Algorithmica (2016) 74:1205–1223
DOI 10.1007/s00453-015-9995-8

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

Pierre Fraigniaud[1] · Magnús M. Halldórsson[2] · Boaz Patt-Shamir[3] · Dror Rawitz[4] · Adi Rosén[1]

Received: 7 November 2013 / Accepted: 27 March 2015 / Published online: 4 April 2015

Abstract We consider two new variants of online integer programs that are duals. 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

Boaz Patt-Shamir and Dror Rawitz were supported in part by the Israeli Ministry of Science Technology, and Space.

A preliminary version was presented at the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 2013.

✉ Dror Rawitz
dror.rawitz@biu.ac.il

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

Magnús M. Halldórsson
mmh@ru.is

Boaz Patt-Shamir
boaz@eng.tau.ac.il

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

1 LIAFA, CNRS, University Paris Diderot, Paris, France

2 ICE-TCS, School of Computer Science, Reykjavik University, 101 Reykjavík, Iceland

3 School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel

4 Faculty of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel

information units broken into multiple packets. The other problem considered is online covering: there is a universe to be covered. Sets arrive online, and we must decide for each set whether we add it to the cover or give it up. The cost of a solution is the total cost of sets taken, plus a penalty for each uncovered element. The number of sets in the solution grows over time, but its cost goes down. 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 Emek et al. (SIAM J Comput 41(4):728–746, 2012) 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.

Keywords Competitive analysis · Randomized algorithm · Packing integer programs · Online set packing · Team formation · Prize-collecting multi-covering

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):

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

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 a *cap* (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: (1) knapsack constraints arrive one by one, and (2) 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 $x_j \ge 0$, for every j. It follows that 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 offline version of the second problem 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,[1] 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, and z_j is the amount of unsatisfied coverage of set j (for which we pay penalty).

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 measure, 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 requirements). We are embarking on a new project that 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 the skills and the hiring cost of the candidate and must then decide irrevocably whether to hire the candidate.

The OPIP problem originates from the following natural networking situation [8]. High-level information units, called *frames*, can be too large to fit in a single network packet, in which case the frames are fragmented into multiple packets. As packets traverse the network, they may arrive at a bottleneck link that 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 delivered in full. If we ignore buffers, this question is precisely our version of OPIP. Namely, in each time step i, a burst of packets arrives, corresponding 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.

1 We misuse the term "set" for simplicity.

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 increases 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, the issue is often less what to include than 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. On the conceptual level, we are the first to formalize the OTF problem, to the best of our knowledge (the OPIP problem was introduced in [8]). On the technical level, we present nearly tight 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. 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 load on constraint i, namely $\sum_j p_j a_{ij}$, to its capacity c_i. Observe that for the case of unit caps (i.e., $p = 1$), $\rho_{\max}$ is the sum of entries in a row i to its capacity c_i. 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 $\Omega(\sqrt{\rho_{\max}})$, where $\rho_{\max}$ is the maximal ratio, over all elements, between the highest and lowest cost of covering a given element. This result holds even for the case where the algorithm may discard a set from its running solution (but never takes back a set that was dismissed). On the other hand, we give a simple deterministic algorithm with a competitive ratio of $O(\sqrt{\rho_{\max}})$. The algorithm requires prior knowledge of the value of $\rho_{\max}$; we show that without such knowledge only the trivial $O(\rho_{\max})$ bound is possible.

We note that our techniques can be used for the variant of OTF in which y_i is bounded (e.g., there is only one copy of a given candidate).

Related Work. Online packing was studied in the past, but traditionally the elements of the universe (equivalently, the constraints) are given ahead of time and sets arrive on-line (e.g., in [2]). In a 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 is known in advance and what arrives online (see also [5]).

Let us first review some results for the offline packing problem PIP. The single constraint case ($m = 1$) is simply the KNAPSACK problem, which is NP-hard and has an FPTAS [17,21]. 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 is NP-hard [18]. Raghavan and Thompson [20] 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 [22] 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 $\varepsilon > 0$ [15]. In terms of the number of elements (constraints, in PIP terms), SET 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 [16].

