Online Set Packing*

Yuval Emek[†]
yuval.emek@tik.ee.ethz.ch

Magnús M. Halldórsson[‡]
mmh@ru.is

Yishay Mansour[§]
mansour@cs.tau.ac.il

Boaz Patt-Shamir[¶ ‖]
boaz@eng.tau.ac.il

Jaikumar Radhakrishnan[**]
jaikumar@tifr.res.in

Dror Rawitz[¶]
rawitz@eng.tau.ac.il

Abstract

In online set packing (OSP), elements arrive online, announcing which sets they belong to, and the algorithm needs to assign each element, upon arrival, to one of its sets. The goal is to maximize the number of sets that are assigned all their elements: a set that misses even a single element is deemed worthless. This is a natural online optimization problem that abstracts allocation of scarce compound resources, e.g., multi-packet data frames in communication networks. We present a randomized competitive online algorithm for the weighted case with general capacity (namely, where sets may have different values, and elements arrive with different multiplicities). We prove a matching lower bound on the competitive ratio for any randomized online algorithm. Our bounds are expressed in terms of the maximum set size and the maximum number of sets an element belongs to. We also present refined bounds that depend on the uniformity of these parameters.

Keywords: online set packing, competitive analysis, randomized algorithm, packet fragmentation, multi-packet frames

*A preliminary version was presented at the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC), 2010. Research supported in part by the Next Generation Video (NeGeV) Consortium, Israel.

[†]Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland

[‡]School of Computer Science, Reykjavik University, 103 Reykjavik, Iceland. Supported in part by the Icelandic Research Fund (grant 90032021).

[§]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886, by a grant from the Israel Science Foundation (grant No. 709/09), grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF) and by Israel Ministry of Science and Technology (grant No. 3-6797). This publication reflects the authors' views only.

[¶]School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.

[‖]Supported in part by the Israel Science Foundation (grant 1372/09) and by Israel Ministry of Science and Technology.

[**]School of Technology and Computer Science, Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai 400005, India.

1 Introduction

Consider the following abstract scenario, which we call *on-line set packing* (OSP) henceforth. There is a collection of sets whose elements are initially unknown. In each time step, a new element arrives, announcing which sets it belongs to. Our job is to assign each element to a bounded number of the sets containing it before the next element arrives. When a set has all its elements assigned to it, we get paid for that set (with no reward for unfinished sets). The goal is to maximize the number, or value, of completed sets.

The problem appears to be a new fundamental online problem that, to the best of our knowledge, was not studied explicitly in the past. Formally, the offline version of the problem we consider is expressed by the following integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} w_i x_i \\ \text{s.t.} \quad & \sum_{i:S_i \ni u_j} x_i \le b_j \qquad \text{for } j = 1, \ldots, n \\ & x_i \in \{0, 1\} \ . \end{aligned} \tag{1}$$

The value of x_i indicates whether set S_i is taken or not, w_i is the benefit obtained by completing set S_i, and b_j is the number of sets element u_j can be assigned to. The online character of OSP is expressed by the following additional assumptions: (1) constraints arrive one by one, (2) the variables x_i can only be decreased from 1 to 0 (i.e., a rejected set can not be taken later), and (3) the target function at a given time is taken only over variables that will not be constrained anymore (we assume that we know when all elements of a set have arrived).

Motivation. Online set packing models a variety of problems of scheduling a bounded-capacity server, where tasks arrive piecewise, and a task is completed only if all its parts are processed. Let us review a few motivating examples for OSP before we state our results.

Video transmission. In video transmission over the Internet, the situation can be roughly described as follows. Data is produced in large units at the source (a video frame may be hundreds of kilobytes long), while networks allow only for small transfer units (e.g., packets sent over Ethernet are limited to 1.5 kilobytes), and therefore in many cases, video frames are broken into a number of small packets. However, a video frame is useful at the receiving end only if *all* its constituent packets arrive.[1] Now consider an outgoing link in a network switch (router): the capacity of the link is bounded, so when a large burst of packets headed to that link arrives, some packets will be dropped (let us ignore buffering for simplicity). The router, in case of an overflow, needs to decide which packets to drop, so as to maximize the number of complete frames delivered; this decision is made online. When packets are parts of large data frames as described above, the question of which packets to drop and which to transmit can be reduced to OSP a follows: elements represent time steps, and sets represent data frames. Time step j is included in data

[1] In reality the situation is more complicated. In MPEG, for example, the smallest useful information unit is actually called a "slice," which is typically a frame part. On the other hand, there are inter-frame dependencies. We ignore these issues here, and assume that the basic information unit is a frame.

frame i if a packet of frame i arrives at time j. The capacity of the link (the number of packets that can be transmitted) at time j is b_j. The w_i parameters model data frames values.

Multihop packet scheduling. Another common scenario in packet-switching networks is the case of packets that need to traverse multiple hops: a packet is delivered only if it is not dropped by any of the switches along its route. This too can be formalized as OSP as follows. Let each pair (t, ℓ) of time t and link ℓ be modeled by an element of the OSP formulation, and let each packet be modeled by a set, whose elements are all time-link pairs which the packet is supposed to cross (again, ignoring buffering). The capacity of link ℓ at time t is represented by b_j, where $j = (t, \ell)$.

Online combinatorial auction. In combinatorial auctions with single-minded bidders, the utility of a bidder is non-zero only if it receives a certain set of items it desires. Consider an online scenario, where items become available over time. The bidders may be reluctant to reveal their exact utility function to the auctioneer, say for reasons of business competition. Initially, each bidder declares a bid, which would be paid if it received the desired set. Later, bidders request items as they appear. (To ensure agents' actions are fair and that the adversary is oblivious, the information on bids and sets can be stored with a third party escrow.)

