Shrinking Maxima, Decreasing Costs: New Online Packing and Covering Problems

Pierre Fraigniaud*
pierre.fraigniaud@liafa.univ-paris-diderot.fr

Magnús M. Halldórsson†
mmh@ru.is

Boaz Patt-Shamir‡
boaz@eng.tau.ac.il

Dror Rawitz‡
rawitz@eng.tau.ac.il

Adi Rosén*
adiro@liafa.univ-paris-diderot.fr

April 17, 2013

Abstract

We consider two new variants of online integer programs. In the packing problem we are given a set of items and a collection of knapsack constraints over these items that are revealed over time in an online fashion. Upon arrival of a constraint we may need to remove several items (irrevocably) so as to maintain feasibility of the solution. Hence, the set of packed items becomes smaller over time. The goal is to maximize the number, or value, of packed items. The problem originates from a buffer-overflow model in communication networks, where items represent information units broken to multiple packets. The other problem considered is online covering: There is a universe we need to cover. Sets arrive online, and we must decide whether we take each set to the cover or give it up, so the number of sets in the solution grows over time. The cost of a solution is the total cost of sets taken, plus a penalty for each uncovered element. This problem is motivated by team formation, where the universe consists of skills, and sets represent candidates we may hire.

The packing problem was introduced in [8] for the special case where the matrix is binary; in this paper we extend the solution to general matrices with non-negative integer entries. The covering problem is introduced in this paper; we present matching upper and lower bounds on its competitive ratio.

*LIAFA, CNRS and University Paris Diderot, France.
†School of Computer Science, Reykjavik University, 103 Reykjavik, Iceland.
‡School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.

1 Introduction

In this paper we study two related online problems based on the classic packing and covering integer programs. The first is a general packing problem called Online Packing Integer Programs (abbreviated opip). In this problem we are given a set of n items and a collection of knapsack constraints over these items. Initially the constraints are unknown and all items are considered packed. In each time step, a new constraint arrives, and the online algorithm needs to remove some items (irrevocably) so as to maintain feasibility of its solution. The goal is to maximize the number, or value, of packed items. Formally, the offline version of the problem we consider is expressed by the following linear integer program ($\mathbb{N}$ denotes the set of non-negative integers):

$$\begin{array}{lll} \max & \sum_{j=1}^{n} b_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \leq c_i & \forall i \\ & x_j \leq p_j & \forall j \\ & x_j \in \mathbb{N} & \forall j \end{array} \tag{PIP}$$

We assume that $A \in \mathbb{N}^{m \times n}$ and $c \in \mathbb{N}^n$. The value of x_j represents the number of copies of item j that are packed, p_j is an upper bound on the number of copies of item j, b_j is the *benefit* obtained by packing item j, and c_i is the *capacity* of the ith constraint. The online character of opip is expressed by the following additional assumptions: (i) knapsack constraints arrive one by one, and (ii) the variables can only be decreased. The special case, where $A \in \{0,1\}^{m \times n}$ and $c = 1^n$ is known as Online Set Packing [8].

An LP-relaxation of (PIP) is obtained by replacing the integrality constraints by $x_j \geq 0$, for every j. Hence, the integral version of the dual of the LP-relaxation is:

$$\begin{array}{lll} \min & \sum_{i=1}^{m} c_i y_i + \sum_{j=1}^{n} p_j z_j & \\ \text{s.t.} & \sum_{i=1}^{m} a_{ij} y_i + z_j \geq b_j & \forall j \\ & y_i \in \mathbb{N} & \forall i \\ & z_j \in \mathbb{N} & \forall j \end{array} \tag{TF}$$

The program (TF) describes the second problem that is considered in this paper, called the Team Formation problem (for reasons that will become apparent below). In this problem we are given n elements, where element j has a covering requirement b_j and a penalty p_j. There are m sets, where the coverage of set i of element j is a_{ij} and its cost is c_i. The solution is a collection of the sets, where multiple copies of sets are allowed. The cost of a solution is the cost of selected sets plus the penalties for unsatisfied covering requirements. In (TF), the value of y_i represents the number of copies of i taken by the solution. Our online version of the Team Formation problem, denoted otf, is as follows. Initially, the elements are uncovered—and hence incur a unit penalty per each unit of uncovered element. Sets with various coverage and cost arrive online. In each time step, a new set arrives, and the algorithm must decide how many copies of the arriving set to add to the solution. The goal is to minimize the total cost of sets taken plus penalties for uncovered elements.

Our main figure of merit, as is customary with online algorithms, is the *competitive ratio*: in the covering case, the ratio of cost incurred by the algorithm (expected cost if the algorithm is randomized) to the best possible cost for the given instance, and in the packing case, the ratio between the benefit earned by the optimum solution to the (expected) benefit earned by the algorithm.

