Online Primal-Dual Algorithms for Covering and Packing

Niv Buchbinder
Microsoft Research, New England Lab
One Memorial Drive, Cambridge, MA 02142
email: nivbuchb@microsoft.com

Joseph (Seffi) Naor[1]
Computer Science Dept.
Technion, Haifa 32000, Israel
email: naor@cs.technion.ac.il

We study a wide range of online covering and packing optimization problems. In an online covering problem, a linear cost function is known in advance, but the linear constraints that define the feasible solution space are given one by one, in rounds. In an online packing problem, the profit function as well as the packing constraints are not known in advance. In each round additional information (i.e., a new variable) about the profit function and the constraints is revealed. An online algorithm needs to maintain a feasible solution in each round; in addition, the solutions generated over the different rounds need to satisfy a monotonicity property. We provide general deterministic primal-dual algorithms for online *fractional* covering and packing problems. We also provide deterministic algorithms for several *integral* online covering and packing problems. Our algorithms are designed via a novel online primal-dual technique and are evaluated via competitive analysis.

Key words: Online Algorithms; Linear Programming; Duality; Competitive Analysis; Covering and Packing; Derandomization

MSC2000 Subject Classification: Primary: 68W05 ; Secondary: 68W40

OR/MS subject classification: Primary: Analysis of algorithms ; Secondary: Linear Programming

---

1. Introduction The primal-dual method is a powerful algorithmic technique that has proved to be extremely useful for a wide variety of problems in the area of approximation algorithms. The method has its origins in the realm of exact algorithms, e.g., for matching and network flow. In the area of approximation algorithms the primal-dual method has emerged as an important unifying design methodology starting from the seminal work of Goemans and Williamson [18].

The focus of this paper is on extending the primal-dual method to the setting of online algorithms and competitive analysis. We study a wide range of online covering and packing optimization problems and provide a unified approach, based on a clean primal-dual method, for the design of online algorithms for these problems. In particular, our analysis uses weak duality rather than a tailor made (i.e., problem specific) potential function. Thus, we believe our results further our understanding of the applicability of the primal-dual method to online algorithms.

In an online problem the input is revealed in parts and the main issue is obtaining good performance in the face of uncertainty. A standard measure for evaluating the performance of an online algorithm is the *competitive ratio*, which compares the performance of an online algorithm to that of an offline algorithm which is given the whole input sequence beforehand. For a given input sequence, consider the ratio between the cost of the solution of an online algorithm and the minimum (optimum) cost solution. The maximum ratio, taken over all input sequences, is defined to be the competitive ratio of the online algorithm.

In an "offline" (fractional) covering problem the objective is to minimize the total cost given by a linear cost function $\sum_{i=1}^{n} c(i)x(i)$. The feasible solution space is defined by a set of m linear constraints of the form $\sum_{i=1}^{n} a(i,j)x(i) \geq b(j)$, where the entries $a(i,j)$ and $b(j)$ are non-negative. The general *online fractional covering problem* is an online version of the covering problem, described as a game between an algorithm and an adversary. In this setting the cost function is known in advance, but the linear constraints that define the feasible solution space are given to the algorithm one by one in an online fashion. In order to maintain a feasible solution to the current set of given constraints, the algorithm is allowed to increase the variables $x(i)$. It may not, however, decrease any previously increased variable. We also extend our study to cases where the value of each variable has an upper bound $u(i)$, referred to as a *box constraint*. The box constraints are known to the online algorithm in advance. This captures online settings in which the amount of resources is limited. The performance of the online algorithm is

compared against to the minimum cost cover of the given problem instance.

We show that the online covering problem is closely related to a dual online packing problem. In an offline packing problem we are given n packing constraints of the type $\sum_{j=1}^{m} a(j,i)y(j) \leq c(i)$. The goal is to find a feasible solution that maximizes a profit function $\sum_{i=1}^{m} b(j)y(j)$. The general *online fractional packing problem* is an online version of this problem. In the online setting the values $c(i)$ $(1 \leq i \leq n)$ are known in advance, but the profit function and the exact packing constraints are not known in advance. In the jth round a new variable $y(j)$ is introduced to the algorithm, along with its set of coefficients $a(i,j)$ $(1 \leq i \leq n)$. Note that other variables that were not yet introduced may also appear in the packing constraints. This means that each packing constraint is gradually revealed to the algorithm. The algorithm can only increase the value of a variable $y(j)$ in the round in which it is given and cannot change the values of any previously given variables. The performance of the algorithm is measured with respect to the maximum possible profit of the packing problem instance. Although this online setting seems a bit unnatural, it generalizes many well known online problems that we list in the sequel.

1.1 Our Results We study both the general online fractional covering problem (with and without box constraints) and the general online fractional packing problem. We show that the two online problems constitute a primal-dual pair and provide general deterministic primal-dual schemes that compute near-optimal fractional solutions for each of the problems.

For the general packing problem we design a scheme that gets as input the desired competitive ratio $B > 0$. It produces a solution violating the packing constraints by a factor of at most

$$O\left(\left(\log n + \log \max_i \frac{a_i(\max)}{a_i(\min)}\right)/B\right),$$

where n denotes the number of packing constraints in the instance, and:

$$(1)\ a_i(\max) = \max_{k=1}^{m}\{a(i,k)\} \qquad (2)\ a_i(\min) = \min_{k=1}^{m}\{a(i,k) | a(i,k) \neq 0\}.$$

In particular, when all coefficients $a(i,j) \in \{0,1\}$, we show an $O(\log n)$-competitive algorithm that does not violate any of the constraints. We show that in general these results are tight for any $B > 0$.

The scheme for the covering problem is very similar. In fact, when each coefficient $a(i,j) \in \{0,1\}$, the same scheme is applicable to both problems. Unfortunately, the scheme designed for the packing problem cannot be used for the general covering problem in which the coefficients $a(i,j) \geq 0$ are not limited to $\{0,1\}$. For this problem we need to design a more complicated scheme. For any $B > 0$, our scheme is $O(\log n/B)$-competitive and covers each constraint up to a factor of $1/B$ (i.e. $\sum_{i=1}^{n} a(i,j)x(i) \geq b(j)/B$). A simple modification of the lower bounds in [2] proves that this result is tight as well. We also extend the scheme to handle box constraints, which define an upper bound on the values of the variables.

The online fractional covering problem with $a(i,j) \in \{0,1\}$ was previously studied in the context of online graph optimization problems [1, 2]. The scheme we propose for this case is simpler than the algorithms of [1, 2], since we do not need to guess the value of the optimum solution, and thus avoid the need for phases. More importantly, our scheme is more precise, and thus improves on the competitive ratio of the algorithms od [1, 2]. Specifically, we get a competitive ratio of $O(\log d)$ instead of $O(\log n)$, where d is the maximum number of variables in each given constraint ($d \leq n$). This improvement immediately reflects on the competitive ratio of the problems studied in [1, 2]. The competitive ratio for the online set-cover improves from $O(\log m \log n)$ to $O(\log d \log n)$, thus replacing m, the total number of sets, by d, the maximum number of sets that an element can belong to. In the connectivity problems studied in [2], we improve the $O(\log m)$ factor, where m is the number of edges, to $O(\log C)$, where C is the size of the maximum cut in the graph. Similarly, in the cut problems studied in [2] we improve the $O(\log m)$ factor to $O(\log L)$, where L is the length of the longest simple path in the graph.

Finally, in Section 5 we show the applicability of our schemes to solving online integral versions of covering and packing problems via randomized rounding. To do so, we use the fractional schemes as a "black box". We then focus on converting the randomized algorithm we obtain into a deterministic algorithm. To this end, we suggest a method for rounding the fractional solution deterministically by transforming an (offline) pessimistic estimator into an online potential function. We note that, in general, the existence of an offline derandomization method does not necessarily yield an online deterministic algorithm. In particular, while all randomized rounding methods in [2] can be derandomized offline, it is

an open question whether online deterministic algorithms for these problems exist. Still, the perspective of deterministic rounding motivates us to consider derandomization methods that were previously suggested for various offline problems, and these enable us to obtain better deterministic online algorithms.

We demonstrate our derandomization method on two online problems, a covering problem and a packing problem. We first consider the online unweighted set-cover problem [1] in Section 5.1. For this problem we provide a better deterministic algorithm in terms of the competitive ratio. We draw on ideas from the improved offline randomized rounding and derandomization methods of [29] and come up with an improved potential function for the online problem. This yields a competitive ratio of $O(\log d \log \frac{n}{OPT})$ instead of $O(\log d \log n)$, where OPT is the optimal number of sets needed to cover the requested elements. The improvement in the competitive ratio is significant when the number of sets needed for the cover is large, and in particular when the sets are small.

We also consider the problem of throughput-competitive online routing of virtual circuits [5]. We show that a deterministic algorithm equivalent to the one from [5] can be derived easily by combining our scheme with the method of conditional expectations of the randomized rounding algorithm presented in [28]. This is an interesting way of viewing the algorithm of [5], which also provides a systematic method for deriving it. In particular, we note that our scheme produces in an online fashion a near-optimal *fractional solution* to the problem that does not violate the capacity constraints. Producing such a fractional solution, unlike an integral solution, is independent of the values of the edge capacities.