OPIP was introduced in [8], assuming that the matrix is binary, namely each set requires either one or zero copies of each item. A randomized algorithm was given for that case with a competitive ratio of $O(k\sqrt{\nu})$, where k is the maximal set size and ν is the maximal ratio, over all items, between the number of sets containing that item to the number of its copies. In OPIP terms this bound is $O(C_{\max}\sqrt{\rho_{\max}})$. A nearly matching lower bound of $\tilde{\Omega}(k\sqrt{\nu})$ was also given for the unit capacities case. This translates to an $\tilde{\Omega}(C_{\max}\sqrt{\rho_{\max}})$ lower bound for OPIP. Subsequent work extended these results to allow for redundancy [19], i.e., when the benefit of a set is earned when at least a β-fraction of its elements are assigned to it, for some fixed $\beta > 0$. For the special case of unit capacity OPIP in which the constraint matrix has the consecutive ones property, a deterministic $O(\log R_{\max})$-competitive algorithm was given in [14], where $R_{\max}$ the maximal sum of entries in a row, as well as a matching lower bound.

Previously, the online packing problem where *sets* arrive online and constraints are fixed was defined in [2], and an $O(\log n)$-competitive algorithm given for the case when each set requires at most 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., [10,12]; further results and references can be found in [3,9]). In this family of problems, n candidates arrive in random order (or with random value), and the goal is to pick k of them (classically, $k = 1$) that optimize some function of the set, such as the probability of picking the candidates with the top k values, or the average rank of selected candidates. The difficulty, similar to our OTF formulation, is that the decision must be taken immediately upon the candidate's 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 Sect. 2 we introduce some notation. In Sect. 3 we describe and analyze our online algorithm for OPIP, and in Sect. 4 we consider 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 $C_{\max} = \max_j C(j)$. Define $\rho(i) = (\sum_j p_j a_{ij})/c_i$, for every i, and $\rho_{\max} = \max_i \rho(i)$.

Observe that if $\sum_j p_j a_{ij} \le c_i$ for some i, in an OPIP instance, then constraint i is redundant. Hence, we assume w.l.o.g. that $\sum_j p_j a_{ij} > c_i$ for every i, which means that $\rho(i) > 1$, for every i.

We assume hereafter that $\gcd(a_{i1}, \ldots, a_{in}, c_i) = 1$, for every i. Otherwise, we may divide $a_{i1}, \ldots, a_{in}$, and c_i by this common factor. 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 $a_{ij} \le c_i$ for every i and j: if $a_{ij} > c_i$ then item j is not a member in any feasible solution.

Given a subset J of items 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 OPT(i) denotes the items in OPT that are active in constraint i. Also, let $R_J(i) = \sum_{j\in J} a_{ij}$, and define the *weighted benefit* of a constraint i as $wb(i) = \sum_j a_{ij} \cdot b_j$.

Given an OTF instance, $R(i) = \sum_j a_{ij}$ is the coverage potential of a single copy of set i, and $\sum_j p_j a_{ij}$ is the potential savings in penalties of a single copy of set i. Hence, $\rho(i)$ is the ratio between the savings and cost of set i, namely it is the *cost effectiveness* of set i. Observe that we may assume that $\rho(i) > 1$, since otherwise we may ignore the set. Intuitively, the cheapest possible way to cover the elements is by sets with maximum cost effectiveness. Hence, ignoring the sets and simply paying the penalties (i.e., the solution $y = 0$ and $z = b$) is a $\rho_{\max}$-approximate solution.

3 Online Packing Integer Programs

In this section we present a randomized algorithm for OPIP whose competitive ratio is $2C_{\max}\sqrt{\rho_{\max}}$. We describe an algorithm for OPIP with unit caps, namely for the case where $p_j = 1$, for every j, that is a slight generalization of the algorithm given in [8], allowing us to deal with non-binary instances. We solve the general case by simply treating each item j as p_j items, namely by duplicating the jth column p_j times. Observe that this transformation does not change $C_{\max}$ or $\rho_{\max}$.