Motivation. The OTF problem is an abstraction of the following situation (corresponding to a binary matrix and binary demands). We are embarking on a new project, which requires some n skills. The requirement for skill j can be satisfied by outsourcing for some cost p_j, or by hiring an employee who possesses skill j. The goal is to minimize the project cost under the following procedure: We interview candidates one by one. After each interview we know what are the skills of the candidate and what is the cost of hiring her, and then we must decide: do we hire the candidate? If we don't, we lose her forever.

The OPIP problem came out of the following natural networking situation [8]. High-level information units, called frames, may be too large to fit in a single network packet, in which case the frames are fragmented to multiple packets. As packets traverse the network, they may arrive at a bottleneck link which cannot deliver them all, giving rise to a basic online question: which packets to drop so as to maximize the number of frames that are completely delivered. If we ignore buffers, this question is precisely our version of OPIP: in each time step i, a burst of packets arrives, which corresponds to the ith constraint in (PIP): a_{ij} is the size of the packet from frame j that arrives at step i, and c_i is the total size that the link can deliver at time i.

Our problems appear unique in the literature of online computation in that solutions get progressively *smaller* with time. Traditionally, the initial solution is expected to be the empty set, and its value or cost only gets larger as the input is progressively presented. In our class of problems, some aspects of the input are known, inducing a naïve initial solution. The presented input progressively elucidates the structure of the instance, adding more constraints (in maximization problems) or providing increasing opportunities for cost reductions or optimizations (in minimization problems). In reality, often the issue is not what to include, but what to keep. We feel that this complementary viewpoint is natural and deserves further treatment.

Contribution and results. The contributions of this paper are twofold. In the conceptual level, to the best of our knowledge we are the first to formalize the OTF problem (the OPIP problem was introduced in [8]).

On the technical level, we present new results for both the OPIP and the OTF problems. For OPIP, we extend the results of [8] from a binary matrix to the case of general non-negative integer demands. This is a useful extension when we consider our motivating network bottleneck scenario: it allows the algorithm to deal with packets of different size, while previous solutions were restricted to uniform-size packets. For the case of unit caps (i.e., $p = 1$), the competitive ratio of our algorithm is $O(C_{\max}\sqrt{\rho_{\max}})$, where $C_{\max}$ the maximal sum of entries in a column, and $\rho_{\max}$ is the maximal ratio of the sum of entries in a row i to its cap c_i. An additional $\sqrt{\max_j p_j}$ factor is incurred for non-unit cap. We remark that the extension is non-trivial, although it uses known techniques.

Regarding OTF, we prove matching upper and lower bounds on the competitive ratio: We show that even randomized algorithms cannot have competitive ratio better than $\Omega(\sqrt{\gamma})$, where γ is the maximal ratio, over all elements, between the highest and lowest cost of covering a given element. On the other hand, we give a simple deterministic algorithm with a competitive ratio of $O(\sqrt{\gamma})$.

Related work. Online packing was studied in the past, but traditionally the elements of the universe (equivalently, the constraints) were given ahead of time and sets arrive on-line (e.g., in [2]). In the similar vein, online set cover was defined in [1] as follows. A collection of sets is given ahead of time. Elements arrive online, and the algorithm is required to maintain a cover of the elements that arrived: if the arriving element is not already covered, then some set from the given

collection must be added to the solution. Our problems have the complementary view of what's known in advance and what arrives online (see also [5]).

As mentioned above, our notion of OPIP was essentially introduced in [8]. Let us first review some results for the off-line packing problem PIP. The single constraint case ($m = 1$) is simply the KNAPSACK problem which is NP-hard and has an FPTAS [20, 16]. If the number of constraints is constant, the offline version of PIP becomes the MULTI-DIMENSIONAL KNAPSACK problem that has a PTAS [11], while obtaining an FPTAS for it is NP-hard [17]. Raghavan and Thompson [19] used randomized rounding to obtain solutions whose benefit is $t_1 = \Omega(\text{OPT}/m^{1/\alpha})$ for PIP, where $\alpha = \min_j \min_i \frac{c_j}{a_{ij}}$. A solution of benefit $t_2 = \Omega(\text{OPT}/m^{1/(\alpha+1)})$ is also given for the case where $A \in \{0,1\}^{m\times n}$. (In this case $\alpha = \min_j c_j$.) Srinivasan [21] improved these results by obtaining solutions whose benefits are $\Omega(t_1^{\alpha/(\alpha-1)})$ and $\Omega(t_2^{\alpha/(\alpha-1)})$. Chekuri and Khanna [6] showed that, for every fixed integer α and fixed $\varepsilon > 0$, PIP with $c = \alpha^m$ and $A \in \{0,1\}^{m\times n}$ cannot be approximated within a factor of $m^{1/(\alpha+1)-\varepsilon}$, unless NP=ZPP. They also showed that PIP with uniform capacities cannot be approximated within a factor of $m^{1/(\alpha+1)-\varepsilon}$, unless NP=ZPP, even with a resource augmentation factor α. (In this case the solution x satisfies $Ax \le \alpha c$.)

