Fractional Set Cover in the Streaming Model*

Piotr Indyk[1], Sepideh Mahabadi[2], Ronitt Rubinfeld[3], Jonathan Ullman[4], Ali Vakilian[5], and Anak Yodpinyanee[6]

1 **CSAIL, MIT, Cambridge, MA, USA**
indyk@mit.edu
2 **CSAIL, MIT, Cambridge, MA, USA**
mahabadi@mit.edu
3 **CSAIL, MIT and TAU, Cambridge, MA, USA**
ronitt@csail.mit.edu
4 **CCIS, Northeastern University, Boston, MA, USA**
jullman@ccs.neu.edu
5 **CSAIL, MIT, Cambridge, MA, USA**
vakilian@mit.edu
6 **CSAIL, MIT, Cambridge, MA, USA**
anak@mit.edu

Abstract

We study the *Fractional Set Cover* problem in the streaming model. That is, we consider the relaxation of the set cover problem over a universe of n elements and a collection of m sets, where each set can be picked fractionally, with a value in $[0, 1]$. We present a randomized $(1+\varepsilon)$-approximation algorithm that makes p passes over the data, and uses $\widetilde{O}(mn^{O(1/p\varepsilon)} + n)$ memory space. The algorithm works in both the set arrival and the edge arrival models. To the best of our knowledge, this is the first streaming result for the fractional set cover problem. We obtain our results by employing the multiplicative weights update framework in the streaming settings.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Streaming Algorithms, Fractional Set Cover, LP relaxation, Multiplicative Weight Update

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.12

1 Introduction

Set Cover is one of the classical NP-hard problems in combinatorial optimization. In this problem the input consists of a set (universe) of n elements $\mathcal{U} = \{e_1, \cdots, e_n\}$ and a collection of m sets $\mathcal{F} = \{S_1, \cdots, S_m\}$. The goal is to find the minimum size *set cover* of $\mathcal{U}$, i.e., a collection of sets in $\mathcal{F}$ whose union is $\mathcal{U}$. The LP relaxation of Set Cover (called SetCover-LP) is also well-studied. It is a continuous relaxation of the problem where each set $S \in \mathcal{F}$ can be selected "fractionally", i.e., assigned a number x_S from $[0, 1]$, such that for each element e its "fractional coverage" $\sum_{S:e\in S} x_S$ is at least 1, and the sum $\sum_S x_S$ is minimized. Both variants are well-studied and have many applications in operations research [23, 25, 11], information retrieval and data mining [34], learning theory [26], web host analysis [15], etc.

* PI, SM and AV are supported by grants from the NSF and the Simons Investigator award. RR is supported by the Israel Science Foundation (ISF) grant 1536/14, the NSF grants CCF-1420692 and CCF-1650733. AY is supprted by the NSF grants CCF-1420692, CCF-1650733, CCF-1065125, and the DPST scholarship, Royal Thai Government.

A natural $\ln n$-approximation greedy algorithm of Set Cover, which in each iteration picks the *best* remaining set, is widely used and known to be the best possible under $\mathbf{P} \neq \mathbf{NP}$ [29, 21, 33, 5, 31, 18]. However, the greedy algorithm is sequential in nature and does not perform efficiently in the standard models developed for *massive data analysis*; in particular, in the *streaming* model. In streaming Set Cover [34], the ground set $\mathcal{U}$ is stored in the memory, the sets $S_1, \cdots, S_m$ are stored consecutively in a read-only repository and the algorithm can only access the sets by performing sequential scans (or passes) over the repository. Moreover, the amount of (read-write) memory available to the algorithm is much smaller than the input size (which can be as large as mn). The objective is to design a *space-efficient* algorithm that returns a (nearly)-optimal feasible cover of $\mathcal{U}$ after performing only a few passes over the data. Streaming Set Cover has witnessed a lot of developments in recent years, and tight upper and lower bounds are known, in both *low space* [20, 13] and *low approximation* [17, 24, 8, 12, 7] regimes.

Despite the above developments, the results for the *fractional* variant of the problem are still unsatisfactory. To the best of our knowledge, it is not known whether there exists an efficient and accurate algorithm for this problem that uses only a logarithmic (or even a *poly logarithmic*) number of passes. This state of affairs is perhaps surprising, given the many recent developments on fast LP solvers [27, 37, 28, 4, 3, 35]. To the best of our knowledge, the only prior results on streaming Packing/Covering LPs were presented in paper [1], which studied the LP relaxation of Maximum Matching.

1.1 Our Results

In this paper, we present the first $(1+\varepsilon)$-approximation algorithm for the fractional Set Cover in the streaming model with constant number of passes. Our algorithm performs p passes over the data stream and uses $\widetilde{O}(mn^{O(\frac{1}{p\varepsilon})}+n)$ memory space to return a $(1+\varepsilon)$ approximate solution of the LP relaxation of Set Cover for positive parameter $\varepsilon \leq 1/2$.

We emphasize that similarly to the previous work on variants of Set Cover in streaming setting, our result also holds for the *edge arrival* stream in which the pair of (S_i, e_j) (edges) are stored in the read-only repository and all elements of a set are not necessarily stored consecutively.

1.2 Related work

Set Cover Problem. The Set Cover problem was first studied in the streaming model in [34], which presented an $O(\log n)$-approximation algorithm in $O(\log n)$ passes and using $\widetilde{O}(n)$ space. This approximation factor and the number of passes can be improved to $O(\log n)$ by adapting the greedy algorithm *thresholding* idea presented in [16] . In the low space regime ($\widetilde{O}(n)$ space), Emek and Rosen [20] designed a *deterministic* single pass algorithm that achieves an $O(\sqrt{n})$-approximation. This is provably the best guarantee that one can hope for in a single pass even considering randomized algorithms. Later Chakrabarti and Wirth [13] generalized this result and provided a *tight* trade-off bounds for Set Cover in multiple passes. More precisely, they gave an $O(pn^{1/(p+1)})$-approximate algorithm in p-passes using $\widetilde{O}(n)$ space and proved that this is the best possible approximation ratio up to a factor of $\text{poly}(p)$ in p passes and $\widetilde{O}(n)$ space.

A different line of work started by Demaine et al. [17] focused on designing a "low" approximation algorithm (between $\Theta(1)$ and $\Theta(\log n)$) in the smallest possible amount of space. In contrast to the results in the $\widetilde{O}(n)$ space regime, [17] showed that randomness is necessary: any constant pass deterministic algorithm requires $\Omega(mn)$ space to achieve constant

approximation guarantee. Further, they provided a $O(4^p \log n)$-approximation algorithm that makes $O(4^p)$ passes and uses $\widetilde{O}(mn^{1/p} + n)$. Later Har-Peled et al. [24] improved the algorithm to a $2p$-pass $O(p \log n)$-approximation with memory space $\widetilde{O}(mn^{1/p} + n)$[1]. The result was further improved by Bateni et al. where they designed a p-pass algorithm that returns a $(1 + \varepsilon) \log n$-approximate solution using $mn^{\Theta(1/p)}$ memory [12].

As for the lower bounds, Assadi et al. [8] presented a lower bound of $\Omega(mn/\alpha)$ memory for any single pass streaming algorithm that computes a α-approxime solution. For the problem of estimating the size of an optimal solution they prove $\Omega(mn/\alpha^2)$ memory lower bound. For both settings, they complement the results with matching tight upper bounds. Very recently, Assadi [7] proved a lower bound for streaming algorithms with multiple passes which is tight up to polylog factors: any α-approximation algorithm for Set Cover requires $\Omega(mn^{1/\alpha})$ space, even if it is allowed polylog(n) passes over the stream, and even if the sets are arriving in a random order in the stream. Further, [7] provided the matching upper bound: a $(2\alpha + 1)$-pass algorithm that computes a $(\alpha + \varepsilon)$-approximate solution in $\widetilde{O}(\frac{mn^{1/\alpha}}{\varepsilon^2} + \frac{n}{\varepsilon})$ memory (assuming exponential computational resource).

Max Cover Problem. The first result on streaming Max k-Cover showed how to compute a $(1/4)$-approximate solution in one pass using $\widetilde{O}(kn)$ space [34]. It was improved by Badanidiyuru et al. [9] to a $(1/2 - \varepsilon)$-approximation algorithm that requires $\widetilde{O}(n/\varepsilon)$ space. Moreover, their algorithm works for a more general problem of Submodular Maximization with cardinality constraints. This result was later generalized for the problem of non-monotone submodular maximization under constraints beyond cardinality [14]. Recently, McGregor and Vu [30] and Bateni et al. [12] independently obtained single pass $(1 - 1/e - \varepsilon)$-approximation with $\widetilde{O}(m/\varepsilon^2)$ space. On the lower bound side, [30] showed a lower bound of $\widetilde{\Omega}(m)$ for constant pass algorithm whose approximation is better than $(1 - 1/e)$. Moreover, [7] proved that any streaming $(1 - \varepsilon)$-approximation algorithm of Max k-Cover in polylog(n) passes requires $\widetilde{\Omega}(m/\varepsilon^2)$ space even on random order streams and the case $k = O(1)$. This bound is also complemented by the $\widetilde{O}(mk/\varepsilon^2)$ and $\widetilde{O}(m/\varepsilon^3)$ algorithms of [12, 30]. For more detailed survey of the results on streaming Max k-Cover refer to [12, 30, 7].

Covering/Packing LPs. The study of LPs in streaming model was first discussed in the work of Ahn and Guha [1] where they used *multiplicative weights update* (MWU) based techniques to solve the LP relaxation of Maximum (Weighted) Matching problem. They used the fact that MWU returns a near optimal fractional solution with small size support: first they solve the fractional matching problem, then solve the actual matching only considering the edges in the support of the returned fractional solution.

Our algorithm is also based on the MWU method, which is one of the main key techniques in designing fast approximation algorithms for Covering and Packing LPs [32, 36, 22, 6]. We note that the MWU method has been previously studied in the context of *streaming* and *distributed* algorithms, leading to efficient algorithms for a wide range of graph optimization problems [1, 10, 2].

For a related problem, *covering integer LP* (covering ILP), Assadi et al. [8] designed a one pass streaming algorithm that estimates the optimal solution of $\{\min \mathbf{c}^\top \mathbf{x} \mid \mathbf{A}^\top \mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \{0,1\}^n\}$ within a factor of α using $\widetilde{O}(\frac{mn}{\alpha^2} \cdot b_{\max} + m + n \cdot b_{\max})$ where $b_{\max}$ denotes

[1] In streaming model, space complexity is of interest and one can assume exponentital computation power. In this case the algorithms of [17, 24] save a factor of $\log n$ in the approximation ratio.

the largest entry of $\mathbf{b}$. In this problem, they assume that columns of $\mathbf{A}$, constriants, are given one by one in the stream.