For the rest of this section we assume that unit item upper bounds, namely that $p = 1$. In particular, we assume that $\rho(i) = R(i)/c_i$, for every i.

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. Initially, we independently choose for each item j a random priority $r(j) \in [0, 1]$ with distribution D_{b_j}. When constraint i arrives, we construct c_i subsets $S_{i1}, \dots, S_{ic_i}$ as follows. Each item j chooses a_{ij} subsets at random. Then, for each subset $S_{i\ell}$, $\ell \in \{1, \dots, c_i\}$, 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.

Example 1 Supposed that the instance contains four items whose priorities are $r(1) = 0.5$, $r(2) = 0.8$, $r(3) = 0.4$, and $r(4) = 0.9$. Upon arrival of the ith constraint: $x_1 + 3x_2 + 2x_3 + 2x_4 \leq 4$, Algorithm RP constructs $c_i = 4$ random subsets: $S_{i1} = \{1, 3\}$, $S_{i2} = \{2, 3, 4\}$, $S_{i3} = \{2, 4\}$, and $S_{i4} = \{2\}$. Item 2 is eliminated due to S_2 and S_3, while Item 3 is eliminated due to S_1 and S_2. Items 1 and 4 are not eliminated by this constraint.

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 use a variation of 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, the algorithm implicitly constructs 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 generated by the algorithm. See example in Fig. 1.

In what follows we use m' to denote the number of rows in A', namely $m' = \sum_i c_i$. Also, we use $R'(i)$ to denote $\sum_j a'_{ij}$, $C'(j)$ to denote $\sum_{i=1}^{m'} a'_{ij}$, and so forth. See example in Fig. 2. Notice that b remains the same, since the item set did not change. However, the weighted benefit of a new constraint i is $wb'(i) = \sum_j a'_{ij} \cdot b_j$. Since A' is binary, $wb'(i)$ is the sum of benefits that correspond to variables that appear in new constraint i.

$$x_1 + 3x_2 + 2x_3 + 2x_4 \le 4 \quad \Longrightarrow \quad \begin{array}{ccccccccc} x_1 & + & & & x_3 & & & \le 1 \\ & & x_2 & + & x_3 & + & x_4 & \le 1 \\ & & x_2 & + & & & x_4 & \le 1 \\ & & x_2 & & & & & \le 1 \end{array}$$

Fig. 1 The inequalities that are induced by the sets in Example 1

$$A = \begin{pmatrix} & \vdots & & \\ 1 & 3 & 2 & 2 \\ & \vdots & & \end{pmatrix} \text{Row } i \quad \Longrightarrow \quad A' = \begin{pmatrix} \vdots & & & \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ \vdots & & & \end{pmatrix} \begin{array}{l} \text{Row } \sum_{j<i} c_j + 1 \\ \\ \\ \text{Row } \sum_{j\le i} c_j \end{array}$$

Fig. 2 The rows of A' that correspond to the inequalities that are given in Fig. 1. In this case we have that $R'(\sum_{j<i} c_j + 1) = 2$

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,

$$\mathbb{E}\left[R'\left(\sum_{t<i} c_t + \ell\right)\right] = \sum_j \mathbb{E}\left[a'_{\sum_{t<i} c_t + \ell, j}\right] = \sum_i \frac{a_{ij}}{c_i} = \frac{R(i)}{c_i} = \rho(i),$$

where the last equality holds in the unit caps case. □

Let $N[j]$ denote the items that are in conflict with item 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 $r(j) > r(k)$, for every $k \in N(j)$.

First, consider the probability of satisfying an item j.