As mentioned before, the special case of PIP, where $A \in \{0,1\}^{m\times n}$ and $c = 1^n$ is known as SET PACKING. This problem is as hard as MAXIMUM INDEPENDENT SET even when all elements have degree 2 (i.e., A contains at most two non-zero entries in each row), and therefore cannot be approximated to within a factor of $O(n^{1-\epsilon})$, for any $\varepsilon > 0$ [14]. In terms of the number of elements (constraints, in PIP terms) SET PACKING packing is $O(\sqrt{m})$-approximable, and hard to approximate within $m^{1/2-\varepsilon}$, for any $\varepsilon > 0$ [13]. When set sizes are at most k (A contains at most k non-zero entries in each column), it is approximable to within $(k+1)/3 + \varepsilon$, for any $\varepsilon > 0$ [7], and within $(k+1)/2$ in the weighted case [4], but known to be hard to approximate to within $o(k/\log k)$-factor [15].

OPIP was introduced in [8], assuming that the matrix is binary, namely each set requires either one or zero copies of each element. In [8], a randomized algorithm was given for that case, obtaining competitive ratio of $O(k\sqrt{\sigma})$, where k is the maximal set size and σ is the maximal ratio, over all elements, between the number of sets containing that element to the number of its copies. In OPIP terms this bound is $O(C_{\max}\sqrt{\rho_{\max}})$. A nearly matching lower bound of $\tilde{\Omega}(k\sqrt{\sigma})$ was also given. Subsequent work extended these results to allow for redundancy [18], i.e., when the benefit of a set is earned by the algorithm even if up to a $\beta < 1$ fraction of its elements are not assigned to it.

Previously, the online packing problem where sets arrive online and constraints are fixed was defined in [2]. They give an algorithm with competitive ratio $O(\log n)$ assuming that no set requires more than a $1/\log n$ fraction of the cap of any element. A matching lower bound shows that this requirement is necessary to obtain a polylogarithmic competitive ratio.

Regarding team formation, we are unaware of any prior formalization of the problem, let alone analysis. The online cover problem defined in [1] has an algorithm with competitive ratio $O(\log n \log m)$. Another related problem is the secretary problem (see, e.g., [12, 10]; some more recent results and references can be found in [9, 3]). In this family of problems a n candidates arrive in random order (or with random value), and the goal is to pick k of them (classically, $k = 1$) which optimizes some function of the value set, such as the probability of picking the candidates with the top k values, or the average rank of selected candidates. The difficulty, similarly to our OTF formulation, is that the decision regarding each candidate must be taken immediately upon

her arrival. However, the stipulation that the input is random makes the secretary problem very different from OTF. Another difference is that unlike OTF, the number of candidates to pick is set in advance.

Paper organization. The remainder of this paper is organized as follows. In Section 2 we formalize the model and introduce some notation. In Section 3 we analyze OPIP, and in Section 4 we analyze OTF.

2 Preliminaries

In this section we define our notation. Given a matrix $A \in \mathbb{N}^{m \times n}$, let $R(i) = \sum_j a_{ij}$ be the sum of entries in the ith row, and let $C(j) = \sum_i a_{ij}$ be the sum of entries in the jth column. Denote $R_{\max} = \max_i R(i)$ and $C_{\max} = \max_j C(j)$.

Given an OPIP instance, define $\rho(i) = R(i)/c_i$. Observe that if $\sum_j a_{ij} \leq c_i$ for some i, then constraint i is redundant. Hence we assume w.l.o.g. that $\sum_j a_{ij} > c_i$ for every i, which means that $\rho(i) > 1$, for every i. We assume hereafter that $\text{lcm}(a_{i1}, \ldots, a_{in}, c_i) = 1$, for every i. This does not change $\rho(i)$, but it may decrease $C_{\max}$ and our bound on the competitive ratio. On the other extreme, we assume that $a_{ij} \leq c_i$ for every i and j: if $a_{ij} > c_i$ then item j is not a member in any feasible solution.

Given a subset of items J and a constraint i, let $J(i) = \{j \in J : a_{ij} > 0\}$ be the subset of items from J that participate in constraint i. For example, if OPT is the set of items in some fixed optimal solution, then $\text{OPT}(i)$ denotes the items in OPT that are active in constraint i. Also, let $R_J(i) = \sum_{j \in J} a_{ij}$.

Given a subset of items J, let $b(J) = \sum_{j \in J} b_j$. Also, we define the *normalized benefit* of a constraint i as $\bar{b}(i) = \sum_j a_{ij} \cdot b_j$.

Given an OTF instance, $1/\rho(i)$ stands for the cost per unit of coverage that may be covered by i. We denote $\gamma_j = \max\left\{p_j \cdot \max_{i:a_{ij}>0} \rho(i), 1\right\}$. In other words, γ_j is the ratio between the penalty for not covering a unit of coverage of j and the minimum possible cost per unit of coverage that may be obtained to cover j. Also, denote $\gamma_{\max} = \max_j \gamma_j$.