In a different regime, [19] studied approximating the feasibility LP in streaming model with additive approximation. Their algorithm performs two passes and is most efficient when the input is dense.

1.3 Our Techniques

Preprocessing. Let k denote the value of the optimal solution. The algorithm starts by picking a uniform *fractional* vector (each entry of value $O(\frac{k}{m})$) which covers all frequently occurring elements (those appearing in $\Omega(\frac{m}{k})$ sets), and updates the uncovered elements in one pass. This step considerably reduces the memory usage as the uncovered elements have now lower occurrence (roughly $\frac{m}{k}$). Note that we do not need to assume the knowledge of the correct value k: in parallel we try all powers of $(1+\varepsilon)$, denoting our guess by ℓ.

Multiplicative Weight Update. To cover the remaining elements, we employ the MWU framework and show how to implement it in the streaming setting. In each iteration of MWU, we have a probability distribution $\mathbf{p}$ corresponding to the constraints (elements) and we need to satisfy the *average* covering constraint. More precisely, we need an *oracle* that assigns values to x_S for each set S so that $\sum_S p_S x_S \geq 1$ subject to $\|\mathbf{x}\|_1 \leq \ell$, where p_S is the sum of probabilities of the elements in the set S. Then, the algorithm needs to update $\mathbf{p}$ according to the amount each element has been covered by the oracle's solution. The simple greedy realization of the oracle can be implemented in the streaming setting efficiently by computing all p_S while reading the stream in one pass, then choosing the heaviest set (i.e., the set with largest p_S) and setting its x_S to ℓ. This approach works, except that the number of rounds T required by the MWU framework is large. In fact, $T = \Omega(\frac{\phi \log n}{\varepsilon^2})$, where ϕ is the width parameter (the maximum amount an oracle solution may over-cover an element), which is $\Theta(\ell)$ in this naïve realization. Next, we show how to decrease T in two steps.

Step 1. A first hope would be that there is a more efficient implementation of the oracle which gives a better width parameter. Nonetheless, no matter how the oracle is implemented, if all sets in $\mathcal{F}$ contain a fixed element e, then the width is inevitably $\Omega(\ell)$. This observation implies that we need to work with a different set system that has small width, but at the same time, it has the same objective value as of the optimal solution. Consequently, we consider the *extended set system* where we replace $\mathcal{F}$ with all subsets of the sets in $\mathcal{F}$. This extended system preserves the optimality, and under this system we may avoid over-covering elements and obtain $T = O(\log n)$ (for constant ε).

In order to turn a solution in our set system into a solution in the extended set system with small width, we need to remove the repeated elements from the sets in the solution so that every covered element appears exactly once, and thereby getting constant width. However, as a side effect, this reduces the total weight of the solution ($\sum_{S \in \text{SOL}} p_S x_S$), and thus the average covering constraint might not be satisfied anymore. In fact, we need to come up with a guarantee that, on one hand, is preserved under the pruning step, and on the other hand, implies that the solution has large enough total weight

Therefore, to fulfill the average constraint under the pruning step, the oracle must instead solve the *maximum coverage* problem: given a budget, choose sets to cover the largest (fractional) amount of elements. We first show that this problem can be solved approximately via the MWU framework using the simple oracle that picks the heaviest set, but this MWU algorithm still requires T passes over the data. To improve the number of

SetCover-LP $\langle\langle$*Input:* $\mathcal{U}, \mathcal{F}\rangle\rangle$

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{F}} x_S & \\ \text{subject to} \quad & \sum_{S: e \in S} x_S \geq 1 & \forall e \in \mathcal{U} \\ & x_S \geq 0 & \forall S \in \mathcal{F} \end{aligned}$$

Figure 1 LP relaxation of Set Cover.

passes, we perform *element sampling* and apply the MWU algorithm to find an approximate maximum coverage of a small number of sampled elements, whose subproblem can be stored in memory. Fortunately, while the number of fractional solutions to maximum coverage is unbounded, by exploiting the structure of the solutions returned by the MWU method, we can limit the number of plausible solutions of this oracle and approximately solve the average constraint, thereby reducing the space usage to $\widetilde{O}(m)$ for a $O(\frac{\log n}{\varepsilon^2})$-pass algorithm.

Step 2. To further reduce the number of required passes, we observe that the weights of the constraints change slowly. Thus, in a single pass, we can sample the elements for multiple rounds in advance, and then perform rejection (sub-)sampling to obtain an unbiased set of samples for each subsequent round. This will lead to a streaming algorithm with p passes and $mn^{O(1/p)}$ space.

Extension. We also extend our result to handle general covering LPs. More specifically, in the LP relaxation of Set Cover, maximize $\mathbf{c}^\top \mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$, $\mathbf{A}$ has entries from $\{0, 1\}$ whereas entries of $\mathbf{b}$ and $\mathbf{c}$ are all ones. If the non-zero entries instead belong to a range $[1, M]$, we increase the number of sampled elements by $\text{poly}(M)$ to handle discrepancies between coefficients, leading to a $\text{poly}(M)$-multiplicative overhead in the space usage.

2 MWU Framework of the Streaming Algorithm for Fractional Set Cover

In this section, we present a basic streaming algorithm that computes a $(1+\varepsilon)$-approximate solution of the LP-relaxation of Set Cover for any $\varepsilon > 0$ via the MWU framework. We will, in the next section, improve it into an efficient algorithm that achieves the claimed $O(p)$ passes and $\widetilde{O}(mn^{1/p})$ space complexity.

Let $\mathcal{U}$ and $\mathcal{F}$ be the ground set of elements and the collection of sets, respectively, and recall that $|\mathcal{U}| = n$ and $|\mathcal{F}| = m$. Let $\mathbf{x} \in \mathbb{R}^m$ be a vector indexed by the sets in $\mathcal{F}$, where x_S denotes the value assigned to the set S. Our goal is to compute an approximate solution to the LP in Figure 1. Throughout the analysis we assume $\varepsilon \leq 1/2$, and ignore the case where some element never appears in any set, as it is easy to detect in a single pass that no cover is valid. For ease of reading, we write $\widetilde{O}$ and $\widetilde{\Theta}$ to hide $\text{polylog}(m, n, \frac{1}{\varepsilon})$ factors.

Outline of the algorithm. Let k denote the optimal objective value, and $0 < \varepsilon \leq 1/2$ be a parameter. The outline of the algorithm is shown in **fracSetCover** (Figure 2). This algorithm makes calls to the subroutine **feasibilityTest**, that given a parameter ℓ, with high probability, either returns a solution of objective value at most $(1+\varepsilon/3)\ell$, or detects that the optimal objective value exceeds ℓ. Consequently, we may search for the right value of ℓ by

fracSetCover(ε):
▷ Finds a feasible $(1+\varepsilon)$-approximate solution in $O(\frac{\log n}{\varepsilon})$ iterations
for $\ell \in \{(1+\varepsilon/3)^i \mid 0 \le i \le \log_{1+\varepsilon/3} n\}$ **do in parallel**: $\mathbf{x}_\ell \leftarrow$ **feasibilityTest**$(\ell, \varepsilon/3)$
return x_{ℓ^*} where $\ell^* \leftarrow \min\{\ell : x_\ell \text{ is not INFEASIBLE}\}$

Figure 2 fracSetCover returns a $(1+\varepsilon)$-approximate solution of $SetCover - LP$, where **feasibilityTest** is an algorithm that returns a solution of objective value at most $(1+\varepsilon/3)\ell$ when $\ell \ge k$.

considering all values in $\{(1+\varepsilon/3)^i \mid 0 \le i \le \log_{1+\varepsilon/3} n\}$. As for some value of ℓ it holds that $k \le \ell \le k(1+\varepsilon/3)$, we obtain a solution of size $(1+\varepsilon/3)\ell \le (1+\varepsilon/3)(1+\varepsilon/3)k \le (1+\varepsilon)k$ which gives an approximation factor $(1+\varepsilon)$. This whole process of searching for k increases the space complexity of the algorithm by at most a multiplicative factor of $\log_{1+\varepsilon/3} n \approx \frac{3\log n}{\varepsilon}$.

The **feasibilityTest** subroutine employs the multiplicative weights update method (MWU) which is described next.

2.1 Preliminaries of the MWU method for solving covering LPs

In the following, we describe the MWU framework. The claims presented here are standard results of the MWU method. For more details, see e.g. Section 3 of [6]. Note that we introduce the general LP notation as it simplifies the presentation later on.

Let $\mathbf{Ax} \ge \mathbf{b}$ be a set of linear constraints, and let $\mathcal{P} \triangleq \{\mathbf{x} \in \mathbb{R}^m : \mathbf{x} \ge \mathbf{0}\}$ be the polytope of the non-negative orthant. For a given error parameter $0 < \beta < 1$, we would like to solve an approximate version of the feasibility problem by doing one of the following:

- Compute $\hat{\mathbf{x}} \in \mathcal{P}$ such that $\mathbf{A}_i\hat{\mathbf{x}} - b_i \ge -\beta$ for *every* constraint i.
- Correctly report that the system $\mathbf{Ax} \ge \mathbf{b}$ has no solution in $\mathcal{P}$.

The MWU method solves this problem assuming the existence of the following oracle that takes a distribution $\mathbf{p}$ over the constraints and finds a solution $\hat{\mathbf{x}}$ that satisfies the constraints on average over $\mathbf{p}$.

▶ **Definition 2.1.** Let $\phi \ge 1$ be a width parameter and $0 < \beta < 1$ be an error parameter. A $(1,\phi)$-*bounded* $(\beta/3)$-*approximate oracle* is an algorithm that takes as input a distribution $\mathbf{p}$ and does one of the following:

- Returns a solution $\hat{\mathbf{x}} \in \mathcal{P}$ satisfying
  - $\mathbf{p}^\top\mathbf{A}\hat{\mathbf{x}} \ge \mathbf{p}^\top\mathbf{b} - \beta/3$, and
  - $\mathbf{A}_i\hat{\mathbf{x}} - b_i \in [-1,\phi]$ for *every* constraint i.
- Correctly reports that the inequality $\mathbf{p}^\top\mathbf{Ax} \ge \mathbf{p}^\top\mathbf{b}$ has no solution in $\mathcal{P}$.

