Solving Coverage Problems on Massive Data: Extended Abstract of Recent Results*

MohammadHossein Bateni
Google Research

Hossein Esfandiari
Harvard University

Vahab Mirrokni
Google Research

Abstract

We study three *coverage problems*—minimum set cover, maximum k-cover, and minimum set cover with outliers—in large-scale settings. Our main contribution is a simple yet powerful *sketch* for these problems that leads to almost optimal algorithms in streaming, MapReduce and RAM models. The optimality is measured in terms of the running time, approximation guarantee, space complexity, as well as number of rounds/passes of computation. These results are complemented by demonstrating why natural sketches are not sufficient to solve these problems. We further study extending the algorithms to several weighted variants of set cover, as well as facility location, dominating set, and a large class of submodular maximization problems.

Our extensive empirical study illustrates the effectiveness of the new algorithms. Here we consider a variety of set-cover instances (bag-of-word document summarization, collaboration networks, dominating set) as well as a real application for feature selection. We observe that using sketches 30–600 times smaller than the input, one can solve the coverage maximization problem with quality very close to that of the state-of-the-art single-machine algorithm.

1 Introduction

Maximum coverage and minimum set cover problems are among the most fundamental problems in optimization and computer science. Not only have they been instrumental for development of new techniques, but they also have numerous applications in theory and practice, either directly or as subroutine in other algorithms. Notably there are a variety of practical applications in machine learning or data mining (say, for data summarization and web mining); see, e.g., [11, 17] and references therein.

More formally we study the following problems. Consider a family $\mathcal{S}$ of n subsets of a ground set I of m items. In the minimum set cover problem, we look for the smallest number of sets in $\mathcal{S}$ that collectively cover I; in other words, their union is exactly I. A "dual" to this task is the maximum k-cover problem, where given an additional parameter k, the goal is to find k sets from $\mathcal{S}$ so as to maximize the size of their union. We also study a less known problem that provides a middle ground between the two. The input to the minimum set cover with outliers problem includes a parameter $0 \leq \lambda \leq 1$ in addition to the set specification $\mathcal{S}$. Here we aim to find the smallest number of sets whose union covers at least a $1 - \lambda$ fraction of I.

Nowadays there is an inevitable need to handle many optimization problems at large scale [47, 46, 20, 8, 16, 15, 20, 32]. Older methods typically cannot cope with huge amounts of data. Firstly,

*This extended abstract summarizes the main results of [12, 13, 14], which contain all the proofs as well as additional discussion. Core sketching results and the discussion of the streaming implementation appear in [12], whereas [13] contains MapReduce and RAM results as well as an extensive empirical study and extension to the weighted setting. Part of the results was presented at SPAA 2017 [14].

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

polynomial running time is no longer a satisfactory test for usability of an algorithm. Second, the entire data set may not fit on the memory of a single machine. Several new computational models have been proposed to address these hurdles. The most famous are the streaming setting, the MapReduce framework, and the RAM model. The first two are quite popular and well-known but we also include the last one, because we think that development of multicore processors and shared memory architectures will once again bring such models into focus.

The streaming setting places the main restriction on the available memory. A single processor receives the data set, one small piece at a time (perhaps in an adversarial order), and is required to output the result at the end, once the stream has been fully processed. The algorithm may in some cases get the opportunity of having multiple passes over the input. However, it can never carry large amount of information. The metrics to optimize here (beside the accuracy of the solution) are running time at each step, total memory requirement, and number of passes.

Two standard arrival models are studied for graph problems in the streaming literature. In the edge-arrival model, edges of the input graph are given to the algorithm one at a time in an arbitrary (possibly adversarial) order. The vertex-arrival model, though, assumes that *vertices* appear one by one, and upon arrival each vertex is given to the algorithm along with all its edges. Often solving a problem in the edge-arrival model is more challenging that doing the same in the vertex-arrival model. In the context of coverage problems, we refer to these models as set- and edge-arrival models, because we sometimes think of a set system as a bipartite graph between sets and items (with edges denoting membership relations).

