaxSetCoverage *with probability at least 0.99 requires $\Omega(m/k^2)$ space even when all the sets have the same size.*

*Proof.* Our proof is a reduction from $\mathsf{DISJ}_k(m)$. Consider a sufficiently large $n$ where $k$ divides $n$. For each $i \in [m]$, let $\mathcal{P}_i$ be a random partition of $[n]$ into $k$ sets $V_1^i, \ldots, V_k^i$ of equal size. Each partition is chosen independently and the players agree on these partitions using public randomness before receiving the input.

For each player $j$, if $i \in S_j$, then she puts $V_j^i$ in the stream. According to the aforementioned assumption, the stream consists of at least $m/4$ sets.

If the input is a NO instance, then for each $i \in [m]$, there is at most one set $V_j^i$ in the stream. Hence, the stream consists of independent random sets of size $n/k$. Therefore, for each $e \in [n]$ and any $k$ sets $V_{j_1}^{i_1}, \ldots, V_{j_k}^{i_k}$ in the stream,

$\Pr\left[e \in V_{j_1}^{i_1} \cup \ldots \cup V_{j_k}^{i_k}\right] = 1-(1-1/k)^k$. By an application of Chernoff bound for negatively correlated boolean random variables [42],

$$\begin{aligned}
&\Pr\left[\left||V_{j_1}^{i_1} \cup \ldots \cup V_{j_k}^{i_k}| - \left(1-\left(1-\frac{1}{k}\right)^k\right)n\right| > \epsilon\left(1-\left(1-\frac{1}{k}\right)^k\right)n\right] \\
&\leq 3\exp\left(-\epsilon^2\left(1-\left(1-\frac{1}{k}\right)^k\right)\frac{n}{3}\right) \\
&\leq 3\exp\left(-\epsilon^2(1-1/e)n/3\right) \\
&\leq \frac{1}{m^{10+k}}\ .
\end{aligned}$$

The last inequality holds when $n$ is a sufficiently large multiple of $k\epsilon^{-2}\log m$. Therefore, the maximum coverage in this case is at most $(1+\epsilon)(1-(1-1/k)^k)n$ with probability at least $1-1/m^{10}$ by taking the union bound over all $\binom{m}{k} \leq m^k$ possible $k$ sets.

If the input is a YES instance, then clearly, the maximum coverage is $n$. This is because there exists $i \in [m]$ such that $i \in S_1 \cap \ldots \cap S_k$ and therefore $V_1^i, \ldots, V_k^i$ are in the stream.

Therefore, any constant pass and $O(s)$-space algorithm that finds a $(1+2\epsilon)(1-(1-1/k)^k)$ approximation of the maximum coverage with probability at least 0.99 implies a protocol to solve the $k$-party disjointness problem using $O(s)$ bits of communication. Thus, $s = \Omega(m/k^2)$ as required. □

Consider the sets $S_1, \ldots, S_r \subseteq [u]$ that satisfy the unique intersection promise as in $\mathsf{DISJ}_r(u)$. Let $X$ be the $r$ by $u$ matrix in which the row $X_i$ is the characteristic vector of $S_i$. Suppose there are $r' = \Omega(r^2)$ players. Chakrabarti et al. [18] showed that if each entry of $X$ is given to a unique player and the order in which the entries are given to the players is random, then the players need to use $\Omega(u/r)$ bits of communication to tell whether the sets is a YES instance or a NO instance with probability at least 0.9. Thus, in any randomized protocol, the maximum message sent by a player contains $\Omega(u/r^3)$ bits. Hence, using the same reduction and assuming constant $k$, we show that the same lower bound holds even for random order stream.

**Theorem 22.** *Assuming $n = \Omega(\epsilon^{-2}k\log m)$, any constant-pass algorithm that finds a $(1+\epsilon)(1-(1-1/k)^k)$ approximation of* MaxSetCoverage *with probability at least 0.99 requires $\Omega(m/k^3)$ space even when all the sets have the same size and arrive in random order.*

Next, we prove a lower bound for the $k$-vertex coverage problem for graphs where the ratio between the minimum degree and the maximum degree is at least $\kappa$. We show that for constant $k$, beating $\kappa$ approximation for constant $\kappa$ requires $\Omega(N)$ space.