The MWU algorithm for solving covering LPs involves T rounds. It maintains the (non-negative) weight of each constraint in $\mathbf{Ax} \ge \mathbf{b}$, which measures how much it has been satisfied by the solutions chosen so far. Let $\mathbf{w}^t$ denote the weight vector at the beginning of round t, and initialize the weights to $\mathbf{w}^1 \triangleq \mathbf{1}$. Then, for rounds $t = 1,\ldots,T$, define the probability vector $\mathbf{p}^t$ proportional to those weights $\mathbf{w}^t$, and use the oracle above to find a solution $\mathbf{x}^t$. If the oracle reports that the system $\mathbf{p}^\top\mathbf{Ax} \ge \mathbf{p}^\top\mathbf{b}$ is infeasible, the MWU algorithm also reports that the original system $\mathbf{Ax} \ge \mathbf{b}$ is infeasible, and terminates. Otherwise, define the cost vector incurred by $\mathbf{x}^t$ as $\mathbf{m}^t \triangleq \frac{1}{\phi}(\mathbf{Ax} - \mathbf{b})$, then update the weights so that $w_i^{t+1} \triangleq w_i^t(1 - \beta m_i^t/6)$ and proceed to the next round. Finally, the algorithm returns the average solution $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^t$.

Feasibility-SC-LP $\langle\langle$*Input:* $\mathcal{U}, \mathcal{F}, \ell\rangle\rangle$

$$\sum_{S \in \mathcal{F}} x_S \leq \ell$$

$$\sum_{S: e \in S} x_S \geq 1 \qquad \forall e \in \mathcal{U}$$

$$x_S \geq 0 \qquad \forall S \in \mathcal{F}$$

(a) LP relaxation of Feasibility Set Cover.

Feasibility-Covering-LP $\langle\langle$*Input:* $\mathbf{A}, \mathbf{b}, \mathbf{c}, \ell\rangle\rangle$

$\mathbf{c}^\top \mathbf{x} \leq \ell$	(objective value)
$\mathbf{A}\mathbf{x} \geq \mathbf{b}$	(covering)
$\mathbf{x} \geq \mathbf{0}$	(non-negativity)

(b) LP relaxation of the Feasibility Covering problem.

Figure 3 LP relaxations of the feasibility variant of set cover and general covering problems.

The MWU theorem (e.g., Theorem 3.5 of [6]) shows that $T = O(\frac{\phi \log n}{\beta^2})$ is sufficient to correctly solve the problem, yielding $\mathbf{A}_i \hat{\mathbf{x}} - b_i \geq -\beta$ for every constraint, where n is the number of constraints. In particular, the algorithm requires T calls to the oracle.

▶ **Theorem 2.2** (MWU Theorem [6]). *For every $0 < \beta < 1, \phi \geq 1$ the* MWU *algorithm either solves the $Feasibility - Covering - LP$ problem up to an additive error of β (i.e., solves $\mathbf{A}_i \mathbf{x} - b_i \geq -\beta$ for every i) or correctly reports that the LP is infeasible, making only $O(\frac{\phi \log n}{\beta^2})$ calls to a $(1, \phi)$-bounded $\beta/3$-approximate oracle of the LP.*

2.2 Semi Streaming MWU-based algorithm for factional Set Cover

Setting up our MWU algorithm. As described in the overview, we wish to solve, as a subroutine, the decision variant of $SetCover - LP$ known as $Feasibility - SC - LP$ given in Figure 3a, where the parameter ℓ serves as the guess for the optimal objective value.

To follow the conventional notation for solving LPs in the MWU framework, consider the more standard form of covering LPs denoted as Feasibility-Covering-LP given in Figure 3b. For our purpose, $\mathbf{A}_{n \times m}$ is the element-set incidence matrix indexed by $\mathcal{U} \times \mathcal{F}$; that is, $A_{e,S} = 1$ if $e \in S$, and $A_{e,S} = 0$ otherwise. The vectors $\mathbf{b}$ and $\mathbf{c}$ are both all-ones vectors indexed by $\mathcal{U}$ and $\mathcal{F}$, respectively. We emphasize that, unconventionally for our system $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, there are n constraints (i.e. elements) and m variables (i.e. sets).

Employing the MWU approach for solving covering LPs, we define the polytope

$$\mathcal{P}_\ell \triangleq \{\mathbf{x} \in \mathbb{R}^m : \mathbf{c}^\top \mathbf{x} \leq \ell \text{ and } \mathbf{x} \geq \mathbf{0}\}.$$

Observe that by applying the MWU algorithm to this polytope $\mathcal{P}$ and constraints $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, we obtain a solution $\bar{\mathbf{x}} \in \mathcal{P}_\ell$ such that $\mathbf{A}_e \left(\frac{\bar{\mathbf{x}}}{1-\beta}\right) \geq \frac{b_e - \beta}{1-\beta} = 1 = b_e$, where $\mathbf{A}_e$ denotes the row of $\mathbf{A}$ corresponding to e. This yields a $(1 + O(\varepsilon))$-approximate solution for $\beta = O(\varepsilon)$.

Unfortunately, we cannot implement the MWU algorithm on the full input under our streaming context. Therefore, the main challenge is to implement the following two subtasks of the MWU algorithm in the streaming settings. First, we need to design an oracle that solves the average constraint in the streaming setting. Moreover, we need to be able to efficiently update the weights for the subsequent rounds.

Covering the common elements. Before we proceed to applying the MWU framework, we add a simple first step to our implementation of **feasibilityTest** (Figure 4) that will greatly reduce the amount of sapce required in implementing the MWU algorithm. This can be interpreted as the fractional version of Set Sampling described in [17]. In our subroutine, we partition the elements into the common elements that occur more frequently, which will be

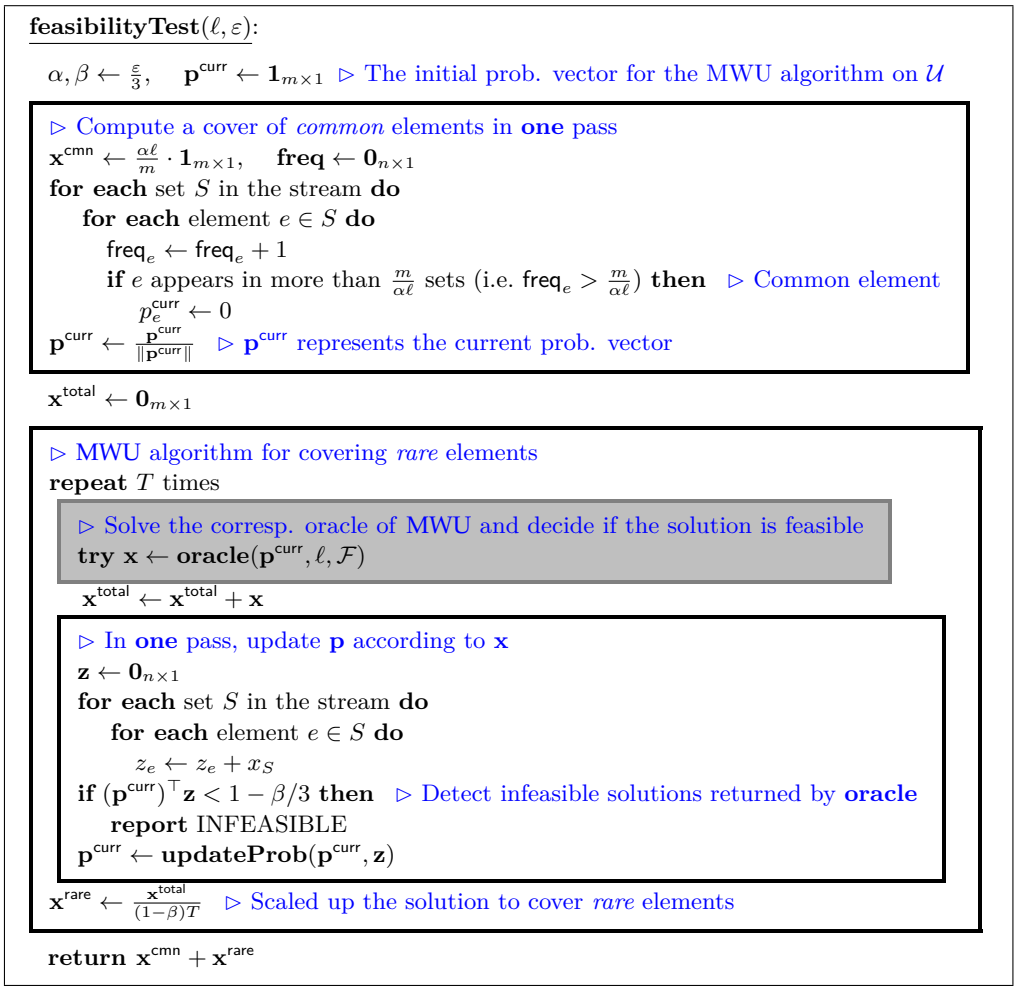

Figure 4 A generic implementation of **feasibilityTest**. Its performance depend on the implementations of **oracle**, **updateProb**. We will investigate different implementations of **oracle** in the gray box.

covered if we simply choose a uniform vector solution, and the rare elements that occur less frequently, for which we perform the MWU algorithm to compute a good solution. In one pass we can find all frequently occurring elements by counting the number of sets containing each element. The amount of required space to perform this task is $O(n \log m)$.

We call an element that appears in at least $\frac{m}{\alpha\ell}$ sets *common*, and we call it *rare* otherwise, where $\alpha = \Theta(\varepsilon)$. Since we are aiming for a $(1+\varepsilon)$-approximation, we can define $\mathbf{x}^{\mathsf{cmn}}$ as a vector whose all entries are $\frac{\alpha\ell}{m}$. The total cost of $\mathbf{x}^{\mathsf{cmn}}$ is $\alpha\ell$ and all common elements are covered by $\mathbf{x}^{\mathsf{cmn}}$. Thus, throughout the algorithm we may restrict our attention to the rare elements.

Our goal now is to construct an efficient MWU-based algorithm, which finds a solution $\mathbf{x}^{\mathsf{rare}}$ covering the rare elements, with objective value at most $\frac{\ell}{1-\beta} \leq (1+\varepsilon-\alpha)\ell$. We note that our implementation does not explicitly maintain the weight vector $\mathbf{w}^t$ described in Section 2.1, but instead updates (and normalizes) its probability vector $\mathbf{p}^t$ in every round.

heavySetOracle$(\mathbf{p}, \ell, \mathcal{F})$:
Compute p_S for every $S \in \mathcal{F}$ while reading the set system ▷ either from stream or memory
$S^* \leftarrow \mathbf{argmax}_{S \in \mathcal{F}} p_S$
if $p_S < (1 - \beta/3)/\ell$ **then report** INFEASIBLE
$\mathbf{x} \leftarrow \mathbf{0}_{n \times 1}, x_S \leftarrow \ell$
return $\mathbf{x}$

Figure 5 heavySetOracle computes p_S of every set given the set system in a stream or stored memory, then returns the solution $\mathbf{x}$ that optimally places value ℓ on the corresponding entry. It reports INFEASIBLE if there is no sufficiently good solution, concluding that the set system is infeasible.