Another popular approach towards the problem of handling large data is the MapReduce framework [22], which serves as the standard in industrial distributed computation. Several attempts have been made to formalize a theoretical model for this framework and we focus on the most famous. A sublinear number of machines, each with sublinear memory, are available to the algorithm. At the beginning of one round, the data is somehow sent (i.e., "mapped") to different machines. Then comes the Map phase, when each machine processes the data it receives and sends the (possibly different) results to certain machines. Next in the Reduce phase, every machine processes the Map phase results sent to it, and outputs the final results. This Map/Reduce computation may go on for many rounds. However, the memory available to each machine, the time for each Map or Reduce phase, as well as the number of rounds should be limited.

In the RAM model [5], we have one processor—can be extended to several—that has random access to any part of the huge data set. The caveat is that each "read" takes some amount of time, hence reading all the input is not practical. In this respect, it is similar to the streaming setting but the algorithm has the opportunity to choose which parts of the input it looks at.

2 Results and related work

Recall that the input consists of n sets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of a possibly much larger universe of size m. We prove the existence of a new set system $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ such that $H_i \subseteq S_i$ for each $1 \leq i \leq n$. The definition of our sketch, $\mathcal{H}$, depends on a constant parameter $\epsilon > 0$. This sketch has two crucial properties:

1. It is *manageable*, i.e., the total size of the sketch is small. More formally $\sum_i |H_i| = O(n \operatorname{poly} \log m)$.
2. It is *applicable*: With high probability, any α-approximate solution to the maximum k-cover instance defined by $\mathcal{H}$ is an $\alpha - \epsilon$-approximate solution to the original instance defined by $\mathcal{S}$.

We further demonstrate how this sketch can indeed be constructed efficiently on each of the three large-scale computational models. This leads to a straightforward algorithm for maximum k-cover: Build the sketch and apply the best sequential algorithm to the sketch. Since the sketching step reduces the instance size to fit in memory (or be within the permissible number of reads in the RAM model), any efficient sequential algorithm may be used in the second step. In particular, though, there are almost linear-time sequential algorithms for maximum k-cover [42].

The three problems discussed in this manuscript are known to be NP-hard [29]. Moreover, we know that they are APX-hard [24]: there exists a constant lower bound on the approximation guarantee of

Table 1: Comparison of our results [13] for MapReduce model to prior work. The first three work for the more general case of submodular maximization.

Problem	Credit	Number of rounds	Approximation factor	Load per machine
k-cover	[38]	$O(\frac{1}{\epsilon\delta}\log m)$	$1-\frac{1}{e}-\epsilon$	$O(mkn^{\delta})$
k-cover	[41]	2	0.54	$\max(mk^2, mn/k)$
k-cover	[21]	$\frac{1}{\epsilon}$	$1-\frac{1}{e}-\epsilon$	$\frac{\max(mk^2,mn/k)}{\epsilon}$
k-cover	[13]	4	$1-\frac{1}{e}-\epsilon$	$\tilde{O}(n)$
set cover with outliers	[13]	4	$(1+\epsilon)\log\frac{1}{\lambda}$	$\tilde{O}(n)$
submodular cover	[43]	$\Omega(n^{\frac{1}{6}})$	$\Omega(n^{\frac{1}{6}})$	$\tilde{O}(mn)$
submodular cover	[44]	$O(\frac{\log n\log m}{\epsilon})$	$(1+\epsilon)\log\frac{1}{\lambda}$	$\tilde{O}(mn)$
set cover with outliers	[13]	4	$(1+\epsilon)\log\frac{1}{\lambda}$	$\tilde{O}(n)$
dominating set	[13]	4	$(1+\epsilon)\log\frac{1}{\lambda}$	$\tilde{O}(n)$

any polynomial-time algorithm, assuming P $\neq$ NP. In fact, the lower bound is logarithmic for set cover and set cover with outliers problems. These negative results, in particular, rule out the existence of a polynomial-time approximation scheme (PTAS).