Since $\kappa$ can be smaller than any constant, this also establishes that $\Omega(N)$ space is required for any constant approximation of MaxVertexCoverage.

**Theorem 23.** *For $\epsilon > 0$, any constant-pass algorithm that finds a $(\kappa + \epsilon)$ approximation of* MaxVertexCoverage *with probability at least 0.99 requires $\Omega(N\kappa^3/k)$ space.*

*Proof.* Initially, assume $k = 1$. We consider the multi-party set disjointness problem $\mathsf{DISJ}_t(N')$ where $t = 1/\kappa$ and $N' = N/t$. Here, there are $t$ players and the input sets are subsets of $[N']$. We consider a bipartite graph where the set of possible nodes are $L \cup R$ where $L = \{u_i\}_{i\in[N']}$ and $R = \{v_{i,j}\}_{i\in[N'],j\in[t]}$. Note that this graph has $(t+1)N' = \Theta(N)$ nodes. However we only consider a node to exist if the stream contains an edge incident to that node.

The $j$-th player defines a set of edges on this graph based on their set $S_j$ as follows. If $i \in S_j$, she puts the edge between $u_i$ and $v_{i,j}$. If $S_1, \ldots, S_t$ is a YES instance, then there must be a node $u_i$ that has degree $t$. If $A$ is a NO instance, then every node in the graph has degree at most 1. Hence the ratio of minimum degree to maximum degree is at least $1/t = \kappa$ as required.

Thus, for $k = 1$, a $1/t$ approximation with probability at least 0.99 on a graph of $N$ nodes implies a protocol to solve $\mathsf{DISJ}_t(N')$. Therefore, the algorithm requires $\Omega(N\kappa^3)$ space. For general $k$, we make $k$ copies of the above construction to deduce the lower bound $\Omega(N\kappa^3/k)$. □

## 5 Summary

In this paper, we studied the maximum $k$-set and $k$-vertex coverage problems in the popular streaming-set model and the dynamic graph stream model. We provided a set of sublinear space algorithms that exhibit numerous trade-offs between the space complexity and the approximation guarantee. Furthermore, our lower bounds suggest that several of our algorithms are tight or nearly-tight.

This paper further investigated the topic of NP-Hard problems in data streams. For the problems we considered, several questions remained open. For example,

1. For maximum $k$-set coverage:
   (a) Is our $\tilde{O}(\epsilon^{-3}k)$ space and $1/2 - \epsilon$ approximation algorithm for the maximum $k$-set coverage tight?
   (b) Can we obtain a better than $1/2 - \epsilon$ approximation (or ultimately $1 - 1/e - \epsilon$ approximation) using $o(m)$ space.
2. For maximum $k$-vertex coverage:
   (a) Identify other interesting families of graphs (beside near-regular graphs) that admit $o(N)$ space algorithm for the maximum $k$-vertex coverage problem.