1.2 Related Work Covering and packing optimization problems, as well as their online versions, have been studied extensively. Examples of such online covering optimization problems are the online Steiner problem that was considered in [19] and the online generalized Steiner problem that was considered in [4, 7]. Recently, Alon et al. [2] suggested a general two-phase approach for a wide class of online network optimization problems. First, a fractional solution to the online problem is generated. The solution is then rounded in an online fashion in the second phase. The two phases, although separated, are run simultaneously. In [2], the first phase is performed by a general $O(\log n)$-competitive method which is applicable to a wide class of online graph optimization problems. The rounding phase, on the other hand, is problem dependent. The approach was shown useful by the development of randomized algorithms for many interesting integral graph optimization problems. Examples include the online group Steiner problem and online non-metric facility location. The same approach was also implicity used in [1], where a deterministic online $O(\log n \log m)$-competitive algorithm for the integral set cover problem was provided. Our methods for generating fractional solutions for covering problems generalize and improve upon the methods of [2].

Garg and Young [16] also considered an online flow control packing problem. Specifically, they considered a similar in spirit online fractional packing problem in which requests arrive online and the objective of the algorithm is to converge to a maximum throughput multicommodity flow. However, in their model an online algorithm is allowed to both increase and decrease flows of existing requests.

There is a long line of work on generating a near-optimal fractional solution for *offline* covering and packing problems, e.g. [26, 23, 30, 15, 13, 22]. Generating such a solution for the *offline* covering problem with upper bounds on the variables was considered in [12, 14]. All these methods take advantage of the offline nature of the problems. We remark that several of these methods use primal-dual analysis, therefore our framework can be viewed as an adaptation of these methods to the context of online computation.

Finally, our online rounding methods are based on their corresponding offline randomized rounding methods. In particular, our integral routing algorithm derives ideas from the rounding methods in [28, 27]. For the design of our online set cover algorithm we derive ideas from the offline rounding in [29].

2. Preliminaries In this section we formally define our problems and discuss their dual nature. In an "offline" (fractional) covering problem the objective is to minimize the total cost given by a linear cost function $\sum_{i=1}^{n} c(i)x(i)$. The feasible solution space is defined by a set of m linear constraints of the form $\sum_{i=1}^{n} a(i,j)x(i) \geq b(j)$, where the entries $a(i,j)$ and $b(j)$ are non-negative. Given an instance of a covering problem we first normalize each constraint to the form: $\sum_{i=1}^{n} a(i,j)x(i) \geq 1$. Any primal covering instance has a corresponding dual packing problem that provides a lower bound on any feasible solution to the instance. A general form of a (normalized) primal covering problem along with its (normalized) dual packing problem is given in Figure 1. Throughout the paper we refer to the covering

Primal (Covering)		Dual (Packing)	
Minimize:	$\sum_{i=1}^{n} c(i)x(i)$	Maximize:	$\sum_{j=1}^{m} y(j)$
Subject to:		Subject to:	
For each $1 \leq j \leq m$:	$\sum_{i=1}^{n} a(i,j)x(i) \geq 1$	For each $1 \leq i \leq n$:	$\sum_{j=1}^{m} a(i,j)y(j) \leq c(i)$
For each $1 \leq i \leq n$:	$x(i) \geq 0$	For each $1 \leq j \leq m$:	$y(j) \geq 0$

Figure 1: Primal (covering) and dual (packing) problems

problem as the "primal problem" and the packing problem as the "dual problem".

The general *online fractional covering problem* is an online version of the covering problem. In this setting the cost function is known in advance, but the linear constraints that define the feasible solution space are given to the algorithm one-by-one, in rounds. In each round a new constraint is given and in order to maintain a feasible solution to the current set of constraints, the online algorithm is allowed to increase the value of the variables. It may not, however, decrease the value of any variable. We also define an online version of the packing problem. In the general *online fractional packing problem* the values $c(i)$ $(1 \leq i \leq n)$ are known in advance. However, the profit function and the exact packing constraints are not known in advance. In the jth round a new variable $y(j)$ is introduced to the algorithm, along with its set of coefficients $a(i,j)$ $(1 \leq i \leq n)$[2]. Note that other variables, that have not yet been introduced, may also appear (in the future) in the packing constraints. This means that each packing constraint is gradually revealed to the algorithm. The algorithm may increase the value of a variable $y(j)$ only in the round where it is given, and may not change the values of any previously given variables.

We observe that these two online settings form a primal-dual pair in the following sense: at any time an algorithm for the general online fractional covering problem maintains a subset of the final linear constraints. This subset defines a *sub-instance* of the final covering instance. The dual packing problem of this sub-instance is a *sub-instance* of the final dual packing problem. In the dual packing sub-instance, only part of the dual variables are known, along with their corresponding coefficients. The two sub-instances form together a *primal-dual pair*. In each round of the general online fractional covering problem a new constraint on the feasible solution space is given. The primal covering sub-instance is updated by adding the new constraint. To update the dual sub-instance we add a new dual variable to the profit function along with its coefficients that are defined by the new primal constraint. Note that the dual update is the same as in the setting of the online fractional packing problem.

The schemes we propose maintain at each step solutions for both the primal and dual sub-instances. When a new constraint is given to the online fractional covering problem, our scheme also considers the new *corresponding* dual variable and its coefficients. When the scheme for the online fractional packing problem receives a new variable along with its coefficients, it also considers the *corresponding* new constraint in the primal sub-instance. In the rest of the paper we use the notion of primal and dual sub-instances and the correspondence between dual variables and primal constraints.

3. The General Online Fractional Packing Problem In this section we describe an online scheme for computing a near-optimal fractional solution for the general online fractional packing problem. The scheme gets the desired competitive ratio $B > 0$ and returns a solution which is within a factor of B of the optimal, and which does not violate the packing constraints by too much (to be made more precise shortly). We prove that the scheme is optimal up to constant factors. Our scheme simultaneously maintains primal (covering) and dual (packing) solutions for the primal and dual sub-instances.

Initially, each variable $x(i)$ is initialized to zero. In each round a new variable $y(j)$ is introduced along with its coefficients $a(i,j)$ $(1 \leq i \leq n)$. In the corresponding primal sub-instance a new constraint is introduced of the form $\sum_{i=1}^{n} a(i,j)x(i) \geq 1$. Without loss of generality, we can assume that this constraint has at least one non-zero coefficient, otherwise it means that there is no bound on the value of $y(j)$ and the profit function is unbounded. The algorithm increases the value of the new variable $y(j)$ and the values of the primal variables $x(i)$ until the new primal constraint is satisfied. The augmentation method is described here in a continuous fashion, but it is not hard to implement the augmentation in a discrete way in any desired accuracy. In our continuous description the variables $x(i)$ behave according to a monotonically increasing function of $y(j)$. To implement the scheme in a discrete fashion, one should

[2]We can always normalize the new variable such that its coefficient in the objective function is 1.

find the minimal $y(j)$ such that the new primal constraint is satisfied. Note that variable $y(j)$ is being increased only in the jth round and the values of the primal variables never decrease. In the following we describe the jth round. The performance of the scheme is analyzed in Theorem 3.1.

(i) $y(j) \leftarrow 0$; For each $x(i)$: $a_i(\max) \leftarrow \max_{k=1}^{j}\{a(i,k)\}$.
(ii) While $\sum_{i=1}^{n} a(i,j)x(i) < 1$:
  (a) Increase $y(j)$ continuously.
  (b) Increase each variable $x(i)$ by the following increment function:

$$x(i) \leftarrow \max\left\{x(i), \frac{1}{na_i(\max)}\left[\exp\left(\frac{B}{2c(i)}\sum_{k=1}^{j} a(i,k)y(k)\right) - 1\right]\right\}.$$

THEOREM 3.1 *For any $B > 0$, the above scheme is a $B-$competitive algorithm for the general online fractional packing problem. Also, for the ith constraint it holds:*

$$\sum_{k=1}^{m} a(i,k)y(k) = c(i) \cdot O\left(\frac{\log n + \log \frac{a_i(\max)}{a_i(\min)}}{B}\right),$$

where, $a_i(\max) = \max_{k=1}^{m}\{a(i,k)\}$, and $a_i(\min) = \min_{k=1}^{m}\{a(i,k)|a(i,k) \neq 0\}$.

When all entries $a(i,j) \in \{0,1\}$, we do not need to update the value of $a_i(\max)$ in Lines (i) and (ii)b, thus simplifying the scheme. In this case, if we know in advance that each primal constraint consists of at most ℓ non-zero coefficients, it is possible to improve on the competitive factor from $O(\log n)$ to $O(\log \ell)$. This is done by replacing the value n in Line (ii)b by ℓ. In addition, the same scheme is applicable to the online fractional covering problem (with $a(i,j) \in \{0,1\}$). The improvement of the competitive ratio to $O(\log \ell)$ immediately reflects on the competitiveness of several corresponding integral covering problems which are mentioned in Section 1.1. We state these observations in the following Theorem. The proof follows along the same lines as the proof of Theorem 3.1.

