Randomized MWU for Positive LPs[*]

Chandra Chekuri Kent Quanrud

Abstract

We describe and analyze a simple randomized multiplicative weight update (MWU) based algorithm for approximately solving positive linear programming problems, in particular, mixed packing and covering LPs.

Given m explicit linear packing and covering constraints over n variables specified by N nonzero entries, Young [36] gave a deterministic algorithm returning an $(1+\epsilon)$-approximate feasible solution (if a feasible solution exists) in $\tilde{O}(N/\epsilon^2)$ time. We show that a simple randomized implementation matches this bound, and that randomization can be further exploited to improve the running time to $\tilde{O}(N/\epsilon + m/\epsilon^2 + n/\epsilon^3)$ (both with high probability). For instances that are not very sparse (with at least $\tilde{\omega}(1/\epsilon)$ nonzeroes per column on average), this improves the running time of $\tilde{O}(N/\epsilon^2)$. The randomized algorithm also gives improved running times for some implicitly defined problems that arise in combinatorial and geometric optimization.

1 Introduction

In this paper we consider fast approximation schemes for positive linear programming problems where all the input data consists of non-negative numbers, and the output solution is also required to be non-negative. The most general class here is *mixed packing and covering* LPs which we state in a *normalized* form below:

$$(\mathsf{NMPC}) \quad \text{find } x \text{ such that } x \geq \mathbb{0},\ Ax \leq \mathbb{1} \text{ and } Bx \geq \mathbb{1},$$

where $A \in \mathbb{R}_{\geq 0}^{\mathcal{P}\times n}$ and $B \in \mathcal{Q}^{\mathcal{C}\times n}$ are nonnegative matrices[*]. Let $m_p = |\mathcal{P}|$ be the number of packing constraints, $m_c = |\mathcal{C}|$ the number of covering constraints, and $m = m_p + m_c$ the total number of constraints. We let N denote the total number of nonzeroes in A and B. A basic special case is finding a nonnegative solution $x \geq \mathbb{0}$ to $Ax = b$, where A and b have nonnegative entries.

A simpler subclass of (NMPC) considers *pure packing problems* of the form

$$(\mathsf{P}) \qquad \max\ \langle c, x\rangle \text{ over } x \geq \mathbb{0} \text{ such that } Ax \leq \mathbb{1}$$

and *pure covering problems* of the form

$$(\mathsf{C}) \qquad \min\ \langle c, x\rangle \text{ over } x \geq \mathbb{0} \text{ such that } Bx \geq \mathbb{1},$$

where again all inputs A, B, and c are assumed to be nonnegative. Pure packing and pure covering LPs are duals of each other. In contrast, the dual of a mixed packing and covering LP is *not* a mixed packing and covering LP.

Postive LPs are a particularly simple class of linear programs that nonetheless have many fundamental applications in computer science and optimization. Of course one can solve these by general purpose LP solvers, but the running time is a non-linear polynomial in the input size. Moreover, there are several applications in which the LP is *implicitly* defined. For several such LPs, using the Ellipsoid method or explicitly writing down the LP and then using a standard solver is prohibitively expensive.

We are interested in algorithms that obtain relatively coarse approximations in substantially less time. The approximation criteria is as follows. Given an error parameter $\epsilon > 0$, if a feasible solution z to (NMPC) exists, then we must return a nonnegative vector $x \geq \mathbb{0}$ such that $Ax \leq (1+\epsilon)\mathbb{1}$ and $Bx \geq (1-\epsilon)\mathbb{1}$. (Clearly, the error can be made one-sided by scaling x up or down.) In a regime where ϵ is moderately large, we seek algorithms that are polynomial in $1/\epsilon$ and very fast in the other parameters m, n, and N. In contrast, interior point algorithms or ellipsoid based methods obtain an additive error of ϵ and the dependence of the run-time is polynomial in $\log(1/\epsilon)$; however, the dependence on m, n and N is much worse.

The past two decades, starting with works of Shahrokhi and Matula [29], Plotkin, Shmoys, and Tardos [28], Grigoriadis and Khachiyan [14], Luby and Nisan [24], and many others (too many to provide a proper accounting here), have produced a substantial amount of literature on iterative methods for solving positive LPs with polynomial dependence on $1/\epsilon$ and otherwise better dependencies on parameters such

[*] Department of Computer Science, University of Illinois, Urbana, IL 61820. {chekuri,quanrud2}@illinois.edu. Work on this paper partly supported by NSF grant CCF-1526799.

[*] The normalized form does not have an objective function since a maximization or minimization objective with non-negative coefficients can be incorporated as a constraint.

as the input size. Here we are interested in width-independent algorithms. One broad technique is Lagrangian relaxation with potential functions (exponential or logarithmic) whose run-time dependence on ϵ is $1/\epsilon^2$. Another technique relies on a reduction to (accelerated) first order methods from convex optimization. These methods improve the dependence on ϵ to $1/\epsilon$; Bienstock and Iyengar [5] demonstrated this building upon Nesterov's accelerated gradient descent technique. The first order methods had worse running times on the other parameters, however, recently there have been several exciting developments that are able to obtain a $1/\epsilon$ dependence while also having near-linear dependence on the other parameters.

For (NMPC) the algorithm of Young [36] returns an ϵ-approximation deterministically in $O\big(N\ln(m)/\epsilon^2\big)$; this is the first nearly-linear time algorithm. Young's algorithm follows the *multiplicative weight update (MWU)* framework, and improves on previous MWU-type algorithms by lazily applying the weight updates via an efficient amortized data structure. Applying Nesterov's accelerated gradient descent technique, Bienstock and Iyengar [5] achieve a running time of $O\big(n^{2.5}d/\epsilon\big)$ where d is the maximum number of nonzeroes in a column.

In the restricted setting of pure packing and covering better results are known. Koufogiannakis and Young [21, 22] obtained $(1+\epsilon)$ approximations to (P) and (C) in $O\big(N + (m+n)\ln(m)/\epsilon^2\big)$ time via a randomized algorithm. The algorithm of Koufogiannakis and Young efficiently simulates a zero-sum game between two players solving the primal and dual problem respectively and draws inspiration from Grigoriadis and Khachiyan [15]. Recently, Allen-Zhu and Orecchia [2] gave a randomized coordinate descent algorithm that returns a $(1+\epsilon)$-relative approximation to (P) in $O(N\ln(N)\ln(\epsilon)/\epsilon)$ time and (C) in $O\big(N\ln(n)\ln(\epsilon)/\epsilon^{1.5}\big)$ time. The running time for (C) was subsequently improved to $O\big(N\ln^2(N/\epsilon)/\epsilon\big)$ by Wang, Rao, and Mahoney [33]. There have been several recent developments in parallel algorithms as well. Since our focus here is only on sequential algorithms, we refer the reader to the Mahoney, Rao, Wang, and Zhang [25] for several pointers to recent and past work.

It is natural to ask whether one can obtained improved running times for (NMPC) more in line with the ones known for (P) and (C). This was explicitly raised in [21, 22, 36]. The high-level goal is to shift the ϵ-factors in the running time off of the dominant term N to lower-order terms such as m and n.

Open Question 1.1. *Is there a (randomized) algorithm for (NMPC) that returns a $(1+\epsilon)$-approximate solution in $\tilde{O}\big(N + (n+m)/\epsilon^2\big)$ time or in $\tilde{O}(N/\epsilon)$ time?*

We note that some ideas that help for pure packing and covering such as coupling do not yet have an analogue for (NMPC) and this makes it challenging to address the preceding question. Our discussion so far has mainly focused on explicitly described LPs. As we mentioned already, there are several applications where the LP is implicitly defined. A canonical example is multicommodity flow which was instrumental in the development of fast approximation schemes for positive LPs. Achieving fast running times for such problems requires a mix of techniques. Our work here is not only motivated by explicit problems of the form (NMPC), but other (implicit) mixed packing and covering problems contingent on the implementation of simple subroutines/oracles. In this regard, one advantage of the MWU-based iterative methods is the relative simplicity of both the algorithm and the analysis, which allows for subroutines to be approximated by heuristics and accelerated by data structures for a better total running time (see, for example, [27, 1, 36, 6, 7]). Our focus in this paper is on augmenting the MWU framework to obtain faster and/or simpler algorithms for several classes of problems.

Our contribution and results: We develop a randomized algorithm based on the MWU framework for (NMPC), perhaps most similar to [36] among the competing algorithms mentioned above. It leads to a width-independent algorithm that simplifies the weight update step via a simple correlated random choice in each iteration. The randomized scheme was used previously in [21, 22] for pure packing and covering in the context of their primal-dual coupled algorithm. Here we apply it to a primal algorithm in the more general setting of mixed packing and covering.

Our first contribution is to analyze this randomized algorithm and prove that it achieves a $(1+\epsilon)$-approximation in $O(m\log m/\epsilon^2)$ iterations with high-probability. The analysis is technical due to the adaptive nature of the step sizes that are needed to achieve width-independence, and the interaction between the packing and covering constraints (the packing weights go up and the covering weights go down).

Our second contribution leverages the randomized MWU algorithm for better approximation algorithms to (NMPC). An easy implementation recovers the $O(N\log m/\epsilon^2)$ run time achieved in [36]; the advantage of the randomized algorithm is that steps are transparent and very simple data structures suffice. In addition the randomization is useful for some implicit problems. We then push randomized techniques further to improve the running time and make measurable progress on Question 1.1. Our improvement is captured by the next theorem.

Theorem 1.1. *Given a normalized mixed packing and covering problem (*NMPC*),* `random-mwu` *returns a ϵ-relative approximation in*

$$O\left(\begin{array}{c} \frac{N\log^2 m}{\epsilon} + \frac{m\log^2 m}{\epsilon^2} \\ + \frac{n\log(m)(\log(m)+\log(n))}{\epsilon^3} \end{array}\right) = \tilde{O}\left(\frac{N}{\epsilon} + \frac{m}{\epsilon^2} + \frac{n}{\epsilon^3}\right)$$

time with probability $1 - 1/\operatorname{poly}(m)$.

The improvement from $\tilde{O}(N/\epsilon^2)$ to $\tilde{O}(N/\epsilon)$ is made possible by a general data structure for approximating nonnegative linear maps that we discuss in detail later.[†] The improvement is meaningful when the input matrix is not extremely sparse. For dense matrices with $N = \tilde{\Omega}(mn)$, Theorem 1.1 improves the running time from $\tilde{O}(mn/\epsilon^2)$ to $\tilde{O}(mn/\epsilon)$. We believe that it may be possible to improve the run time further to achieve a bound of $\tilde{O}(N/\epsilon + (m+n)/\epsilon^2)$. Note that MWU-based algorithms cannot escape some $1/\epsilon^2$ dependence via the lower bound of Klein and Young [19].

Applications to implicit problems: Finally, our randomized algorithm yields faster algorithms for solving some implicit LPs that arise as relaxations in combinatorial and geometric optimization. We note that the randomized MWU algorithm is useful not just for mixed packing and covering but also pure packing and covering. In some recent work [6, 7] we had developed faster algorithms for implicit packing problems by adapting the MWU framework via a combination of data structures. Our randomized variant of MWU was initially motivated by some examples of implicit problems in combinatorial and geometric settings that were not amenable to the data structure ideas in [6]. In Section 5 we sketch a few applications of explicit and implicit problems and highlight how the randomized variant allows for handling new implicit problems as well as generalizing some prior results in [6] to the mixed packing and covering setting. Our focus in this paper is on the high-level randomized algorithm and its analysis for explicit problems. Details of implicit applications are deferred, partly due to space constraints, and partly due to the fact that these applications require domain-specific data structures and other ideas which are beyond the scope of this paper.

2 Randomized MWU and Overview of Techniques

The proposed algorithm, called `random-mwu` and sketched in Figure 1, is (at a high-level) a variant of a deterministic algorithm for mixed packing and covering originally proposed in [35] and refined in [36]. The algorithm falls in the broad framework of Lagrangian relaxation algorithms that iteratively solve a relaxation of the original problem as follows. The algorithm maintains weights for each constraint (which can be interpreted as dual variables). We let v denote the weight vector for packing constraints and w denote the vector for covering constraints; v and w are both initialized to the all-1's vector $\mathbb{1}$. In each iteration the algorithm finds a feasible solution y for the relaxed problem which is obtained by collapsing all the packing constraints into a single constraint by taking a weighted combination, and similarly collapsing all the covering constraints into a single constraint:

$$\text{(2.1)}\quad \text{find } x \geq \mathbb{0} \text{ s.t. } \langle v, Ax\rangle \leq \langle v, \mathbb{1}\rangle, \langle w, Bx\rangle \geq \langle w, \mathbb{1}\rangle$$

Note that if the relaxed problem is infeasible then the original problem is infeasible. `random-mwu` uses an approximate oracle rather than exact oracle. Relaxing the oracle does not upset the the final analysis for a $(1+\epsilon)$-approximate solution, and offers flexibility that leads to improvements in the running time. A key observation is the relative simplicity of (2.1) compared to (NMPC). For $1 \leq i \leq n$, let $\alpha_i = \langle v, A\rangle_i / \langle v, \mathbb{1}\rangle$ and let $\beta_i = \langle w, B\rangle_i / \langle w, \mathbb{1}\rangle$. (2.1) is feasible iff there exists an i such that $\alpha_i/\beta_i \leq 1$. Then setting $y_i = 1/\beta_i$ and all other coordinates to zero is a feasible solution. In each iteration, we can assume without loss of generality that only one coordinate is updated.

The algorithm adds the solution y to the current solution (which is initialized to $\mathbb{0}$) with an appropriate step size δ. Our algorithm follows the "timed" framework from [8] which increments time from 0 to 1 and the step size and other parameters are appropriately normalized. To obtain a width-independent running time two key existing ideas are needed: (i) non-uniform step sizes [13] and (ii) dropping covering constraints that are already satisfied. See [36]. After each iteration the packing and covering weights are updated multiplicatively (packing weights are increased in an exponential fashion and covering weights are decreased).