Interestingly even in the non-distributed setting, there are simple greedy algorithms that achieve approximation factors essentially matching the lower bounds: $1-\frac{1}{e}$ for k-cover and $O(\log n)$ for the other two problems. The algorithms summarized have similar, *optimal* approximation factors.

We further demonstrate that our algorithms use almost optimal space. Specifically we prove that in the streaming setting, it is impossible to have an approximation factor better than $\frac{1}{2}$ for k-cover in $o(n)$ space. (There is no complexity assumption here, as the proof relies on a communications complexity argument.) We claim almost optimal space consumption, because our algorithms use *quasi-linear* space: $\tilde{O}(n) = O(n\,\mathrm{poly}(\log n, \log m))$.

Coverage problems have been studied extensively in the context of set-arrival models [8, 47, 46, 25, 16]. Most of these give suboptimal approximation guarantees. In particular, Saha and Getoor [47] provide a $\frac{1}{4}$-approximation algorithm for k-cover in one pass using $\tilde{O}(m)$ space. The same technique gives a $\Theta(\log m)$ approximation algorithm for set cover in $\Theta(\log m)$ passes, using $\tilde{O}(m)$ space. On the hardness side, interestingly, Assadi et al. [8] show that there is no α-approximation one-pass streaming algorithm for set cover using $o(nm/\alpha)$ space. Demaine et al. [23] provide (for any positive integer r) a $4^r\log m$-approximation algorithm for the set cover problem in 4^r passes using $\tilde{O}(nm^{1/r}+m)$ space[2]. Recently, Har-Peled et al. improves this result and provide a p-pass $O(p\log m)$-approximation algorithm in $\tilde{O}(nm^{O(1/p)}+m)$ space[2]. Indeed, all the above results hold only for the set-arrival model, whereas our results are for the more general edge-arrival model. Tables 1 and 2 summarize our theoretical results and compare them to prior work.

Often in the graph streaming problems, while the size of the input is $\tilde{O}(|E|)$ for a graph $G(V,E)$, the solution size may be as large as $\Omega(|V|)$. The best hope then is to find the solution in $\tilde{O}(|V|)$ space. Algorithms fitting this description are called semi-streaming [45], and many graph problems have been studied in this setting [2, 3, 26, 28, 35, 36, 37]. On the other hand, the extensive work on edge-arrival streaming [4, 7, 9, 18, 27, 33, 34] had not (prior to our work) studied coverage problems.

3 A new sketch

One strategy to solve the problem is to build an *oracle* that allows one to estimate the quality of any given solution. Let us define the coverage function $f: 2^{\mathcal{S}} \mapsto \mathbb{R}$ as $f(\mathcal{A}) = |\bigcup_{A\in\mathcal{A}} A|$. Our goal then is to find a subcollection of $\mathcal{S}$ of size k that maximizes f.

Suppose we have access to an oracle that computes $f_\epsilon: 2^{\mathcal{S}} \mapsto \mathbb{R}$ such that

$$(1-\epsilon)f(\mathcal{A}) \leq f_\epsilon(\mathcal{A}) \leq (1+\epsilon)f(\mathcal{A}) \qquad \forall \mathcal{A}\subseteq\mathcal{S}.$$

[2]The space bounds claimed in [23, 31] assume $m = O(n)$, hence stated differently.

Table 2: Comparison of results [14] for streaming model to prior work. Note that all our results for edge-arrival model also hold for the set-arrival model.

Problem	Credit	No. of passes	Approx. factor	Space	Arrival
k-cover	[47]	1	$1/4$	$\tilde{O}(m)$	set
k-cover	[11]	1	$1/2$	$\tilde{O}(n+m)$	set
k-cover	[14]	1	$1-1/e-\epsilon$	$\tilde{O}(n)$	edge
set cover with outliers	[25, 16]	p	$O(\min(n^{\frac{1}{p+1}}, e^{-\frac{1}{p}}))$	$\tilde{O}(m)$	set
set cover with outliers	[14]	1	$(1+\epsilon)\log\frac{1}{\lambda}$	$\tilde{O}_\lambda(n)$	edge
set cover	[16, 47]	p	$(p+1)m^{\frac{1}{p+1}}$	$\tilde{O}(m)$	set
set cover	[23]	4^r	$4^r \log m$	$\tilde{O}(nm^{\frac{1}{r}}+m)$	set
set cover	[31]	p	$O(p \log m)$	$\tilde{O}(nm^{O(\frac{1}{p})}+m)$	set
set cover	[14]	p	$(1+\epsilon)\log m$	$\tilde{O}(nm^{O(\frac{1}{p})}+m)$	edge