3 Online Packing Integer Programs

In this section we describe a randomized algorithm for OPIP with unit caps, namely for the case where $p_j = 1$, for every j. The competitive ratio of our algorithm is $2C_{\max}\sqrt{\rho_{\max}}$. We note that the algorithm is a slight generalization of the algorithm given in [8], which allows us to deal with non-binary instances. We note that one may solve the general case by treating each item j as p_j items, but this simplistic approach results in an additional multiplicative factor of $\sqrt{\max_j p_j}$ to the competitive ratio.

Random variables. For $w > 0$, let $D_w : \mathbb{R} \to [0, 1]$ be a (cumulative) distribution function of a random variable Z that is defined by

$$D_w(z) \;=\; \Pr[Z \leq z] \;=\; \begin{cases} 0 & \text{if } z < 0; \\ z^w & \text{if } 0 \leq z < 1; \\ 1 & \text{if } 1 \leq z. \end{cases}$$

Note that D_1 is the uniform distribution over $[0,1]$ and, in general, for a positive integer q, D_q is the distribution of the maximum of q independent and identically distributed variables, each uniformly distributed over $[0,1]$.

Algorithm RP. For each item j, we independently choose a random priority $p(j) \in [0,1]$ with distribution D_{b_j}. When constraint i arrives, we construct c_i subsets $S_{i1}, \ldots, S_{ic_i}$ as follows. Each item j chooses a_{ij} subsets at random. Then, for each subset $S_{i\ell}$, $\ell \in \{1, \ldots, c_i\}$, we reject all items but the one with the highest priority. Observe that an item survives only if it has the highest priority in all of its chosen sets.

Intuitively, the approach is to prefer items with high priority. In the special case where $a_{ij} \in \{0,1\}$, one may simply choose the c_i items with highest priority. A somewhat more subtle approach, based on a reduction to the unit capacity case is used in [8]: Items are randomly partitioned into c_i equal-size subsets; from each subset only the top priority item survives. Our Algorithm RP extends this approach: we construct c_i subsets whose expected sizes are equal, such that item j is contained in exactly a_{ij} subsets.

Analysis. Observe that each subset $S_{i\ell}$ induces the following constraint: $\sum_{j \in S_{i\ell}} x_j \leq 1$. Hence, we construct a new uniform capacity OPIP instance by defining the matrix $A' \in \{0,1\}^{(\sum_i c_i) \times n}$ as follows: $a_{\sum_{t<i} c_k + \ell, j} = 1$ if and only if $j \in S_{i\ell}$. Each row of A' corresponds to one of the random constraints created by the algorithm.

Observation 1. *$C(j) = C'(j)$, for every j, and $\mathbb{E}[R'(\sum_{t<i} c_t + \ell)] = \rho(i)$, for every i and ℓ.*

Proof. $C(j) = C'(j)$, since the item j appears in a_{ij} new constraints with coefficient 1, for every such constraint i. Each item j participates in the ℓth new constraint corresponding to original constraint i with probability a_{ij}/c_i. Hence, $\mathbb{E}[R'(\sum_{t<i} c_t + \ell)] = \sum_j \mathbb{E}[a'_{\sum_{t<i} c_t + \ell, j}] = \sum_i \frac{a_{ij}}{c_i} = \frac{R(i)}{c_i}$. □

Let $N[j]$ denote the items that are in conflict with j, namely $N[j] = \{k : \exists i, \ell \text{ s.t. } j, k \in S_{i\ell}\}$. Notice that $j \in N[j]$. We also define $N(j) = N[j] \setminus \{j\}$. Clearly, item j is satisfied by the algorithm if and only if its priority is higher than that of all other items with whom it competes, i.e., if $p(j) > p(k)$, for every $k \in N(j)$.

First, we consider the probability of satisfying an item j.

Lemma 2. $\Pr[p(j) > \max\{p(k) : k \in N(j)\}] = \mathbb{E}\left[\frac{b_j}{b(N[j])}\right]$.

Proof. Supposed that $N(j) = N$ and let $p_{\max} = \max\{p(k) : k \in N\}$. Then, for $z \in [0,1]$ we have

$$\Pr[p_{\max} < z] \;=\; \prod_{k \in N} \Pr[p(k) < z] \;=\; \prod_{k \in N} z^{b_k} \;=\; z^{\sum_{k \in N} b_k} \;=\; z^{b(N)} \;,$$

that is, $p_{\max}$ has distribution $D_{b(N)}$. Hence,

$$\Pr[p(j) > p_{\max}] \;=\; \int_0^1 \Pr[p_{\max} < z] \cdot f_{p(j)}(z) dz \;=\; \int_0^1 z^{b(N)} \cdot b_j z^{b_j - 1} dz \;=\; \frac{b_j}{b(N) + b_j} \;.$$

It follows that

$$\begin{aligned}\Pr[p(j) > \max\{p(k) : k \in N(j)\}] &= \sum_N \Pr[N(j) = N] \cdot \Pr[p(j) > \max\{p(k) : k \in N\} | N(j) = N] \\ &= \sum_N \Pr[N(j) = N] \cdot \frac{b_j}{b(N) + b_j} \\ &= \mathbb{E}\left[\frac{b_j}{b(N(j)) + b_j}\right]\end{aligned}$$

as required. □

Next, we provide a lower bound on the expected performance of the algorithm.