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

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

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

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

$$\begin{aligned}\Pr[r(j) > r_{\max}] = \int_0^1 \Pr[r_{\max} < z] \cdot f_{r(j)}(z)dz \;&=\; \int_0^1 z^{b(N)} \cdot b_j z^{b_j - 1} dz \\ = \frac{b_j}{b(N) + b_j},&\end{aligned}$$

where $f_{r(j)}$ denotes the probability density function of the random variable $r(j)$. It follows that

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

as required. □

Next, we provide a lower bound on the expected performance of Algorithm RP. We abuse notation by referring to the output of the algorithm by RP, as well.

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

Proof By Lemma 2 and by linearity of expectation we obtain

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

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, $\mathbb{E}\left[\frac{1}{X}\right] \geq \frac{1}{\mathbb{E}[X]}$) then implies that

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

and the lemma follows. □

Our next step is to bound $\sum_{j\in J} b(N[j])$. Recall that, since A' is binary, $wb'(i)$ is the sum of benefits that appear in new constraint i. Hence, if j appears in new constraint i, its weighted competition is at most $wb'(i)$.

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

Proof Observe that

$$\begin{aligned}
\sum_{j\in J} b(N[j]) &= \sum_{j\in J}\sum_{k\in N[j]} b_k && (1)\\
&\le \sum_{j\in J}\sum_{(i,\ell):j\in S_{i\ell}}\sum_{k\in S_{i\ell}} b_k && (2)\\
&= \sum_{j\in J}\sum_{(i,\ell):j\in S_{i\ell}} b(S_{i\ell}) && (3)\\
&= \sum_{i=1}^{m}\sum_{\ell=1}^{c_i} |S_{i\ell}\cap J|\cdot b(S_{i\ell})\\
&= \sum_{i=1}^{m'} R'_J(i)\cdot wb'(i)\ ,
\end{aligned}$$

where (1) and (3) are by definition, and (2) is since there can be more than one collision. □

To complete the analysis we derive appropriate upper bounds for the denominator when $J = [n]$ and when $J = \mathrm{OPT}$.

Lemma 5

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{[n]}(i)\bar{b}'(i)\right] < 2\sum_{i=1}^{m}\rho(i)\cdot wb(i)\ , \quad (4)$$

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{\mathrm{OPT}}(i)\bar{b}'(i)\right] \le \sum_{j\in[n]} C(j)b_j + \sum_{j\in\mathrm{OPT}} C(j)b_j \le 2\sum_{j\in[n]} C(j)b_j\ . \quad (5)$$

Proof Consider $i' \in [m']$ that corresponds to the ℓth new constraint of original constraint i, and two items $j \ne k$. 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

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

It follows that

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_J(i) \cdot wb'(i)\right] \le \sum_i \rho_J(i) \cdot wb(i) + \sum_i wb_J(i). \tag{6}$$

Since $\rho(i) > 1$, for every i, Inequality (4) is obtained by assigning $J = [n]$ in (6).

To prove Inequality (5) we assign $J = \text{OPT}$. In this case, $\rho_{\text{OPT}}(i) \le 1$, for every i, since OPT is a feasible solution. Hence

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

and the lemma follows. □

Lemma 5 implies that

Theorem 1

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

Theorem 1 implies the following:

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

Proof By definition,

$$\sum_i \rho(i) \cdot wb(i) \le \rho_{\max} \sum_i \sum_j a_{ij} b_j = \rho_{\max} \sum_j b_j C(j) \le \rho_{\max} b([n]) C_{\max},$$

and

$$\sum_j C(j) b_j \le C_{\max} b([n]).$$

Hence, it follows from Theorem 1 that

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

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 Online Algorithm

Our algorithm generates a monotonically growing collection of sets based on a simple deterministic threshold rule. Recall that $\rho_{\max}$ is the maximum cost effectiveness, over all sets. Algorithm THRESHOLD assumes knowledge of $\rho_{\max}$ and works as follows. Let y be the set vector constructed by THRESHOLD, and define $z_j^i = \max\left\{b_j - \sum_{\ell \le i} a_{\ell j} y_\ell, 0\right\}$, i.e., z_j^i is the amount of missing coverage for element j after the introduction of set i. Note that z_j^i is monotone non-increasing with i.