It might sound reasonable that such an oracle suffices for solving the problem: In particular, if such an oracle can be created quickly with small space (say, via sampling), then the resulting algorithm would be a good distributed algorithm for k-cover.

Unfortunately, though, we prove that this is not possible: Any α-approximate algorithm for k-cover that only uses f_ϵ to access the data requires $\exp\left(\Omega(n\epsilon^2\alpha^2 - \log n)\right)$ queries to the oracle. Therefore, there is no polynomial-time $n^{-0.49}$-approximation algorithm for k-cover using the oracle f_ϵ as a black box.

We remark that our sketch is fairly similar to ℓ_0 sketches [19], which are essentially defined to estimate the value of coverage functions. Indeed, one may maintain n instances of the ℓ_0 sketch, and estimate the value of the coverage function of a single feasible solution of size k with high probability. However, having $\binom{n}{k}$ different choices for a solution of size k leads to a huge blow-up on the failure probability of at least one such solution. We show a straightforward analysis to approximate k-cover using ℓ_0 sketches with $\tilde{O}(nk)$ space, which is quite larger than our sketch.

We present a new sketching technique that does not suffer from the shortcomings identified above. Conceptually this is done in two stages, and relies on two parameters $0 < p \leq 1$ and $0 < \Delta \leq n$. In the first stage, we sample a subset $I' \subseteq I$ by selecting each item independently with probability p. The other items are removed from the set system. In other words, this turns $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ into another family $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$, where $F_i = S_i \cap I'$ for $1 \leq i \leq n$. Then the second stage removes some "edges" (between sets and items) or in other words restricts the frequency of items in the set system: Items are arbitrarily removed from sets, so that no item appears in more than Δ sets. More formally, we construct a family $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ such that

1. $H_i \subseteq F_i$ for $1 \leq i \leq n$;
2. $|\{H \in \mathcal{H} \mid e \in H\}| \leq \Delta$ for each $e \in I'$; and
3. $\mathcal{H}$ is *maximal* in the sense that adding any item to a set in $\mathcal{H}$ violates one of the above properties.

Both stages are necessary to guarantee the desired properties of the sketch. Roughly speaking, the first reduces the number of items to be near-linear in terms of the number of sets. Concentration bounds show that with high probability, this operation preserves (up to a factor p) the size of the union of every subcollection of sets. However, the total size of the produced sets might still be large, as the frequency of certain items may be large. The second stage is based on the observation that, roughly speaking, a logarithmic maximum frequency for items suffices to preserve the union size for the optimal solution to k-cover.

The following theorem summarizes our sketch and lies at the core of all our algorithms.

Theorem 1 ([12]). *Consider a set system $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ on a ground set of size m. Given parameters $\delta \geq 1$, $0 < \epsilon < 1$ and $1 \leq k \leq n$, we build the sketch $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ with parameters p and Δ defined as follows. Let U_k be the maximum size of the union of k sets from $\mathcal{H}$,*

and define

$$p = \frac{6k\delta \log\log_{1+\epsilon} m}{\epsilon^2 U_k}, \qquad \Delta = \frac{n \log \frac{1}{\epsilon}}{\epsilon k}.$$

Then with probability $1 - 3e^{-\delta}$ *we obtain the following two properties.*

1. *There is an upper bound on the total size of the* $\mathcal{H}$*:*

$$\sum_{i=1}^{n} |H_i| \leq g(n, m, \delta) = \frac{24n\delta \log \frac{1}{\epsilon} \log\log_{1+\epsilon} m \log n}{(1-\epsilon)\epsilon^3}.$$