The efficiency of the algorithm depends on the number of iterations and the work done in each iteration. One can show that the deterministic variant terminates in $O(m \log m/\epsilon^2)$ iterations. Each iteration requires two main steps: (i) finding a solution to (2.1), and (ii) updating the weights. These are non-trivial bottlenecks to achieving an overall near-linear running time. This was accomplished in [36] using a careful amortized data structure to lazily update the weights among other ideas. Some of these ideas were shown to be effective for implicit problems as well [6, 7].

[†] For ease of notation, we use $\tilde{O}(\cdots)$ to suppress logarithmic factors.

```
random-mwu($A \in \mathbb{R}^{\mathcal{P}\times n}_{\geq 0}$, $B \in \mathbb{R}^{\mathcal{C}\times n}_{\geq 0}$, $\epsilon$)
    $\eta = (\ln m)/\epsilon$ // parameter to control step size
    $v \leftarrow \mathbb{1}$, $w \leftarrow \mathbb{1}$ // $v$, $w$: packing & covering weights
    $\mathcal{Q} \leftarrow \mathcal{C}$ // $\mathcal{Q}$: active covering constraints
    // for $a, b \in \mathbb{R}^{\mathcal{C}}$, let $\langle a, b\rangle_{\mathcal{Q}} = \sum_{i \in \mathcal{Q}} a_i b_i$
    $t \leftarrow 0$ // time goes from $0$ to $1$
[1] while $t \leq 1$ and $\mathcal{Q} \neq \emptyset$
[2]   choose $y \in \mathbb{R}^n_{\geq 0}$ such that
         (*) $\langle v, Ay\rangle \leq (1 + O(\epsilon))\langle v, \mathbb{1}\rangle$
        (**) $\langle w, By\rangle_{\mathcal{Q}} \geq (1 - O(\epsilon))\langle w, \mathbb{1}\rangle_{\mathcal{Q}}$
         // $y$ is approx soln to Lagrangean relaxation with weights $v, w$
      if no $y \in \mathbb{R}^n_{\geq 0}$ satisfies (*) and (**) then return ‘‘infeasible’’
[3]   $\delta \leftarrow$ max value $\delta > 0$ such that // step size
         (*) $\delta\eta Ay \leq \epsilon$
        (**) $\delta\eta By \leq \epsilon$
       (***) $t + \delta \leq 1$
      $x \leftarrow x + \delta y$ // increment current solution with $\delta y$
      $t \leftarrow t + \delta$ // increment time
      pick $\theta \in [0, 1]$ uniformly at random
      for $i \in \mathcal{P}$ // update/increase packing weights
        // approximate $v_i \leftarrow \exp(\delta\eta\langle e_i, Ay\rangle)v_i$
        if $(\theta \leq \delta\eta\langle e_i, Ay\rangle/\epsilon)$
[4]       $v_i \leftarrow \exp(\epsilon)v_i$
      end for
      for $i \in \mathcal{Q}$ // update/decrease active covering weights
        // approximate $w_i \leftarrow \exp(-\delta\eta\langle e_i, By\rangle)$
        if $(\theta \leq \delta\eta\langle e_i, By\rangle/\epsilon)$
          $w_i \leftarrow \exp(-\epsilon)w_i$
[5]       if $w_i \leq \exp(-\eta)$
            $\mathcal{Q} \leftarrow \mathcal{Q} - i$ // $i$ made inactive if weight small enough
      end for
    end while
    return $x$
```

Figure 1: *A randomized implementation of the MWU framework for mixed packing and covering.*

The primary difference in the algorithm we propose from that in [36] is that the multiplicative weight updates are now *randomized*. We borrow this idea from [22] where it was applied to pure packing and covering problems. Where the standard deterministic update might increase a weight by a multiplicative factor of $\exp(\epsilon p)$ for some $p \in [0, 1]$, random-mwu increases the weight by a multiplicative factor of $\exp(\epsilon)$ with probability p. In expectation, random-mwu makes the appropriate update with respect to the logarithm of the weight. The key to implementation efficiency is that all the weight updates are *correlated* via a single random variable θ. This cleanly addresses the efficiency issue in updating the weights at the expense of moving the burden to proving the correctness of the algorithm. On the other hand, randomized weight updates do *not* address the issue of solving (2.1) in each iteration, which remain a bottleneck. We use randomization again in a different way to improve that step as well in Section 4.

Figure 1 is incomplete, as we leave the implementation of lines [2] and [3] unspecified until Section 4. This part will also be randomized and the details are deferred primarily for ease of exposition. A secondary reason is that in some implicit packing and covering problems, [2] and [3] can be supplied by a more efficient and domain-specific oracle. Some other low-level implementation details are omitted from Figure 1. The additional code allows us to list all the entries of a column above a threshold in time slightly faster than presorting all the nonnegative entries and doing a binary search. We defer this discussion to the end because the techniques are well-known and the speedup is by only a logarithmic factor. Full implementation details are eventually provided in Section 4.

We prove that `random-mwu` terminates both successfully and efficiently with high probability.

Theorem 2.1. *Let $A \in \mathbb{R}^{m_p \times n}_{\geq 0}$ and $B \in \mathbb{R}^{m_c \times n}_{\geq 0}$ be nonnegative matrices for which there exists a nonnegative $x \in \mathbb{R}^n_{\geq 0}$ such that $Ax \leq \mathbb{1}$ and $Bx \geq \mathbb{1}$. Let $m = m_p + m_c$, and let N be the total number of nonzero coefficients in A and B.*

With probability $1 - 1/\operatorname{poly}(m)$, `random-mwu(`$A$`,`$B$`,`$\epsilon$`)` returns a point $\hat{x}$ such that $A\hat{x} \leq (1 + O(\epsilon))\mathbb{1}$ and $B\hat{x} \geq (1 - O(\epsilon))\mathbb{1}$ in $O\big((m_c + \min\{m_p, n\}) \ln(n)/\epsilon^2\big)$ iterations and, excluding the time spent in lines `[2]` and `[3]`, $\tilde{O}\big(N + m \ln(n)/\epsilon^2\big)$ time. Each packing weight $v_i, i \in \mathcal{P}$ increases along integral powers of $\exp(\epsilon)$ from 1 to (at most) $\exp(\ln(m_p)/\epsilon)$. Each covering weight $w_i, i \in \mathcal{C}$ decreases along integral powers of $\exp(\epsilon)$ from 1 to $\exp(-\ln(m_c)/\epsilon)$.

Remark 2.1. *In the preceding theorem we assumed the existence of a feasible solution for the sake of simplicity. In fact the algorithm either outputs a $(1+\epsilon)$-approximate solution in the claimed run time or correctly reports that the given system has no feasible solution.*

As stated, the running time of `random-mwu` is assured with high probability (and can be shown to be finite with probability 1), but technically speaking is unbounded. `random-mwu` can be made a proper Monte Carlo algorithm that always terminates in $O\big(N + m \ln(m)/\epsilon^2\big)$ time (excluding calls to the oracle) by killing the algorithm if it runs for too long.

Online Chernoff: Chernoff bounds for sums of independent, bounded and *non-negative* random variables are ubiquitous in computer science. However, in processes or algorithms where the choice in a step depends on decisions made in previous steps, it is necessary to use some form of martingale analysis. However, typical martingale inequalities are stated in the setting of bounded random variables with zero mean, and in this setting the concentration bounds depend on the number of variables (which correspond to iterations in our case) or the sum of their variances. These are not suitable for our needs. The following concentration bound reformulates Chernoff inequalities for online settings, and is the workhorse of the proofs in this paper. The theorem in a slightly stronger form with a stopping time is stated and proved in [22].

Theorem 2.2. ([22, Lemma 10]) *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_n \in [0, 1]$ be random variables and let $\epsilon \in [0, 1/2)$ be a sufficiently small constant.*

(a) If $\operatorname{E}[X_i \mid X_1, \ldots, X_{i-1}, Y_1, \ldots, Y_i] \leq Y_i$ for $i \in [n]$, then for any $\delta > 0$,

$$\operatorname{P}\left[\sum_{i=1}^n X_i \geq (1 + \epsilon) \sum_{i=1}^n Y_i + \delta\right] \leq (1 + \epsilon)^{-\delta}$$

(b) If $\operatorname{E}[X_i \mid X_1, \ldots, X_{i-1}, Y_1, \ldots, Y_i] \geq Y_i$ for each i, then for any $\delta > 0$,

$$\operatorname{P}\left[\sum_{i=1}^n X_i \leq (1 - \epsilon) \sum_{i=1}^n Y_i - \delta\right] \leq (1 - \epsilon)^{\delta}.$$

Note that concentration bound depends only on the additive term δ and not on n. Theorems such as the preceding one are related to work on drift analysis in random processes [23].

2.1 Approximating monotonic linear maps

`random-mwu` with some basic bucketing tricks allows us to recover an implementation with a running time of $O\big(N \log m/\epsilon^2\big)$ for (NMPC). In order to obtain the running time in Theorem 1.1 we need the next component of our work. Recall that the randomization in the algorithm allows us to efficiently update v and w; the entries are probabilistically updated so that we can charge the cost of an update to a relatively big change in the value. However, implementing `[2]` efficiently requires us to dynamically maintain a $(1 \pm \epsilon)$-relative approximation for every coordinate of $A^T v$ and $B^T w$. Since the coordinates of v and w are updated infrequently, the basic approach already maintains $A^T v$ and $B^T w$ deterministically in $O\big(N \log(m)/\epsilon^2\big)$ time total. Here we develop a randomized data structure that improves this bound. We introduce the ideas from a more fundamental perspective, as the results are interesting in their own right.

Recall that a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear if $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$ for any $x, y \in \mathbb{R}^n$ and $\alpha, \beta \in \mathbb{R}$. A linear function $f : \mathbb{R}^m \to \mathbb{R}^n$ can be represented by a matrix $A \in \mathbb{R}^{m \times n}$ such that $f(x)_i = \sum_{j=1}^n A_{ij} x_j$ for any input $x \in \mathbb{R}^n$ and output coordinate $i \in [m]$. The function f is monotonically increasing iff the coefficients A_{ij} are nonnegative. Given the matrix representation A of a linear function f, we can compute the vector Ax exactly in $O(N)$ time, where N is the number of nonzeroes in A, via the above sum. We want to maintain the vector Ax as x varies; i.e., $x = z_1 + \cdots + z_k \in \mathbb{R}^n$ over a sequence of updates $z_i \in \mathbb{R}^n$. In the offline setting, if all the vectors $z_1, \ldots, z_k$ are provided as input, then we can simply compile their sum x and compute Ax in $O(N)$ time plus the time it takes to read in $z_1, \ldots, z_k$. In many applications, the vectors are delivered online and adversarially. Having already computed $A(z_1 + \cdots + z_k)$, we are given z_{k+1}, and want to compute the new vector, $A(z_1 + \cdots + z_k + z_{k+1})$. By linearity, $A(z_1 + \cdots + z_k + z_{k+1}) = A(z_1 + \cdots + z_k) + Az_{k+1}$, and hence we can computing $A_{z_{k+1}}$ and add it to the existing solution $A(z_1 + \cdots + z_k)$. Doing this for every vector takes total time $O(kN)$ which is significantly slow when k is large.

The natural approach above is seemingly best possible as far as exact and deterministic algorithms go. In some modern settings, the problem is more relaxed where an approximation to Ax is sufficient. When implementing *fast* approximation algorithms, dynamically updating Ax may even be a bottleneck. This is precisely the setting in `random-mwu`, so we consider the problem of maintaining Ax approximately rather than exactly.

REMARK 2.2. *In mixed packing and covering, the covering weights go down, and hence we also need to handle the approximate maintenance of Ax as x monotonically* decreases *with the guarantee that Ax stays nonnegative.*

In particular, we are interesting in *uniform approximations* where each coordinate $\langle e_i, Ax\rangle$ $(i \in [m])$ should be approximated well. We focus on the positive and monotone setting where $A \in \mathbb{R}^{m\times n}_{\geq 0}$, $x \geq \mathbb{0}$ and x is either monotonically increasing or monotonically decreasing. A simplified form of our basic result is as follows.

THEOREM 2.3. *Let $\epsilon > 0$ with ϵ sufficiently small, and let $A \in \mathbb{R}^{m\times n}_{\geq 0}$ be a nonnegative matrix with N total nonzeroes, and let $L \in \mathbb{N}$ and $\beta > 0$ be fixed parameters. One can initialize a data structure in $O(N)$ time with the following guarantee. Consider any sequence of L increments $\alpha_1 e_{j_1}, \alpha_2 e_{j_2}, \ldots, \alpha_\ell e_{j_L} \in \mathbb{R}^n_{\geq 0}$ delivered online satisfying the following*

(i) Either $\alpha_\ell \geq 0$ for all ℓ or $\alpha_\ell \leq 0$ for all ℓ

(ii) Letting $x^\ell = \mathbb{1} + \displaystyle\sum_{k=1}^{\ell} \alpha_k e_{j_k}$ for each $\ell \in [L]$, we have $\dfrac{1}{\beta} A\mathbb{1} \leq Ax^\ell \leq \beta A\mathbb{1}$ for all $\ell \in [L]$ (i.e., the online sequence is constrained as such).

Then the data structure maintains a nonnegative vector $y \in \mathbb{R}^m_{\geq 0}$ that with probability at least $1 - 1/\operatorname{poly}(m, L)$, satisfies $(1-\epsilon)Ax^\ell \leq y \leq (1+\epsilon)Ax_\ell$ for all $\ell \in [L]$ in $\Big(L(\log(m) + \log\log\beta) + N(\log(m) + \log\log\beta)\log\beta + m^{\frac{\log(\beta)(\log m + \log L)}{\epsilon^2}}\Big)$ total time.

Organization of the rest of the paper: The rest of the paper is devoted to proving the theorems that we outlined so far. We start by proving Theorem 2.1 in Section 3. Section 4 proves Theorem 2.3. We put together the ingredients to prove Theorem 1.1 in Section 4.1. An overview of applications is provided in Section 5.

We organized the paper in a sequential fashion with full proofs rather than move them to the appendix. The reader is encouraged to skim the high-level ideas and skip the low-level proofs as they see fit.