2.3 First Attempt: Simple Oracle and Large Width

A greedy solution for the oracle. We implement the oracle for MWU algorithm such that $\phi = \ell$, and thus requiring $\Theta(\ell \log n/\beta^2)$ iterations (Theorem 2.2). In each iteration, we need an oracle that finds some solution $\mathbf{x} \in \mathcal{P}_\ell$ satisfying $\mathbf{p}^\top \mathbf{A}\mathbf{x} \geq \mathbf{p}^\top \mathbf{b} - \beta/3$, or decides that no solution in $\mathcal{P}_\ell$ satisfies $\mathbf{p}^\top \mathbf{A}\mathbf{x} \geq \mathbf{p}^\top \mathbf{b}$.

Observe that $\mathbf{p}^\top \mathbf{A}\mathbf{x}$ is maximized when we place value ℓ on x_{S^*} where S^* achieves the maximum value $p_S \triangleq \sum_{e \in S} p_e$. Further, for our application, $\mathbf{b} = \mathbf{1}$ so $\mathbf{p}^\top \mathbf{b} = 1$. Our implementation **heavySetOracle** of **oracle** given in Figure 5 below is a deterministic greedy algorithm that finds a solution based on this observation. As $\mathbf{A}_e \mathbf{x} \leq \|\mathbf{x}\|_1 \leq \ell$, **heavySetOracle** implements a $(1, \ell)$-bounded $(\beta/3)$-approximate oracle. Therefore, the implementation of **feasibilityTest** with **heavySetOracle** computes a solution of objective value at most $(\alpha + \frac{1}{1-\beta})\ell < (1 + \frac{\varepsilon}{3})\ell$ when $\ell \geq k$ as promised.

Finally, we track the space usage which concludes the complexities of the current version of our algorithm: it only stores vectors of length m or n, whose entries each requires a logarithmic number of bits, yielding the following theorem.

▶ **Theorem 2.3.** *There exists a streaming algorithm that w.h.p. returns a $(1+\varepsilon)$-approximate fractional solution of SetCover $-$ LP$(\mathcal{U}, \mathcal{F})$ in $O(\frac{k \log n}{\varepsilon^2})$ passes and using $\widetilde{O}(m+n)$ memory for any positive $\varepsilon \leq 1/2$. The algorithm works in both set arrival and edge arrival streams.*

The presented algorithm suffers from large number of passes over the input. In particular, we are interested in solving the fractional Set Cover in constant number of passes using sublinear space. To this end, we first reduce the required number of rounds in MWU by a more complicated implementation of **oracle**.

3 Max Cover Problem and its Application to Width Reduction

In this section, we improve the described algorithm in the previous section and prove the following result.

▶ **Theorem 3.1.** *There exists a streaming algorithm that w.h.p. returns a $(1+\varepsilon)$-approximate fractional solution of SetCover $-$ LP$(\mathcal{U}, \mathcal{F})$ in p passes and uses $\widetilde{O}(mn^{O(1/p\varepsilon)} + n)$ memory for any $2 \leq p \leq \text{polylog}(n)$ and $0 < \varepsilon \leq 1/2$. The algorithm works in both set arrival and edge arrival streams.*

Recall that in implementing **oracle**, we must find a solution $\mathbf{x}$ of total size $\|\mathbf{x}\|_1 \leq \ell$ with a sufficiently large weight $\mathbf{p}^\top \mathbf{A}\mathbf{x}$. Our previous implementation chooses only one good entry x_S and places its entire *budget* ℓ on this entry. As the width of the solution is roughly the

maximum amount an element is over-covered by $\mathbf{x}$, this implementation induces a width of ℓ. In this section, we design an oracle that returns a solution in which the budget is distributed more evenly among the entries of $\mathbf{x}$ to reduce the width. To this end, we design an implementation of **oracle** of the MWU approach based on the Max ℓ-Cover problem (whose precise definition will be given shortly). The solution to our Max ℓ-Cover aids in reducing the width of our **oracle** solution to a constant, so the required number of rounds of the MWU algorithm decreases to $O(\frac{\log n}{\varepsilon^2})$, independent of ℓ. Note that, if the objective value of an optimal solution of Set Cover$(\mathcal{U}, \mathcal{F})$ is ℓ, then a solution of width $o(\ell)$ may not exist, as shown in Lemma 3.2 (whose proof is given in Section A.1). This observation implies that we need to work with a different set system. Besides having small width, an optimal solution of the Set Cover instance on the new set system should have the same objective value of the optimal solution of Set Cover$(\mathcal{U}, \mathcal{F})$.

▶ **Lemma 3.2.** *There exists a set system in which, under the direct application of the MWU framework in computing a $(1+\varepsilon)$-approximate solution, induces width $\phi = \Omega(k)$, where k is the optimal objective value. Moreover, the exists a set system in which the approach from the previous section (which handles the frequent and rare elements differently) has width $\phi = \Theta(n) = \Theta(\sqrt{m/\varepsilon})$.*

Extended Set System. First, we consider the *extended set system* $(\mathcal{U}, \breve{\mathcal{F}})$, where $\breve{\mathcal{F}}$ is the collection containing all subsets of sets in $\mathcal{F}$; that is,

$$\breve{\mathcal{F}} \triangleq \{R : R \subseteq S \text{ for some } S \in \mathcal{F}\}.$$

It is straightforward to see that the optimal objective value of Set Cover over $(\mathcal{U}, \breve{\mathcal{F}})$ is equal to that of $(\mathcal{U}, \mathcal{F})$: we only add subsets of the original sets to create $\breve{\mathcal{F}}$, and we may replace any subset from $\breve{\mathcal{F}}$ in our solution with its original set in $\mathcal{F}$. Moreover, we may *prune* any collection of sets from $\mathcal{F}$ into a collection from $\breve{\mathcal{F}}$ of the same cardinality so that, this pruned collection not only covers the same elements, but also each of these elements is covered exactly once. This extended set system is defined for the sake of analysis only: we will never explicitly handle an exponential number of sets throughout our algorithm.

We define *ℓ-cover* as a collection of sets of total weight ℓ. Although the pruning of an ℓ-cover reduces the width, the total weight $\mathbf{p}^\top \mathbf{A}\mathbf{x}$ of the solution will decrease. Thus, we consider the weighted constraint of the form

$$\sum_{e \in \mathcal{U}} \left(p_e \cdot \min\{1, \sum_{S : e \in S} x_S\} \right) \geq 1;$$

that is, we can only gain the value p_e without any multiplicity larger than 1. The problem of maximizing the left hand side is known as the *weighted max coverage* problem: for a parameter ℓ, find an ℓ-cover such that the total value p_e's of the covered elements is maximized.

3.1 The Maximum Coverage Problem

In the design of our algorithm, we consider the *weighted* Max k-Cover problem, which is closely related to Set Cover. Extending upon the brief description given earlier, we fully specify the LP relaxation of this problem. In the weighted Max k-Cover$(\mathcal{U}, \mathcal{F}, \ell, \mathbf{p})$, given a ground set of elements $\mathcal{U}$, a collection of sets $\mathcal{F}$ over the ground set, a budget parameter ℓ, and a weight vector $\mathbf{p}$, the goal is to return ℓ sets in $\mathcal{F}$ whose weighted *coverage*, the total weight of all covered elements, is maximized. Moreover, since we are aiming for a

MaxCover-LP $\langle\langle$*Input:* $\mathcal{U}, \mathcal{F}, \ell, \mathbf{p}\rangle\rangle$

$$\begin{aligned}
\text{maximize} \quad & \sum_{e\in\mathcal{U}} p_e z_e \\
\text{subject to} \quad & \sum_{S: e\in S} x_S \geq z_e \qquad \forall e \in \mathcal{U} \\
& \sum_{S\in\mathcal{F}} x_S = \ell \\
& 0 \leq z_e \leq 1 \qquad \forall e \in \mathcal{U} \\
& x_S \geq 0 \qquad \forall S \in \mathcal{F}
\end{aligned}$$

Figure 6 LP relaxation of weighted Max k-Cover.

fractional solution of Set Cover, we consider the LP relaxation of weighted Max k-Cover, $MaxCover - LP$ (see Figure 6); in this LP relaxation, z_e denotes the fractional amount that an element is covered, and hence is capped at 1.

As an intermediate goal, we aim to compute an approximate solution of $MaxCover - LP$, given that the optimal solution covers all elements in the ground set, or to correctly detect that no solution has weighted coverage of more than $(1 - \varepsilon)$. In our application, the vector $\mathbf{p}$ is always a probability vector: $\mathbf{p} \geq \mathbf{0}$ and $\sum_{e\in\mathcal{U}} p_e = 1$. We make the following useful observation.

▶ **Observation 3.3.** *Let k be the value of an optimal solution of $SetCover - LP(\mathcal{U}, \mathcal{F})$ and let $\mathbf{p}$ be an arbitrary probability vector over the ground set. Then there exists a fractional solution of $MaxCover - LP(\mathcal{U}, \mathcal{F}, \ell, \mathbf{p})$ whose weighted coverage is one if $\ell \geq k$.*

δ-integral near optimal solution of MaxCover-LP. Our plan is to solve MaxCover-LP over a randomly projected set system, and argue that with high probability this will result in a valid **oracle**. Such an argument requires an application of the union bound over the set of solutions, which is generally of unbounded size. To this end, we consider a more restrictive domain of *δ-integral* solutions: this domain has bounded size, but is still guaranteed to contain a sufficiently good solution.

▶ **Definition 3.4** (δ-integral solution)**.** A fractional solution $\mathbf{x}_{n\times 1}$ of an LP is δ-integral if $\frac{1}{\delta} \cdot \mathbf{x}$ is an integral vector. That is, for each $i \in [n]$, $x_i = v_i\delta$ where each v_i is an integer.

Next we claim that **maxCoverOracle** given in Figure 7 below, which is the MWU algorithm with **heavySetOracle** for solving $MaxCover - LP$, results in a δ-integral solution. The proof of the following lemma is given in Section A.2.