Lemma 3. *For any subset of items J, $\mathbb{E}[b(\mathrm{RP})] \geq \frac{b(J)^2}{\mathbb{E}\left[\sum_{j\in J} b(N[j])\right]}$.*

Proof. By Lemma 2, $\Pr[j \in \mathrm{RP}] = \mathbb{E}\left[\frac{b_j}{b(N[j])}\right]$. Thus, by linearity of expectation, we obtain

$$\mathbb{E}[b(\mathrm{RP})] = \sum_{j\in J} b_j \cdot \mathbb{E}\left[\frac{b_j}{b(N[j])}\right] = \mathbb{E}\left[\sum_{j\in J} \frac{b_j^2}{b(N[j])}\right] \geq \mathbb{E}\left[\frac{(\sum_{j\in J} b_j)^2}{\sum_{j\in J} b(N[j])}\right] ,$$

where the inequality is due to the following consequence of the Cauchy-Schwarz inequality (with b_j for α_j and $b(N[j])$ for β_j): for positive reals $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$, we have $\sum_j \frac{\alpha_j^2}{\beta_j} \geq \frac{\left(\sum_j \alpha_j\right)^2}{\sum_j \beta_j}$. Jensen's inequality (for a non-negative random variable X, $\mathbb{E}\left[\frac{1}{X}\right] \geq \frac{1}{\mathbb{E}[X]}$) implies that

$$\mathbb{E}[b(\mathrm{RP})] \geq \mathbb{E}\left[\frac{(\sum_{j\in J} b_j)^2}{\sum_{j\in J} b(N[j])}\right] \geq \frac{(\sum_{j\in J} b_j)^2}{\mathbb{E}\left[\sum_{j\in J} b(N[j])\right]} ,$$

and the lemma follows. □

Our next step is to bound $\sum_{j\in J} b(N[j])$.

Lemma 4. *Let J be a subset of items. Then, $\sum_{j\in J} b(N[j]) \leq \sum_{i=1}^{m'} R'_J(i)\bar{b}'(i)$.*

Proof. Observe that

$$\begin{aligned}\sum_{j\in J} b(N[j]) = \sum_{j\in J} \sum_{k\in N[j]} b_k \leq \sum_{j\in J} \sum_{(i,\ell):j\in S_{i\ell}} \sum_{k\in S_{i\ell}} b_k = \sum_{j\in J} \sum_{(i,\ell):j\in S_{i\ell}} b(S_{i\ell}) &= \sum_{i=1}^{m} \sum_{\ell=1}^{c_i} |S_{i\ell} \cap J| \cdot b(S_{i\ell}) \\ &= \sum_{i=1}^{m'} R'_J(i)\bar{b}'(i) ,\end{aligned}$$

as required. □

To complete the analysis we find appropriate upper bounds for the denominator when $J = [n]$ and when $J = \textsc{opt}$.

Lemma 5.

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{[n]}(i)\bar{b}'(i)\right] < 2\sum_{i=1}^{m}\rho(i)\bar{b}(i)\ , \tag{1}$$

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{\text{OPT}}(i)\bar{b}'(i)\right] \leq \sum_{j\in[n]} C(j)b_j + \sum_{j\in\text{OPT}} C(j)b_j \leq 2\sum_{j\in[n]} C(j)b_j\ . \tag{2}$$

Proof. Consider $i' \in [m']$ that corresponds to the ℓth new constraint of original constraint i, and two items $j \neq k$. We have that $\Pr[j,k \in S_{i\ell}] = \Pr[j \in S_{i\ell}] \cdot \Pr[k \in S_{i\ell}] = \frac{a_{ij}}{c_i} \cdot \frac{a_{ik}}{c_i}$, due to the independence of the random choices of j and k. Hence, for $i \in [m]$ we have that

$$\begin{aligned}
\mathbb{E}\left[\sum_{\ell=1}^{c_i} R'_J\left(\sum_{t<i} c_t + \ell\right)\cdot \bar{b}'\left(\sum_{t<i} c_t + \ell\right)\right] &= \sum_{j\in J(i)}\sum_{k}\sum_{\ell=1}^{c_i} b_k \Pr[j,k\in S_{i\ell}] \\
&= \sum_{j\in J(i)}\frac{a_{ij}}{c_i}\sum_{k\neq j} c_i b_k \frac{a_{ik}}{c_i} + \sum_{j\in J(i)} c_i b_j \frac{a_{ij}}{c_i} \\
&\leq \sum_{j\in J(i)}\frac{a_{ij}}{c_i}\cdot\bar{b}(i) + \sum_{j\in J(i)} a_{ij}\cdot b_j \\
&\leq \rho_J(i)\bar{b}(i) + \bar{b}_J(i)\ .
\end{aligned}$$

It follows that

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_J(i)\bar{b}'(i)\right] \leq \sum_i \rho_J(i)\bar{b}(i) + \sum_i \bar{b}_J(i)\ . \tag{3}$$

Since $\rho(i) > 1$, for every i, Inequality (1) is obtained by assigning $J = [n]$ in (3).