Other related work: There is a vast literature on fast approximation schemes for LPs and convex programming problems. Several important techniques have been developed by many authors. It is infeasible to do justice to this literature here. We refer the reader to Arora, Hazan, and Kale [4] for a broad survey on the utility of MWU in theoretical computer science. Our work is inspired by several ideas and results in the papers of Young [36, 22, 34, 35] which incorporate ideas from other papers including Garg and Könemann [13] and Fleischer [11]. Some of our motivation for revisiting MWU based methods comes from applications to implicit problems that arise in combinatorial and geometric settings [1, 6, 7]. Several papers in computational geometry exploit MWU (where it is referred to as reweighting technique) for covering and hitting set problems; typically the algorithms do not explicitly refer to the LP relaxation and combine the rounding and LP solving in a single framework for deriving (approximation) algorithms. We believe that separating the problem of solving the LP from that of rounding, and separating the generic high-level MWU framework from its specific implementation for a concrete application, is helpful.

Although we have mainly focused on linear programming problems here, Lagrangian relaxation based algorithms are applicable to the broader context of convex programming [18], and also for submodular function maximization [8]. We believe that our randomized variant should also generalize to these settings. Lagrangian relaxation via logarithmic barrier function also yield width-independent algorithms with $O(1/\epsilon^2)$ iterations — see [9, 17] and references therein. These methods are less well-known in the computer science algorithms literature, and appear to be more difficult to exploit via data structures and other techniques.

3 Randomized multiplicative weight updates in the oracle model

In this section, we analyze `random-mwu` in the oracle model with line `[2]` unspecified and prove Theorem 2.1.

`random-mwu` is an iterative algorithm that (as written) is indexed by a continuous "time" variable t that increases from 0 to 1. For each iteration $\ell \in \mathbb{N}$, let v^ℓ, w^ℓ, and t^ℓ denote the values of v, w, t at the beginning of the ℓth iteration. We let y^ℓ and δ^ℓ denote the values of y and δ computed during the ℓth iteration. We let $\hat{\ell} \in \mathbb{N}$ denote the total number of iterations (when it terminates), and denote $\hat{v} = \lim_{\ell\to\infty} v^\ell$, $\hat{w} = \lim_{\ell\to\infty} w^\ell$, and $\hat{t} = \lim_{\ell\to\infty} t^\ell$ denote the values of v, w, and t at the end of the algorithm (or the possibly unbounded limit if it does not terminate). For $\ell \in \mathbb{N}$, we let $\mathcal{Q}^\ell$ denote the value of $\mathcal{Q}$ at the beginning of the ℓth iteration if the algorithm has not yet terminated, or to $\mathcal{Q}^{\hat{\ell}}$ if the algorithm has terminated and $\ell > \hat{\ell}$.

The analysis shows that the packing constraints are not violated by more than a $(1+\epsilon)$-factor and that the covering constraints are satisfied to within a $(1-\epsilon)$ factor with high probability. This is non-trivial even in the deterministic setting. In our analysis we use the time variable t to link the evolution of the packing and covering weights. The main task in the analysis is to show that the randomized algorithm's state closely follows the deterministic invariants. For this purpose we rely repeatedly on Theorem 2.2. We consider packing and covering constraints separately, and then consider the number of iterations after which we tie things together.

3.1 Packing constraints

We begin by showing that the randomized weight update follows the approximates MWU framework in expectation.

Lemma 3.1. *For each iteration ℓ, with the outcomes of iterations 1 through $\ell-1$ fixed, and for each packing constraint $i \in \mathcal{P}$, we have*

$$\begin{aligned} &\mathrm{E}\left[\ln\left(v_i^{\ell+1}\right)\right] = \ln\left(v_i^\ell\right) + \delta^\ell \eta \langle e_i, Ay^\ell\rangle \\ \textit{and} \quad &\mathrm{E}\left[v_i^{\ell+1}\right] \le \exp\left((1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle\right) v_i^\ell. \end{aligned}$$

Proof. The first inequality is immediate from the randomized step in the algorithm. For the second, we have

$$\begin{aligned} &\mathrm{E}\left[v_i^{\ell+1}\right] \\ &= \left(\delta^\ell\eta\langle e_i, Ay^\ell\rangle/\epsilon\right)\exp(\epsilon)v_i^\ell + \left(1 - \delta^\ell\eta\langle e_i, Ay^\ell\rangle/\epsilon\right)v_i^\ell \end{aligned}$$

by `[4]`,

$$= v_i^\ell + (\exp(\epsilon)-1)\left(\delta^\ell\eta\langle e_i, Ay^\ell\rangle/\epsilon\right)v_i^\ell,$$

$$\le v_i^\ell + (1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle v_i^\ell$$

since $\exp(\epsilon) \le 1+\epsilon+\epsilon^2$ for ϵ sufficiently small,

$$\le \exp\left((1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle\right)v_i^\ell$$

since $1+z \le \exp(z)$ for all z. ■

The second lemma analyzes the sum of weights $\langle v, \mathbb{1}\rangle$ for the packing constraints. In particular, it shows that the expected increase in the logarithm of the sum of packing constraints during an iteration is proportional to the step size.

Lemma 3.2. *Let $\ell \in \mathbb{N}$ be an iteration and fix the outcomes of iterations 1 through $\ell-1$. We have,*

$$\begin{aligned} &\mathrm{E}\left[\langle v^{\ell+1}, \mathbb{1}\rangle\right] \le \exp\left((1+O(\epsilon))\delta^\ell\eta\right)\langle v^\ell, \mathbb{1}\rangle \\ \textit{and} \quad &\mathrm{E}\left[\ln\left(\langle v^{\ell+1}, \mathbb{1}\rangle\right)\right] \le (1+O(\epsilon))\delta^\ell\eta + \ln\left(\langle v^\ell, \mathbb{1}\rangle\right). \end{aligned}$$

Proof. By Lemma 3.1 and linearity of expectation, we have

$$\mathrm{E}\left[\langle v^{\ell+1}, \mathbb{1}\rangle\right] \le \sum_{i\in\mathcal{P}} \exp\left((1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle\right)v_i^\ell.$$

The sum on the right hand side is essentially the sum for the standard (exact and deterministic) multiplicative weight update. Standard analysis of MWU, given at the end of the proof for the sake of completeness, implies the following inequality (3.2).

$$\sum_{i\in\mathcal{P}} \exp\left((1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle\right)v_i^\ell \le \exp\left((1+O(\epsilon))\delta^\ell\eta\right)\langle v^\ell, \mathbb{1}\rangle \tag{3.2}$$

Assuming (3.2), we have,

$$\mathrm{E}\left[\langle v^{\ell+1}, \mathbb{1}\rangle\right] \le \exp\left((1+O(\epsilon))\delta^\ell\eta\right)\langle v^\ell, \mathbb{1}\rangle,$$

for the first inequality. To obtain the second inequality, applying Jensen's inequality, we have

$$\begin{aligned} \mathrm{E}\left[\ln\left(\langle v^{\ell+1}, \mathbb{1}\rangle\right)\right] &\le \ln\left(\mathrm{E}\left[\langle v^{\ell+1}, \mathbb{1}\rangle\right]\right) \\ &\le (1+O(\epsilon))\delta^\ell\eta + \ln\left(\langle v^\ell, \mathbb{1}\rangle\right), \end{aligned}$$

as desired.

We now prove (3.2). Let $\alpha_i = \langle e_i, Ay^\ell\rangle$. The choice of δ^ℓ ensures that e $\delta^\ell\eta\alpha_i \le \epsilon$ for all $i \in \mathcal{P}$. Using the fact that $\exp(z) \le 1+z+z^2$ for $z \in [0, 1/2)$ and ϵ is sufficiently small,

$$\begin{aligned} &\sum_{i\in\mathcal{P}} \exp\left((1+\epsilon)\delta^\ell\eta\langle e_i, Ay^\ell\rangle\right)v_i^\ell \\ &= \sum_{i\in\mathcal{P}} \exp\left((1+\epsilon)\delta^\ell\eta\alpha_i\right)v_i^\ell \end{aligned}$$

$$\begin{aligned}
&\le \sum_{i\in\mathcal{P}} v_i^\ell \big(1 + (1+\epsilon)\delta^\ell\eta\alpha_i + ((1+\epsilon)\delta^\ell\eta\alpha_i)^2\big) \\
&= \sum_{i\in\mathcal{P}} v_i^\ell + (1+\epsilon)\sum_{i\in\mathcal{P}} v_i^\ell\delta^\ell\alpha_i(1+(1+\epsilon)\delta^\ell\eta\alpha_i) \\
&\le \sum_{i\in\mathcal{P}} v_i^\ell + (1+O(\epsilon))\delta^\ell\eta\sum_{i\in\mathcal{P}} v_i^\ell\alpha_i \\
&= \langle v^\ell, \mathbb{1}\rangle + (1+O(\epsilon))\delta^\ell\eta\langle v, Ay^\ell\rangle \\
&\le \sum_{i\in\mathcal{P}} v_i^\ell + (1+O(\epsilon))\delta^\ell\eta\langle v^\ell, \mathbb{1}\rangle
\end{aligned}$$

as y^ℓ satisfies $\langle v, Ax\rangle \le \langle v, \mathbb{1}\rangle$,

$$\begin{aligned}
&= (1 + (1+O(\epsilon))\delta^\ell\eta)\langle v^\ell, \mathbb{1}\rangle \\
&\le \exp\big((1+O(\epsilon))\delta^\ell\eta\big)\langle v^\ell, \mathbb{1}\rangle,
\end{aligned}$$

as desired. The last inequality uses the fact that $1+z \le \exp(z)$. ■

In the deterministic version of MWU the weight of a packing constraint i is exponential in the load of constraint i. The next lemma shows that the randomized weight tracks the load closely. Here it is more convenient to work with the logarithm of the weight.

Lemma 3.3. *For sufficiently small $\epsilon > 0$, any i, any $L \in \mathbb{N}$, and any $\zeta > 0$,*

$$\mathrm{P}\big[(1-\epsilon)\eta\langle e_i, Ax^{L+1}\rangle \ge \ln\big(v_i^{L+1}\big) + \zeta\big] \le (1-\epsilon)^{\zeta/\epsilon}.$$

Proof. For each index $\ell \in [L]$, let

$$X_\ell = \frac{\ln\big(v_i^{\ell+1}\big) - \ln\big(v_i^\ell\big)}{\epsilon} \text{ and } Y_\ell = \frac{\eta\langle e_i, Ay^\ell\rangle}{\epsilon}.$$

Then

$$\epsilon\sum_{\ell=1}^L X_\ell = \ln\big(v_i^{L+1}\big) \text{ and } \epsilon\sum_{\ell=1}^L Y_\ell = \eta\langle e_i, Ax_L^{\ell+1}\rangle,$$

so

$$\begin{aligned}
&\mathrm{P}\big[(1-\epsilon)\eta\langle e_i, Ax^{L+1}\rangle \ge \ln\big(v_i^{L+1}\big) + \zeta\big] \\
&\quad = \mathrm{P}\left[\sum_{\ell=1}^L X_\ell \le (1-\epsilon)\sum_{\ell=1}^L Y_\ell - \frac{\zeta}{\epsilon}\right].
\end{aligned}$$

Moreover, by line [4], we have $X_\ell \in [0,1]$, and by Lemma 3.1, we have $\mathrm{E}[X_\ell \mid X_1, \ldots, X_{\ell-1}, Y_1, \ldots, Y_\ell] = Y_\ell$. Therefore, by Theorem 2.2 (b), we have

$$\mathrm{P}\left[\sum_{\ell=1}^L X_\ell \le (1-\epsilon)\sum_{\ell=1}^L Y_\ell - \frac{\zeta}{\epsilon}\right] \le (1-\epsilon)^{\zeta/\epsilon},$$

as desired. ■

The next lemma shows that logarithm of the total sum of packing weights closely tracks the time with appropriate normalization.

Lemma 3.4. *For sufficiently small $\epsilon > 0$, any $L \in \mathbb{N}$, and any $\zeta > 0$,*

$$\begin{aligned}
&\mathrm{P}\big[\ln\big(\langle v^{L+1}, \mathbb{1}\rangle\big) \ge \ln(m) + (1+O(\epsilon))\eta t^{L+1} + \zeta\big] \\
&\quad \le (1+\epsilon)^{-\zeta/\epsilon}.
\end{aligned}$$

Proof. For each index $\ell \in [L]$, let

$$X_\ell = \frac{\ln\big(\langle v^{\ell+1}, \mathbb{1}\rangle\big) - \ln\big(\langle v^\ell, \mathbb{1}\rangle\big)}{\epsilon} \text{ and } Y_\ell = \frac{\eta\delta^\ell}{\epsilon},$$

where $X_\ell = Y_\ell = 0$ for indices $\ell > \hat{\ell}$ after the algorithm has ended. Then

$$\epsilon\sum_{\ell=1}^L X_\ell = \ln\big(\langle v^{L+1}, \mathbb{1}\rangle\big) - \ln(m) \text{ and } \epsilon\sum_{\ell=1}^L Y_\ell = \eta t^{L+1},$$

so

$$\begin{aligned}
&\mathrm{P}\big[\ln\big(\langle v^{L+1}, \mathbb{1}\rangle\big) \ge \ln(m) + (1+O(\epsilon))\eta t^{L+1} + \zeta\big] \\
&\quad = \mathrm{P}\left[\sum_{\ell=1}^L X_\ell \ge (1+O(\epsilon))\sum_{\ell=1}^L Y_\ell + \frac{\zeta}{\epsilon}\right].
\end{aligned}$$

Moreover, for each index ℓ, we have $X_\ell \in [0,1]$ by line [4], and $\mathrm{E}[X_\ell] \le (1+O(\epsilon))Y_\ell$ by Lemma 3.2. Therefore, by Theorem 2.2 (a),

$$\mathrm{P}\left[\sum_{\ell=1}^L X_\ell \ge (1+O(\epsilon))\sum_{\ell=1}^L Y_\ell + \frac{\zeta}{\epsilon}\right] \le (1+\epsilon)^{-\zeta/\epsilon},$$

as desired. ■

Lemma 3.5. *For sufficiently small $\epsilon > 0$, any i, any $L \in \mathbb{N}$, and $\eta \ge \ln(m)/\epsilon$,*

$$\mathrm{P}\big[\langle e_i, Ax^{L+1}\rangle \ge 1 + O(\epsilon)\big] \le \frac{1}{\mathrm{poly}(m)}.$$

Proof. Let $L \in \mathbb{N}$, and let $\zeta > 0$ be a parameter to be fixed later. We have

$$(1-\epsilon)\eta\langle e_i, Ax^{L+1}\rangle \le \ln\big(v_i^{L+1}\big) + \zeta$$

by Lemma 3.3,

$$\le \ln\big(\langle v^{L+1}, \mathbb{1}\rangle\big) + \zeta$$

since $\hat{v} \ge \mathbb{1}$,

$$\le \ln(m) + \eta t^{L+1} + 2\zeta$$

by Lemma 3.4,

$$\le \ln(m) + \eta + 2\zeta$$

since $t^{L+1} \le 1$, with total probability of failure, by the union bound, of at most $(1+\epsilon)^{-\zeta/\epsilon} + (1-\epsilon)^{\zeta/\epsilon}$. For $\zeta = O(\ln m)$, the claim follows. ■

3.2 Covering constraints

We now prove that `random-mwu` approximately satisfies the covering constraints with high probability. The first lemma observes that the randomized weight update does the "right thing" (per the MWU framework) in expectation.