THEOREM 3.2 *If the entries $a(i,j) \in \{0,1\}$, then there exists an $O(\log \ell)$-competitive algorithm for the fractional packing problem which does not violate the constraints. The same algorithm is $O(\log \ell)$-competitive for the fractional covering problem.*

Another useful property of our scheme is that it can handle an unbounded number of new dual variables in each round, in the following sense. In some cases the new dual variables, as well as the new primal constraints, do not appear explicitly. Instead, the scheme is only given an oracle for Line 2 that either returns an unsatisfied primal constraint or states that there is no such constraint. It can easily be verified that such an oracle suffices in order to implement the online scheme.

PROOF. [Proof of Theorem 3.1] Let $X(j)$ and $Y(j)$ be the values of the primal and dual solutions, respectively, obtained in round j. We prove the following claims on $X(j)$ and $Y(j)$:

(i) In each round j: $Y(j) \geq X(j)/B$.
(ii) The primal solution produced by the scheme is feasible.
(iii) For any dual constraint:

$$\sum_{k=1}^{m} a(i,k)y(k) \leq c(i) \cdot \frac{2\log\left(1 + \frac{na_i(\max)}{a_i(\min)}\right)}{B} = c(i) \cdot O\left(\frac{\log n + \log \frac{a_i(\max)}{a_i(\min)}}{B}\right).$$

The proof of the theorem then follows directly from weak duality.

Proof of (1): Note first that when the value of $a_i(\max)$ increases, the value of the primal solution does not change. Thus, the value of the primal solution only increases when the dual solution increases.

Initially, the values of the primal and dual solutions are zero. Consider the jth round in which $y(j)$ is being increased continuously. We prove that $\frac{\partial X(j)}{\partial y(j)} \leq B\frac{\partial Y(j)}{\partial y(j)}$, concluding that $X(j) \leq B \cdot Y(j)$.

$$\begin{aligned}
\frac{\partial X(j)}{\partial y(j)} &= \sum_{i=1}^{n} c(i)\frac{\partial x(i)}{\partial y(j)} \\
&\leq \sum_{i=1}^{n} \frac{c(i)Ba(i,j)}{2c(i)} \frac{1}{na_i(\max)} \exp\left(\frac{B}{2c(i)}\sum_{k=1}^{j} a(i,k)y(k)\right) \qquad (1)\\
&= \frac{B}{2}\sum_{i=1}^{n} a(i,j)\left[\frac{1}{na_i(\max)}\left(\exp\left(\frac{B}{2c(i)}\sum_{k=1}^{j} a(i,k)y(k)\right)-1\right)+\frac{1}{na_i(\max)}\right] \\
&\leq \frac{B}{2}\sum_{i=1}^{n} a(i,j)\left[x(i)+\frac{1}{na_i(\max)}\right] \leq \frac{B}{2}(1+1) = B = B\cdot\frac{\partial Y(j)}{\partial y(j)}, \qquad (2)
\end{aligned}$$

where (1) follows by taking the derivative of $x(i)$, and (2) follows since:

(i) $\sum_{i=1}^{n} a(i,j)x(i) < 1$.

(ii) $x(i) \geq \frac{1}{na_i(\max)}\left[\exp\left(\frac{B}{2c(i)}\sum_{k=1}^{j} a(i,k)y(k)\right)-1\right]$.

(iii) $\frac{1}{n}\sum_{i=1}^{n}\frac{a(i,j)}{a_i(\max)} \leq 1$.

The final equality follows since the value of the dual is $\sum_{k=1}^{j} y(k)$, and so $\frac{\partial Y(j)}{\partial y(j)} = 1$.

Proof of (2): This claim is trivial since we increase the primal variables until the current primal constraint becomes feasible. We never decrease any variable $x(i)$, so (feasible) constraints remain feasible.

Proof of (3): Consider the ith dual constraint of the form $\sum_{k=1}^{j} a(i,k)y(k) \leq c(i)$. Each time a variable $y(k)$ with coefficient $a(i,k) > 0$ is increased, the primal variable $x(i)$ is increased too. Let $a_i(\min) = \min_{k=1}^{m}\{a(i,k)|a(i,k) \neq 0\}$ and $a_i(\max) = \max_{k=1}^{m}\{a(i,k)\}$ be as defined previously. During the run of the algorithm, $x(i) \leq 1/a_i(\min)$, since if equality holds, then each primal constraint ($1 \leq j \leq m$) with $a(i,j) > 0$ is already feasible. Thus, we get the following:

$$\frac{1}{na_i(\max)}\left[\exp\left(\frac{B}{2c(i)}\sum_{k=1}^{j} a(i,k)y(k)\right)-1\right] \leq x(i) \leq 1/a_i(\min).$$

Simplifying, we obtain:

$$\sum_{k=1}^{j} a(i,k)y(k) \leq \frac{2\log\left(1+\frac{na_i(\max)}{a_i(\min)}\right)}{B}\cdot c(i).$$

□

Discussion. The reader may wonder at this point how did we decide to choose the function used in the algorithm for updating the primal and dual variables. We attempt to give here a systematic way of deriving this function. To understand the basic idea it suffices to consider the simpler case in which the variables $a(i,j) \in \{0,1\}$, and the jth primal constraint is simply $\sum_{i\in S(j)} x_i \geq 1$, where $S(j)$ is the set of indices i for which $a(i,j) = 1$. Also, suppose that our goal is to maintain both primal and dual feasiblity. Consider the point in time in which the jth primal constraint is given and assume that it is not satisfied. Our goal is to bound the derivative of the primal cost (denoted by P) as a function of the dual profit (denoted by D). That is, show that

$$\frac{\partial P}{\partial y_j} = \sum_{i\in S(j)} c_i \cdot \frac{\partial x_i}{\partial y_j} \leq \alpha \cdot \frac{\partial D}{\partial y_j},$$

where α is going to be the competitive factor. Suppose that the derivative of the primal cost satisfies:

$$\sum_{i\in S(j)} c_i \cdot \frac{\partial x_i}{\partial y_j} = A \cdot \sum_{i\in S(j)}\left(x_i + \frac{1}{n}\right). \qquad (3)$$

Then, since $\sum_{i \in S(j)} x_i \leq 1$, $\sum_{i \in S(j)} \frac{1}{n} \leq 1$, and $\frac{\partial D}{\partial y_j} = 1$, we get that

$$A \cdot \sum_{i \in S(j)} \left(x_i + \frac{1}{n} \right) \leq 2A \cdot \frac{\partial D}{\partial y_j}.$$

Thus, $\alpha = 2A$. Now, satisfying Equality (3) requires solving the following differential equation for each $i \in S(j)$:

$$\frac{\partial x_i}{\partial y_j} = \frac{A}{c_i} \cdot \left(x_i + \frac{1}{n} \right).$$

It is easy to verify that the solution is a family of functions of the following form:

$$x_i = C \cdot \exp\left(\frac{A}{c_i} \sum_{\ell \ | i \in S(j)} y_\ell \right) - \frac{1}{n},$$

where C can assume any value. Next, we have the following two boundary conditions on the solution:

- Initially, $x_i = 0$, and this happens when $\frac{1}{c_i} \sum_{j | i \in S(j)} y_j = 0$.
- If $\frac{1}{c_i} \cdot \sum_{j | i \in S(j)} y_j = 1$, (i.e., the dual constraint is tight), then $x_i = 1$. (Then, the primal constraint is also satisfied.)

The first boundary condition gives $C = \frac{1}{n}$. The second boundary condition gives us $A = \ln(n+1)$. Putting everything together we get the exact function used in the algorithm.

3.1 Lower Bounds In this section we prove two simple lower bounds showing that our scheme is optimal (up to constant factors).

LEMMA 3.1 *There is an instance of the general fractional packing problem with a single constraint such that*

$$\sum_{j=1}^{m} a(i,j) y(j) \geq \frac{H(a(\max)/a(\min))}{B}$$

for any online $B-$competitive algorithm, where $H(m)$ denotes the mth harmonic number, and $a(\max)/a(\min)$ is the ratio between the maximum and minimum non-zero entries in the constraint.