To prove Inequality (2) we assign $J = \text{OPT}$. In this case, $\rho_{\text{OPT}}(i) \leq 1$, for every i, since OPT is a feasible solution. Hence

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{m'} R'_{\text{OPT}}(i)\bar{b}'(i)\right] \leq \sum_i \bar{b}(i) + \sum_i \bar{b}_{\text{OPT}}(i) &= \sum_i\sum_j a_{ij}b_j + \sum_i\sum_{j\in\text{OPT}} a_{ij}b_j \\
&= \sum_j b_i\sum_i a_{ij} + \sum_{j\in\text{OPT}} b_j\sum_i a_{ij} \\
&= \sum_j b_j C(j) + \sum_{j\in\text{OPT}} b_j C(j)\ ,
\end{aligned}$$

and the lemma follows. □

Lemma 5 implies that

Theorem 1.

$$\mathbb{E}[b(\text{RP})] \geq \max\left\{\frac{b([n])^2}{2\sum_i \rho(i)\bar{b}(i)}, \frac{b(\text{OPT})^2}{2\sum_j C(j)b_j}\right\} \geq \frac{b([n])b(\text{OPT})}{2\sqrt{\sum_i \rho(i)\bar{b}(i)\cdot\sum_j C(j)b_j}}\ .$$

Theorem 1 implies the following:

Corollary 2. *There is an* OPIP *algorithm with competitive ratio at most* $2C_{\max}\sqrt{\rho_{\max}}$.

Proof. $\sum_i \rho(i)\bar{b}(i) = \rho_{\max}\sum_i\sum_j a_{ij}b_j = \rho_{\max}\sum_j b_j C(j) \leq \rho_{\max} b([n])C_{\max}$, and $\sum_j C(j)b_j \leq C_{\max}b([n])$. Hence,

$$\mathbb{E}[b(\text{RP})] \geq \frac{b([n])b(\text{OPT})}{2\sqrt{\rho_{\max}b([n])C_{\max}\cdot C_{\max}b([n])}} = \frac{b(\text{OPT})}{2C_{\max}\sqrt{\rho_{\max}}} ,$$

and we are done. □

4 Competitive Team Formation

In this section we provide a deterministic online algorithm for OTF and a matching lower bound that holds even for randomized algorithms. Furthermore, our lower bound holds for a more general case, where the commitment of the online algorithm is only "one way" in the following sense. Once a set is dismissed it cannot be recruited again, but a set in the solution at one point may be thrown out of the solution later.

4.1 An Algorithm

Our solution algorithm generates a monotonically growing collection of sets based on a simple deterministic threshold rule. Recall that γ is the maximal ratio, over all elements, between the highest and lowest cost of covering a given element. Algorithm THRESHOLD assumes knowledge of γ and works as follows. Let y be the vector that is constructed by THRESHOLD, and define $z_j = \max\{b_j - \sum_i a_{ij}y_i, 0\}$, namely z_j is the number of missing coverage units for element j (initially, $y = 0$ and hence $z = b$). Upon arrival of a new candidate i, assign $y_i = v$, such that v is the maximum integer that satisfies

$$v \cdot c_i \leq \frac{\sum_j \min\{v \cdot a_{ij}, z_j\} \cdot p_j}{\sqrt{\gamma}} . \tag{4}$$

Intuitively, we take the maximum possible number of units of set i that allows us to save a factor of at least $\sqrt{\gamma}$ over the penalties it replaces. Note that $\min\{va_{ij}, z_j\}$ is the amount of coverage v copies of set i adds to element j. Hence, the total amount of penalties that are saved by v copies of set i is $\sum_j \min\{va_{ij}, z_j\}p_j$. Also notice that v is well defined because (4) is always satisfied by $v = 0$.

Theorem 3. *Algorithm* THRESHOLD *is* $2\sqrt{\gamma}$*-competitive.*

Proof. Let (y^*, z^*) be an optimal (integral) solution. Let (y, z) denote the solution that was computed by THRESHOLD, and let (y^i, z^i) be the solution that is induced by THRESHOLD after the ith step (initially, $y^0 = 0$ and $z^0 = b$).