Lemma 3.6. *Let $\ell \in [L]$ be an iteration and let the outcomes of iterations 1 through $\ell - 1$ be fixed. For each remaining covering constraint $i \in \mathcal{Q}^\ell$, we have*

$$\mathrm{E}\left[\ln\left(w_i^{\ell+1}\right)\right] \geq \ln\left(w_i^\ell\right) - \delta^\ell \eta \langle e_i, By^\ell \rangle$$

and

$$\mathrm{E}\left[w_i^{\ell+1}\right] \leq \exp\left(-(1-\epsilon)\delta^\ell \eta \langle e_i, By^\ell \rangle\right) w_i^\ell.$$

Proof. The first inequality is immediate from the algorithm. For the second, we have

$$\begin{aligned}
&\mathrm{E}\left[w_i^{\ell+1}\right] \\
&= \frac{\delta^\ell \eta \langle e_i, By^\ell \rangle}{\epsilon} \exp(-\epsilon) w_i^\ell + \left(1 - \frac{\delta^\ell \eta \langle e_i, By^\ell \rangle}{\epsilon}\right) w_i^\ell \\
&= w_i^\ell - (1 - \exp(-\epsilon)) \left(\frac{\delta^\ell \eta \langle e_i, By^\ell \rangle}{\epsilon}\right) w_i^\ell \\
&\leq w_i^\ell - (1-\epsilon)\delta^\ell \eta \langle e_i, By^\ell \rangle w_i^\ell
\end{aligned}$$

since $\exp(-\epsilon) \leq 1 - \epsilon + \epsilon^2$ for $\epsilon > 0$,

$$\leq \exp\left(-(1-\epsilon)\delta^\ell \eta \langle e_i, By^\ell \rangle\right) w_i^\ell$$

since $1 - z \leq \exp(-z)$ for $z \geq 0$, as desired. ∎

We now consider the sum of covering weights of the active constraints and the expectation of the change in a single iteration.

Lemma 3.7. *Let $\ell \in [L]$ be an iteration and let the outcomes of iterations 1 through $\ell - 1$ be fixed. Then*

$$\mathrm{E}\left[\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right] \leq \exp\left(-(1 - O(\epsilon))\eta\delta^\ell\right) \left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell} \tag{3.3}$$

and

$$\mathrm{E}\left[\ln\left(\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right)\right] \leq \ln\left(\left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right) - (1 - O(\epsilon))\eta\delta^\ell.$$

Proof. By Lemma 3.6 and linearity of expectation, we have

$$\mathrm{E}\left[\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right] \leq \sum_{i \in \mathcal{Q}^\ell} \exp\left((1-\epsilon)\delta^\ell \eta \langle e_i, By^\ell \rangle\right) w_i^\ell.$$

The right hand side is essentially the sum if the weight update were executed exactly and deterministically. The standard analysis for (deterministic) MWU, similar to the proof of (3.2) above, implies the following inequality:

$$\begin{aligned}
&\sum_{i \in \mathcal{Q}^\ell} \exp\left(-(1-\epsilon)\delta^\ell \eta \langle e_i, By^\ell \rangle\right) w_i^\ell \\
&\quad \leq \exp\left(-(1 - O(\epsilon))\delta^\ell \eta\right) \left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}
\end{aligned} \tag{3.4}$$

By (3.4), we have,

$$\mathrm{E}\left[\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right] \leq \exp\left(-(1 - O(\epsilon))\delta^\ell \eta\right) \left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell},$$

as desired. For the second inequality, by Jensen's inequality, we have,

$$\begin{aligned}
&\mathrm{E}\left[\ln\left(\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right)\right] \\
&\leq \ln\left(\mathrm{E}\left[\left\langle w^{\ell+1}, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right]\right) \\
&\leq \ln\left(\exp\left(-(1 - O(\epsilon))\eta\delta^\ell\right) \left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right) \quad \text{by (3.3),} \\
&= \ln\left(\left\langle w^\ell, \mathbb{1} \right\rangle_{\mathcal{Q}^\ell}\right) - (1 - O(\epsilon))\eta\delta^\ell,
\end{aligned}$$

as desired. ∎

The lemma below shows that the logarithm of the covering weight w_i tracks the load of covering constraint i closely.

Lemma 3.8. *For sufficiently small $\epsilon > 0$, any i, any $L \in \mathbb{N}$, and any $\zeta > 0$,*

$$\mathrm{P}\left[(1+\epsilon)\eta \langle e_i, Bx^{L+1} \rangle \leq \ln\left(w_i^{L+1}\right) - \zeta\right] \leq (1-\epsilon)^{\zeta/\epsilon}.$$

Proof. For each index $\ell = 1, \dots, L$, let

$$X_\ell = \frac{\ln\left(w_i^\ell\right) - \ln\left(w_i^{\ell+1}\right)}{\epsilon} \text{ and } Y_\ell = \frac{\delta^\ell \eta \langle e_i, By^\ell \rangle}{\epsilon},$$

where $X_\ell = Y_\ell = 0$ for indices $\ell > \hat{\ell}$ after the algorithm terminates. Then

$$\epsilon \sum_{\ell=1}^{L} X_\ell = \ln\left(w_i^1\right) - \ln(\hat{w}_i) = \ln(\hat{w}_i) \text{ and}$$

$$\epsilon \sum_{\ell=1}^{L} Y_\ell = \eta \langle e_i, Bx^{L+1} \rangle,$$

so

$$\begin{aligned}
&\mathrm{P}\left[(1+\epsilon)\eta \langle e_i, B\hat{x} \rangle \leq -\ln(\hat{w}) - \zeta\right] \\
&= \mathrm{P}\left[\sum_{\ell=1}^{L} X_\ell \geq (1+\epsilon)\sum_{\ell=1}^{L} Y_\ell + \frac{\zeta}{\epsilon}\right].
\end{aligned}$$

By line `[5]`, $X_\ell \in [0,1]$ for each iteration ℓ. By Lemma 3.6, for each iteration ℓ,

$\mathrm{E}[X_\ell \mid X_1, \ldots, X_{\ell-1}, Y_1, \ldots, Y_\ell] = Y_\ell$. Therefore, by Theorem 2.2 (a), we have

$$\mathrm{P}\left[\sum_{\ell=1}^{L} X_\ell \geq (1+\epsilon) \sum_{\ell=1}^{L} Y_\ell + \frac{\zeta}{\epsilon}\right] \leq (1+\epsilon)^{-\zeta/\epsilon},$$

as desired. ■

Lemma 3.9. *For $L \in \mathbb{N}$, and any i, $-(1+\epsilon)\ln\big(w_i^{L+1}\big) \geq -\ln\Big(\big\langle \mathbb{1}, w^{L+1}\big\rangle_{\mathcal{Q}^{L+1}}\Big)$.*

Proof. Since $\mathcal{Q}^{L+1} \neq \emptyset$, and $w_i^{L+1} \geq \exp(-(1+\epsilon)\eta)$ for any $i \in \mathcal{Q}^{L+1}$, we have

$$\ln\Big(\big\langle \mathbb{1}, w^{L+1}\big\rangle_{\mathcal{Q}^{L+1}}\Big) \geq -(1+\epsilon)\eta.$$

If $i \notin \hat{\mathcal{Q}}$, then $w_i^{L+1} \leq \exp(-\eta)$ because i is made inactive only if w_i drops below $\exp(-\eta)$. Therefore,

$$-(1+\epsilon)\ln\big(w_i^{L+1}\big) \geq (1+\epsilon)\eta \geq -\ln\Big(\big\langle \mathbb{1}, w^{L+1}\big\rangle_{\mathcal{Q}^{L+1}}\Big).$$

If $i \in \hat{\mathcal{Q}}$, then $w_i^{L+1} \leq \big\langle w^{L+1}, \mathbb{1}\big\rangle_{\mathcal{Q}^{L+1}}$ since all weights are nonnegative, and

$$-\ln\big(w_i^{L+1}\big) \geq -\ln\Big(\big\langle w^{L+1}, \mathbb{1}\big\rangle_{\mathcal{Q}^{L+1}}\Big),$$

as desired. ■

The lemma below relates the evolution of the logarithm of the sum of covering weights and the time variable.

Lemma 3.10. *For sufficiently small $\epsilon > 0$, any $L \in \mathbb{N}$, and any $\zeta > 0$,*

$$\mathrm{P}\Big[\ln\Big(\big\langle \mathbb{1}, w^{L+1}\big\rangle_{\mathcal{Q}^{L+1}}\Big) \geq \ln(m) - (1-O(\epsilon))\eta t^{L+1} + \zeta\Big]$$

$$\leq (1-\epsilon)^{\zeta/\epsilon}.$$

Proof. For $\ell = 1, \ldots, L$, let

$$X_\ell \overset{\text{def}}{=} \frac{\ln\Big(\big\langle w^\ell, \mathbb{1}\big\rangle_{\mathcal{Q}^\ell}\Big) - \ln\Big(\big\langle w^{\ell+1}, \mathbb{1}\big\rangle_{\mathcal{Q}^\ell}\Big)}{\epsilon}, \text{ and}$$

$$Y_\ell \overset{\text{def}}{=} \eta\delta^\ell,$$

where each variable takes value 0 on indices $\ell > \hat{\ell}$ after the algorithm terminates. Then

$$\sum_{\ell=1}^{L} X_\ell \leq \ln(m) - \ln\Big(\big\langle \mathbb{1}, w^{L+1}\big\rangle_{\mathcal{Q}^{L+1}}\Big) \text{ and}$$

$$\sum_{\ell=1}^{L} Y_\ell = \frac{\eta}{\epsilon} t^{L+1},$$

so

$$\mathrm{P}\Big[\ln\big(\big\langle \mathbb{1}, w^{L+1}\big\rangle\big)_{\mathcal{Q}^{L+1}} \geq \ln(m) - (1-O(\epsilon))\eta t^{L+1} + \zeta\Big]$$

$$= \mathrm{P}\Big[\ln(m) - \ln\big(\big\langle \mathbb{1}, w^{L+1}\big\rangle\big)_{\mathcal{Q}^{L+1}} \leq (1-O(\epsilon))\eta t^{L+1} - \zeta\Big]$$

$$\leq \mathrm{P}\left[\sum_{\ell=1}^{L} X_\ell \leq (1-O(\epsilon)) \sum_{\ell=1}^{L} Y_\ell - \zeta/\epsilon\right].$$

For each ℓ, we have $X_\ell \in [0,1]$, and by Lemma 3.7,

$$\mathrm{E}[X_\ell \mid X_1, \ldots, X_{\ell-1}, Y_1, \ldots, Y_\ell] \geq (1-O(\epsilon))Y_\ell.$$

By Theorem 2.2, we have

$$\mathrm{P}\left[\sum_{\ell=1}^{L} X_\ell \leq (1-O(\epsilon)) \sum_{\ell=1}^{L} Y_\ell - \zeta/\epsilon\right] \leq (1-\epsilon)^{\zeta/\epsilon},$$

as desired. ■

Now we relate the load on covering constraint i to time.

Lemma 3.11. *For sufficiently small $\epsilon > 0$, and $\eta = O(\ln(m)/\epsilon)$, and any i,*

$$\mathrm{P}\big[\big\langle e_i, Bx^{L+1}\big\rangle \geq (1-O(\epsilon))t^{L+1} - O(\epsilon)\big] \leq \frac{1}{\operatorname{poly}(m)}.$$

Proof. Let $\zeta > 0$ be a parameter to be specified later. We have

$$\begin{aligned}
&(1+\epsilon)\eta\big\langle e_i, Bx^{L+1}\big\rangle \\
&\geq -\ln\big(w_i^{L+1}\big) - \zeta && \text{by Lemma 3.8,} \\
&\geq -(1-\epsilon)\ln\Big(\big\langle w^{L+1}, \mathbb{1}\big\rangle_{\mathcal{Q}^{L+1}}\Big) - \zeta && \text{by Lemma 3.9,} \\
&\geq (1-O(\epsilon))(\eta + \ln m)t^{L+1} - 2\zeta
\end{aligned}$$

with total probability of failure $\leq (1-\epsilon)^{\zeta/\epsilon} + (1+\epsilon)^{-\zeta/\epsilon}$ by the union bound. For $\zeta = O(\ln m)$, we have $\zeta/\eta = O(\epsilon)$ and $(1-\epsilon)^{\zeta/\epsilon} + (1+\epsilon)^{-z\eta/\epsilon} = 1/\operatorname{poly}(m)$, as desired. ■

The preceding lemma shows that all covering constraints will be satisfied with high probability if the algorithm terminates with $t \geq (1-O(\epsilon))$.