▶ **Lemma 3.5.** *Consider a $MaxCover - LP$ with the optimal objective value* OPT *(where the weights of elements form a probability vector). There exists a $\Theta(\frac{\varepsilon_{\text{MC}}^2}{\log n})$-integral solution of $MaxCover - LP$ whose objective value is at least $(1 - \varepsilon_{\text{MC}})$*OPT*. In particular, if an optimal solution covers all elements $\mathcal{U}$ ($\ell \geq k$),* **maxCoverOracle** *returns a solution whose weighted coverage is at least $1 - \varepsilon_{\text{MC}}$ in polynomial time.*

Pruning a fractional ℓ-cover. In our analysis, we aim to solve the Set Cover problem under the extended set system. We claim that any solution $\mathbf{x}$ with coverage $\mathbf{z}$ in the actual set system may be turned into a pruned solution $\breve{\mathbf{x}}$ in the extended set system that provides

maxCoverOracle$(\mathcal{U}, \mathcal{F}, \ell)$:
  $\mathbf{x} \leftarrow$ MWU solution of Set Cover LP relaxation implemented with **heavySetOracle**
  return $\mathbf{x}$

Figure 7 **maxCoverOracle** returns a fractional ℓ-cover with weighted coverage at least $1 - \beta/3$ w.h.p. if $\ell \geq k$. It provides no guarantee on its behavior if $\ell < k$.

the same coverage $\mathbf{z}$, but satisfies the strict equality $\sum_{\breve{S} \in \breve{\mathcal{F}}: e \in \breve{S}} \breve{x}_{\breve{S}} = z_e$. Since $z_e \leq 1$, the pruned solution satisfies the condition for an oracle with width *one*. We give an algorithm **prune** for pruning $\mathbf{x}$ into $\breve{\mathbf{x}}$ in Section A.3 and only state the property of this algorithm here.

▶ **Lemma 3.6.** *A fractional ℓ-cover $\mathbf{x}$ of $(\mathcal{U}, \mathcal{F})$ can be converted, in polynomial time, to a fractional ℓ-cover $\breve{\mathbf{x}}$ of $(\mathcal{U}, \breve{\mathcal{F}})$ such that for each element e, its coverage $z_e = \sum_{\breve{S} \in \breve{\mathcal{F}}: e \in \breve{S}} \breve{x}_{\breve{S}} = \min(\sum_{S: e \in S} x_S, 1)$.*

We remark that in order to update the weights in the MWU framework, it is sufficient to know the vector $\mathbf{z}$, which has a simple formula given in the lemma above. The actual solution $\breve{\mathbf{x}}$ is not necessary.

3.2 Sampling-Based Oracle for Fractional Max Coverage

In the previous section, we simply needed to compute the values p_S's in order to construct a solution for the **oracle**. Here as we aim to bound the width of **oracle**, our new task is to find a fractional ℓ-cover $\mathbf{x}$ whose weighted coverage is at least $1 - \beta/3$. The *element sampling* technique, which is also known from prior work in streaming Set Cover and Max k-Cover, is to sample a few elements and solve the problem over the sampled elements only. Then, by applying the union bound over all possible candidate solutions, it is shown that w.h.p. a nearly optimal cover of the sampled elements also covers a large fraction of the whole ground set. This argument applies to the aforementioned problems precisely because there are standard ways of bounding the number of all integral candidate solutions (e.g. ℓ-covers).

However, in the fractional setting, there are infinitely many solutions. Consequently, we employ the notion of δ-integral solutions where the number of such solutions is bounded. In Lemma 3.6, we showed that there always exists a δ-integral solution to $MaxCover - LP$ whose coverage is at least a $(1 - \varepsilon_{\text{MC}})$-fraction of an optimal solution. Moreover, the number of all possible solutions is bounded by the number of ways to divide the budget ℓ into ℓ/δ equal parts of value δ and distribute them (possibly with repetition) among m entries:

▶ **Observation 3.7.** *The number of feasible δ-integral solutions to $MaxCover - LP(\mathcal{U}, \mathcal{F}, \ell, \mathbf{p})$ is $O(m^{\ell/\delta})$ for any multiple ℓ of δ.*

Next, we design our algorithm using the element sampling technique: we show that a $(1 - \beta/3)$-approximate solution of $MaxCover - LP$ can be computed using the projection of all sets in $\mathcal{F}$ over a set of elements of size $\Theta(\frac{\ell \log n \log mn}{\beta^4})$ picked according to $\mathbf{p}$. For every fractional solution $(\mathbf{x}, \mathbf{z})$ and subset of elements $\mathcal{V} \subseteq \mathcal{U}$, let $\mathcal{C}_{\mathcal{V}}(\mathbf{x}) \triangleq \sum_{e \in \mathcal{V}} p_e z_e$ denote the coverage of elements in $\mathcal{V}$ where $z_e = \min(1, \sum_{S: e \in S} x_S)$. We may omit the subscript $\mathcal{V}$ in $\mathcal{C}_{\mathcal{V}}$ if $\mathcal{V} = \mathcal{U}$.

The following lemma, which is essentially an extension of the Element Sampling lemma of [17] for our application, MaxCover-LP, shows that a $(1 - \varepsilon_{\text{MC}})$-approximate ℓ-cover over a set of sampled elements of size $\Theta(\ell \log n \log mn/\gamma^4)$ w.h.p. has a weighted coverage of at least $(1 - 2\gamma)(1 - \varepsilon_{\text{MC}})$ if there exists a fractional ℓ-cover whose coverage is 1. Thus,

choosing $\varepsilon_{\text{MC}} = \gamma = \beta/9$ yields the desired guarantee for **maxCoverOracle**, leading to the performance given in Theorem 3.9. The omitted proofs are given in Section A.4-A.5.

▶ **Lemma 3.8.** *Let ε_{MC} and γ be parameters. Consider the $MaxCover - LP(\mathcal{U}, \mathcal{F}, \ell, \mathbf{p})$ with optimal solution of value* OPT*, and let $\mathcal{L}$ be a multi-set of $s = \Theta(\ell \log n \log(mn)/\gamma^4)$ elements sampled independently at random according to the probability vector $\mathbf{p}$. Let $\mathbf{x}^{\text{SOL}}$ be a $(1-\varepsilon_{\text{MC}})$-approximate $\Theta(\frac{\gamma^2}{\log n})$-integral ℓ-cover over the sampled elements. Then with high probability, $\mathcal{C}(\mathbf{x}^{\text{SOL}}) \geq (1-2\gamma)(1-\varepsilon_{\text{MC}})\text{OPT}$.*

▶ **Theorem 3.9.** *There exists a streaming algorithm that w.h.p. returns a $(1+\varepsilon)$-approximate fractional solution of $SetCover - LP(\mathcal{U}, \mathcal{F})$ in $O(\log n/\varepsilon^2)$ passes and uses $\widetilde{O}(m/\varepsilon^6 + n)$ memory for any positive $\varepsilon \leq 1/2$. The algorithm works in both set arrival and edge arrival streams.*

3.3 Final Step: Running Several MWU Rounds Together

We complete our result by further reducing the number of passes at the expense of increasing the required amount of memory, yielding our full algorithm **fastFeasibilityTest** in Figure 8. More precisely, aiming for a p-pass algorithm, we show how to execute $R \triangleq \frac{T}{\Theta(p)} = \Theta(\frac{\log n}{p\beta^2})$ rounds of the MWU algorithm in a single pass. We show that this task may be accomplished with a multiplicative factor of $f \cdot \Theta(\log mn)$ increase in memory usage, where $f \triangleq n^{\Theta(1/(p\beta))}$.

Advance sampling. Consider a sequence of R consecutive rounds $i = 1, \ldots, R$. In order to implement the MWU algorithm for these rounds, we need (multi-)sets of sampled elements $\mathcal{L}_1, \ldots, \mathcal{L}_R$ according to probabilities $\mathbf{p}^1, \ldots, \mathbf{p}^R$, respectively (where $\mathbf{p}^i$ is the probability corresponding to round i). Since the probabilities of subsequent rounds are not known in advance, we circumvent this problem by choosing these sets $\mathcal{L}_i$'s with probabilities according to $\mathbf{p}^1$, but the number of samples in each set will be $|\mathcal{L}_i| = s \cdot f \cdot \Theta(\log mn)$ instead of s. Then, once $\mathbf{p}^i$ is revealed, we sub-sample the elements from $\mathcal{L}_i$ to obtain $\mathcal{L}'_i$ as follow: for a (copy of) sampled element $\hat{e} \in \mathcal{L}_i$, add $\hat{e}$ to $\mathcal{L}'_i$ with probability $\frac{p^i_{\hat{e}}}{p^1_{\hat{e}} f}$; otherwise, simply discard it. Note that it is still left to be shown that the probability above is indeed at most 1.

Since each e was originally sampled with probability p^1_e, then in $\mathcal{L}'_i$, the probability that a sampled element $\hat{e} = e$ is exactly p^i_e/f. By having $f \cdot \Theta(\log mn)$ times the originally required number of samples s in the first place, in expectation we still have $\mathbf{E}[|\mathcal{L}'_i|] = |\mathcal{L}_i| \sum_{e \in \mathcal{U}} \frac{p^i_e}{f} = (s \cdot f \cdot \Theta(\log mn))\frac{1}{f} = s \cdot \Theta(\log mn)$. Due to the $\Theta(\log mn)$ factor, by the Chernoff bound, we conclude that with w.h.p. $|\mathcal{L}'_i| \geq s$. Thus, we have a sufficient number of elements sampled with probability according to $\mathbf{p}^i$ to apply Lemma 3.8, as needed.

Change in probabilities. As noted above, we must show that the probability that we sub-sample each element is at most 1; that is, $p^i_e/p^1_e \leq f = n^{\Theta(1/(p\beta))}$ for every element e and every round $i = 1, \ldots, R$. We bound the multiplicative difference between the probabilities of two consecutive rounds as follows (see Section A.6 for proof).

▶ **Lemma 3.10.** *Let $\mathbf{p}$ and $\mathbf{p}'$ be the probability of elements before and after an update. Then for every element e, $p'_e \leq (1 + O(\beta))p_e$.*

Therefore, after $R = \Theta(\frac{\log n}{p\beta^2})$ rounds, the probability of any element may increase by at most a factor of $(1 + O(\beta))^{\Theta(\frac{\log n}{p\beta^2})} \leq e^{\Theta(\frac{\log n}{p\beta})} = n^{\Theta(1/(p\beta))} = f$, as desired. This concludes the proof of Theorem 3.1.