We first show that $\sum_i c_i y_i \leq \sqrt{\gamma} \cdot \sum_i c_i y_i^* + \sum_j p_j z_j^*$. Using condition (4), we then have that

$$\sum_i c_i y_i \leq \frac{1}{\sqrt{\gamma}} \sum_i \sum_j \min\{a_{ij} y_i, z_j^{i-1}\} \cdot p_j = \frac{1}{\sqrt{\gamma}} \sum_j p_j \sum_i \min\{a_{ij} y_i, z_j^{i-1}\} \leq \frac{1}{\sqrt{\gamma}} \sum_j p_j b_j \ ,$$

where the second inequality follows since $\min\{a_{ij} y_i, z_j^{i-1}\}$ is the amount of coverage that is added to j in the ith round, and therefore the total coverage of j, $\sum_i \min\{a_{ij} y_i, z_j^{i-1}\}$ is at most $b_j - z_j \leq b_j$. On the other hand, by the definition of γ_j we have that $\gamma_j \geq p_j \cdot \rho(i) = p_j \frac{R(i)}{c_i}$, for any i such that $a_{ij} > 0$. Hence,

$$\sum_i c_i y_i^* = \sum_i \frac{c_i}{R(i)} y_i^* \sum_j a_{ij} \geq \sum_i \sum_j y_i^* \frac{p_j}{\gamma_j} a_{ij} \geq \frac{1}{\gamma} \sum_j p_j \sum_i y_i^* a_{ij} \geq \frac{1}{\gamma} \sum_j p_j (b_j - z_j^*) \ .$$

It follows that

$$\sum_i c_i y_i^* + \sum_j p_j z_j^* \geq \frac{1}{\gamma} \sum_j p_j (b_j - z_j^*) + \sum_j p_j z_j^* \geq \frac{1}{\gamma} \sum_j p_j b_j \geq \frac{1}{\sqrt{\gamma}} \sum_i c_i y_i \ .$$

Now, we turn to bound the penalties that (y, z) pays and (y^*, z^*) does not pay, namely we bound $\sum_j p_j \max\{z_j - z_j^*, 0\}$. Define $\Delta_i = \max\{y_i^* - y_i, 0\}$. If $\Delta = 0$, then $z_j \leq z_j^*$, for every j, and we are done. Otherwise, let i be an index such that $\Delta_i > 0$. Due to condition (4) in the ith step, we have that

$$c_i y_i \leq \frac{\sum_j \min\{a_{ij} y_i, z_j^{i-1}\} \cdot p_j}{\sqrt{\gamma}} \qquad \text{while} \qquad c_i y_i^* > \frac{\sum_j \min\{a_{ij} y_i^*, z_j^{i-1}\} \cdot p_j}{\sqrt{\gamma}} \ .$$

Observe that j's coverage increases by $\min\{a_{ij} y_i, z_j^{i-1}\} = z_j^i - z_j^{i-1}$ in the ith step. If we further increase y_i to y_i^* we may gain $\min\{\Delta_i a_{ij}, z_j^i\}$ additional coverage for item j. Hence,

$$c_i \Delta_i = c_i y_i^* - c_i y_i > \frac{\sum_j \min\{a_{ij} \Delta_i, z_j^i\} \cdot p_j}{\sqrt{\gamma}} \geq \frac{\sum_j \min\{a_{ij} \Delta_i, z_j\} \cdot p_j}{\sqrt{\gamma}} \ .$$

It follows that

$$\sqrt{\gamma} \sum_i c_i \Delta_i > \sum_i \sum_j \min\{a_{ij} \Delta_i, z_j\} \cdot p_j \geq \sum_j p_j \min\left\{ \sum_i a_{ij} \Delta_i, z_j \right\} \geq \sum_j p_j \max\{z_j - z_j^*, 0\} \ ,$$

where the last inequality follows from the fact that $y + \Delta \geq y^*$ and therefore Δ covers at least $\max\{z_j - z_j^*, 0\}$, for every j. Hence,

$$\sum_j p_j \max\{z_j - z_j^*, 0\} \leq \sqrt{\gamma} \sum_i c_i \Delta_i \leq \sqrt{\gamma} \sum_i c_i y_i^* \ .$$

Putting it all together, we get that

$$\begin{aligned} \sum_i c_i y_i + \sum_j p_j z_j \ &\leq \ \sum_i c_i y_i + \sum_j p_j z_j^* + \sum_j p_j \max\{z_j - z_j^*, 0\} \\ &\leq \ 2\sqrt{\gamma} \sum_i c_i y_i^* + (\sqrt{\gamma} + 1) \sum_j p_j z_j^* \ , \end{aligned}$$

as required. □

4.2 A Lower Bound

We present a matching lower bound, which holds for randomized algorithm, and even for the case where the algorithm may discard a set from its running solution (but never take back a set that was dismissed).

Theorem 4. *The competitive ratio of any randomized algorithms for* OTF *is in* $\Omega(\sqrt{\gamma})$. *The bound holds also in the binary case, where all demands, coverages, penalties and costs are either* 0 *or* 1.