3.3 Iterations

In this section, we analyze the number of iterations taken by `random-mwu`, which we have yet to even show is finite.

Lemma 3.12. *With probability $1 - 1/\operatorname{poly}(m)$, `random-mwu` terminates within $O\big((m_c + \min\{m_p, n\})\ln(m)/\epsilon^2\big)$ iterations and within $O\big(m\ln(m)/\epsilon^2\big)$ individual updates to weights.*

Proof. Consider the first L iterations, where $L \in \mathbb{N}$ is chosen to be sufficiently large as specified in the statement. We will argue that the algorithm will terminate with high probability in less than L iterations.

Every iteration $\ell \in [L]$, by choice of δ^ℓ in line `[3]`, there exists either a packing constraint $i \in \mathcal{P}$ such that

$$\delta^\ell \eta \langle e_i, Ay^\ell \rangle = \epsilon,$$

or a covering constraint $i \in \mathcal{Q}^\ell$ such that

$$\delta^\ell \eta \langle e_i, By^\ell \rangle = \epsilon.$$

This constraint has its weight updated by a multiplicative factor of $\exp(\epsilon)$ *deterministically.* That is, every iteration updates at least one weight deterministically.

Since the algorithm terminates once $w \leq \exp(-\eta)\mathbb{1}$, each covering constraint w_i is only updated by a factor of $\exp(\epsilon)$ at most $\eta/\epsilon = O(\ln(m)/\epsilon^2)$ times before before i is removed from $\mathcal{Q}^\ell$. Thus, the total number of weight updates to covering constraints is at most $O(m \ln(m)/\epsilon^2)$. Now we consider weight updates to packing constraints. With high probability, by Lemma 3.4, we have $\langle v^{L+1}, \mathbb{1} \rangle \leq m^{O(1/\epsilon)}$. In particular, we have $v_i^{L+1} \leq m^{O(1/\epsilon)}$ with high probability for each packing constraint $i \in \mathcal{P}$. In such an event, a packing constraint v_i can only be increased by a factor of $\exp(\epsilon)$ at most $\ln_{\exp(\epsilon)}(m^{O(1/\epsilon)}) = O(\ln(m)/\epsilon^2)$ before $v_i \geq m^{O(1/\epsilon)}$.

Thus, with high probability, each constraint has its weight updated by a factor of $\exp(\epsilon)$ at most $O(\ln(m)/\epsilon^2)$ times. The total number of weight updates, then, is with high probability, $O(m \ln(m)/\epsilon^2)$. If we charge each iteration to a constraint updated by a factor of $\exp(\epsilon)$, then there are at most $O(m \ln(m)/\epsilon^2)$ iterations. Thus the algorithm terminates, with high probability, in $O(m \ln(m)/\epsilon^2)$ iterations.

We obtain a refined bound as follows. Recall that each iteration ℓ picks a solution y^ℓ which has a single non-zero coordinate $j \in [n]$. Every time we select j, we either decreases a covering weight by a $(1+\epsilon)$-multiplicative factor, or increase the *same* packing weight by a $(1+\epsilon)$ multiplicative-factor; this packing constraint corresponds to the bottleneck packing constraint for j (the row with the largest coefficient in j's column). If a packing weight is increased by an $(1+\epsilon)$-multiplicative factor, then we can only select coordinate j $O(\ln(m)/\epsilon^2)$ times before this particular packing weight hits the upper bound. Thus, charging each iteration to either a covering weight decreasing by a $(1+\epsilon)$-multiplicative factor, or increase the same packing weight per coordinate by a $(1+\epsilon)$-multiplicative factor, there are at most $O((m_c + n)\ln(m)/\epsilon^2)$ iterations. Taking the minimum of the two upper bounds gives the upper bound we seek. ■

Remark 3.1. *The two concentration bounds – one ensuring correctness, and the other bounding the running time – are obtained not separately but jointly: bottleneck operations are amortized against the same invariants of the framework that ensure the correctness of the output.*

3.4 Tying it all together

We restate and complete the proof of Theorem 2.1.

Theorem 2.1 1. *Let $A \in \mathbb{R}_{\geq 0}^{m_p \times n}$ and $B \in \mathbb{R}_{\geq 0}^{m_c \times n}$ be nonnegative matrices for which there exists a nonnegative $x \in \mathbb{R}_{\geq 0}^n$ such that $Ax \leq \mathbb{1}$ and $Bx \geq \mathbb{1}$. Let $m = m_p + m_c$, and let N be the total number of nonzero coefficients in A and B.*

With probability $1 - 1/\operatorname{poly}(m)$, `random-mwu(A,B,`$\epsilon$`)` returns a point $\hat{x}$ such that $A\hat{x} \leq (1+O(\epsilon))\mathbb{1}$ and $B\hat{x} \geq (1-O(\epsilon))\mathbb{1}$ in $O((m_c + \min\{m_p, n\})\ln(n)/\epsilon^2)$ iterations and, excluding the time spent in lines `[2]` and `[3]`, $\tilde{O}(N + m\ln(n)/\epsilon^2)$ time. Each packing weight $v_i, i \in \mathcal{P}$ increases along integral powers of $\exp(\epsilon)$ from 1 to (at most) $\exp(\ln(m_p)/\epsilon)$. Each covering weight $w_i, i \in \mathcal{C}$ decreases along integral powers of $\exp(\epsilon)$ from 1 to $\exp(-\ln(m_c)/\epsilon)$.

Proof. Let $L = \Theta((m_c + \min\{m_p, n\})\ln(m)/\epsilon^2)$ be a sufficiently large but fixed number. Lemma 3.12 shows that with probability $1 - 1/\operatorname{poly}(m)$ the algorithm terminates in less than L iterations. Let $\mathcal{E}$ be the event that the algorithm terminates before L iterations. Conditioning on $\mathcal{E}$, there is a finite and well-defined total number of iterations, $\hat{\ell} \leq L$, a final time, $\hat{t} = t^{\hat{\ell}+1}$, and output $\hat{x} = x^{\hat{\ell}+1}$. Note that the algorithm terminates only if the time variable reaches 1 or if Q is empty. Hence $\hat{t} = 1$ or $\mathcal{Q}^{\hat{\ell}} = \emptyset$.

Conditioning on $\mathcal{E}$, Lemma 3.5, Lemma 3.11, Lemma 3.8 when applied to the fixed iteration L imply that they hold for the last iteration $\hat{\ell} < L$ since all the variables are frozen after the algorithm terminates.

By Lemma 3.5 and union bound, $A\hat{x} \leq (1+O(\epsilon))$. By Lemma 3.11, $B\hat{x} \geq (1-O(\epsilon))\hat{t} - O(\epsilon)$. If $\hat{t} = 1$, then $B\hat{x} \geq (1-O(\epsilon))$, as desired. If $\hat{t} < 1$, then $w_i^{\hat{\ell}+1} \leq \exp(-\eta)$ for all $i \in [m_c]$, so $B\hat{x} \geq (1-O(\epsilon))$ by Lemma 3.8 and the union bound. Thus, conditioned on $\mathcal{E}$, we have $A\hat{x} \leq (1+O(\epsilon))$ and $B\hat{x} \geq (1-O(\epsilon))$ with probability $1 - 1/\operatorname{poly}(m)$. Since $\mathrm{P}[\mathcal{E}] \geq 1 - 1/\operatorname{poly}(m)$ we have that the desired claim on both the correctness of the output and the number of iterations. ■

4 Approximating nonnegative matrices

We recall Theorem 2.3 which describes a data structure for online maintenanace of Ax as x monotonically increases. Since we also need to handle the case when

x monotonically decreases we set up the data structure and the setting in more generality.

REMARK 4.1. *For notational simplicity we choose the dimensions as* $A \in \mathbb{R}^{m}_{\geq 0}\times$ *and* $x \in \mathbb{R}^n_{\geq 0}$. *However, when we apply the results in this section to analyze* `rand-mwu`*, we will be using it for the transpose of* $m \times n$ *matrices, and* m *and* n *are swapped in the time bounds.*

The data structure has two main routines. `init` takes as input a nonnegative matrix $A \in \mathbb{R}^{m\times n}_{\geq 0}$, error tolerance ϵ, and δ which controls the failure probability. `inc` takes two inputs, a real value α and a coordinate $j \in [n]$ to which the increment of α is to be applied. The data structure maintains at all times maintains an estimate y of Ax which can be directly accessed. Here we state the setting with additional details that will be useful in our application.

SETTING 4.1. *Let* $\epsilon, \delta > 0$ *with* ϵ *sufficiently small and* δ *sufficiently large. Let* $A \in \mathbb{R}^{m\times n}_{\geq 0}$ *be a nonnegative matrix with* N *nonzeroes. Consider an instance of* `apx-matrix` *initialized by* `init(`ϵ`,`δ`,`A`)`. *Let* L *be a fixed parameter, and let* `inc(`α_1`,`j_1`)`*, ...,* `inc(`α_L`,`j_L`)` *be a sequence of calls to* `inc` *delivered online. For* $\ell = 0, \ldots, L$, *let* $x^\ell = \mathbb{1} + \sum_{k=1}^{\ell} \alpha_k e_{j_k}$. *The online sequence is constrained in two ways.*

(i) Every α_i *has the same sign; i.e.,* x^ℓ *is either monotonically increasing or monotonically decreasing.*

(ii) For a fixed parameter $\beta > 0$, *for every* $\ell \in [L]$, *and every nonzero* $A_{ij_\ell} \neq 0$ *incident to the* ℓ*th update* `inc(`α_ℓ`,`e_{j_ℓ}`)`*, we have*

$$\frac{1}{\beta}\langle e_i, A\mathbb{1}\rangle \leq \left\langle e_i, Ax^{\ell-1}\right\rangle \leq \beta\langle e_i, A\mathbb{1}\rangle.$$

`apx-matrix` *maintains an estimate* y *of* Ax. *For* $\ell \in [L]$, *let* y^ℓ *be the value of* y *after the* ℓ*th call to* `inc`.

High-level idea: The data structure is based in a simple idea related to online randomized maintenance of counters [26, 10]. Consider the online maintenance of a single number x (the one dimensional case) as we provide increments $\alpha_1, \ldots, \alpha_k$. We maintain an estimate y. If the current increment α_i is *small relative to the current estimate* y we update y probabilistically; the cost here is to actually add to y while we ignore the time to check α_i/y. On the other hand if α_i is *large* we update y deterministically for otherwise we would incur too much variance. The key is take this idea to higher-dimensional setting where y is a vector and each output coordinate is influenced by multiple coordinates of A. To handle this we maintain the relative importance of each coordinate with respect to the current estimate y (these are the values θ_{ij}) and periodically *reset and rescale* when the estimate y changes significantly (by a constant factor relatively) in any coordinate. Given increment α to coordinate j we cannot afford to evaluate all the nonzero coordinates in row i for that would defeat the purpose of improving the run-time. The same idea of correlated random choice for weight updates is again used here.

Now we formally prove that our scheme maintains the estimate y correctly with high probability when parameters are set appropriately. We will subsequently analyze the running time.

LEMMA 4.1. *Assuming Setting 4.1,*

$$\text{(4.5)} \quad \mathrm{P}\left[y^\ell \notin (1 \pm O(\epsilon))Ax^\ell \textit{ for any } \ell \in L\right] \leq L^2 m \cdot \mathrm{poly}(\delta).$$

Proof. Fix a row $i \in [m]$. We analyze the probability that $y_i \approx \langle e_i, Ax\rangle$ after each increment, and then take a union bound over all the rows at the end.

Let a "phase" be the sequence of calls to `inc` between consecutive calls to `reset(`i`)` (including the last `inc` that triggers `reset(`i`)`). The last phase does not necessarily end with a `reset`. At the beginning of a phase, y_i is recomputed exactly in line `[6]`. We argue that, for a fixed phase, `apx-matrix` maintains y_i close to $\langle e_i, Ax_i\rangle$ throughout the phase with high probability; we then take a union bound over all the phases.

Fix a phase. Let $\bar{x}^0$ be the value of x at the beginning of the phase, and let $\bar{y}_i^0 = A\bar{x}^0$ mark the (recomputed) value of y_i at the beginning of the phase. Let $\alpha_1 e_{j_1}, \alpha_2 e_{j_2}, \ldots$ be the increments during the phase. By assumption, there are at most L increments in a phase, and padding the end of the sequence with "zero" increments, we simply assume there are exactly L increments. For $\ell = 1, \ldots, L$, let $\bar{x}^\ell = \bar{x}^0 + \sum_{k=1}^{\ell} \alpha_k e_{j_k}$ and $\bar{y}_i^\ell$ the value of y_i after the ℓth increment is processed. When the all the increments are positive or all the increments are negative, both $\left\{\bar{y}_i^\ell\right\}$ and $\left\{\bar{x}^\ell\right\}$ are monotonic sequences in the same direction as the increments. Moreover, once y_i^ℓ leaves the ranges $[\bar{y}_i^0/2, 2\bar{y}_i^0]$, the phase ends and all the increments thereafter are zero. We want to show that $\bar{y}_i^\ell \approx \bar{x}^\ell$ for every $\ell \in [L]$ with high probability.