The solution is constructed as follows. Upon arrival of a new candidate i, assign $y_i \leftarrow v$, where v is the maximum integer that satisfies

$$v \cdot c_i \le \frac{\sum_j \min\left\{v \cdot a_{ij}, z_j^{i-1}\right\} \cdot p_j}{\sqrt{\rho_{\max}}}. \tag{7}$$

Intuitively, we take the maximum possible number of units of set i that allows us to save a factor of at least $\sqrt{\rho_{\max}}$ over the penalties it replaces. Note that $\min\{va_{ij}, z_j^{i-1}\}$ is the amount of coverage that v copies of set i add to element j. Hence, the total amount of penalties that are saved by v copies of set i is $\sum_j \min\{va_{ij}, z_j^{i-1}\}p_j$. Also notice that v is well-defined because (7) is always satisfied by $v = 0$.

We show that the competitive ratio of THRESHOLD is at most $2\sqrt{\rho_{\max}} - 1$.

Theorem 3 *Let* (y, z) *be the solution computed by Algorithm* THRESHOLD, *and let* (y^*, z^*) *be an optimal (integral) solution. Then,*

$$\sum_i c_i y_i + \sum_j p_j z_j \;\le\; \left(2\sqrt{\rho_{\max}} - 1\right) \sum_i c_i y_i^* + \sum_j p_j z_j^*.$$

Proof We first bound $\sum_i c_i y_i$. By condition (7),

$$\begin{aligned}\sum_i c_i y_i &\le \frac{1}{\sqrt{\rho_{\max}}} \sum_i \sum_j \min\left\{a_{ij} y_i, z_j^{i-1}\right\} \cdot p_j \\ &= \frac{1}{\sqrt{\rho_{\max}}} \sum_j p_j \sum_i \min\left\{a_{ij} y_i, z_j^{i-1}\right\} \\ &\le \frac{1}{\sqrt{\rho_{\max}}} \sum_j p_j (b_j - z_j),\end{aligned}$$

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 $b_j - z_j$.

On the other hand, since $\rho(i) = (\sum_j p_j a_{ij}/c_i)$, for every i, we have that

$$\begin{aligned}\sum_i c_i y_i^* &= \sum_i \frac{1}{\rho(i)} y_i^* \sum_j p_j a_{ij} \;\ge\; \frac{1}{\rho_{\max}} \sum_j p_j \sum_i y_i^* a_{ij} \\ &\ge \frac{1}{\rho_{\max}} \sum_j p_j \left(b_j - z_j^*\right).\end{aligned}$$

It follows that

$$\begin{aligned}\sum_i c_i y_i &\le \frac{1}{\sqrt{\rho_{\max}}} \sum_j p_j (b_j - z_j) \\ &= \frac{1}{\sqrt{\rho_{\max}}} \left(\sum_j p_j (b_j - z_j^*) + \sum_j p_j z_j^* - \sum_j p_j z_j \right) \\ &\le \sqrt{\rho_{\max}} \sum_i c_i y_i^* + \frac{1}{\sqrt{\rho_{\max}}} \sum_j p_j \left(z_j^* - z_j\right).\end{aligned}$$

Next, 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

$$\Delta_i = \max\left\{y_i^* - y_i, 0\right\}.$$

If $\Delta = 0$, then $z_j \le z_j^*$, for every j, and we are done. Otherwise, let i be an index such that $\Delta_i > 0$. Due to condition (7) in the ith step, we have that

$$c_i y_i \le \frac{\sum_j \min\left\{a_{ij} y_i, z_j^{i-1}\right\} \cdot p_j}{\sqrt{\rho_{\max}}}$$

while

$$c_i y_i^* > \frac{\sum_j \min\{a_{ij} y_i^*, z_j^{i-1}\} \cdot p_j}{\sqrt{\rho_{\max}}}.$$