PROOF. Consider the following instance, for any m:

$$\max \quad \sum_{j=1}^{m} y(j)$$

$$\text{subject to:} \quad \sum_{j=1}^{m} (m-j+1) y(j) \leq 1.$$

Note that $a(\max)/a(\min) = m$. The variables $y(j)$ arrive one by one. After the jth round (for each j), the optimal offline value is $1/(m-j+1)$. Thus, the value of the objective function given by a B-competitive algorithm must be at least $1/(B(m-j+1))$. This yields the following sequence of inequalities:

$$\begin{aligned} y(1) &\geq 1/(Bm) \\ y(1)+y(2) &\geq 1/(B(m-1)) \\ y(1)+y(2)+y(3) &\geq 1/(B(m-2)) \\ &\vdots \quad \vdots \quad \vdots \\ y(1)+y(2)+y(3)+\ldots+y(m) &\geq 1/B \end{aligned}$$

Summing up over all m inequalities we get the desired bound:

$$\sum_{j=1}^{m} (m-j+1) y(j) \geq \frac{1}{B} \sum_{j=1}^{m} \frac{1}{m-j+1} = \frac{H(m)}{B}.$$

□

The proof of the next lemma is essentially the same as the proof of Lemma 4.1 in [5].

LEMMA 3.2 *There is an instance of the general online fractional packing problem with n constraints and $a(i,j) \in \{0,1\}$, such that for any $B-$competitive online algorithm there exists a constraint such that $\sum_{j=1}^{m} a(i,j)y(j) \geq c(i) \cdot \frac{\log n}{2B}$.*

4. The General Online Fractional Covering Problem In this section we describe our online scheme for computing a near-optimal fractional solution for the online fractional covering problem. As stated in Theorem 3.2, if the coefficients $a(i,j) \in \{0,1\}$, then the scheme in Section 3 is also applicable to the online fractional covering problem. Unfortunately, the scheme is not applicable to the general online fractional covering problem in case the coefficients $a(i,j) \geq 0$ are not limited to $\{0,1\}$. This happens since the scheme described in Section 3 does not always produce a feasible dual solution that can bound the primal solution efficiently. In this section we design a more elaborate scheme for the general online fractional covering problem.

Our scheme for the general online fractional covering problem gets a parameter $B > 0$. With $B > 0$ the competitive ratio of the scheme is $O(\frac{\log n}{B})$ and the following holds for each constraint: $\sum_{i=1}^{n} a(i,j)x(i) \geq \frac{1}{B}$. The scheme works in phases: When the first constraint is introduced, the scheme generates the first lower bound:

$$\alpha(1) \leftarrow \frac{1}{B} \cdot \min_{i=1}^{n} \left\{ \frac{c(i)}{a(i,1)} \right\} \leq \frac{OPT}{B}.$$

During the rth phase, it is assumed that the lower bound on the optimum is $\alpha(r)$, as long as the total primal cost does not exceed $\alpha(r)$. When the primal cost exceeds this bound, the scheme "forgets" about all the values given to the primal and dual variables so far, and starts a new phase in which the lower bound is doubled, i.e., $\alpha(r+1) \leftarrow 2\alpha(r)$. Nevertheless, the values of the "forgotten" variables are accounted for in the total cost of the solution. That is, the algorithm maintains in each phase r a new set of variables $x(i,r)$. However, since the variables of the linear program are required to be monotonically non-decreasing, the value of each variable $x(i)$ is actually set to $\max_r\{x(i,r)\}$ (or alternatively $\sum_r x(i,r)$). The cost of maintaining the variables of the linear program is, thus, at most the cost of maintaining the new variables in each phase. When we start processing a new phase we also set to zero all dual variables, and start processing again all primal constraints, starting from the first constraint. Thus, in each such phase, our algorithm produces "fresh" primal and dual solutions.

In the following we describe one round of our scheme in the rth phase. Let $\sum_{i=1}^{n} a(i,j)x(i) \geq 1$ be the new primal constraint that is introduced and let $y(j)$ be the corresponding dual variable. The values of the primal and dual variables are increased as follows. Note that during each phase $x(i)$ only increases. The performance of the scheme is analyzed in Theorem 4.1.

(i) $y(j) \leftarrow 0$
(ii) While $\sum_{i=1}^{n} a(i,j)x(i) < \frac{1}{B}$:
  (a) increase $y(j)$ continuously.
  (b) Increase each variable $x(i)$ by the following increment function:

$$x(i) \leftarrow \frac{\alpha(r)}{2nc(i)} \exp\left(\frac{\log 2n}{c(i)} \sum_{k=1}^{j} a(i,k)y(k) \right).$$

THEOREM 4.1 *For any $B > 0$, the scheme for the general online fractional covering problem achieves a competitive ratio of $O(\frac{\log n}{B})$, such that for each constraint $\sum_{i=1}^{n} a(i,j)x(i) \geq \frac{1}{B}$.*

PROOF. Let $X(r)$ and $Y(r)$ be the values of the primal and dual solutions, respectively, generated during the rth phase. We prove the following claims on $X(r)$ and $Y(r)$:

(i) For each finished phase r: $Y(r) \geq \frac{B\alpha(r)}{2\log 2n}$.
(ii) The dual solution generated during the rth phase is feasible.
(iii) The total cost of the primal solutions generated from the first phase until the rth phase is less than $2\alpha(r)$.
(iv) For any primal constraint given to the algorithm, $\sum_{i=1}^{n} a(i,j)x(i) \geq \frac{1}{B}$.

From the first three claims, together with weak duality, we conclude that the total cost of the primal solutions in all the phases up to phase r is at most:

$$2\alpha(r) \leq 4\alpha(r-1) \leq 8 \cdot \frac{\log 2n}{B} \cdot Y(r-1) \leq 8 \cdot \frac{\log 2n}{B} \cdot OPT.$$

Notice that if the scheme finishes in the first phase, then the total cost is at most $\alpha(1) \leq \frac{OPT}{B}$.

Proof of (1): Initially, $x(i) = \frac{\alpha(r)}{2nc(i)}$, and so $X(r)$ is initially at most $\alpha(r)/2$. The total profit of the dual solution is initially zero. From then on, the primal cost increases only when some dual variable $y(j)$ is increased. When the phase ends, $X(r) \geq \alpha(r)$. Thus, it suffices to prove that during the phase

$$\frac{\partial X(r)}{\partial y(j)} \leq \frac{\log 2n}{B} \frac{\partial Y(r)}{\partial y(j)}.$$

This follows since,

$$\begin{aligned}
\frac{\partial X(r)}{\partial y(j)} &= \sum_{i=1}^{n} c(i) \frac{\partial x(i)}{\partial y(j)} \\
&= \sum_{i=1}^{n} \frac{c(i)\log(2n)a(i,j)}{c(i)} \frac{\alpha(r)}{2nc(i)} \exp\left(\frac{\log 2n}{c(i)} \sum_{k=1}^{j} a(i,k)y(k)\right) \\
&= \log 2n \sum_{i=1}^{n} a(i,j)x(i) \leq \frac{\log 2n}{B} = \frac{\log 2n}{B} \cdot \frac{\partial Y(j)}{\partial y(j)}, \qquad (4)
\end{aligned}$$

where (4) follows since $\sum_{i=1}^{n} a(i,j)x(i) \leq \frac{1}{B}$. The final equality follows since the value of the dual solution is $\sum_{k=1}^{j} y(k)$ and thus $\frac{\partial Y(j)}{\partial y(j)} = 1$.

Proof of (2): Consider the ith dual constraint of the form $\sum_{k=1}^{m} a(i,k)y(k) \leq c(i)$. Each time variable $y(k)$ with coefficient $a(i,k) > 0$ is increased, the corresponding primal variable $x(i)$ is increased too. During the rth phase of the algorithm, $x(i) \leq \frac{\alpha(r)}{c(i)}$, since otherwise it would have contributed to the cost of the primal solution more than $\alpha(r)$, and the current phase would have ended. Thus, we get the following equation:

$$x(i) = \frac{\alpha(r)}{2nc(i)} \exp\left(\frac{\log 2n}{c(i)} \sum_{k=1}^{j} a(i,k)y(k)\right) \leq \frac{\alpha(r)}{c(i)}.$$

Simplifying, we get the desired result:

$$\sum_{k=1}^{m} a(i,k)y(k) \leq c(i).$$

Proof of (3): We bound the total cost paid by the online algorithm. The total primal cost in the rth phase is at most $\alpha(r)$. Since the ratio between $\alpha(k)$ and $\alpha(k-1)$ is 2, we get that the total cost until the rth phase is at most $\sum_{k=1}^{r} \alpha(k) \leq 2\alpha(r)$.

Proof of (4): The claim is trivial, since each round terminates only when the value of the left hand side of the new primal constraint is at least $1/B$. The value of each variable $x(i)$ never decreases, thus all previous primal constraints remain feasible. □

4.1 Adding Box Constraints In this section we extend our methods to handle upper bounds, or *box constraints*, on the variables $x(i)$. Let $u(i)$ denote the upper bound on variable $x(i)$. Note that upper bounds on the variables may result in an infeasible instance of a covering problem. We next sketch the main ideas and changes that are needed to deal with upper bounds. Adding box constraints to a covering problem results in new negative variables $z(i)$, $1 \leq i \leq n$, in the dual program. The primal covering with box constraints and the new corresponding dual program are described in Figure 2. The performance of the scheme is analyzed in Theorem 4.2.