We first consider the increasing case, where $\alpha_\ell \geq 0$ for each ℓ. Then $\left\{\bar{x}^\ell\right\}$ and $\left\{\bar{y}_i^\ell\right\}$ are monotonically increasing, and the phase ends when $\bar{y}_i^\ell \geq 2\bar{y}_i^0$. Noting that $\gamma = 1/\delta$, we define random variables $Z_1, Z_2, \ldots Z_L \geq 0$ by $Z_\ell = \frac{\ln(\gamma)\left(\bar{y}_i^\ell - \bar{y}_i^{\ell-1}\right)}{\epsilon^2 \bar{y}_i^0}$. For each $\ell \in [L]$, we have $\mathrm{E}[Z_\ell] = \frac{\ln(\gamma)\alpha_\ell A_{ij_\ell}}{\epsilon^2 \bar{y}_i^0}$. If $\alpha_\ell \neq 0$, then $y_i^\ell \leq 2\bar{y}_i^0$. By choice of the importance sampling thresh-

```
apx-matrix

init($\epsilon$,$\delta$,$A$)
    $x \leftarrow \mathbb{1}$
    $\gamma \leftarrow 1/\delta$
    for each $i \in [m]$
        reset($i$)

reset($i$)
[6] $y_i \leftarrow \sum_{A_{ij} \neq 0} A_{ij} x_j$
    $\tilde{y}_i \leftarrow y_i$
    for each $A_{ij} \neq 0$
[7]     $\log \theta_{ij} \leftarrow \left\lfloor \log \frac{\epsilon^2 \tilde{y}_i}{\ln(\gamma) A_{ij}} \right\rfloor$

inc($\alpha \in \mathbb{R}$,$j \in [n]$)
    $x_j \leftarrow x_j + \alpha$
    let $\pi \in [0,1]$ random
[8] for each $i$ s.t. $\frac{|\alpha|}{\theta_{ij}} \geq \pi$
[9]     $y_i \leftarrow y_i + \mathrm{sign}(\alpha) A_{ij} \max\{\theta_{ij}, |\alpha|\}$
        unless $y_i/2 \leq \tilde{y}_i \leq 2y_i$
            reset($i$)
```

Figure 2: apx-matrix *is a randomized data structure that, given a nonnegative matrix $A \in \mathbb{R}^{m \times n}_{\geq 0}$, efficiently maintains a relative and uniform approximation for Ax over the lifetime of a nonnegative vector $x \in \mathbb{R}^n_{\geq 0}$ initialized to $\mathbb{1}$ and monotonically either increasing or decreasing (see Theorem 4.1).*

olds θ_{ij} in line [7], we have $\frac{\ln(\delta)\theta_{ij_\ell} A_{ij_\ell}}{\epsilon^2 \bar{y}_i^0} \leq 1$, hence

$$Z_\ell > 1 \text{ only if } |\alpha_\ell| > \theta_{ij_\ell} \text{ only if } Z_\ell = \alpha_\ell A_{ij_\ell} \text{ deterministically.}$$

Each Z_ℓ, conditional on the preceding $\ell - 1$ increments, still satisfies $\mathrm{E}\left[(1+\epsilon)^{Z_\ell}\right] \leq (1+\epsilon)^{\mathrm{E}[Z_\ell]}$ and $\mathrm{E}\left[(1-\epsilon)^{Z_\ell}\right] \leq (1-\epsilon)^{\mathrm{E}[Z_\ell]}$, so the conclusion of Theorem 2.2 still holds (and can be re-derived with standard techniques). By Theorem 2.2, we have

$$\begin{aligned}
&\mathrm{P}\left[y_i^\ell \geq (1+\epsilon)\langle e_i, A\bar{x}^\ell\rangle\right] \\
&= \mathrm{P}\left[\bar{y}_i^\ell - \bar{y}_i^0 \geq (1+\epsilon)\langle e_i, A\bar{x}^\ell\rangle - \bar{y}_i^0\right] \\
&= \mathrm{P}\left[\bar{y}_i^\ell - \bar{y}_i^0 \geq (1+\epsilon)\langle e_i, A(\bar{x}^\ell - \bar{x}^0)\rangle + \epsilon\bar{y}_i^0\right] \\
&= \mathrm{P}\left[\sum_{k=1}^{\ell} Z_k \geq \frac{(1+\epsilon)\ln(\gamma)}{\epsilon^2 \bar{y}_i^0} \sum_{k=1}^{\ell} \alpha_k A_{ij_k} + \frac{\ln(\gamma)}{\epsilon}\right] \\
&\leq (1+\epsilon)^{-\ln(\gamma)/\epsilon} = \mathrm{poly}(\delta)
\end{aligned}$$

and

$$\begin{aligned}
&\mathrm{P}\left[\bar{y}_i^\ell \leq (1-\epsilon)\langle e_i, A\bar{x}^\ell\rangle\right] \\
&\leq \mathrm{P}\left[\bar{y}_i^\ell - y_o^0 \leq (1-\epsilon)\langle e_i, A(\bar{x}^\ell - \bar{x}^0)\rangle - \epsilon\bar{y}_i^0\right] \\
&= \mathrm{P}\left[\bar{y}_i^\ell - \bar{y}_i^0 \leq (1-\epsilon)\langle e_i, A(\bar{x}^\ell - \bar{x}^0)\rangle - \epsilon\bar{y}_i^0\right] \\
&= \mathrm{P}\left[\sum_{k=1}^{\ell} Z_k \leq \frac{(1+\epsilon)\ln(\gamma)}{\epsilon^2 \bar{y}_i^0} \sum_{k=1}^{\ell} \alpha_k A_{ij_k} - \frac{\ln(\gamma)}{\epsilon}\right] \\
&\leq (1+\epsilon)^{-\ln(\gamma)/\epsilon} = \mathrm{poly}(\delta).
\end{aligned}$$

There are at most L increments in the phase. By the union bound, we have

(4.6)
$$\mathrm{P}\left[\bar{y}_i^\ell \notin (1 \pm \epsilon)\langle e_i, A\bar{x}^\ell\rangle \text{ for any } \ell \in [L]\right] \leq 2L\,\mathrm{poly}(\delta).$$

There are also at most L phases. By the union bound, all increments in all phases are accurate in the sense of (4.6) with probability of failure $L \cdot 2L\,\mathrm{poly}(\delta) = 2L^2\,\mathrm{poly}(\delta)$. Finally, we take a union bound over all coordinates $i \in [m]$, and conclude that the probability of any coordinate failing is at most $L^2 m\,\mathrm{poly}(\delta)$.

The decreasing case, where $\alpha_\ell \leq 0$ for each ℓ, is somewhat symmetric, with additional care required at the end. We confine our attention to a single phase. The sequences $\{\bar{y}_i^\ell\}$ and $\{\bar{x}^\ell\}$ are monotonically decreasing, and freeze when $\bar{y}_i^\ell \leq \bar{y}_i^0/2$. We define $Z_1, Z_2, \dots \geq 0$ by the scaled decrease, $Z_\ell = \frac{\ln(\gamma)(\bar{y}_i^{\ell-1} - \bar{y}_i^\ell)}{\epsilon^2 \bar{y}_i^0}$. For each $\ell \in [L]$, we have $\mathrm{E}[Z_\ell] = \frac{\ln(\gamma)\alpha_\ell A_{ij_\ell}}{\epsilon^2 \bar{y}_i^0}$. By choice of the sampling threshold θ_{ij_ℓ} in [7], we have $\frac{\ln(\delta)\theta_{ij_\ell} A_{ij_\ell}}{\epsilon^2 \bar{y}_i^0} \leq 1$, hence

$$Z_\ell > 1 \text{ only if } |\alpha| > \theta_{ij_\ell} \text{ only if } Z_\ell = \frac{\ln(\delta) A_{ij_\ell}\alpha_\ell}{\epsilon^2 \bar{y}_i^0} \text{ deterministically.}$$

Each Z_ℓ, conditional on the previous $\ell-1$ iterations, still satisfies $\mathrm{E}\left[(1+\epsilon)^{Z_\ell}\right] \leq (1+\epsilon)^{\mathrm{E}[Z_\ell]}$ and $\mathrm{E}\left[(1-\epsilon)^{Z_\ell}\right] \leq (1-\epsilon)^{\mathrm{E}[Z_\ell]}$, so the proof and theorem of Theorem 2.2

still holds. By Theorem 2.2, then, we have

$$\begin{aligned}
&\mathrm{P}\left[\bar{y}_i^\ell \leq \langle e_i, A\bar{x}^\ell \rangle - 2\epsilon \bar{y}_i^0\right] \\
&\quad = \mathrm{P}\left[\bar{y}_i^0 - \bar{y}_i^\ell \geq (1+\epsilon)\langle e_i, A(\bar{x}^0 - \bar{x}^\ell)\rangle - \epsilon \bar{y}_i^0\right] \\
&\quad = \mathrm{P}\left[\sum_{k=1}^{\ell} Z_k \geq \frac{(1+\epsilon)\ln(\gamma)}{\epsilon^2 \bar{y}_i^0} \sum_{k=1}^{\ell} \alpha_j A_{ij_k} + \frac{\ln(\gamma)}{\epsilon}\right] \\
&\quad \leq (1+\epsilon)^{-\ln(\gamma)/\epsilon} = \mathrm{poly}(\delta)
\end{aligned}$$

and

$$\begin{aligned}
&\mathrm{P}\left[\bar{y}_i^\ell \geq \langle e_i, A\bar{x}^\ell \rangle + 2\epsilon \bar{y}_i^0\right] \\
&\quad = \mathrm{P}\left[\bar{y}_i^0 - \bar{y}_i^\ell \leq (1-\epsilon)\langle e_i, A(\bar{x}^0 - \bar{x}^\ell)\rangle - \epsilon \bar{y}_i^0\right] \\
&\quad = \mathrm{P}\left[\sum_{k=1}^{\ell} Z_k \leq \frac{(1-\epsilon)\ln(\gamma)}{\epsilon^2 \bar{y}_i^0} \sum_{k=1}^{\ell} \alpha_k A_{ij_k} - \frac{\ln(\gamma)}{\epsilon}\right] \\
&\quad \leq (1-\epsilon)^{\ln(\gamma)/\epsilon} = \mathrm{poly}(\delta).
\end{aligned}$$

If $\bar{y}_i^\ell < \bar{y}_i^0/2$, then `apx-matrix` will recompute $\bar{y}_i^\ell$ and ensure exact accuracy. If $\bar{y}_i^\ell \geq \bar{y}_i^0/2$, then we have

$$\begin{gathered}
(1-4\epsilon)\bar{y}_i^\ell \geq \langle e_i, A\bar{x}^\ell \rangle \text{ only if } \bar{y}_i^\ell \geq \langle e_i, A\bar{x}^\ell \rangle + \epsilon \bar{y}_i^0, \text{ and} \\
(1+4\epsilon)\bar{y}_i^\ell \leq \langle e_i, A\bar{x}^\ell \rangle \text{ only if } \bar{y}_i^\ell \leq \langle e_i, A\bar{x}^\ell \rangle - \epsilon \bar{y}_i^0,
\end{gathered}$$

as desired. As in the increasing case, via the union bound, the above holds for all L increments of all L phases and over all coordinates $i \in [m]$ with probability $L^2 m \,\mathrm{poly}(\delta)$. (4.5) then follows. ■

Now we analyze the running time of the data structure for a sequence of L increments. Note that the parameter β does not play a role in the correctness but it does play a role in the running time. This is natural for the following reason. Imagine a setting in which all increments are double the current sum. Then the algorithm will be forced to deterministically compute the exact sum in each step to be accurate. The lemma below bounds the number of times that `reset` is called for each row $i \in [m]$.

LEMMA 4.2. *Assuming Setting 4.1, with probability* $1 - \mathrm{poly}(m, L, \delta)$, `reset(`$i$`)` *is called* $O(\log \beta)$ *times for each row* $i \in [m]$.

Proof. With probability $1 - \mathrm{poly}(m, L)\,\mathrm{poly}(\delta)$, we have $y \in (1 \pm \epsilon)Ax$ at all times. Assume this is the case, and fix $i \in [m]$.

In the increasing case, where $\alpha_\ell \geq 0$ for all ℓ, we have $\langle e_i, A\mathbb{1}\rangle \leq \langle e_i, Ax\rangle \leq \beta\langle e_i, A\mathbb{1}\rangle$ for any constraint i. `reset(`i`)` is invoked only when $y_i > 2\tilde{y}_i$, where $\tilde{y}_i$ was the exact value of $\langle e_i, Ax\rangle$ earlier in the process, and then both y_i and $\tilde{y}_i$ are reset to the current, exact value of $\langle e_i, Ax\rangle$. Consider a call to `inc(`α`,`j`)` that triggers a `reset(`i`)`. Let y_i' and x' denote the values of y_i and x before the call, and let y_i'' and x'' denote the values of y_i and x after line [9] (but before invoking `reset(`i`)`). If $y_i'' - y_i' \leq \frac{2\epsilon^2}{\delta} y_i'$, then

$$\begin{aligned}
\langle e_i, Ax''\rangle &\geq \langle e_i, Ax'\rangle \geq (1-\epsilon)y_i' \\
&\geq (1 - O(\epsilon))y_i'' \geq 2(1 - O(\epsilon))\tilde{y}_i,
\end{aligned}$$

hence $\langle e_i, Ax\rangle$ has increased by a (constant) multiplicative factor of $(1 - O(\epsilon))2$. If $y_i'' - y_i' \geq \frac{2\epsilon^2}{\delta} y_i'$, then $\alpha > \theta_{ij}$ and $y_i'' - y_i' = \alpha A_{ij}$, hence

$$\begin{aligned}
\langle e_i, Ax''\rangle &= \langle e_i, Ax'\rangle + y_i'' - y_i' \\
&\geq (1 - O(\epsilon))y_i' + 2\tilde{y}_i - y_i' \\
&\geq 2(1 - O(\epsilon))\tilde{y}_i,
\end{aligned}$$

so again $\langle e_i, Ax\rangle$ has increased by a multiplicative factor of $2(1 - O(\epsilon))$. It follows that `reset(`i`)` is called at most $O\left(\log_{(1-O(\epsilon))2} \beta\right) = O(\log \beta)$ times.