Proof. Our lower bound construction uses affine planes defined as follows. Let $n = q^2$, where q is prime. In our construction, each pair $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$ corresponds to an element. Sets will correspond to lines: a line in this finite geometry is a collection of pairs $(x, y) \in \mathbb{Z}_q \times \mathbb{Z}_q$ satisfying either $y \equiv ax + b \pmod{q}$, for some given $a, b \in \mathbb{Z}_q$, or of the form $(c, *)$ for some given $c \in \mathbb{Z}_q$. There are $q^2 + q = \Theta(n)$ such lines. The important properties we use are the following.

1. All points can be covered by q disjoint (parallel) lines.
2. Two lines that intersect in more than a single point are necessarily identical.

We now describe the lower bound scenario. The elements correspond (in a 1-1 fashion) to the points in the affine plane. All elements have unit penalty and unit covering requirement, i.e., $p_j = 1$ and $b_j = 1$, for every j. The input sequence starts with a sequence of $q^2 + q$ sets corresponding to all distinct lines of the plane, each with unit cost. Fix any deterministic online algorithm ALG. We proceed by cases, depending on the expected number r of these sets that ALG retains at this point. If $r \leq \sqrt{n}/2$ or $r > n/2$, then we are already done: at this time the cost to the algorithm is $\Omega(n)$ (due either to penalties or to the cost of sets retained), while the optimal cost at this time is $\sqrt{n}$ by virtue of Property (1) above.

Otherwise, $\sqrt{n}/2 < r \leq n/2$. Let L be a line chosen uniformly at random. The probability that L is retained by the algorithm is at most $1/2$, since $r \leq n/2$. We now extend the input sequence by one more set $L^c \stackrel{\text{def}}{=} \{1, ..., n\} \setminus L$, and assign L_c unit cost. Note that by Property (2), if L is not retained by the algorithm, then the number of other lines that cover the points of L cannot be smaller than $|L| = \sqrt{n}$, and hence the expected cost of ALG due only to the points of L (either by covering set costs or by incurred penalties) is at least $\sqrt{n}/2$. Obviously, throwing out any set from the solution at this time will not help to reduce the cost. On the other hand, the optimal solution to this scenario is the sets L and L^c, whose cost is 2, and hence the competitive ratio is at least $\Omega(\sqrt{n})$. □

Remarks. First, we note that in the proof above, the unit-cost set L^c can be replaced by $\sqrt{n} - 1$ sets, where each set covers $\sqrt{n}$ elements and costs $\frac{1}{\sqrt{n}-1}$. Second, we note that one may be concerned that in the first case, the actual γ of the instance is not n. This can be easily remedied as follows. Let the instance consist of $2n$ elements: n elements in the affine plane as in the proof, and another n dummy elements. The dummy elements will be all covered by a single set that arrives first in the input sequence. The remainder of the input sequence is as in the proof. This allows us to argue that the *actual* γ is indeed n, whatever the ensuing scenario is, while decreasing the lower bound by no more than a constant factor.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[2] B. Awerbuch, Y. Azar, and S. A. Plotkin. Throughput-competitive on-line routing. In *Proc. 34th FOCS*, pages 32–40, 1993.

[3] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam. Submodular secretary problem and extensions. In *Proc. APPROX'10*, pages 39–52, 2010.

[4] P. Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. *Nord. J. Comput.*, 7(3):178–184, 2000.

[5] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

[6] C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.

[7] M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. arXiv report 1304.1424, April 2013.

[8] Y. Emek, M. M. Halldórsson, Y. Mansour, B. Patt-Shamir, J. Radhakrishnan, and D. Rawitz. Online set packing. *SIAM Journal on Computing*, 41(4):728–746, 2012.

[9] M. Feldman, J. S. Naor, and R. Schwartz. Improved competitive ratios for submodular secretary problems. In *Proc. APPROX '11*, pages 218–229, 2011.

[10] P. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

[11] A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional $0-1$ knapsack problem: worst-case and probabilistic analyses. *Eur. J. Oper. Res.*, 15:100–109, 1984.

[12] J. P. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *J. Amer. Statist. Assoc.*, 61:35–73, 1966.

[13] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle. Independent sets with domination constraints. *Discrete Applied Mathematics*, 99(1–3):39–54, 2000.

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[15] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, 2006.

[16] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[17] M. J. Magazine and M.-S. Chern. A note on approximation schemes for multidimensional knapsack problems. *Math. Oper. Res.*, 9(2):244–247, 1984.

[18] Y. Mansour, B. Patt-Shamir, and D. Rawitz. Overflow management with multipart packets. In *IEEE INFOCOM*, 2011.

[19] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[20] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22(1):115–124, 1975.

[21] A. Srinivasan. Improved approximations of packing and covering problems. In *27th Annual ACM Symposium on the Theory of Computing*, pages 268–276, 1995.