Primal (Covering)		Dual (Packing)	
Minimize:	$\sum_{i=1}^{n} c(i)x(i)$	Maximize:	$\sum_{j=1}^{m} y(j) - \sum_{i=1}^{n} u(i)z(i)$
Subject to:		Subject to:	
$\forall j : 1 \leq j \leq m$:	$\sum_{i=1}^{n} a(i,j)x(i) \geq 1$	$\forall i : 1 \leq i \leq n$:	$\sum_{j=1}^{m} a(i,j)y(j) - z(i) \leq c(i)$
$\forall i : 1 \leq i \leq n$:	$0 \leq x(i) \leq u(i)$	$\forall i, j$:	$y(j), z(i) \geq 0$

Figure 2: Primal (covering) with box constraints and its dual (packing) problem.

THEOREM 4.2 *For any $B > 0$, the scheme for the general online fractional covering problem with box constraints achieves a competitive ratio of $O(\frac{\log n}{B})$, such that for the jth constraint, $\sum_{i=1}^{n} a(i,j)x(i) \geq \frac{1}{B}$, and for the ith variable, $x(i) \leq \frac{u(i)}{B}$.*

Our proposed scheme works in phases, similarly to Section 4, where each phase has an upper bound on the total cost. When variable $x(i)$ reaches the value of $\frac{u(i)}{B}$, we start increasing its corresponding dual variable $z(i)$. Increasing $z(i)$ stops the increase of primal variable $x(i)$. Let X be the set of *tight variables* with value $\frac{u(i)}{B}$. If all variables in some unsatisfied constraint are tight, the scheme returns that no feasible solution exists. A single round during the rth phase is described in the following:

(i) $y(j) \leftarrow 0$
(ii) While $\sum_{i=1}^{n} a(i,j)x(i) < \frac{1}{B}$:
  (a) Increase $y(j)$ continuously.
  (b) For each variable $x(i) \in X$ (i.e., $x(i) = u(i)/B$), increase $z(i)$ at rate $a(i,j)y(j)$.
  (c) Increase each variable $x(i)$ by the following increment function:

$$x(i) \leftarrow \min\left\{\frac{u(i)}{B}, \frac{\alpha(r)}{2nc(i)}\right\} \cdot \exp\left(\log 2n \left[\frac{1}{c(i)} \sum_{k=1}^{j} a(i,k)y(k) - z(i)\right]\right).$$

Note that during each phase r the initial value of variable $x(i)$ is $\min\{\frac{u(i)}{B}, \frac{\alpha(r)}{2nc(i)}\}$ and it can only increase during the phase. It is also easy to verify that during the run of the algorithm $x(i) \leq \frac{u(i)}{B}$ is maintained (due to the augmentation of $z(i)$). We next provide the proof of Theorem 4.2.

PROOF. [Proof of Theorem 4.2] The proof is essentially along the same lines as the proof of Theorem 4.1. We prove that:

(i) For each finished phase r: $Y(r) \geq \frac{B\alpha(r)}{2\log 2n}$.
(ii) The dual solution generated during the rth phase is feasible.
(iii) The total cost of the primal solutions generated from the first phase until the rth phase is less than $2\alpha(r)$.
(iv) For any primal constraint given to the algorithm, $\sum_{i=1}^{n} a(i,j)x(i) \geq \frac{1}{B}$.

Proof of (1): The proof is the same as in Theorem 4.1. However, in this case we should consider variables that are tight (i.e. $x(i) = \frac{u(i)}{B}$), and hence are not increased anymore. The derivative of the dual profit becomes:

$$\frac{\partial Y(r)}{\partial y(j)} = 1 - \sum_{x(i) \in X} u(i)a(i,j).$$

Note that since $\sum_{i=1}^{n} a(i,j)x(i) < \frac{1}{B}$, and $x(i) \leq \frac{u(i)}{B}$, then the derivative is always non-negative. If all the primal variables belong to X, it means that $\sum_{i=1}^{n} a(i,j)\frac{u(i)}{B} < \frac{1}{B}$, and so there is no feasible solution to the original instance. Otherwise, similarly to the proof of Theorem 4.1, we get:

$$\frac{\partial X(r)}{\partial y(j)} = \sum_{i=1}^{n} c(i)\frac{\partial x(i)}{\partial y(j)}$$

$$
\begin{aligned}
&= \sum_{x(i)\notin X} \frac{c(i)\cdot(\log 2n)\cdot a(i,j)}{c(i)} \frac{\alpha(r)}{2nc(i)} \exp\left(\frac{\log 2n}{c(i)} \sum_{k=1}^{j} a(i,k)y(k) - z(i)\right) \\
&= \log 2n \sum_{x(i)\notin X} a(i,j)x(i) \\
&\leq \frac{\log 2n}{B}\left(1 - \sum_{x(i)\in X} a(i,j)u(i)\right) = \frac{\log 2n}{B}\frac{\partial Y(r)}{\partial y(j)}, \qquad (5)
\end{aligned}
$$

where Inequality (5) follows since

$$\sum_{x(i)\notin X} a(i,j)x(i) < \frac{1}{B} - \sum_{x(i)\in X} a(i,j)x(i) = \frac{1}{B} - \sum_{x(i)\in X} a(i,j)\frac{u(i)}{B}.$$

Proof of (2): Consider the ith dual constraint of the form $\sum_{k=1}^{j} a(i,k)y(k) - z(i) \leq c(i)$. If the corresponding primal variable is tight, then the value of the left hand side of the constraint remains the same, since the variable $z(i)$ is increased. Until that time variable $x(i)$ is increased in the same way as in the case where there are no box constraints. In particular, each time some $y(k)$ with coefficient $a(i,k) > 0$ is increased, the corresponding primal variable $x(i)$ is increased too. During the rth phase of the algorithm, $x(i) \leq \frac{\alpha(r)}{c(i)}$, since otherwise it contributes to the cost of the primal solution more than $\alpha(r)$, and the current phase ends. Thus, we get the following equation:

$$x(i) = \frac{\alpha(r)}{2nc(i)} \exp\left(\log 2n \left[\frac{1}{c(i)} \sum_{k=1}^{j} a(i,k)y(k) - z(i)\right]\right) \leq \frac{\alpha(r)}{c(i)}.$$

Simplifying we get the desired result:

$$\sum_{k=1}^{j} a(i,k)y(k) - z(i) \leq c(i).$$

The rest of the proof is essentially the same as the proof of Theorem 4.1. □

5. Integral Online Problems: Derandomization In this section we show the applicability of our fractional schemes to the solution of online integral problems. To do so, we use the schemes as a "black box" and deterministically round the fractional solutions obtained. The deterministic rounding is obtained by transforming an (offline) pessimistic estimator into an online potential function. This idea appeared implicity in [1]. We note that the existence of an offline derandomization method does not necessarily yield an online deterministic rounding algorithm. For example, derandomizing (online) algorithms for problems like the online multicast problem on trees and the online group Steiner problem considered in [2] is still an open problem. (They all do have offline derandomizations.) Thus, we remark that the ability to transform a (offline) pessimistic estimator into an online potential function is quite surprising.

We demonstrate our derandomization method on two online problems. We first consider the online unweighted set-cover problem [1]. For this problem we provide a better deterministic algorithm in terms of the competitive ratio. In particular the algorithm we design is $O(\log d \log(n/OPT))$-competitive, where d is the maximum frequency of an element (i.e., the maximum number of sets an element belong to), and n is the number of elements. The previous algorithm in [1] is $O(\log m \log n)$-competitive, where m is the number of sets. To do so, we draw on ideas from the improved offline rounding and derandomization methods of [29] to generate an improved potential function. We then consider the online problem of throughput-competitive routing of virtual circuits. We show that the algorithm presented in [5] can be derived by a deterministic rounding of the fractional solution produced by our packing scheme.

The above approach provides us with the following insight to the competitive factors obtained by [5]. The $O(\log m)$ competitive factor follows from an online generation of a fractional solution. The minimum edge capacity determines the scaling of the fractional solution that is needed in order to guarantee a high probability of success in the rounding phase. In contrast, we note that producing a near-optimal fractional solution that does not violate capacity constraints can be done independently of the values of the edge capacities.

5.1 Improved Competitive Ratio for Online Unweighted Set Cover We define the online unweighted set cover problem as follows. Let $X = \{e_1, \ldots, e_n\}$ be a ground set of n elements, and let $\mathcal{S}$ be a family of subsets of X, $|\mathcal{S}| = m$. A *cover* is a collection of sets such that their union is X. Clearly, this problem belongs to the covering-packing framework described in Figure 1, where for each i, $1 \leq i \leq m$, $c(i) = 1$, and $a(i,j) = 1$ if and only if element j belongs to subset i of $\mathcal{S}$. The goal is to find a cover of minimum cardinality.