In the decreasing case, where $\alpha_\ell \geq 0$ for all ℓ, we have $\frac{1}{\beta}\langle e_i, Ax\rangle \leq \langle e_i, Ax\rangle \leq \langle e_i, Ax\rangle$ for any constraint i. We claim that each time we call `reset(`i`)`, $\langle e_i, Ax\rangle$ has decreased by a $(1 \pm O(\epsilon))2$ multiplicative factor. Indeed, consider a call to `inc(`α`,`e_j`)` that triggers a `reset(`i`)`. Let y_i' and x' denote the values of y_i and x before the call, and let y_i'' and x'' denote the values of y_i and x after line [9] (but before invoking `reset(`i`)`). If $y_i' - y_i'' \leq \frac{2\epsilon^2}{\delta} y_i'$, then

$$\langle e_i, Ax'\rangle = (1+\epsilon)y_i' \leq (1 + O(\epsilon))y_i'' \leq \frac{(1 + O(\epsilon))\tilde{y}_i}{2},$$

so $\langle e_i, Ax''\rangle$ has dropped by a (constant) multiplicative factor of $(1 - O(\epsilon))2$. If $y_i' - y_i'' \geq \frac{2\epsilon^2}{\delta} y_i'$, then $\alpha > \theta_{ij}$ and $y_i' - y_i'' = \alpha A_{ij}$, so

$$\begin{aligned}
\langle e_i, Ax''\rangle &= \langle e_i, Ax'\rangle + y_i'' - y_i' \\
&\leq (1 + O(\epsilon))y_i' + \frac{\tilde{y}_i}{2} - y_i' \\
&\leq \frac{(1 + O(\epsilon))\tilde{y}_i}{2},
\end{aligned}$$

so $\langle e_i, Ax''\rangle$ is a constant multiple of $(1 - O(\epsilon))2$ smaller than $\tilde{y}_i$. Since $\langle e_i, Ax\rangle$ lies in a range contained in a multiplicative factor of β, `reset(`i`)` is invoked at most $O\left(\log_{(1-O(\epsilon))2}(\beta)\right) = O(\log(\beta))$ times. ■

The final step in the analysis is to specify how the loop in line [8] is implemented.

LEMMA 4.3. *Fix* $j \in [n]$. *One can organize the thresholds* $\{\theta_{ij} : A_{ij} \neq 0\}$ *such that for a call* `inc(`α`,`j`)`, *all coordinates* i *satisfying line* [8] *can be listed in* $O(\log m)$ *time plus* $O(1)$ *per satisfying coordinate. The total time to initialize and maintain the data structure is* $O(N \log(\beta) \log(m))$.

Proof. Setting 4.1 implies that each coordinate of Ax that is still subject to change lies within a β-multiplicative factor of its initial value in $A\mathbb{1}$. For each column j, we build a balanced binary tree over the nonzeroes $A_{ij} \neq 0$ keyed by θ_{ij}. All the trees can be built in $O(N \log m)$ time total. For a call `inc(`α`,`j`)`, the first coordinate i satisfying line `[8]` can be found in $\log(m)$ time in the tree; each subsequent coordinate takes $O(1)$ time per coordinate. Whenever a threshold θ_{ij} changes, the corresponding tree can be updating in $O(\log m)$ time. Each θ_{ij} is updated at most $\log \beta$ times. The running time follows. ■

To finish the analysis we note that each y_i is updated $O(\frac{\log(1/\delta)}{\epsilon^2})$ time between two reset operations. Thus the total work for satisfied coordinates is $O\Big(m \log \beta \frac{\log(1/\delta)}{\epsilon^2}\Big)$. Combining the above, we have the following.

Theorem 4.1. *Assuming Setting 4.1, for $\delta = 1/\operatorname{poly}(m, L)$,* `apx-matrix` *maintains $y \in (1 \pm O(\epsilon))Ax$ in $O\Big(L(\log(m) + \log\log\beta) + N(\log(m) + \log\log\beta)\log\beta + m\frac{\log(\beta)(\log m + \log L)}{\epsilon^2}\Big)$ total time with probability of failure $1/\operatorname{poly}(m, L)$.*

4.1 Finalizing `random-mwu`

In this section, we gather the different components developed in Section 3 and Section 4 to prove Theorem 1.1. We first obtain crude bounds on the variation of nonzeroes within each column of A or B.

Lemma 4.4. *Without loss of generality, for each $j \in [n]$,*

$$\frac{\max_i A_{ij}}{\min_i\{A_{ij} : A_{ij} \neq 0\}} = O(\operatorname{poly}(n, 1/\epsilon)),$$
$$\frac{\max_i\{B_{ij}\}}{\min_i\{B_{ij} : B_{ij} \neq 0\}} = O(\operatorname{poly}(n, 1/\epsilon)).$$

Proof. For each j, let $\alpha_j = \max_i A_{ij}$. Then $x \leq 1/\alpha$ for any feasible solution x to (NMPC), and $x \leq (1 + O(\epsilon))/\alpha$ for any $(1 \pm O(\epsilon))$-feasible solution.

Let $A' \in \mathbb{R}_{\geq 0}^{m_p \times n}$ be defined by

$$A'_{ij} = \begin{cases} A_{ij} & \text{if } A_{ij} \geq \frac{\alpha_j}{\operatorname{poly}(n,1/\epsilon)}, \\ 0 & \text{otherwise.} \end{cases}$$

Any x satisfying $A'x \leq (1 + O(\epsilon))\mathbb{1}$ must satisfy $x \leq (1 + O(\epsilon))\alpha$. Moreover, since $0 \leq A_{ij} - A'_{ij} \leq \alpha_j/\operatorname{poly}(n, 1/\epsilon)$, for any x with $A'x \leq (1 + O(\epsilon))$, we have

$$\begin{aligned} Ax = A'x + (A - A')x &\leq A'x + \frac{1}{\operatorname{poly}(n, 1/\epsilon)}\mathbb{1} \\ &\leq (1 + O(\epsilon))\mathbb{1}. \end{aligned}$$

Replacing A with A', it suffices to assume that $\alpha_j A_{ij} \geq \alpha_j/\operatorname{poly}(n, 1/\epsilon)$ for every nonzero A_{ij}.

As for B, we first observe that $B_{ij} \geq \operatorname{poly}(n, 1/\epsilon)\alpha$, then adding $(1/\operatorname{poly}(n, 1/\epsilon)\alpha)e_j$ meets the covering constraint for row i, while $Ae_j/\alpha \operatorname{poly}(n, 1/\epsilon) \leq \epsilon/n$ has negligible effect on the packing constraints. Removing any row i with such a large B_{ij} and rewriting the system as though we have taken $(1/\operatorname{poly}(n, 1/\epsilon)\alpha)e_j$ can be done in linear time, and allows us to assume that $B_{ij} \leq \operatorname{poly}(n, 1/\epsilon)\alpha_j$ for all i, j. Finally, define $B' \leq B$ by

$$B'_{ij} = \begin{cases} B_{ij} & \text{if } B'_{ij} \geq \alpha/\operatorname{poly}(n, 1/\epsilon), \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $Bx \geq (1 - O(\epsilon))$ only if $B'x \geq 1 - O(\epsilon)$. On the flip side, if $Ax \leq 1 + O(\epsilon)$ but $\langle e_i, B'x\rangle \leq 1 + O(\epsilon)$ for some i, then $x \leq (1 + O(\epsilon))/\alpha$,

$$\begin{aligned} \langle e_i, Bx\rangle &\leq \langle e_i, B'x\rangle + \alpha\langle e_i, (B - B')\mathbb{1}\rangle \\ &\leq 1 - O(\epsilon) + \operatorname{poly}(\epsilon/n) \leq (1 - O(\epsilon)). \end{aligned}$$

Thus we may work with B' instead of B without loss of generality. ■

We now restate and prove Theorem 1.1.

Theorem 1.1 2. *Given a normalized mixed packing and covering problem (NMPC),* `random-mwu` *returns a ϵ-relative approximation in*

$$O\left(\begin{array}{c} \frac{N\log^2 m}{\epsilon} + \frac{m\log^2 m}{\epsilon^2} \\ + \frac{n\log(m)(\log(m)+\log(n))}{\epsilon^3} \end{array}\right) = \tilde{O}\left(\frac{N}{\epsilon} + \frac{m}{\epsilon^2} + \frac{n}{\epsilon^3}\right)$$

time with probability $1 - 1/\operatorname{poly}(m)$.

Proof. By Theorem 2.1, it remains to implement lines `[2]` and `[3]` within the running time stated.

Recall that if the original problem (NMPC) is feasible, then any feasible solution to (NMPC) satisfies `(*)` and `(**)` of line `[2]` even without the $(1 \pm O(\epsilon))$ factors. Moreover, if (NMPC) is feasible, then `[2]` is solved by finding an approximately best bang-for-buck coordinate j such that

$$\frac{\langle w, Be_j\rangle_{\mathcal{Q}}}{\langle v, Ae_j\rangle} \geq (1 - O(\epsilon)) \max_{\hat{j} \in [n]} \frac{\langle w, Be_{\hat{j}}\rangle_{\mathcal{Q}}}{\langle v, Ae_{\hat{j}}\rangle},$$

and taking

$$y = \frac{\langle w, B\mathbb{1}\rangle_{\mathcal{Q}}}{\langle w, Be_j\rangle_{\mathcal{Q}}} e_j.$$

We employ the `apx-matrix` data structure to maintain $(1 \pm O(\epsilon))$-relative coordinate-wise approximations of $A^T v$ and $B^T w$. By Lemma 4.4, all the nonzero coefficients within a row of A or within a row of B lie within a $\operatorname{poly}(n, 1/\epsilon)$-multiplicative factor of one another. The vectors v and w both start at $\mathbb{1}$, and any nonzero coordinate is in the range $m^{-O(1/\epsilon)}$ or $m^{O(1/\epsilon)}$. We also know that there are at most $O\big(m \log(m)/\epsilon^2\big)$ iterations. Thus, maintaining $A^T v$ and $B^T w$ approximately corresponds to Setting 4.1 for $\beta = m^{O(1/\epsilon)} \operatorname{poly}(n, 1/\epsilon)$ and $L = O\big((\min\{m_p, n\} + m_c) \log(m)/\epsilon^2\big)$. Also recall that since we are working with A^T and B^T the roles of m, n have to be interchanged in the bounds guaranteed by Theorem 4.1. By Theorem 4.1, the data structure maintains an $(1 \pm \epsilon)$-approximation of every coordinate of $A^T v$ and $B^T w$ at all times in

$$O\left(\begin{array}{c} \frac{(\min\{m_p,n\}+m_c)\log^2 m}{\epsilon^2} \\ +\frac{N\log^2 m}{\epsilon} \\ +\frac{n\log(m)(\log(m)+\log(n))}{\epsilon^3} \end{array}\right) = \tilde{O}\left(\frac{N}{\epsilon} + \frac{m}{\epsilon^2} + \frac{n}{\epsilon^3}\right)$$

total time, with probability of failure $1/\operatorname{poly}(m)$, as desired. We observe that the randomness in the internals of the data structure is independent of the randomness in weight update step.

Beyond maintaining each coordinate of $A^T v$ and $B^T w$, implementing the oracle in `[2]` within the stated bounds is easy by the following well-known "lazy-greedy" technique. To satisfy `[2]`, it suffices to find a coordinate j approximately maximizing the ratio

$$\frac{\langle w, Be_j\rangle_{\mathcal{Q}}}{\langle v, Ae_j\rangle} \text{ and returning } y = \frac{\langle w, \mathbb{1}\rangle_{\mathcal{Q}}}{\langle w, Be_j\rangle} e_j.$$

The coordinate-wise approximations provided by `apx-matrix` allow us to compute a $(1 \pm O(\epsilon))$-approximation of a coordinates ratio in constant time. To repeatedly find an approximately best coordinate, let $\lambda = \sup_j \langle \mathbb{1}, Be_j\rangle / \langle \mathbb{1}, Ae_j\rangle$ be the maximum ratio initially. When line `[2]` is invoked, we process the coordinates in round robin order until we either find a coordinate j with ratio $\geq (1 - O(\epsilon))\lambda$ or, should no satisfying coordinate j, setting $\lambda \leftarrow (1 - O(\epsilon))\lambda$ and repeating the search. (A round-robin sweep always begins where the last round-robin search ended in the previous oracle call.) Every time we compute a ratio can be charged to the oracle call (if the coordinate is returned) or the current value of λ. Moreover, the range of values taken by λ lies within a β^2-multiplicative factor of its initial value, where $\beta = m^{O(1/\epsilon)} \operatorname{poly}(n/\epsilon)$. Therefore, λ is decreased at most

$$O\Big(\log_{1/(1-O(\epsilon))} \beta^2\Big) = O\left(\frac{\log(m)}{\epsilon^2} + \frac{\log(n)}{\epsilon}\right)$$

times over the course of the algorithm. It follows that the total time processing coordinates is $O\big(m\big(\log(m)/\epsilon^2 + \log(n)/\epsilon\big)\big)$ plus $O(1)$ for each iteration, as desired. ■

5 An overview of applications

In this section we give a brief overview of some abstract and concrete applications where mixed packing and covering problems arise. Theorem 1.1 applies to any explicitly given problem and gives a speed up if the matrices A, B are sufficiently dense. Here we also point out implicit instances where the randomized version is useful.

5.1 Linear system solving in the positive orthant

Consider the problem of solving $Ax = \mathbb{1}, x \geq \mathbb{0}$ where $A \in \mathbb{R}_{\geq 0}^{m \times n}$ is a non-negative matrix. It is easy to see that it is a special case of (NMPC). Young [35] points out applications of this problem to x-ray tomography. We also observe that this formulation captures the LP relaxation of the perfect b-matching problem in graphs and hypergraphs. See [12] for an application of hypergraph matching in the field of medicine via the solution of the LP relaxation and randomized rounding.

5.2 Separating over non-negative polytopes and an application to spanning trees

Another basic application of solving $Ax = \mathbb{1}, x \geq 0$ is the following problem. Suppose we are given a polytope P and a point p, both in the non-negative orthant $\mathbb{R}_{\geq 0}^d$. We would like to decide if $p \in P$ and if it is, express p as a convex combination of the vertices of P. Depending on the application, P can be given explicitly as a the convex hull of a given set of of n points $V = \{v_1, v_2, \ldots, v_n\}$ in $\mathbb{R}_{\geq 0}^d$, or implicitly. We consider the explicit case first. p is in the convex hull of V iff the following system is feasible: $Ax = p, \sum_i x_i = 1, x \geq 0$ where the columns of A are $v_1, \ldots, v_n$. Thus, it is a special case of (NMPC).