**Acknowledgements** We thank Sagar Kale for discussions of related work.

## References


1. Ageev, A.A., Sviridenko, M.: Approximation algorithms for maximum coverage and max cut with given sizes of parts. In: IPCO, *Lecture Notes in Computer Science*, vol. 1610, pp. 17–30. Springer (1999)
2. Ageev, A.A., Sviridenko, M.: Pipage rounding: A new method of constructing algorithms with proven performance guarantee. J. Comb. Optim. **8**(3), 307–328 (2004)
3. Ahn, K.J., Cormode, G., Guha, S., McGregor, A., Wirth, A.: Correlation clustering in data streams. In: ICML, *JMLR Workshop and Conference Proceedings*, vol. 37, pp. 2237–2246. JMLR.org (2015)
4. Ahn, K.J., Guha, S., McGregor, A.: Analyzing graph structure via linear measurements. In: SODA, pp. 459–467. SIAM (2012)
5. Ahn, K.J., Guha, S., McGregor, A.: Graph sketches: sparsification, spanners, and subgraphs. In: PODS, pp. 5–14. ACM (2012)
6. Ahn, K.J., Guha, S., McGregor, A.: Spectral sparsification in dynamic graph streams. In: APPROX-RANDOM, *Lecture Notes in Computer Science*, vol. 8096, pp. 1–10. Springer (2013)
7. Anagnostopoulos, A., Becchetti, L., Bordino, I., Leonardi, S., Mele, I., Sankowski, P.: Stochastic query covering for fast approximate document retrieval. ACM Trans. Inf. Syst. **33**(3), 11:1–11:35 (2015)
8. Assadi, S.: Tight space-approximation tradeoff for the multi-pass streaming set cover problem. In: PODS, pp. 321–335. ACM (2017)
9. Assadi, S., Khanna, S., Li, Y.: Tight bounds for single-pass streaming complexity of the set cover problem. In: STOC, pp. 698–711. ACM (2016)
10. Assadi, S., Khanna, S., Li, Y., Yaroslavtsev, G.: Maximum matchings in dynamic graph streams and the simultaneous communication model. In: SODA, pp. 1345–1364. SIAM (2016)
11. Ausiello, G., Boria, N., Giannakos, A., Lucarelli, G., Paschos, V.T.: Online maximum k-coverage. Discrete Applied Mathematics **160**(13-14), 1901–1913 (2012)
12. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: massive data summarization on the fly. In: KDD, pp. 671–680. ACM (2014)
13. Badanidiyuru, A., Vondrák, J.: Fast algorithms for maximizing submodular functions. In: SODA, pp. 1497–1514. SIAM (2014)
14. Bateni, M., Esfandiari, H., Mirrokni, V.S.: Almost optimal streaming algorithms for coverage problems. CoRR **abs/1610.08096** (2016)
15. Bhattacharya, S., Henzinger, M., Nanongkai, D., Tsourakakis, C.E.: Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In: STOC, pp. 173–182. ACM (2015)
16. Bonnet, É., Escoffier, B., Paschos, V.T., Stamoulis, G.: A 0.821-ratio purely combinatorial algorithm for maximum k-vertex cover in bipartite graphs. In: LATIN, *Lecture Notes in Computer Science*, vol. 9644, pp. 235–248. Springer (2016)
17. Caskurlu, B., Mkrtchyan, V., Parekh, O., Subramani, K.: On partial vertex cover and budgeted maximum coverage problems in bipartite graphs. In: IFIP TCS, *Lecture Notes in Computer Science*, vol. 8705, pp. 13–26. Springer (2014)
18. Chakrabarti, A., Cormode, G., McGregor, A.: Robust lower bounds for communication and stream computation. Electronic Colloquium on Computational Complexity (ECCC) **18**, 62 (2011)
19. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In: IEEE Conference on Computational Complexity, pp. 107–117. IEEE Computer Society (2003)
20. Chakrabarti, A., Wirth, A.: Incidence geometries and the pass complexity of semi-streaming set cover. In: SODA, pp. 1365–1373. SIAM (2016)
21. Chekuri, C., Gupta, S., Quanrud, K.: Streaming algorithms for submodular function maximization. In: ICALP (1), *Lecture Notes in Computer Science*, vol. 9134, pp. 318–330. Springer (2015)
22. Chekuri, C., Kumar, A.: Maximum coverage problem with group budget constraints and applications. In: APPROX-RANDOM, *Lecture Notes in Computer Science*, vol. 3122, pp. 72–83. Springer (2004)