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,

$$c_i \Delta_i \;=\; c_i y_i^* - c_i y_i \;>\; \frac{\sum_j \min\{a_{ij} \Delta_i, z_j^i\} \cdot p_j}{\sqrt{\rho_{\max}}} \;\ge\; \frac{\sum_j \min\left\{a_{ij} \Delta_i, z_j\right\} \cdot p_j}{\sqrt{\rho_{\max}}}.$$

It follows that

$$\begin{aligned} \sqrt{\rho_{\max}} \sum_i c_i \Delta_i &> \sum_i \sum_j \min\left\{a_{ij} \Delta_i, z_j\right\} \cdot p_j \\ &\ge \sum_j p_j \min\left\{\sum_i a_{ij} \Delta_i, z_j\right\} \\ &\ge \sum_j p_j \max\left\{z_j - z_j^*, 0\right\}, \end{aligned}$$

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

$$\sum_j p_j \max\left\{z_j - z_j^*, 0\right\} \;\le\; \sqrt{\rho_{\max}} \sum_i c_i \Delta_i \;\le\; \sqrt{\rho_{\max}} \sum_i c_i y_i^*.$$

Putting it all together, we get that

$$\sum_i c_i y_i + \sum_j p_j z_j \le \sum_i c_i y_i + \sum_j p_j z_j^* + \sum_j p_j \max\{z_j - z_j^*, 0\}$$

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

as required. □

This leads us to an upper bound on the competitive ratio.

Corollary 4 *Algorithm* THRESHOLD *is* $(2\sqrt{\rho_{\max}} - 1)$*-competitive.*

We note that the same approach would work for the variant of OTF in which there is an upper bound u_i on the number of copies of set i that can be used, i.e., $y_i \leq u_i$. In this case the value of v in condition (7) is also bounded by u_i. The rest of the details are omitted.

4.2 A Lower Bound

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

We start with a couple of simple constructions. In the first construction, the input consists of sets of size one, and in the second all costs and penalties are the same.

Theorem 5 *The competitive ratio of any randomized online algorithm for* OTF *is* $\Omega(\sqrt{\rho_{\max}})$*. This bound holds for inputs with only two elements and sets of size one, with unit coverage and uniform penalties.*

Proof Let ALG be a randomized algorithm. Consider an input sequence consisting of two elements with unit covering requirement and penalty p. The arrival sequence is composed of two or three sets. The first set to arrive is $\{1\}$ of cost 1. (The goal of the first set is to make sure that the ratio between the penalty and the minimum cost is p.) The second set is $\{2\}$ of cost $\sqrt{p}$. If ALG takes this set with probability less than half, then the sequence ends; otherwise, the third set $\{2\}$ of cost 1 arrives.

In the first case the optimal cost is $1 + \sqrt{p}$, while ALG pays at least $1 + \frac{1}{2}p$. Otherwise, the optimal cost is 2, while ALG pays at least $1 + \frac{1}{2}\sqrt{p}$. Notice that we may repeat the second part of this sequence as many times as needed. Finally, notice that $\rho_{\max} = p$. □

Theorem 6 *The competitive ratio of any randomized online algorithm for* OTF *is* $\Omega(\sqrt{\rho_{\max}})$*. This bound holds for inputs with unit costs and penalties.*

Proof Let ALG be a randomized algorithm. Assume unit penalties and unit coverage requirements. Consider the input sequence that starts with $\sqrt{n}$ candidates, each with $\sqrt{n}$ fresh skills and cost 1. Let ℓ be the expected number of candidates ALG takes from this sequence. If $\ell < \sqrt{n}/2$, this is the whole input. In this case the expected cost of ALG is at least $\frac{1}{2}n$, whereas the optimal cost is $\sqrt{n}$. If $\ell \geq \frac{1}{2}\sqrt{n}$, then we add an omnipotent candidate (who has all skills) at the end, with cost 1. It follows that ALG pays at least $\frac{1}{2}\sqrt{n}$ in expectation, while OPT pays only 1. Finally, notice that $\rho_{\max} = n$. □

Next, we give a lower bound construction that applies to the more general setting in which the algorithm may discard a set from its solution.