In an online setting, the elements, or covering constraints, are given one-by-one by an adversary. For each new element given, the algorithm has to cover it by some set of $\mathcal{S}$ containing it. Denote by $X' \subseteq X$ the set of elements given by the adversary. In general, the algorithm does not know in advance the set of elements X' given by the adversary, i.e., X' may be a strict subset of X, however, our assumption is that the set cover instance (i.e., X and $\mathcal{S}$) is known in advance. Using the techniques of Section 3 we can get online a fractional solution which is $O(\log d)$-competitive. Recall that d is the maximum frequency of an element (i.e., the maximum number of sets an element belongs to). Let $\mathcal{C}$ denote the family of sets in $\mathcal{S}$ that the algorithm chooses, and let C denote the set of elements covered by sets belonging to $\mathcal{C}$.

We show how to get online an integral solution which is $O(\log d \log(n/OPT))$-competitive. We assume that the value OPT is known in advance. This assumption is justified since we can guess the value of OPT by doubling. We start by guessing $OPT = 1$, and then run the algorithm with this value of the optimal solution. If it turns out that the cardinality of the optimal solution is already at least twice our current guess for it, (that is, the cardinality of $\mathcal{C}$ exceeds $\Theta(OPT \log d \log(n/OPT))$), then we can "forget" about all the sets chosen so far to $\mathcal{C}$, update the value of OPT by doubling it, and restart the algorithm. Of course, any set that was already chosen is not thrown away in reality. Therefore, the sum of costs in all rounds is an upper bound on the real cost of the algorithm. It can be easily verified that this can only change the competitive ratio by a constant factor.

Our main technique is transforming the pessimistic estimator that appears in [29] into an online potential function. Denote by $w(s)$ the fractional weight of set s as given by the algorithm of Section 3. (Recall this weight is monotonically increasing over time). For each element e_i $(1 \leq i \leq n)$ define:

$$f(e_i) = \min\left\{1, \exp\left(-\alpha + \alpha \sum_{s|e_i \in s} w(s)\right)\right\}.$$

Define the potential function $\Phi = \Phi_1 + \Phi_2$, where:

$$\Phi_1 = \left[1 - \prod_{e_i|e_i \notin C}(1 - f(e_i))\right] \quad \text{and} \quad \Phi_2 = \exp\left(\sum_{s \in \mathcal{S}} \left((\ln 2) \cdot \chi_{\mathcal{C}}(s) - \alpha w(s)\right) - OPT\right).$$

The function $\chi_{\mathcal{C}}(s) = 1$ if $s \in \mathcal{C}$, and $\chi_{\mathcal{C}}(s) = 0$ otherwise. The parameter $\alpha = O(\log(n/OPT))$ will determine our competitive ratio. The first term of the potential function guarantees that each element that is given to the algorithm is covered. The second term is used to bound the cardinality of the solution.

We now run the online algorithm presented in Section 3 to produce a fractional solution. Note that since all coefficients $a(i,j) \in \{0,1\}$, the simpler scheme in Section 3 is applicable (Theorem 3.2). The above potential function Φ is used to determine which sets to pick to the cover:

- Each time the weight of a set s is augmented in Line 2(b) of the algorithm, add s to the cover $\mathcal{C}$ if its addition does not increase the value of the potential function Φ.

At this point it is not even clear whether the above algorithm produces a feasible integral solution. We prove this fact along with the competitive ratio in Lemma 5.2. In the next lemma we prove crucial properties of the potential function that are useful for proving Lemma 5.2.

LEMMA 5.1 *The potential function Φ satisfies the following properties:*

(i) *At start $\Phi \leq 1$, and at any time during the run of the algorithm $\Phi > 0$.*

(ii) *When a set is augmented, then either taking it to $\mathcal{C}$, or excluding it, does not increase Φ.*

PROOF. We prove the two claims:

Proof of (1): Since for each element e_i, $0 \leq f(e_i) \leq 1$, the value of the product in the term Φ_1 is at most 1, yielding that $\Phi_1 \geq 0$. The term Φ_2 is trivially positive. Initially, for each set s, $w(s) = 0$. We choose

$$\alpha = \max\left\{1, \ln\left(\frac{rn}{OPT}\right)\right\} = O\left(\log\left(\frac{n}{OPT}\right)\right),$$

where $r = e\ln(e/e-1)$. Thus, the value of the potential function initially is:

$$\Phi \quad = \quad 1 - (1-e^{-\alpha})^n + \exp(-OPT) \leq 1 - \exp(-dne^{-\alpha}) + \exp(-OPT) \leq 1,$$

where the inequalities follow since $\alpha \geq 1$ and $\alpha \geq \ln(\frac{dn}{OPT})$.

Proof of (2): We use a probabilistic argument. Consider a point in time where $w(s)$ is augmented by $0 \leq \delta_s \leq 1$. Consider the event of adding set s to the cover with probability $1 - \exp(-\alpha\delta_s)$. We prove that choosing the set with such probability does not increase the expected value of Φ, and thus the claim follows. Let $\Phi^{\text{start}} = \Phi_1^{\text{start}} + \Phi_2^{\text{start}}$ and $\Phi^{\text{end}} = \Phi_1^{\text{end}} + \Phi_2^{\text{end}}$ be the respective values of the potential function Φ before and after the probabilistic experiment. Consider first the term Φ_2. We bound the expected value of Φ_2 after the probabilistic experiment. If s is already in $\mathcal{C}$ then Φ_2 only decreases. Otherwise:

$$\begin{aligned} E\left[\Phi_2^{\text{end}}\right] &= \Phi_2^{\text{start}}\left([1-\exp(-\alpha\delta_s)]\exp\left(\ln 2 - \alpha\delta_s\right) + \exp(-\alpha\delta_s)\exp(-\alpha\delta_s)\right) \\ &= \Phi_2^{\text{start}}\exp(-\alpha\delta_s)\left([1-\exp(-\alpha\delta_s)]\,2 + \exp(-\alpha\delta_s)\right) \\ &= \Phi_2^{\text{start}}\exp(-\alpha\delta_s)\left(2 - \exp(-\alpha\delta_s)\right) \leq \Phi_2^{\text{start}}, \end{aligned}$$

where the last inequality follows since $x(2-x) \leq 1$ for all x.

Next consider the term Φ_1 of the potential function. If s is in $\mathcal{C}$ then Φ_1 does not change. Otherwise, let $N(s)$ be the elements that are covered by s (and were augmented during this round). Let $f(e)$ and $f'(e)$ be the contribution of element e to Φ_1 before and after the weight augmentation step, respectively. Let $0 \leq A \leq 1$ be the product of the still uncovered elements that are not covered by s. The contribution of these elements to Φ_1 remains the same before and after the weight augmentation. We prove that Φ_1 decreases, by induction on the number of elements in $N(s)$. If $|N(s)| = 1$ (i.e. s covers only one additional element e) then:

$$\begin{aligned} E\left[\Phi_1^{\text{end}}\right] &= [1-\exp(-\alpha\delta_s)]\,(1-A) + \exp(-\alpha\delta_s)(1 - A(1-f'(e)) \\ &= 1 - A(1-\exp(-\alpha\delta_s)f'(e)) \leq 1 - A(1-f(e)) = \Phi_1^{\text{start}}. \end{aligned}$$

We next assume that the claim holds for $|N(s)| = k$ and prove it for $|N(s)| = k+1$ Assume that the additional elements s covers are $e_1, e_2, \ldots, e_{k+1}$, then:

$$\begin{aligned} E\left[\Phi_1^{\text{end}}\right] &= [1-\exp(-\alpha\delta_s)]\,(1-A) + \exp(-\alpha\delta_s)(1 - A\prod_{i=1}^{k+1}(1-f'(e_i))) \\ &= [1-\exp(-\alpha\delta_s)]\,(1-A) + \exp(-\alpha\delta_s)(1 - A\prod_{i=1}^{k}(1-f'(e_i))) \\ &\qquad + A\cdot\exp(-\alpha\delta_s)f'(e_{k+1})\prod_{i=1}^{k}(1-f'(e_i)) \\ &\leq 1 - A\prod_{i=1}^{k}(1-f(e_i)) + A\cdot\exp(-\alpha\delta_s)f'(e_{k+1})\prod_{i=1}^{k}(1-f'(e_i)) \qquad (6) \\ &\leq 1 - A\prod_{i=1}^{k}(1-f(e_i)) + A\cdot f(e_{k+1})\prod_{i=1}^{k}(1-f(e_i)) \qquad (7) \\ &= 1 - A\prod_{i=1}^{k+1}(1-f(e_i)) = \Phi_1^{\text{start}}, \qquad (8) \end{aligned}$$

Primal		Dual	
Minimize:	$\sum_{e\in E} u(e)x(e) + \sum_{r_i} Z(r_i)$	Maximize:	$\sum_{r_i} \sum_{P\in\mathbb{P}(r_i)} f(r_i,P)$
Subject to:		Subject to:	
$\forall r_i \in \mathbb{R}, P \in \mathbb{P}(r_i)$:	$\sum_{e\in P} x(e) + Z(r_i) \geq 1$	$\forall r_i \in \mathbb{R}$:	$\sum_{P\in\mathbb{P}(r_i)} f(r_i,P) \leq 1$
		$\forall e \in E$:	$\sum_{r_i\in\mathbb{R}, P\in\mathbb{P}(r_i)\mid e\in P} \sum_{r_i} f(r_i,P) \leq u(e)$

Figure 3: The splittable routing problem (Maximize) and its corresponding primal problem.

where Inequality (6) follows from the induction hypothesis and Inequality (7) follows from the properties of the function $f(e)$. □

LEMMA 5.2 *The online algorithm satisfies the following:*

(*i*) *Each element that is given to the algorithm is covered.*

(*ii*) *The cardinality of* $\mathcal{C}$ *is at most* $OPT \cdot O(\log d \log(n/OPT))$.