23. Chitnis, R., Cormode, G., Esfandiari, H., Hajiaghayi, M., McGregor, A., Monemizadeh, M., Vorotnikova, S.: Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In: SODA, pp. 1326–1344. SIAM (2016)
24. Cormode, G., Datar, M., Indyk, P., Muthukrishnan, S.: Comparing data streams using hamming norms (how to zero in). IEEE Trans. Knowl. Data Eng. **15**(3), 529–540 (2003)
25. Cormode, G., Karloff, H.J., Wirth, A.: Set cover algorithms for very large datasets. In: CIKM, pp. 479–488. ACM (2010)
26. Emek, Y., Rosén, A.: Semi-streaming set cover - (extended abstract). In: ICALP (1), *Lecture Notes in Computer Science*, vol. 8572, pp. 453–464. Springer (2014)
27. Feige, U.: A threshold of ln $n$ for approximating set cover. J. ACM **45**(4), 634–652 (1998)
28. Guha, S., McGregor, A., Tench, D.: Vertex and hyperedge connectivity in dynamic graph streams. In: PODS, pp. 241–247. ACM (2015)
29. Har-Peled, S., Indyk, P., Mahabadi, S., Vakilian, A.: Towards tight bounds for the streaming set cover problem. In: PODS, pp. 371–383. ACM (2016)
30. Kapralov, M., Lee, Y.T., Musco, C., Musco, C., Sidford, A.: Single pass spectral sparsification in dynamic streams. In: FOCS, pp. 561–570. IEEE Computer Society (2014)
31. Kapralov, M., Woodruff, D.P.: Spanners and sparsifiers in dynamic streams. In: PODC, pp. 272–281. ACM (2014)
32. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. Theory of Computing **11**, 105–147 (2015)
33. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Inf. Process. Lett. **70**(1), 39–45 (1999)
34. Kogan, D., Krauthgamer, R.: Sketching cuts in graphs and hypergraphs. In: 6th Innovations in Theoretical Computer Science (2015)
35. Konrad, C.: Maximum matching in turnstile streams. In: ESA, *Lecture Notes in Computer Science*, vol. 9294, pp. 840–852. Springer (2015)
36. Krause, A., Guestrin, C.: Near-optimal observation selection using submodular functions. In: AAAI, pp. 1650–1654. AAAI Press (2007)
37. Kumar, R., Moseley, B., Vassilvitskii, S., Vattani, A.: Fast greedy algorithms in mapreduce and streaming. TOPC **2**(3), 14 (2015)
38. McGregor, A.: Graph stream algorithms: a survey. SIGMOD Record **43**(1), 9–20 (2014)
39. McGregor, A., Tench, D., Vorotnikova, S., Vu, H.T.: Densest subgraph in dynamic graph streams. In: MFCS (2), *Lecture Notes in Computer Science*, vol. 9235, pp. 472–482. Springer (2015)
40. McGregor, A., Vorotnikova, S., Vu, H.T.: Better algorithms for counting triangles in data streams. In: PODS, pp. 401–411. ACM (2016)
41. McGregor, A., Vu, H.T.: Better streaming algorithms for the maximum coverage problem. In: ICDT, *LIPIcs*, vol. 68, pp. 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
42. Panconesi, A., Srinivasan, A.: Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. SIAM J. Comput. **26**(2), 350–368 (1997)
43. Radhakrishnan, J., Shannigrahi, S.: Streaming algorithms for 2-coloring uniform hypergraphs. In: Algorithms and Data Structures - 12th International Symposium, WADS 2011, New York, NY, USA, August 15-17, 2011. Proceedings, pp. 667–678 (2011). DOI 10.1007/978-3-642-22300-6_57. URL `http://dx.doi.org/10.1007/978-3-642-22300-6_57`
44. Saha, B., Getoor, L.: On maximum coverage in the streaming model & application to multi-topic blog-watch. In: SDM, pp. 697–708. SIAM (2009)
45. Schmidt, J.P., Siegel, A., Srinivasan, A.: Chernoff-hoeffding bounds for applications with limited independence. SIAM J. Discrete Math. **8**(2), 223–250 (1995)
46. Spielman, D.A., Teng, S.: Spectral sparsification of graphs. SIAM J. Comput. **40**(4), 981–1025 (2011)
47. Srinivasan, A.: Distributions on level-sets with applications to approximation algorithms. In: FOCS, pp. 588–597. IEEE Computer Society (2001)
48. Sun, H.: Counting hypergraphs in data streams. CoRR **abs/1304.7456** (2013). URL `http://arxiv.org/abs/1304.7456`
49. Yu, H., Yuan, D.: Set coverage problems in a one-pass data stream. In: SDM, pp. 758–766. SIAM (2013)

Author, Article title, Journal, Volume, page numbers (year) Author, Book title, page numbers. Publisher, place (year)