Theorem 7 *The competitive ratio of any randomized online algorithm for* OTF *is* $\Omega(\sqrt{\rho_{\max}})$. *This bound holds even if the algorithm is allowed to discard sets. Furthermore, it 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 randomized 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 $L^c \overset{\text{def}}{=} \{1, \ldots, n\} \setminus L$, 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 $\rho_{\max}$ 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* $\rho_{\max}$ is indeed n, whatever the ensuing scenario is, while decreasing the lower bound by no more than a constant factor.

The above theorems hold even if $\rho_{\max}$ is known to the algorithm. However, if $\rho_{\max}$ is unknown, and discarding sets is not allowed, then we get a stronger lower bound.

Theorem 8 *The competitive ratio of any randomized online algorithm for* OTF *is* $\Omega(\rho_{\max})$*, if the algorithm cannot discard sets and has no knowledge of* $\rho_{\max}$*. It holds even in the case of unit penalties, demands and coverage.*

Proof Let ALG be a randomized algorithm. Suppose for the sake of deriving a contradiction that there is an arbitrarily slow growing invertible function h such that ALG has competitive ratio at most $\rho_{\max}/h(\rho_{\max})$.

For every sufficiently large x, we shall construct an instance I with $\rho_{\max} = \rho_{\max}(I) \geq x$ for which the performance ratio of ALG is at least $2\rho_{\max}/h(\rho_{\max})$. This contradicts the assumption of the competitive ratio of ALG, implying the theorem.

Let x be value satisfying $h(x) \geq 4$ and let $f(x) = h(x)/4$. The instance we construct as follows has only one element with unit penalty. A set arrives with cost $1/x$. If ALG takes the set with probability less than $\frac{1}{2}$, we stop. Otherwise, we present a second set with cost $1/f^{-1}(x)$.

In the former case, $\rho_{\max} = x$, the expected cost of ALG is at least $\frac{1}{2}$, and OPT pays $1/x$, for a competitive ratio of

$$\frac{\mathbb{E}[\text{ALG}]}{\text{OPT}} \geq \frac{1/2}{1/x} = \frac{\rho_{\max}}{2} \geq \frac{2\rho_{\max}}{h(\rho_{\max})}.$$

In the latter case, $\rho_{\max} = f^{-1}(x)$, which means that $x = f(\rho_{\max})$. The expected cost of ALG is at least $1/(2x) = 1/(2f(\rho_{\max}))$, while the cost of OPT is $1/f^{-1}(x) = 1/\rho_{\max}$. The performance ratio is then at least $\rho_{\max}/(2f(\rho_{\max})) \geq 2\rho_{\max}/h(\rho_{\max})$. In both cases, we obtain a contradiction, implying the theorem. □

It follows that when $\rho_{\max}$ is unknown, it will not be possible to obtain an $O(\sqrt{\rho_{\max}})$-competitive algorithm without the ability to discard sets.

5 Conclusion

As mentioned in the introduction, the special case of OPIP in which the matrix is binary (i.e., each set requires either one or zero copies of each item) was considered in [8], where an upper bound and an almost tight lower bound on the competitive ratio of randomized algorithms where presented. We have shown that a variant of the algorithm from [8] applies to general OPIP. We note that the above lower bound applies to the unit capacities case. However, there is no lower bound for OPIP with non-unit uniform capacities.