2. *Any* α*-approximate solution for* k-cover *on* $\mathcal{H}$ *produces an* $\alpha - 12\epsilon$*-approximate solution for the same problem on the original set system* $\mathcal{S}$*.*

4 Algorithms

To solve the k-cover problem, one sketch from Theorem 1 is sufficient: Build a sketch and solve the problem on the sketch. However, to solve the other two problems (set cover and set cover with outliers), we construct logarithmically many such sketches (simultaneously) and solve the problem on each. This is done because we do not know the proper value for k in these problems. It is guaranteed that one sketch is built with a good approximation of k, hence the resulting solution from that sketch is our desired solution.

It remains to show how to construct the sketch in each of the large-scale settings, because the resulting sketch is small enough to run the algorithm on at the end (on MapReduce, streaming or the RAM model). Here we discuss sketch construction only for streaming. Refer to [13] for the other two models.

There is a caveat in Theorem 1. It seems that the "correct" value of p depends on the optimal coverage U_k. Next we demonstrate how the knowledge of U_k is not necessary. Let us parameterize the sketch by the probability p as $\mathcal{H}_p$. Picking a larger value for p than the correct one does not harm the second property in the theorem statement, but it could possibly violate the first property (and lead to superquasilinear sketch size). Let p^* be the smallest value for p that makes total sketch size exceed $g(n, m, \delta)$. (For simplicity of exposition we are ignoring the probabilistic nature of the sketch size.) Then the theorem suggests that the correct value of p cannot be larger than p^*. Therefore we can safely use p^* in the construction of the sketch.

Solving the other two problems requires additional ideas. For a given set system $\mathcal{S}$, let $\mathsf{Opt}(\mathcal{S}, k)$ denote the maximum coverage of k sets from $\mathcal{S}$. Define $\mathsf{Greedy}(\mathcal{S}, k)$ as the coverage of the greedy algorithm on an instance defined by the set system $\mathcal{S}$ and parameter k. The $1 - \frac{1}{e}$ approximation guarantee of the greedy algorithm implies $\mathsf{Greedy}(\mathcal{S}, k) \geq (1 - \frac{1}{e})\mathsf{Opt}(\mathcal{S}, k)$. Slight changes to the argument gives the following for any $\alpha > 1$:

$$\mathsf{Greedy}(\mathcal{S}, k \log \alpha) \geq (1 - 1/\alpha)\mathsf{Opt}(\mathcal{S}, k). \tag{1}$$

Roughly speaking, one can repeat the greedy algorithm $\log \alpha$ times to achieve this.

Let q be the size of the optimal solution for set cover with λ outliers on set system $\mathcal{S}$; i.e., $\mathsf{Opt}(\mathcal{S}, q) \geq (1 - \lambda)|I|$ where I is the ground set over which $\mathcal{S}$ is defined. Combining (1) with our k-cover algorithm guarantees that, for any fixed $\epsilon > 0$, we can find a solution of size $q \log \frac{1}{\lambda'}$ covering $1 - \lambda' - \epsilon$ fraction of the items with high probability. Roughly speaking we can search through a logarithmic number of possibilities (in a geometric progression) and find a solution of size at most $O(q \log \frac{1}{\lambda})$ that covers a $1 - \lambda$ fraction of the items. We show in [12] how to set the parameters λ' and ϵ properly.

The algorithm for set cover solves $r - 1$ instances of set cover with λ outliers (for $\lambda = m^{-\frac{1}{2+r}}$), each on the residual instance of yet uncovered items. At most $m\lambda^{r-1} = m^{\frac{3}{2+r}}$ items may still be uncovered by the union of the solutions from these $r - 1$ instances. A sequential set cover algorithm is used at the end to cover all the remaining items. We show that the total memory used in each round of this algorithm is bounded by $\tilde{O}(nm^{O(1/r)} + m)$, hence there is a tradeoff between the number of rounds and the space usage.