Implementation details. We make a few remarks about the implementation given in Figure 8. First, even though we perform all sampling in advance, the decisions of **maxCoverOracle** do not depend on any $\mathcal{L}_i$ of later rounds, and **updateProb** is entirely deterministic: there is no dependency issue between rounds. Next, we only need to perform **updateProb** on the sampled elements $\mathcal{L} = \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_R$ during the current R rounds. We therefore denote the probabilities with a different vector $\mathbf{q}^i$ over the sampled elements $\mathcal{L}$ only. Probabilities of elements outside $\mathcal{L}$ are not required by **maxCoverOracle** during these rounds, but we simply need to spend one more pass after executing R rounds of MWU to aggregate the new probability vector $\mathbf{p}$ over all (rare) elements. Similarly, since **maxCoverOracle** does not have the ability to verify, during the MWU algorithm, that each solution $\mathbf{x}^i$ returned by the oracle indeed provides a sufficient coverage, we check all of them during this additional pass. Lastly, we again remark that this algorithm operates on the extended set system: the solution $\mathbf{x}$ returned by **maxCoverOracle** has at least the same coverage as $\check{\mathbf{x}}$. While $\check{\mathbf{x}}$ is not explicitly computed, its coverage vector $\mathbf{z}$ can be computed exactly.

3.4 Extension to general covering LPs

We remark that our MWU-based algorithm can be extended to solve a more general class of covering LPs. Consider the problem of finding a vector $\mathbf{x}$ that minimizes $\mathbf{c}^\top \mathbf{x}$ subject to constraints $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. In terms of the Set Cover problem, $A_{e,S} \geq 0$ indicates the multiplicity of an element e in the set S, $b_e > 0$ denotes the number of times we wish e to be covered, and $c_S > 0$ denotes the cost per unit for the set S. Now define

$$L \triangleq \min_{(e,S):A_{e,S}\neq 0} \frac{A_{e,S}}{b_e c_S} \quad \text{and} \quad U \triangleq \max_{(e,S)} \frac{A_{e,S}}{b_e c_S}.$$

Then, we may modify our algorithm to obtain the following result.

▶ **Theorem 3.11.** *There exists a streaming algorithm that w.h.p. returns a $(1+\varepsilon)$-approximate fractional solution to general covering LPs in p passes and using $\widetilde{O}(\frac{mU}{\varepsilon^6 L} \cdot n^{O(\frac{1}{p\varepsilon})} + n)$ memory for any $3 \leq p \leq \mathrm{polylog}(n)$, where parameters L and U are defined above. The algorithm works in both set arrival and edge arrival streams.*

Proof. We modify our algorithm and provide an argument of its correctness as follows. First, observe that we can convert the input LP into an equivalent LP with all entries $b_e = c_S = 1$ by simply replacing each $A_{e,S}$ with $\frac{A_{e,S}}{b_e c_S}$. Namely, let the new parameters be $\mathbf{A}', \mathbf{b}'$ and $\mathbf{c}'$, and we consider the variable $\mathbf{x}'$ where $x'_S = c_S x_S$. It is straightforward to verify that $\mathbf{c}'^\top \mathbf{x}' = \mathbf{c}^\top \mathbf{x}$ and $\mathbf{A}'_e \mathbf{x}' = \frac{\mathbf{A}_e \mathbf{x}}{b_e}$, reducing the LP into the desired case. Thus, we may afford to record $\mathbf{b}$ and $\mathbf{c}$, so that each value $\frac{A_{e,S}}{b_e c_S}$ may be computed on-the-fly. Henceforth we assume that all entries $b_e = c_S = 1$ and $A_{e,S} \in \{0\} \cup [L, U]$. Observe as well that the optimal objective value k may be in the expanded range $[1/U, n/L]$, so the number of guesses must be increased from $\frac{\log n}{\varepsilon}$ to $\frac{\log(nU/L)}{\varepsilon}$.

Next consider the process for covering the rare elements. We instead use a uniform solution $\mathbf{x}^{\mathsf{cmn}} = \frac{\alpha \ell L}{m} \cdot \mathbf{1}$. Observe that if an element occurs in at least $\frac{m}{\alpha \ell L}$ sets, then $\mathbf{A}_e \mathbf{x}^{\mathsf{cmn}} = \sum_{S:e\in S} A_{e,S} \cdot \frac{\alpha \ell}{m} \geq \frac{m}{\alpha \ell L} \cdot L \cdot \frac{\alpha \ell}{m} = 1$. That is, we must adjust our definition so that an element is considered common if it appears in at least $\frac{m}{\alpha \ell L}$ sets. Consequently, whenever we perform element sampling, the required amount of memory to store information of each element increases by a factor of $1/L$.

Next consider Lemma 3.5, where we show an existence of integral solutions via the MWU algorithm with a greedy oracle. As the greedy implementation chooses a set S and places the entire budget ℓ on x_S, the amount of coverage $A_{e,S} x_S$ may be as large as ℓU as $A_{e,S}$ is

fastFeasibilityTest(ℓ, ε):

$\alpha, \beta \leftarrow \frac{\varepsilon}{3}$, $\mathbf{p}^{\mathsf{curr}} \leftarrow \mathbf{1}_{m\times 1}$ ▷ The initial prob. vector for the MWU algorithm on $\mathcal{U}$

Compute a cover of *common* elements in **one** pass ▷ See Fig. 4's **feasibilityTest** block

$\mathbf{x}^{\mathsf{total}} \leftarrow \mathbf{0}_{m\times 1}$

▷ MWU algorithm for covering *rare* elements
repeat p times
$R \leftarrow \Theta(\frac{\log n}{p\beta^2})$ ▷ Number of MWU iterations performed together

▷ In **one** pass, projects all sets in $\mathcal{F}$ over the collections of samples $\mathcal{L}_1, \cdots \mathcal{L}_R$
sample $\mathcal{L}_1, \ldots, \mathcal{L}_R$ according to $\mathbf{p}^{\mathsf{curr}}$ each of size $\ell n^{\Theta(1/(p\beta))}\,\mathrm{poly}(\log mn)$
$\mathcal{L} \leftarrow \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_R$, $\mathcal{F}_{\mathcal{L}} \leftarrow \emptyset$ ▷ $\mathcal{L}$ is a set whereas $\mathcal{L}_1, \ldots, \mathcal{L}_R$ are multi-sets
for each set S in the stream **do** $\mathcal{F}_{\mathcal{L}} \leftarrow \mathcal{F}_{\mathcal{L}} \cup \{S \cap \mathcal{L}\}$

▷ Each pass simulates R rounds of MWU
for each $e \in \mathcal{L}$ **do** $q_e^1 \leftarrow p_e^{\mathsf{curr}}$ ▷ Project $\mathbf{p}^{\mathsf{curr}}_{n\times 1}$ to $\mathbf{q}^1_{|\mathcal{L}|\times 1}$ over sampled elements
$\mathbf{q}^1 \leftarrow \frac{\mathbf{q}^1}{\|\mathbf{q}^1\|}$
for each round $i = 1, \ldots, R$ **do**
$\mathcal{L}'_i \leftarrow$ **sample** each elt $e \in \mathcal{L}_i$ with probab. $\frac{q_e^i}{q_e^1 n^{\Theta(1/(p\beta))}}$ ▷ Rejection Sampling
$\mathbf{x}^i \leftarrow$ **maxCoverOracle**$(\mathcal{L}'_i, \mathcal{F}_{\mathcal{L}}, \ell)$ ▷ w.h.p. $\mathcal{C}(\mathbf{x}^i) \geq 1 - \beta/3$ when $\ell \geq k$
▷ In **no additional** pass, updates probab. $\mathbf{q}$ over sampled elts according to $\mathbf{x}^i$
$\mathbf{z} \leftarrow \mathbf{0}_{|\mathcal{L}|\times 1}$ ▷ Compute coverage over *sampled* elements
for each element-set pair $e \in S$ where $S \in \mathcal{F}_{\mathcal{L}}$ **do** $z_e \leftarrow \min(z_e + x^i_S, 1)$
$\mathbf{q}^{i+1} \leftarrow$ **updateProb**$(\mathbf{q}^i, \mathbf{z})$ ▷ Only update weights of elements in $\mathcal{L}$

▷ In **one** pass, updates probab. $\mathbf{p}^{\mathsf{curr}}$ over *all* (rare) elts according to $\mathbf{x}^1, \ldots, \mathbf{x}^R$
$\mathbf{z}^1, \ldots, \mathbf{z}^R \leftarrow \mathbf{0}_{n\times 1}$ ▷ Compute coverage over *all* (rare) elements
for each element-set pair $e \in S$ in the stream **do**
for each round $i = 1, \ldots, R$ **do** $z_e^i \leftarrow \min(z_e^i + x^i_S, 1)$
for each round $i = 1, \ldots, R$ **do**
if $(\mathbf{p}^{\mathsf{curr}})^\top \mathbf{z}^i < 1 - \beta/3$ **then** ▷ Detect infeasible solutions
report INFEASIBLE
$\mathbf{x}^{\mathsf{total}} \leftarrow \mathbf{x}^{\mathsf{total}} + \mathbf{x}^i$, $\mathbf{p}^{\mathsf{curr}} \leftarrow$ **updateProb**$(\mathbf{p}^{\mathsf{curr}}, \mathbf{z}^i)$ ▷ Perform actual updates

$\mathbf{x}^{\mathsf{rare}} \leftarrow \frac{\mathbf{x}^{\mathsf{total}}}{(1-\beta)T}$ ▷ Scaled up the solution to cover *rare* elements

return $\mathbf{x}^{\mathsf{cmn}} + \mathbf{x}^{\mathsf{rare}}$

Figure 8 An efficient implementation of **feasibilityTest** which performs in p passes and consumes $\widetilde{O}(mn^{O(\frac{1}{p\varepsilon})} + n)$ space.

no longer bounded by 1. Thus this application of the MWU algorithm has width $\phi = \Theta(\ell U)$ and requires $T = \Theta(\frac{\ell U \log n}{\varepsilon_{\mathrm{MC}}^2})$ rounds. Consequently, its solution becomes $\Theta(\frac{\ell}{T}) = \Theta(\frac{\varepsilon_{\mathrm{MC}}^2}{U \log n})$-integral. As noted in Observation 3.7, the number of potential solutions from the greedy oracle increases by a power of U. Then, in Lemma 3.8, we must reduce the error probability of each solution by the same power. We increase the number of samples s by a factor of U to account for this change, increasing the required amount of memory by the same factor.