PROOF. To prove the first claim, note first that each element e_i that is given to the algorithm is fractionally covered (as guaranteed in Section 3). Thus, by definition $f(e_i) = 1$. If e_i is not covered and thus $e_i \notin C$, then $\Phi_1 = 1$. Since $\Phi_2 > 0$, then $\Phi > 1$, contradicting Lemma 5.1 if e_i is not covered.

To prove the second claim note that $\Phi_1 > 0$ always holds. Since by Lemma 5.1 $\Phi \leq 1$, we conclude that also $\Phi_2 \leq 1$. Thus, we get that:

$$\exp\left(\sum_{s\in\mathcal{S}} (\ln 2 \cdot \chi_{\mathcal{C}}(s) - \alpha w(s)) - OPT\right) \leq 1.$$

Simplifying, we get that:

$$\sum_{s\in\mathcal{S}} \chi_{\mathcal{C}}(s) \leq \frac{1}{\ln 2} \cdot \alpha \cdot \sum_{s\in\mathcal{S}} w(s) + \frac{1}{\ln 2} \cdot OPT = O(\log d \log(n/OPT)) \cdot \mathrm{OPT},$$

where the inequality follows from the value of α and the guarantee on the competitiveness of the fractional algorithm. □

5.2 Throughput-Competitive Online Routing of Virtual Circuits The online problem of maximizing the throughput of scheduled virtual circuits was studied in [5]. In its simplest version we are given a graph with capacities $u(e)$ defined on the edge set. A set of requests $r_i = (s_i, t_i)$ $(1 \leq i \leq n)$ arrive in an online fashion. To serve a request, the algorithm chooses a path between s_i and t_i and allocates a bandwidth of 1 on this path. The total bandwidth allocated on any edge is not allowed to exceed its capacity. The total profit of the algorithm is the number of requests served and the performance is measured with respect to the maximum number of requests that could have received service (offline).

In a fractional version of the problem the allocation of bandwidth to a request is not restricted to be from $\{0, 1\}$; instead, each request can be allocated a fractional bandwidth in the range $[0, 1]$. In addition, the bandwidth allocated to a request can be divided between several paths. This is an online version of the maximum multi-commodity flow problem. We describe the problem as a packing problem in Figure 3. For $r_i = (s_i, t_i)$, let $\mathbb{P}(r_i)$ be the set of simple paths between s_i and t_i. For each $P \in \mathbb{P}(r_i)$, the variable $f(r_i, P)$ corresponds to the amount of flow (service) given to request r_i on the path P. The first set of constraints guarantees that each client gets at most a fractional flow (bandwidth) of 1. The second set of constraints guarantees the capacity constraints on the edges. In the primal problem we assign a variable $Z(r_i)$ to each request r_i and a variable $x(e)$ to each edge e in the graph. The primal problem is a special case of the general online fractional covering problem with $a(i, j) \in \{0, 1\}$. We note that our methods can generate a fractional solution to an extension of the problem with any non-negative coefficients $a(i, j)$. The extension can be viewed as giving each edge a different capacity with respect to the clients that are using it. This models a more complex relationship between clients and network capacities.

A competitive algorithm for the problem was proposed in [5]. We provide here an alternate algorithm that achieves the same competitive factor as [5]. We build our algorithm systematically by using the two

phase approach. First, we use our scheme to generate a *feasible* fractional solution which is a factor of $O(\log P(\max))$ away from the optimum, where $P(\max)$ is the length of the longest path in the graph. Next, we convert the standard pessimistic estimator designed for the offline version of the problem into an online potential function. A randomized method to achieve a feasible integral solution is to scale down all the flows by some factor $B \geq 1$. We then choose to route the flow on a path P, carrying a fractional flow of $f(P)$, with probability $f(P)/B$. This can be done, for example, by mapping the values $f(P)/B$ given to a request to the interval $[0,1]$, and then choosing a random number that would determine which path to choose (or to reject the request altogether).

When B is large enough, there is a positive probability that no edge capacity is violated, and the total integral flow is large enough [28]. This algorithm was derandomized using the method of conditional expectations via a pessimistic estimator [27]. We transform this pessimistic estimator into an online potential function. When the flow corresponding to a request is increased, the algorithm serves the request (integrally) on a path if by doing so the following potential function does not increase. In what follows we prove that there is always a good choice of either a path for a request, or a rejection of a request, that would not increase the potential function. Our online potential function $\Phi = \Phi_1 + \Phi_2$ is the sum of the following two functions:

$$\Phi_1 = \frac{1}{2} \exp\left\{ \left[\sum_{r_i} \sum_{P \in \mathbb{P}(r_i)} \frac{f(r_i,P)}{2B} - \ln 2 \cdot \sum_{r_i} \chi(r_i) \right] \right\}$$

$$\Phi_2 = \frac{1}{2m} \sum_{e \in E} \exp\left\{ \left(1 + \frac{\ln 2m}{u(e)}\right) \chi(e) - \sum_{P|e \in P} \sum_{r_i} f(r_i,P) \right\},$$

where $\chi(r_i)$ is the characteristic function of request r_i (i.e. $\chi(r_i) = 1$ iff it is serviced), $\chi(e)$ is the total number of paths that use the edge, and m is the number of edges. Choosing $B = \exp\{(1 + \frac{\ln 2m}{u(\min)})\} - 1$, where $u(\min)$ is the minimum capacity of an edge, suffices to prove Lemma 5.3, which in turn is used to prove Lemma 5.4. Note also that $B \geq 1$ always holds.

LEMMA 5.3 *Initially, $\Phi \leq 1$, and throughout the algorithm $\Phi > 0$. In addition, each time the flow for a request is increased, then either serving the request on some flow path or rejecting the request altogether does not increase the potential function Φ.*

PROOF. It is easy to verify that initially $\Phi = 1$. Also, since Φ is a sum of exponential functions it is always positive.

To prove the second part of the claim we use a probabilistic argument. Specifically, we consider a single round in which the flow to some request r_i is increased on several paths, such that

$$f(r_i) \triangleq \sum_{P \in \mathbb{P}(r_i)} f(r_i,P).$$

Then we consider the following random trial: we map the values $f(r_i,P)/B$ to the interval $[0,1]$ and then choose a random number in $[0,1]$. We choose to fully serve the request r_i through the path that comes up in the trial. If the random number we chose falls outside the flow values, we choose to reject the request. Note that the probability that we served the request is exactly $f(r_i)/B$. Also, for each edge the probability that we use it to serve the request is exactly the amount of flow that is going through the edge divided by B. We prove that the expected value of Φ does not increase in this random trial. This means that either there exists a path such that serving the request on this path does not increase the potential function, or rejecting the request does not increase the potential function. By linearity of expectation we can analyze each term in the potential function separately.

Consider the first term of the potential function. Let Φ_1^{start} and Φ_1^{end} be the respective start and end values of Φ_1, before increasing the flow and after the random trial. By the above observation the expected value of Φ_1^{end} is:

$$E\left[\Phi_1^{\text{end}}\right] = \frac{f(r_i)}{B} \cdot \left(\Phi_1^{\text{start}} \cdot \exp\left\{\frac{f(r_i)}{2B} - \ln 2\right\}\right) + \left(1 - \frac{f(r_i)}{B}\right) \cdot \left(\Phi_1^{\text{start}} \cdot \exp\left\{\frac{f(r_i)}{2B}\right\}\right)$$

$$\begin{aligned} &= \Phi_1^{\text{start}} \cdot \left[\frac{f(r_i)}{2B} \cdot \exp\left\{ \frac{f(r_i)}{2B} \right\} + \left(1 - \frac{f(r_i)}{B}\right) \cdot \exp\left\{ \frac{f(r_i)}{2B} \right\} \right] \\ &= \Phi_1^{\text{start}} \cdot \exp\left\{ \frac{f(r_i)}{2B} \right\} \cdot \left[1 - \frac{f(r_i)}{2B} \right] \leq \Phi_1^{\text{start}}, \end{aligned}$$

where the last inequality follows since for any $x \geq 0$, $1 - x \leq e^{-x}$.