5 Extensions

We extend our results to three classes of submodular functions:

1. In element-weighted k-cover, a weight w_v is associated with each element $v \in I$, and the objective is to maximize the total weight of covered elements.
2. An instance of facility location problem contains a quantity $\alpha_{u,v} \in [0,1]$ for each $S \in \mathcal{S}, v \in I$, denoting that set S covers $\alpha_{S,v}$ fraction[3] of element v. A solution $\mathcal{S}' \subseteq \mathcal{S}$ covers $\max_{S \in \mathcal{S}'} \alpha_{S,v}$ fraction of element v. Here the objective is to find a solution $\mathcal{S}' \subseteq \mathcal{S}$ of size k that maximizes $\sum_{v \in I} \max_{S \in \mathcal{S}'} \alpha_{S,v}$.
3. Finally in probabilistic k-cover, quantity $\alpha_{S,v} \in [0,1]$ is provided for each pair of $S \in \mathcal{S}$ and $v \in I$: set S covers element v with probability $\alpha_{S,v}$. A solution $\mathcal{S}' \subseteq \mathcal{S}$ covers $1 - \prod_{S \in}(1 - \alpha_{S,v})$ fraction of element v. The objective then is to find a solution $\mathcal{S}' \subseteq \mathcal{S}$ of cardinality k that maximizes $\sum_{v \in I} \left(1 - \prod_{S \in \mathcal{S}'}(1 - \alpha_{S,v})\right)$.

In [13] we present distributed algorithms for these problems.

Theorem 2. *For each of the three variants defined above (i.e., element-weighted k-cover, facility location and probabilistic k-cover), and for every fixed $\epsilon > 0$, there exists a four-round distributed algorithm that uses $\tilde{O}(n)$ space per machine and finds a $1 - \frac{1}{e} - \epsilon$ approximate solution with probability $1 - O(\frac{1}{n})$.*

We remark at this point that extending to these variants is possible only because the size of our new sketch is independent of (or to be more precise, depends logarithmically on) the number of elements. In fact, the reductions we present for the above problems significantly increase the number of elements in the instance.

6 Empirical study

We perform a comprehensive empirical study of our algorithm for k-cover (implemented in MapReduce) on a variety of data sets. We consider social graphs such as LiveJournal [48], collaboration networks such as DBLP [10] and Wikipedia [39], bag-of-words data sets such as news20 (discussed in [6]), reuters [40] and one based on bigrams in books from Project Gutenberg [1], as well as planted set-cover instances that are known to be hard theoretically. These data sets have up to 4 million sets, 200 million elements, and 73 billion set-element edges; to our best knowledge, these are an order of magnitude larger than the data sets used in other reported k-cover experiments.

Recall that our sketch is parameterized by p and Δ. The theoretical values for these lead to relatively large sketches for the big data sets we study. However we observe that using sketches 30–600 times smaller than the input, one can still solve the coverage maximization problem with quality very close to that of the state-of-the-art single-machine algorithm.

Our algorithm is applicable to the feature-selection problem, which is a first step in many learning-based applications [30], where it is often too expensive to work with the entire matrix or there might be overfitting concerns. Typically a small subset of "representative" features are picked carefully, so as not to affect the overall learning quality. In practice, we gauge the performance of feature selection by *reconstruction error* or *prediction accuracy*; see [6] for details of evaluation criteria.

In order to compare our preliminary results to previous work [6], we model the problem as a maximum k-cover instance by treating columns (i.e., features) as sets and *pairs of rows* (i.e., pairs of sample points) as elements. We say a row *covers* a pair of rows, if that column (feature) is active for both rows (sample points), and seek to pick k columns that *cover as many pairs of rows* as possible.

Comparison with previous work illustrates the scalability and quality of our algorithm. While previous algorithms could only run on a 8% sample of the dataset, our method comfortably processes the entire data. The quality of our results unsurprisingly is better than four of the five methods explained in [6]; their other algorithm, whose quality is on par with ours, is much slower than our distributed algorithm.

[3] In general, in facility location, $\alpha_{u,v}$'s are real numbers, but here, we normalize all $\alpha_{u,v}$'s to a number in $[0,1]$.