As in the previous case, any solution $\mathbf{x}$ may always be pruned so that the width is reduced to 1: our algorithm **prune** still works as long as the entries of $\mathbf{A}$ are non-negative (Section A.3). Therefore, the fact that entries of $\mathbf{A}$ may take on values other than 0 or 1 does not affect the number of rounds (or passes) of our overall application of the MWU framework. Thus, we may handle general covering LPs using a factor of $\widetilde{O}(U/L)$ larger

memory within the same number of passes. In particular, if the non-zero entries of the input are bounded in the range $[1, M]$, this introduces a factor of $\widetilde{O}(U/L) \leq \widetilde{O}(M^3)$ overhead in memory usage. ◀

References

1. K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Proc. 38th Int'l Colloq. Automata Lang. Prog.* (ICALP), pages 526–538. Springer, 2011.
2. K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *Proc. 27th ACM Symp. Parallel Alg. Arch.* (SPAA), pages 202–211, 2015.
3. Z. Allen-Zhu and L. Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proc. 47th Annual ACM Symp. Theory Comput.* (STOC), pages 229–236, 2015.
4. Z. Allen-Zhu and L. Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proc. 26th ACM-SIAM Symp. Discrete Algs.* (SODA), pages 1439–1456, 2015.
5. N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algo.*, 2(2):153–177, 2006.
6. S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.
7. S. Assadi. Tight space-approximation tradeoff for the multi-pass streaming set cover problem. In *Proc. 36th ACM Symp. on Principles of Database Systems* (PODS), pages 321–335, 2017.
8. S. Assadi, S. Khanna, and Y. Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proc. 48th Annual ACM Symp. Theory Comput.* (STOC), pages 698–711, 2016.
9. A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.
10. B. Bahmani, A. Goel, and K. Munagala. Efficient primal-dual graph algorithms for mapreduce. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 59–78. Springer, 2014.
11. N. Bansal, A. Caprara, and M. Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009.
12. M. Bateni, H. Esfandiari, and V. S. Mirrokni. Almost optimal streaming algorithms for coverage problems. *Proc. 29th ACM Symp. Parallel Alg. Arch.* (SPAA), 2017.
13. A. Chakrabarti and A. Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proc. 27th ACM-SIAM Symp. Discrete Algs.* (SODA), pages 1365–1373, 2016.
14. C. Chekuri, S. Gupta, and K. Quanrud. Streaming algorithms for submodular function maximization. In *Proc. 42st Int'l Colloq. Automata Lang. Prog.* (ICALP), pages 318–330. Springer, 2015.
15. F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *Proc. 19th Int. Conf. World Wide Web* (WWW), pages 231–240, 2010.
16. G. Cormode, H. J. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proc. 19th ACM Conf. Info. Know. Manag.* (CIKM), pages 479–488, 2010.

17 E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *Proc. 28th Int'l Symp. Dist. Comp.* (DISC), volume 8784 of *Lect. Notes in Comp. Sci.*, pages 484–498, 2014.

18 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proc. 46th Annual ACM Symp. Theory Comput.* (STOC), pages 624–633. ACM, 2014.

19 P. Drineas, R. Kannan, and M. W. Mahoney. Sampling sub-problems of heterogeneous max-cut problems and approximation algorithms. In *Proc. 37th Annual ACM Symp. Theory Comput.* (STOC), pages 57–68. Springer, 2005.

20 Y. Emek and A. Rosén. Semi-streaming set cover. In *Proc. 41st Int'l Colloq. Automata Lang. Prog.* (ICALP), volume 8572 of *Lect. Notes in Comp. Sci.*, pages 453–464, 2014.

21 U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

22 N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing (SIAM)*, 37(2):630–652, 2007.

23 T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Euro. J. Oper. Res.*, 101(1):81–92, 1997.

24 S. Har-Peled, P. Indyk, S. Mahabadi, and A. Vakilian. Towards tight bounds for the streaming set cover problem. In *Proc. 35th ACM Symp. on Principles of Database Systems* (PODS), pages 371–383, 2016.

25 N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 312–320. IEEE, 1982.

26 M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory.* MIT press, 1994.

27 C. Koufogiannakis and N. E. Young. A nearly linear-time PTAS for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014.

28 Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in $O(vrank)$ iterations and faster algorithms for maximum flow. In *Proc. 55th Annual IEEE Symp. Found. Comput. Sci.* (FOCS), pages 424–433, 2014.

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

30 A. McGregor and H. T. Vu. Better streaming algorithms for the maximum coverage problem. In *20th International Conference on Database Theory, ICDT*, pages 22:1–22:18, 2017.

31 D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 276–287. Springer, 2012.

32 S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

33 R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th Annual ACM Symp. Theory Comput.* (STOC), 1997.

34 B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proc. SIAM Int. Conf. Data Mining* (SDM), pages 697–708, 2009.

35 D. Wang, S. Rao, and M. W. Mahoney. Unified acceleration method for packing and covering problems via diameter reduction. In *Proc. 43st Int'l Colloq. Automata Lang. Prog.* (ICALP), pages 50:1–50:13, 2016.

36 N. E. Young. Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 170–178, 1995.

37 N. E. Young. Nearly linear-work algorithms for mixed packing/covering and facility-location linear programs. *arXiv preprint arXiv:1407.3015*, 2014.

A Omitted Proofs

A.1 Proof of Lemma 3.2

Proof. For the first claim, we consider an arbitrary set system, then modify it by adding a common element e to all sets. Recall that the MWU framework returns an average of the solutions from all rounds. Thus there must exist a round where the oracle returns a solution $\mathbf{x}$ of size $\|\mathbf{x}\|_1 = \Theta(k)$. For the added element e, this solution has $\sum_{S:e\in S} x_S = \sum_{S\in\mathcal{F}} x_S = \Theta(k)$, inducing width $\phi = \Omega(k)$.

For the second claim, consider the following set system with $k = \sqrt{m/\varepsilon}$ and $n = 2k+1$. For $i = 1, \ldots, k$, let $S_i = \{e_i, e_{k+i}, e_{2k+1}\}$, whereas the remaining $m-k$ sets are arbitrary subsets of $\{e_1, \ldots, e_k\}$. Observe that e_{k+i} is contained only in S_i, so $x_{S_i} = 1$ in any valid set cover. Consequently the solution $\mathbf{x}$ where $x_{S_1} = \cdots = x_{S_k} = 1$ and $x_{S_{k+1}} = \cdots = x_{S_{2k+1}} = 0$ forms the unique (fractional) minimum set cover of size $k = \sqrt{m/\varepsilon}$. Next, recall that an element is considered rarely occurring if it appears in at most $\frac{m}{\alpha\ell} > \frac{m}{\varepsilon k}$ sets. As $e_{k+1}, \ldots, e_{2k}$ each only occurs once, and e_{2k+1} only appears in $k = \sqrt{m/\varepsilon} = \frac{m}{\varepsilon k}$ sets, these $k+1$ elements are deemed rare and thus handled by the MWU framework.

The solution computed by the MWU framework satisfies $\sum_{S:e\in S} x_S \geq 1-\beta$ for every e, and in particular, for each $e \in \{e_{k+1}, \ldots, e_{2k}\}$. Therefore, the average solution places a total weight of at least $(1-\beta)\cdot\Theta(k)$ on $x_{S_1}, \ldots, x_{S_k}$, so there must exist a round that places at least the same total weight on these sets. However, these k sets all contain e_{2k+1}, yielding $\sum_{S:e_{2k+1}\in S} x_S \geq (1-\beta)\cdot\Theta(k) = \Omega(k)$, implying a width of $\Omega(k) = \Omega(\sqrt{m/\varepsilon})$. ◀

A.2 Proof of Lemma 3.5

Proof. Let $(\mathbf{x}^*, \mathbf{z}^*)$ denote the optimal solution of value OPT to $MaxCover - LP$, which implies that $\|\mathbf{x}^*\|_1 \leq \ell$ and $\mathbf{A}\mathbf{x}^* \geq \mathbf{z}^*$. Consider the following covering LP: minimize $\|\mathbf{x}\|_1$ subject to $\mathbf{A}\mathbf{x} \geq \mathbf{z}^*$ and $\mathbf{x} \geq \mathbf{0}$. Clearly there exists an optimal solution of objective value ℓ, namely $\mathbf{x}^*$. This covering LP may be solved via the MWU framework. In particular, we may use the oracle that picks one set S with maximum weight (as maintained in the MWU framework) and places its entire budget on x_S. For an accurate guess $\ell' = \Theta(\ell)$ of the optimal value, this algorithm returns an average of $T = \Theta(\frac{\ell' \log n}{\varepsilon_{\text{MC}}^2}) = \Theta(\frac{\ell \log n}{\varepsilon_{\text{MC}}^2})$ oracle solutions. Observe that the outputted solution $\mathbf{x}$ is of the form $x_S = \frac{v_S \ell'}{T} = v_S\delta$ where v_S is the number of rounds in which S is chosen by the oracle, and $\delta = \frac{\ell'}{T} = \frac{\ell'\varepsilon_{\text{MC}}^2}{\ell \log n} = \Theta(\frac{\varepsilon_{\text{MC}}^2}{\log n})$. In other words, $\mathbf{x}$ is $(\frac{\varepsilon_{\text{MC}}^2}{\log n})$-integral. By Theorem 2.2, $\mathbf{x}$ satisfies $\mathbf{A}\mathbf{x} \geq (1-\varepsilon_{\text{MC}})\mathbf{z}^*$. Then in $MaxCover - LP$, the solution $(\mathbf{x}, (1-\varepsilon_{\text{MC}})\mathbf{z}^*)$ yields coverage at least $\mathbf{p}^\top((1-\varepsilon_{\text{MC}})\mathbf{z}^*) = (1-\varepsilon_{\text{MC}})\mathbf{p}^\top\mathbf{z}^* = (1-\varepsilon_{\text{MC}})\text{OPT}$. ◀

A.3 Proof of Lemma 3.6

Proof. Consider the algorithm **prune** in Figure 9. As we pick a valid amount $r \leq x_S$ to move from x_S to $\breve{x}_{\breve{S}}$ at each step, $\breve{\mathbf{x}}$ must be an ℓ-cover (in the extended set system) when **prune** finishes. Observe that if $\sum_{S:e\in S} x_S < 1$ then e will never be removed from any $\breve{S}$,

prune($\mathbf{x}$):
  $\breve{\mathbf{x}} \leftarrow \mathbf{0}_{|\breve{\mathcal{F}}|\times 1}$, $\mathbf{z} \leftarrow \mathbf{0}_{n\times 1}$ ▷ Maintain the pruned solution and its coverage amount
  for each $S \in \mathcal{F}$ **do**
    $\breve{S} \leftarrow S$
    while $x_S > 0$ **do**
      $r \leftarrow \min(x_S, \min_{e\in\breve{S}}(1 - z_e))$ ▷ Weight to be moved from x_S to $\breve{x}_{\breve{S}}$
      $x_S \leftarrow x_S - r$, $x_{\breve{S}} \leftarrow x_{\breve{S}} + r$ ▷ Move weight to the pruned solution
      for each $e \in \breve{S}$ **do** $z_e \leftarrow z_e + r$ ▷ Update coverage accordingly
      $\breve{S} \leftarrow \breve{S} \setminus \{e \in \breve{S} : z_e = 1\}$ ▷ Remove e with $z_e = 1$ from $\breve{S}$
  return $\mathbf{z}$

Figure 9 The **prune** subroutine lifts a solution in $\mathcal{F}$ to a solution in $\breve{\mathcal{F}}$ with the same MaxCover-LP objective value and width 1. The subroutine returns $\mathbf{z}$, the amount by which members of $\breve{\mathcal{F}}$ cover each element. The actual pruned solution $\breve{\mathbf{x}}$ may be computed but has no further use in our algorithm and thus not returned.

so z_e is increased by x_S for every S, and thus $z_e = \sum_{S:e\in S} x_S$. Otherwise, the condition $r \le 1 - z_e$ ensures that z_e stops increasing precisely when it reaches 1. Each S takes up to $n + 1$ rounds in the while loop as one element $e \in S$ is removed at the end of each round. There are at most m sets, so the algorithm must terminate (in polynomial time).