We next analyze the expected value of each term in Φ_2. Let Φ_2^{start} and Φ_2^{end} be the respective start and end values of Φ_2, before increasing the flow and after the random trial. Consider a term corresponding to an edge e. Let $f(r_i, e)$ be the total flow through the edge e that serves request r_i. By the properties of the random trial the probability that we serve request r_i through edge e is exactly $\frac{f(r_i,e)}{B}$. Thus, the expected value of this term is:

$$\begin{aligned} E\left[\Phi_2^{\text{end}}\right] &= \frac{f(r_i,e)}{B} \cdot \left(\Phi_2^{\text{start}} \cdot \exp\left\{ \left(1 + \frac{\ln 2m}{u(e)} - f(r_i,e)\right) \right\} \right) \\ &+ \left(1 - \frac{f(r_i,e)}{B}\right) \cdot \left(\Phi_2^{\text{start}} \cdot \exp\left\{ -f(r_i,e) \right\} \right) \\ &= \Phi_2^{\text{start}} \cdot \exp\left\{ -f(r_i,e) \right\} \cdot \left[\frac{f(r_i,e)}{B} \cdot \exp\left\{ 1 + \frac{\ln 2m}{u(e)} \right\} + \left(1 - \frac{f(r_i,e)}{B}\right) \right] \\ &= \Phi_2^{\text{start}} \cdot \exp\left\{ -f(r_i,e) \right\} \cdot \left[\frac{f(r_i,e)}{B} \cdot \left(\exp\left\{ 1 + \frac{\ln 2m}{u(e)} \right\} - 1 \right) + 1 \right] \\ &\leq \Phi_2^{\text{start}} \cdot \exp\left\{ -f(r_i,e) \right\} \cdot \left[f(r_i,e) + 1 \right] \leq \Phi_2^{\text{start}}. \end{aligned}$$

The last inequalities follow since $B = \exp\left\{1 + \frac{\ln 2m}{u(\min)}\right\} - 1 \geq \exp\left\{1 + \frac{\ln 2m}{u(e)}\right\} - 1$, and since for any $x \geq 0$, $1 + x \leq e^x$. □

LEMMA 5.4 *The algorithm is* $O(\log P(\max) \cdot \left[\exp\{(1 + \frac{2 \ln m}{u(\min)})\} - 1\right])$*-competitive and does not violate the capacity constraints.*

We remark that when $u(\min) \geq \log n$ we get that the algorithm is $O(\log P(\max))$-competitive. The competitiveness is exactly as in [5].

PROOF. To see that the algorithm does not violate the capacity constraints we consider the second part of the potential function. From Lemma 5.3 each term in the potential function Φ is at most 1. Therefore, for each edge e:

$$\frac{1}{2m} \exp\left\{ \left(1 + \frac{\ln 2m}{u(e)}\right) \chi(e) - \sum_{P|e \in P} \sum_{r_i} f(r_i, P) \right\} \leq 1.$$

Since the fractional solution is feasible, we get that $\sum_{P|e \in P} \sum_{r_i} f(r_i, P) \leq u(e)$. Therefore, simplifying the inequality we get that

$$\chi(e) \;\leq\; u(e) \cdot \frac{\ln 2m + \sum_{P|e \in P} \sum_{r_i} f(r_i, P)}{\ln 2m + u(e)} \;\leq\; u(e).$$

To see that the algorithm is competitive consider Φ_1. By Lemma 5.3 $\Phi_1 \leq 1$. Therefore, we get that:

$$\frac{1}{2} \exp\left\{ \left[\sum_{r_i} \sum_{P \in \mathbb{P}(r_i)} \frac{f(r_i, P)}{2B} - \ln 2 \cdot \sum_{r_i} \chi(r_i) \right] \right\} \leq 1.$$

Simplifying we get that:

$$\sum_{r_i} \sum_{P \in \mathbb{P}(r_i)} \frac{f(r_i, P)}{2B} - \ln 2 \cdot \sum_{r_i} \chi(r_i) \leq \ln 2,$$

which means that:

$$\sum_{r_i} \chi(r_i) \geq -1 + \frac{1}{2B \ln 2} \cdot \sum_{r_i} \sum_{P \in \mathbb{P}(r_i)} f(r_i, P).$$

Since the fractional algorithm guarantees that

$$\sum_{r_i} \sum_{P \in \mathbb{P}(r_i)} f(r_i, P) \geq \frac{1}{O(\log P(\max))} \cdot OPT,$$

we get the desired bound. □

6. Conclusions and Further Results We have developed in this paper a general recipe for the design and analysis of online algorithms, applicable to a wide range of interesting online problems. The competitive analysis is direct and without the use of a potential function appearing "out of nowhere". Also, the competitiveness of our online algorithms is shown to be with respect to an optimal fractional solution. We note that the use of a linear program for online problems helps detecting the difficulties of the problem in hand.

Since the preliminary version of this paper appeared in [9], several new applications of the online primal-dual method were discovered, establishing it as an important paradigm for the design of online algorithms.

Covering Problems: The online set-cover problem [1], along with the graph optimization problems in [2], naturally fall within the framework of online covering problems. In fact, these problems motivated our work and were its starting point. It was shown in [11] that classic online problems such as the ski rental problem and the dynamic TCP-acknowledgement problem [20] can be viewed as online covering problems, and the well known randomized competitive algorithms for these problems can be derived via the primal-dual framework presented here. The parking permit problem studied by Meyerson [25] can also be cast as an online covering problem. Thus, the randomized algorithm presented in [25] can be obtained using the two phase approach: first, a fractional solution which has a logarithmic competitive factor is generated using the primal-dual framework. Then, by randomized rounding, an integral solution is obtained similarly to [25]. More recently, an extension of the ideas that appear in our work were used to obtain a randomized $O(\log k)$-competitive algorithm for the weighted paging problem [6], where k is the cache size. This is the first randomized $o(k)$-algorithm for the problem. We note that weighted paging is a special case of the more general k-server problem (see [8] for more details).

Packing Problems: The problems of throughput-competitive online routing of virtual circuits and online load balancing studied in [5, 3] can be viewed as online packing problems. In this work we showed that an alternate routing algorithm (achieving the same competitive factor as the algorithm presented in [5]) can be derived systematically using our approach. A more delicate algorithm, also based on a primal-dual approach, was recently used to derive improved routing goals in many settings [10]. (In particular, [10] considered the fairness goals studied previously by Goel et al. in [17].) Recently, it was shown in [11] that the online (fractional) bipartite matching problem [21] and the more general problem of allocating ad-auctions [24] can also be solved using the primal-dual framework. This yields optimal as well as simple online algorithms for these problems and their extensions.

Acknowledgments. We thank Yossi Azar for many helpful discussions. We would like to thank the anonymous referees for a very thorough reading of the paper and helpful comments.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proceedings of the 35th annual ACM Symposium on the Theory of Computation*, pages 100–105, 2003.

[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.

[3] J. Aspnes, Y. Azar, A. Fiat, S. A. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.

[4] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. In *Proc. of the 7th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 68–74, 1996.

[5] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *Proc. of 34th FOCS*, pages 32–40, 1993.

[6] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. A primal-dual randomized algorithm for weighted paging. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 507–517, 2007.

[7] P. Berman and C. Coulston. On-line algorithms for Steiner tree problems. In *Proc. of the 29th annual ACM Symp. on the Theory of Computation*, pages 344–353, 1997.

[8] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.

[9] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *13th Annual European Symposium on Algorithms*, 2005.

[10] N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *Proceedings of the 47th annual ieee Symposium on Foundations of Computer Science (FOCS 2006)*, 2006.

[11] Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA '07: Proceedings of the 15th Annual European Symposium*, pages 253–264, 2007.

[12] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1001–1010, 2004.

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

[14] N. Garg and R. Khandekar. Fractional covering with upper bounds on the variables: solving LPs with negative entries. In *Proceedings, 12th European Symposium on Algorithms (ESA)*, volume 3221 of *LNCS*, pages 371–382, 2004.

[15] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[16] Naveen Garg and Neal E. Young. On-line end-to-end congestion control. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 303–312, 2002.

[17] A. Goel, A. Meyerson, and S. A. Plotkin. Combining fairness with throughput: online routing with multiple objectives. *J. Comput. Syst. Sci.*, 63(1):62–79, 2001.

[18] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[19] M. Imase and B.M. Waxman. Dynamic Steiner tree problem. *SIAM Journal Discrete Math.*, 4:369–384, 1991.

[20] A. R. Karlin, C. Kenyon, and D. Randall. Dynamic TCP acknowledgement and other stories about e/(e-1). In *ACM Symposium on Theory of Computing*, pages 502–509, 2001.

[21] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *In Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[22] S. G. Kolliopoulos and N. E. Young. Tight approximation results for general covering integer programs. In *IEEE Symposium on Foundations of Computer Science*, pages 522–528, 2001.

[23] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *ACM Symposium on Theory of Computing*, pages 448–457, 1993.

[24] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, 2005.

[25] A. Meyerson. The parking permit problem. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–284, 2005.

[26] S. Plotkin, D. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[27] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comput. Syst. Sci.*, 37(2):130–143, 1988.

[28] P. Raghavan and C.D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[29] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.

[30] N. E. Young. Randomized rounding without solving the linear program. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 170–178, 1995.