References

[1] Free ebooks by Project Gutenberg. https://www.gutenberg.org/. Accessed: 05-19-2016.

[2] Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *ICALP (2)*, pages 328–338, 2009.

[3] Kook Jin Ahn and Sudipto Guha. Laminar families and metric embeddings: Non-bipartite maximum matching problem in the semi-streaming model. *CoRR*, abs/1104.4058, 2011.

[4] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *APPROX*, pages 1–10, 2013.

[5] Alfred V Aho and John E Hopcroft. *Design & Analysis of Computer Algorithms*. Pearson Education India, 1974.

[6] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab S. Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *ICML*, pages 2539–2548, 2016.

[7] Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards (1+ ε)-approximate flow sparsifiers. In *SODA*, pages 279–293, 2014.

[8] Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *STOC*, pages 698–711, 2016.

[9] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *SODA*, pages 1345–1364, 2016.

[10] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: Membership, growth, and evolution. In *KDD*, 2006.

[11] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *KDD*, 2014.

[12] MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. Almost optimal streaming algorithms for coverage problems. *CoRR*, abs/1610.08096, 2016.

[13] MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. Distributed coverage maximization via sketching. *CoRR*, abs/1612.02327, 2016.

[14] MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. Almost optimal streaming algorithms for coverage problems. In *SPAA*, 2017.

[15] Guy E. Blelloch, Harsha Vardhan Simhadri, and Kanat Tangwongsan. Parallel and I/O efficient set covering algorithms. In *SPAA*, pages 82–90, 2012.

[16] Amit Chakrabarti and Anthony Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *SODA*, pages 1365–1373, 2016.

[17] Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-Cover in Map-Reduce. In *WWW*, pages 231–240, 2010.

[18] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemiz adeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *SODA*, pages 1326–1344, 2016.

[19] Graham Cormode, Mayur Datar, Piotr Indyk, and S Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *IEEE Trans. Knowl. Data Eng.*, 15(3):529–540, 2003.

[20] Graham Cormode, Howard J. Karloff, and Anthony Wirth. Set cover algorithms for very large datasets. In *CIKM*, pages 479–488, 2010.

[21] Rafael da Ponte Barbosa, Alina Ene, Huy L. Nguyen, and Justin Ward. A new framework for distributed submodular maximization. *CoRR*, abs/1507.03719, 2015.

[22] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[23] Erik D Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. On streaming and communication complexity of the set cover problem. In *DISC*, pages 484–498, 2014.

[24] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014.

[25] Yuval Emek and Adi Rosén. Semi-streaming set cover. In *ICALP*, pages 453–464, 2014.

[26] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

[27] Hossein Esfandiari, Mohammad T Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *SODA*, pages 1217–1233, 2015.

[28] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, 2005.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[30] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[31] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards tight bounds for the streaming set cover problem. In *PODS*, 2016.

[32] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab Mirrokni. Composable core-sets for diversity and coverage maximization. In *PODS*, 2014.

[33] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *SODA*, pages 734–751, 2014.

[34] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *SODA*, pages 1263–1282, 2015.

[35] Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. In *STACS*, pages 440–451, 2011.

[36] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *APPROX*, pages 231–242, 2012.

[37] Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. In *ICALP*, pages 637–649, 2013.

[38] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in MapReduce and streaming. In *SPAA*, pages 1–10, 2013.

[39] Jure Leskovec, Dan Huttenlocher, and Jon Kleinberg. Governance in social media: A case study of the Wikipedia promotion process. In *AAAI*, 2010.

[40] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[41] Vahab S. Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *STOC*, pages 153–162, 2015.

[42] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. *CoRR*, abs/1409.7938, 2014.

[43] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *NIPS*, 2015.

[44] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *NIPS*, pages 3594–3602, 2016.

[45] Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[46] Noam Nisan. The communication complexity of approximate set packing and covering. In *ICALP*, pages 868–875, 2002.

[47] Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *SDM*, volume 9, pages 697–708, 2009.

[48] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, 2012.