We note that in Section 3.4, we need to adjust **prune** to instead achieves the condition $z_e = \min(\mathbf{A}_e\mathbf{x}, 1)$ where entries of $\mathbf{A}$ are arbitrary non-negative values. We simply make the following modifications: choose $r \leftarrow \min(x_S, \min_{e\in\breve{S}} \frac{1-z_e}{A_{e,S}})$ and update $z_e \leftarrow z_e + r \cdot A_{e,S}$, and the same proof follows. ◀

Remark that to update the weights in the MWU framework, it is sufficient to have the coverage $\sum_{\breve{S}\in\breve{\mathcal{F}}:e\in\breve{S}} \breve{x}_{\breve{S}}$, which are the z_e's returned by **prune**; the actual solution $\breve{\mathbf{x}}$ is not necessary. Observe further that our MWU algorithm can still use $\mathbf{x}$ instead of $\breve{\mathbf{x}}$ as its solution because $\mathbf{x}$ has no worse coverage than $\breve{\mathbf{x}}$ in every iteration, and so does the final, average solution. Lastly, notice that the coverage $\mathbf{z}$ returned by **prune** has the simple formula $z_e = \min(\sum_{S:e\in S} x_S, 1)$. That is, we introduce **prune** to show an existence of $\breve{\mathbf{x}}$, but will never run **prune** in our algorithm.

A.4 Proof of Lemma 3.8

Proof. Consider the $MaxCover - LP(\mathcal{U}, \mathcal{F}, \ell, \mathbf{p})$ with optimal solution $(\mathbf{x}^{\text{OPT}}, \mathbf{z}^{\text{OPT}})$ of value OPT, and let $\mathbf{x}^{\text{SOL}}$ be a $(1 - \varepsilon_{\text{MC}})$-approximate $\Theta(\frac{\gamma^2}{\log n})$-integral ℓ-cover over the sampled elements and $\mathbf{z}^{\text{SOL}}$ be its corresponding coverage vector. Denote the sampled elements with $\mathcal{L} = \{\hat{e}_1, \cdots, \hat{e}_s\}$. Observe that by defining each X_i as a random variable that takes the value $z^{\text{OPT}}_{\hat{e}_i}$ with probability $p_{\hat{e}_i}$ and 0 otherwise, the expected value of $\mathsf{X} = \sum_{i=1}^{s} \mathsf{X}_i$ is

$$\mathbf{E}[\mathsf{X}] = \sum_{i=1}^{s} \mathbf{E}[\mathsf{X}_i] = s \sum_{e\in\mathcal{U}} p_e \cdot z^{\text{OPT}}_e = s \cdot \mathcal{C}(\mathbf{x}^{\text{OPT}}) = s \cdot \text{OPT}.$$

Let $\tau = s(1-\gamma)\text{OPT}$. Since $\mathsf{X}_i \in [0, 1]$, by applying Chernoff bound on X, we obtain

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{C}_{\mathcal{L}}(\mathbf{x}^{\text{OPT}}) \le \tau\right] &= \mathbf{Pr}[\mathsf{X} \le (1-\gamma)\mathbf{E}[\mathsf{X}]] \\ &\le e^{-\frac{\gamma^2\mathbf{E}[\mathsf{X}]}{3}} \le e^{-\frac{\Omega(\ell\log(mn)\log n/\gamma^2)}{3}} = (mn)^{-\Omega(\ell\log n/\gamma^2)}.\end{aligned}$$

Therefore, since $\mathbf{x}^{\text{SOL}}$ is a $(1-\varepsilon_{\text{MC}})$-approximate solution of $MaxCover - LP(\mathcal{L}, \mathcal{F}, \ell, \mathbf{p})$, with probability $1-(mn)^{-\Omega(\ell \log n/\gamma^2)}$, we have $\mathcal{C}_{\mathcal{L}}(\mathbf{x}^{\text{SOL}}) \geq (1-\varepsilon_{\text{MC}})\tau$.

Next, by a similar approach, we show that for any fractional solution $\mathbf{x}$, if $\mathcal{C}_{\mathcal{L}}(\mathbf{x}) \geq \mathcal{C}_{\mathcal{L}}(\mathbf{x}^{\text{OPT}})$, then with probability $1-(mn)^{-\Omega(\ell \log n/\gamma^2)}$, $\mathcal{C}(\mathbf{x}) \geq \left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT}$. Consider a fractional ℓ-cover $(\mathbf{x}, \mathbf{z})$ whose coverage is less than $\left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT}$. Let Y_i denote a random variable that takes value $z_{\hat{e}_i}$ with probability $p_{\hat{e}_i}$, and define $\mathsf{Y} = \sum_{i=1}^{s} \mathsf{Y}_i$. Then, $\mathbf{E}[\mathsf{Y}_i] = \mathcal{C}(\mathbf{x}) < \left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT}$. For ease of analysis, let each $\bar{\mathsf{Y}}_i \in [0,1]$ be an auxiliary random variable that stochastically dominates Y_i with expectation $\mathbf{E}[\bar{\mathsf{Y}}_i] = \left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT}$, and $\bar{\mathsf{Y}} = \sum_{i=1}^{s} \bar{\mathsf{Y}}_i$ which stochastically dominates $\bar{\mathsf{Y}}$ with expectation $\mathbf{E}[\bar{\mathsf{Y}}] = s \cdot \left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT} = \frac{(1-\varepsilon_{\text{MC}})\tau}{1+\gamma}$. We then have

$$\begin{aligned}\mathbf{Pr}[\mathcal{C}_{\mathcal{L}}(\mathbf{x}) > (1-\varepsilon_{\text{MC}})\tau] &= \mathbf{Pr}[\mathsf{Y} > (1-\varepsilon_{\text{MC}})\tau] = \mathbf{Pr}\big[\mathsf{Y} > (1+\gamma)\mathbf{E}[\bar{\mathsf{Y}}]\big] \\ &\leq \mathbf{Pr}\big[\bar{\mathsf{Y}} > (1+\gamma)\mathbf{E}[\bar{\mathsf{Y}}]\big] \leq e^{-\frac{\gamma^2 \mathbf{E}[\bar{\mathsf{Y}}]}{3}} \leq (mn)^{-\Omega(\ell \log n/\gamma^2)},\end{aligned}$$

using the fact that $\left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}}) = \Theta(1)$ for our interested range of parameters. Thus,

$$\mathbf{Pr}\left[\mathcal{C}(\mathbf{x}) \leq \left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT} \text{ and } \mathcal{C}_{\mathcal{L}}(\mathbf{x}) > (1-\varepsilon_{\text{MC}})\tau\right] \leq (mn)^{-\Omega(\ell \log n/\gamma^2)}.$$

In other words, except with probability $(mn)^{-\Omega(\ell \log n/\gamma^2)}$, a chosen solution $\mathbf{x}$ that offers at least as good empirical coverage over $\mathcal{L}$ as $\mathbf{x}^{\text{OPT}}$ (namely $\mathbf{x}^{\text{SOL}}$) does have actual coverage of at least $\left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT}$.

Since the total number of $\Theta(\frac{\gamma^2}{\log n})$-integral ℓ-covers is $O(m^{\ell \log n/\gamma^2})$ (Observation 3.7), applying union bound, with probability at least $1 - O(m^{\ell \log n/\gamma^2}) \cdot (mn)^{-\Omega(\ell \log n/\gamma^2)} = 1 - \frac{1}{\text{poly}(mn)}$, a $(1-\varepsilon_{\text{MC}})$-approximate $\Theta(\frac{\gamma^2}{\log n})$-integral solution of $\textsf{Max}\,k\textsf{-Cover}(\mathcal{L}, \mathcal{F}, \ell, \mathbf{p})$ has weighted coverage of at least $\left(\frac{1-\gamma}{1+\gamma}\right)(1-\varepsilon_{\text{MC}})\text{OPT} > (1-2\gamma)(1-\varepsilon_{\text{MC}})\text{OPT}$ over $\mathcal{U}$. ◀

A.5 Proof of Theorem 3.9

Proof. The algorithm clearly requires $\Theta(T)$ passes to simulate the MWU algorithm. The required amount of memory, besides $\widetilde{O}(n)$ for counting elements, is dominated by the projected set system. In each pass over the stream, we sample $\Theta(\ell \log mn \log n/\varepsilon^4)$ elements, and since they are rarely occurring, each is contained in at most $\Theta(\frac{m}{\varepsilon\ell})$ sets. Finally, we run $\log_{1+\Theta(\varepsilon)} n = O(\log n/\varepsilon)$ instances of the MWU algorithm in parallel to compute a $(1+\varepsilon)$-approximate solution. In total, our space complexity is $\Theta(\ell \log mn \log n/\varepsilon^4) \cdot \Theta(\frac{m}{\varepsilon\ell}) \cdot O(\log n/\varepsilon) = \widetilde{O}(m/\varepsilon^6)$. ◀

A.6 Proof of Lemma 3.10

Proof. Recall the weight update formula $w_e^{t+1} = w_e^t(1 - \frac{\beta(\breve{\mathbf{A}}_e \breve{\mathbf{x}} - b_e)}{6\phi})$ for the MWU framework, where $\breve{\mathbf{A}}_{n \times |\breve{\mathcal{F}}|}$ represents the membership matrix corresponding to the extended set system $(\mathcal{U}, \breve{\mathcal{F}})$. In our case, the desired coverage amount is $b_e = 1$. By construction, we have $\breve{\mathbf{A}}_e \breve{\mathbf{x}} = z_e \leq 1$; therefore, our width is $\phi = 1$, and $-1 \leq \breve{\mathbf{A}}_e \breve{\mathbf{x}} - b_e \leq 0$. That is, the weight of each element cannot decrease, but may increase by at most a multiplicative factor of $1 + \beta/6$, before normalization. Thus even after normalization no weight may increase by more than a factor of $1 + \beta/6 = 1 + O(\beta)$. ◀