In the implicit setting there are a number of applications in combinatorial optimization. We give a concrete example. Consider the following problem. Given a graph $G = (V, E)$ let $\mathrm{ST}(G)$ be the spanning tree polytope of G; in other words it is the convex hull of the characteristic vectors of the spanning trees of G (note that the dimension of the polytope is m where m is the number of edges of G). Given $z \in \mathbb{R}_{\geq 0}^m$ we would like to know whether $z \in \mathrm{ST}(G)$ and if so decompose z into a convex combination of spanning trees. This a fundamental problem and there are a number of applications including recent rounding algorithms for TSP and ATSP (see [32]).

In recent work [6] we developed a near-linear time approximation scheme to pack the maximum number of spanning trees in a given graph by using fast MWU based algorithm. As a corollary, [6] obtains a separation oracle for the dominant of $\mathrm{ST}(G)$. In particular, if $z \in \mathrm{ST}(G)$, the algorithm outputs a convex combination of spanning trees such that the load on any edge e is at most $(1+\epsilon)z_e$. However, it does not guarantee that the load on e is at least $(1-\epsilon)z_e$. Hence many edges with strictly positive z_e value can be unused in the convex decomposition. This is acceptable in several applications but not in others. Via the algorithm in this paper for (NMPC), and some of the ideas in [6], we can obtain a near-linear time randomized algorithm for separating over $\mathrm{ST}(G)$, which in particular implies that we can approximately decompose a given point $z \in \mathrm{ST}(G)$ into a convex combination of spanning trees in near-linear time. The decomposition guarantees a convex combination such that for each edge e the load on e from the convex combination is between $(1-\epsilon)z_e$ and $(1+\epsilon)z_e$.

5.3 Min-Max Linear/Integer Programs and Resource Allocation

A number of applications in resource allocation, routing, and scheduling can be cast as min-max integer programs and their LP relaxations turn out to be mixed packing and covering LPs. Randomized rounding via sophisticated techniques including the Lovasz-Local-Lemma and its constructive versions have led to a number of important results in approximation — see [30, 16] for the general framework, rounding algorithms, and applications. Here we point out a simple application, namely, routing paths to minimize congestion which finds applications in several areas including VLSI and optical networks. The input consists of a (directed) graph $G = (V, E)$ and k source-sink pairs (s_i, t_i), $i \in [k]$. In the explicit setting we are given for each pair (s_i, t_i) a collection of ℓ_i paths $\mathcal{P}_i$ that connect s_i to t_i. The goal is to choose for each pair i, exactly one path from $\mathcal{P}_i$, to minimize the edge congestion from the chosen paths. We can model this as follows. For each pair i we have ℓ_i indicator variables x_{ij}, $j \in [\ell_i]$. The goal is to solve the following integer program

$$\min \lambda$$

$$Ax \leq \lambda \mathbb{1}$$

$$\sum_{j \in [\ell_i]} x_{ij} \geq 1 \quad i \in [k]$$

$$x_{ij} \in \{0,1\} \quad i \in [k], j \in [\ell_i]$$

where A is the incidence matrix between the path collection $\cup_i \mathcal{P}_i$ and the edges of G. The LP relaxation is a mixed packing and covering LP (for a given guess of λ) and is extensively used in approximation algorithm design. Using the LP one can obtain an $O(\log d/\log\log d)$-approximation for the minimum congestion problem where d is the maximum path length in $\cup_i \mathcal{P}_i$ [30, 16]. In the version we described above, the paths are explicitly given for each pair. In the implicit setting $\mathcal{P}_i$ is the set of all paths from s_i to t_i (or say all paths with at most some prescribed length), and the resulting LP is the maximum concurrent multicommodity flow problem which has been extensively studied.

Another way to interpret the allocation constraints in min-max integer program is the following. We think of them as inducing a partition matroid base constraint on the variables. One can further extend the framework by allowing an arbitrary matroid base constraint on the variables. These constraints also result in a mixed packing LP. One can then use a variety of dependent randomized rounding techniques.

5.4 Constrained Set Cover

Set cover is a fundamental and abstract problem that arises in numerous applications. In the explicit version we are given a collection of sets $S_1, S_2, \ldots, S_m$ over a universe $\mathcal{U}$ of n elements and the goal is to find a minimum cardinality or minimum cost sub-collection of the given sets whose union is $\mathcal{U}$. One can express the LP relaxation of this problem as a pure covering problem and there is extensive work in analyzing the integrality gap in various settings. We note that when expressed as a covering LP, the number of nonzeroes N in the matrix corresponds to the number of set-element edges in the bipartite graph representation of the set system.

There are several natural and simple generalizations of set cover that result in a mixed packing and covering LP. One variant is the multicover problem with bounds on the sets. Here each element $e \in \mathcal{U}$ has an integer requirement r_e for the number of distinct sets that e should be covered by. Moreover each set S_i has an upper bound u_i on the number of copies it can be used for; the simplest setting being that $u_i = 1$ for all i. The upper bounds naturally lead to a mixed packing and covering LP. A generalization of this problem to covering integer programs with upper bounds was considered in Kolliopoulos and Young [20]. We also mention that more sophisticated upper bound constraints on the sets, such as partition matroid constraints, appear in various applications [31].

5.5 Implicit problems with structured incidences

In [6] we consider the following simple geometric problem. We are given n closed intervals $I_1, \ldots, I_n$ on the real line specified by their endpoints $I_i = [a_i, b_i]$. Each interval has a nonnegative value v_i and a non-negative size d_i. We are also given m points $p_1, \ldots, p_m \in \mathbb{R}$ on the real line, and each point p_j has a capacity $g_j > 0$. The goal is to choose a subset of the intervals of maximum value such that the total size of chosen intervals at any point is at most the capacity of the point. This is equivalent to the well-studied unsplittable flow problem (UFP) on paths. These problems and their variants have been well-studied in a variety of contexts and have numerous applications. Note that the problem description size is $O(m+n)$. The underlying LP relaxation for the preceding problem, if written explicitly, can have $N = \Omega(mn)$ nonzeroes corresponding to the incidences between the intervals and points. [6] showed that MWU techniques for packing problems with some data structures can be adapted to obtain a $(1+\epsilon)$ approximation to the LP relaxation in $\tilde{O}\big((m+n)/\epsilon^2\big)$ time, avoiding the need to explicitly work with the incidence matrix. The techniques in this paper can extend those results for mixed packing and covering setting; for instance, the points can have lower bounds on how many intervals need to cover them. The main data structure that enabled the speedup in the case of intervals is the segment tree which interfaces cleanly with the needs of the deterministic MWU framework in [6].

As we remarked, the randomized version of MWU in this paper was motivated by applications that did not admit a similar improvement. Consider the problem where we have disks and points in the plane and we wish to solve problems such as the maximum weight independent set or the minimum cost set cover. Here too the problem description size is considerably more compact than the explicit matrix that encodes the incidences between the given disks and points. Unlike intervals and rectangular boxes, disks and other more complex objects do not admit the same type of dynamic range search data structures that can be used in the framework of [6]. In contrast, `random-mwu` and `apx-mtarix` considerably simplify the requirements on the data structure. This is enabled by the particularly simple threshold based sampling used in both of them. For instance, one can use ideas from emptiness and approximate depth estimation oracles [3] instead of using dynamic range reporting data structures that have strong lower bounds even in 2 and 3 dimensions. We defer formal details to a future paper.

5.6 Facility Location and k-median

Facility location and k-median are extensively studied optimization problems. The natural LP for facility location and k-median is not a positive LP. However, one can reformulate facility location as an implicit covering LP. Young [36] shows that one can obtain an $\tilde{O}(nm/\epsilon^2)$ run-time to solve a facility location LP with n facilities and m clients. The k-median and its minimum cost versions can be cast as implicit mixed packing and covering LPs and [34] gives a $(1+\epsilon)$-approximation in $\tilde{O}\big(kmn/\epsilon^2\big)$ time. We note that the assumption here is that all distances between facilities and clients are explicitly given and they may not form a metric. Here the natural problem size is mn. In future work we plan to address whether we can obtain a running time of the form $\tilde{O}(mn/\epsilon + (m+n)/\epsilon^3)$ for these problems via the techniques in this paper.

Acknowledgments: CC thanks Jayram Thathachar and Jan Vondrák for discussions during their work on [8]. The deterministic version of `rand-mwu` is in the timed framework of that paper. We thank Neal Young for comments and clarifications on his extensive work on Lagrangian relaxation algorithms, and encouragement. We thank Sariel Har-Peled for discussions and clarifications on geometric data structures.

References

[1] P. K. Agarwal and J. Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proc. 30th Annu. Sympos. Comput. Geom.* (SoCG), page 271, 2014.

[2] Z. Allen-Zhu and L. Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proc. 47th Annu. ACM Sympos. Theory Comput.* (STOC), pages 229–236, 2015.

[3] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.

[4] 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.

[5] D. Bienstock and G. Iyengar. Approximating fractional packings and coverings in O(1/epsilon) iterations. *SIAM J. Comput.*, 35(4):825–854, 2006.

[6] C. Chekuri and K. Quanrud. Near-linear time approximation schemes for some implicit fractional packing problems. In *Proc. 28th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 801–820, 2017.

[7] C. Chekuri and K. Quanrud. Approximating the Held-Karp bound for Metric TSP in nearly–linear time. In *Proc. of IEEE FOCS*, 2017.

[8] C. Chekuri, T. Jayram, and J. Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *Proc. 6th Conf. Innov. Theoret. Comp. Sci.* (ITCS), pages 201–210, 2015.

[9] F. Diedrich and K. Jansen. Faster and simpler approximation algorithms for mixed packing and covering problems. *Theoretical computer science*, 377(1-3):181–204, 2007.

[10] P. Flajolet. Approximate counting: A detailed analysis. *BIT*, 25(1):113–134, 1985.

[11] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

[12] H. Fohlin, L. Kliemann, and A. Srivastav. *Randomized algorithms for mixed matching and covering in hypergraphs in 3D seed reconstruction in brachytherapy*, chapter 4, pages 71–102. Springer New York, New York, NY, 2008.

[13] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37 (2):630–652, 2007. Preliminary version in Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci. *(FOCS)*(1998).

[14] M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM J. Optim.*, 4(1):86–107, 1994.

[15] M. D. Grigoriadis and L. G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Oper. Res. Lett.*, 18:53–58, 1995.

[16] B. Haeupler, B. Saha, and A. Srinivasan. New constructive aspects of the lovÁsz local lemma. *J. ACM*, 58(6):28:1–28:28, Dec. 2011. ISSN 0004-5411. doi: 10.1145/2049697.2049702. URL `http://doi.acm.org/10.1145/2049697.2049702`.

[17] K. Jansen and H. Zhang. Approximation algorithms for general packing problems and their application to the multicast congestion problem. *Mathematical Programming*, 114(1):183–206, 2008.

[18] R. Khandekar. *Lagrangian relaxation based algorithms for convex programming problems.* PhD thesis, Indian Institute of Technology Delhi, 2004.

[19] P. N. Klein and N. E. Young. On the number of iterations for Dantzig-Wolfe optimization and packing-covering approximation algorithms. *SIAM J. Comput.*, 44(4):1154–1172, 2015.

[20] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *J. Comput. Syst. Sci.*, 71(4):495–505, 2005. doi: 10.1016/j.jcss.2005.05.002. URL `https://doi.org/10.1016/j.jcss.2005.05.002`.

[21] C. Koufogiannakis and N. E. Young. Beating simplex for fractional packing and covering linear programs. In *Proc. 48th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 494–506, 2007.

[22] 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. Preliminary version in Proc. 48th Annu. IEEE Sympos. Found. Comput. Sci. *(FOCS)*, 2007.

[23] P. K. Lehre and C. Witt. Concentrated hitting times of randomized search heuristics with variable drift. In *Proceedings of 25th International Symposium on Algorithms and Computataion (ISAAC)*, pages 686–697, 2014.

[24] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of ACM Symposium on Theory of Computing*, STOC '93, pages 448–457, 1993.

[25] M. W. Mahoney, S. Rao, D. Wang, and P. Zhang. Approximating the solution to mixed packing and covering lps in parallel o (epsilon^{-3}) time. In *Proceedings of ICALP, LIPIcs-Leibniz International Proceedings in Informatics*, volume 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[26] R. Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978.

[27] A. Mądry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proc. 42nd Annu. ACM Sympos. Theory Comput.* (STOC), pages 121–130, 2010.

[28] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. of Oper. Res.*, 20 (2):257–301, 1995.

[29] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. Assoc. Comput. Mach.*, 37(2):318–334, 1990.

[30] A. Srinivasan. An extension of the lovász local lemma, and its applications to integer programming. *SIAM J. Comput.*, 36(3):609–634, 2006. doi: 10.1137/S0097539703434620. URL https://doi.org/10.1137/S0097539703434620.

[31] S. Umetani, M. Arakawa, and M. Yagiura. Relaxation heuristics for the set multicover problem with generalized upper bound constraints. *CoRR*, abs/1705.04970, 2017. URL http://arxiv.org/abs/1705.04970.

[32] J. Vygen. New approximation algorithms for the TSP. *OPTIMA*, 90:1–12, 2012.

[33] D. Wang, S. Rao, and M. W. Mahoney. Unified acceleration method for packing and covering problems via diameter reduction. In *Proc. 43rd Internat. Colloq. Automata Lang. Prog.* (ICALP), pages 50:1–50:13, 2016.

[34] N. E. Young. K-medians, facility location, and the chernoff-wald bound. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 86–95. Society for Industrial and Applied Mathematics, 2000.

[35] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 538–546. IEEE, 2001.

[36] N. E. Young. Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs. *CoRR*, abs/1407.3015, 2014. URL http://arxiv.org/abs/1407.3015.

A Jensen's inequality

Lemma A.1. (Jensen's inequality) *Let $X \in \mathbb{R}$ be a random variable and $f : \mathbb{R} \to \mathbb{R}$ a function.*

(a) If f is convex, then $f(\mathrm{E}[X]) \leq \mathrm{E}[f(X)]$.

(b) If f is concave, then $f(\mathrm{E}[X]) \geq \mathrm{E}[f(X)]$.