We have proven matching upper and lower bounds on the competitive ratio for OTF. We have shown that even randomized algorithms cannot have competitive ratio better than $\Omega(\sqrt{\rho}_{\max})$. The lower bound holds even if $\rho_{\max}$ is known, and even if one is allowed to drop previously selected sets. On the other hand, the upper bound is obtained due to a simple deterministic algorithm that does not drop sets. Unfortunately, our algorithm is based on the prior knowledge of $\rho_{\max}$. It remains an open question whether there is an $O(\sqrt{\rho_{\max}})$-competitive algorithm that has no knowledge of $\rho_{\max}$. We have eliminated the possibility of an $O(\sqrt{\rho_{\max}})$ upper bound for an algorithm that is not allowed to discard sets.

Acknowledgments We thank Moti Medina for going to Berkeley to represent us.

References

1. Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, J.: The online set cover problem. SIAM J. Comput. **39**(2), 361–370 (2009)
2. Awerbuch, B., Azar, Y., Plotkin, S.A.: Throughput-competitive on-line routing. In: 34th IEEE Annual Symposium on Foundations of Computer Science, pp. 32–40 (1993)
3. Bateni, M., Hajiaghayi, M., Zadimoghaddam, M.: Submodular secretary problem and extensions. In: 13th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Volume 6302 of LNCS, pp. 39–52 (2010)
4. Berman, P.: A $d/2$ approximation for maximum weight independent set in d-claw free graphs. Nord. J. Comput. **7**(3), 178–184 (2000)
5. Buchbinder, N., Naor, J.: Online primal-dual algorithms for covering and packing. Math. Oper. Res. **34**(2), 270–286 (2009)
6. Chekuri, C., Khanna, S.: On multidimensional packing problems. SIAM J. Comput. **33**(4), 837–851 (2004)
7. Cygan, M.: Improved approximation for 3-dimensional matching via bounded pathwidth local search. In: 54th IEEE Annual Symposium on Foundations of Computer Science, pp. 509–518 (2013)
8. Emek, Y., Halldórsson, M.M., Mansour, Y., Patt-Shamir, B., Radhakrishnan, J., Rawitz, D.: Online set packing. SIAM J. Comput. **41**(4), 728–746 (2012)
9. Feldman, M., Naor, J.S., Schwartz, R.: Improved competitive ratios for submodular secretary problems. In: 14th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Volume 6845 of LNCS, pp. 218–229 (2011)
10. Freeman, P.: The secretary problem and its extensions: a review. Int. Stat. Rev. **51**(2), 189–206 (1983)
11. Frieze, A.M., Clarke, M.R.B.: Approximation algorithms for the m-dimensional 0–1 knapsack problem: worst-case and probabilistic analyses. Eur. J. Oper. Res. **15**, 100–109 (1984)
12. Gilbert, J.P., Mosteller, F.: Recognizing the maximum of a sequence. J. Am. Stat. Assoc. **61**(313), 35–73 (1966)
13. Halldórsson, M.M., Kratochvíl, J., Telle, J.A.: Independent sets with domination constraints. Discrete Appl. Math. **99**(1–3), 39–54 (2000)
14. Halldórsson, M.M., Patt-Shamir, B., Rawitz, D.: Online scheduling with interval conflicts. Theory Comput. Syst. **53**(2), 300–317 (2013)
15. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**(1), 105–142 (1999)
16. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-set packing. Comput. Complex. **15**(1), 20–39 (2006)
17. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM **22**(4), 463–468 (1975)
18. Magazine, M.J., Chern, M.-S.: A note on approximation schemes for multidimensional knapsack problems. Math. Oper. Res. **9**(2), 244–247 (1984)

19. Mansour, Y., Patt-Shamir, B., Rawitz, D.: Competitive router scheduling with structured data. Theor. Comput. Sci. **530**, 12–22 (2014)
20. Raghavan, P., Thompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica **7**(4), 365–374 (1987)
21. Sahni, S.: Approximate algorithms for the 0/1 knapsack problem. J. ACM **22**(1), 115–124 (1975)
22. Srinivasan, A.: Improved approximations of packing and covering problems. In: 27th Annual ACM Symposium on the Theory of Computing, pp. 268–276 (1995)