Camera refurbishing. Consider a business that refurbishes cameras by assembling together the good parts of defective products. There are many models of cameras, but some models share parts. The mode of operation is as follows: When some part becomes available, the proprietor decides what model it will be used for, and installs that part in that partly-done camera. When a camera is complete, it is sold. In the language of OSP, camera parts are elements, and cameras are sets.

Service center. A service center (say, for printers) sells the following service package: The customer gets to make up to k service calls. The customer is charged only if each of his calls is serviced on the same day. The center can service at most b_j calls on day j.

Distinction: Centralized and distributed algorithms. In many applications it is reasonable to assume that the algorithm is *centralized* in the sense that a decision regarding an arriving element may depend on the complete history of the algorithm thus far. However, some applications require *distributed* decision making, in the sense that elements may arrive at different locations, so that it may be hard (or impossible) to know which elements have arrived thus far, and—if the algorithm is not deterministic—which sets they were assigned to. For example, in the video transmission case, packets may traverse different routes in the network, and different routers will have to decide which packets to drop and which to retain. Obviously, a distributed algorithm is more desirable, but centralized algorithms may have better performance.

Our contributions. In this paper we introduce the problem of online set packing and present a randomized competitive algorithm for it. Our algorithm is naturally distributed. On the negative side, we prove nearly matching lower bounds on the competitive ratio of any randomized online algorithm for it, which holds even in the centralized case. Our bounds on the competitive ratio are expressed in terms of the size of the sets and the *degree* of the elements, where the degree of an element is the number of sets containing it. Specifically, let $k_{\max}$ denote the maximal set size and $\sigma_{\max}$ denote the

maximal element degree. Our first main result is a randomized distributed algorithm that guarantees to complete sets of total expected value at least $\text{OPT}/(k_{\max}\sqrt{\sigma_{\max}})$, where OPT is the maximal value of any feasible solution. The result extends to the general capacity case: Assume that each element j arrives with multiplicity $b_j > 0$, which allows it to be assigned to b_j sets. In this case, our upper bound generalizes using adjusted degree, defined as the degree of element j divided by b_j. The algorithm is inspired by the probabilistic version of Turán's theorem of Alon and Spencer [3]; in fact, our algorithm reduces to theirs in the unweighted case when $\sigma = 2$.

We also derive more refined bounds on the competitive ratio which are sensitive to the variability of set sizes and element degrees: the more uniform they are, the better bounds we get. For example, if all elements have the same degree, then the competitive ratio drops to $k_{\max}$.

Our second main result is a lower bound that shows that no randomized online algorithm (including centralized algorithms) can have competitive ratio much better than $k_{\max}\sqrt{\sigma_{\max}}$, even in the unweighted case, and also show a simple lower bound of $(\sigma_{\max})^{k_{\max}-1}$ for deterministic online algorithms. Our construction for the randomized case is a bit involved, and our technique uses combinatorial designs based on affine planes.

Related work. Let us first consider *off-line* set packing. In this case set packing is as hard as Maximum Independent Set even when all elements have degree 2, and therefore cannot be approximated to within $O(m^{1-\epsilon})$-factor, where m denotes the number of sets and ϵ is any positive constant [7]. In terms of the number of elements, denoted n, offline set packing is $O(\sqrt{n})$-approximable, and hard to approximate within $n^{1/2-\epsilon}$, for any $\epsilon > 0$ [9]. When set sizes are at most k, it is approximable to within $k/2 + \epsilon$ for any $\epsilon > 0$ [12] and within $(k+1)/2$ in the weighted case [5], but known to be hard to approximate to within $\Omega(k/\log k)$-factor [11]. An important application for weighted set packing is in combinatorial auctions (see, e.g., [21]).

Buchbinder and Naor have studied online primal-dual algorithms for covering and packing (see, e.g., [6]). In their model, constraints of the primal (covering) program arrive one by one, and the variables can only be increased by the algorithm. This approach was applied to online set cover [1], and to the following variant of packing: In each step, a new set is introduced by listing all its elements; the algorithm may accept the set only if it is disjoint to previously accepted sets, but it may also reject it. If a set is accepted, the algorithm collects the set value immediately. In our setting, elements arrive one by one, and the benefit is earned only after a set is complete. When specified as linear programs (more precisely, as integer programs), both the packing framework of [6] and our formulation of OSP share the same matrix (Eq. (1) above), but in [6] columns (variables) arrive online, while in our formulation rows (constraints) arrive online.

In fact, all previous online algorithms for packing disjoint structures (possibly partially), whether it be sets, vertices, or paths, assume that decisions are made on already completed structures. In this case, a factor k-approximation is trivially obtained by a greedy algorithm for unweighted set packing. When $k = 2$, we obtain an online matching problem; numerous results are known for a similar but different problem, starting with the $e/(e-1)$-competitive randomized algorithm of [14], that works for a weighted bipartite version with a certain restriction on arrival order. For online independent set problems, that relate to packing paths in graphs, very strong lower bounds generally hold [2, 8].

The problem of online throughput maximization of multi-packet frames (our video transmission motivation) was introduced in [15], with the additional complication of having a finite-capacity buffer. The results there are not comparable to ours because of the buffering, and due to the following

additional differences: on one hand no bound on element degree is assumed in [15] (i.e., arbitrarily many packets may arrive simultaneously at the server), but on the other hand it assumed that the arrival process is "well ordered" in some rather restrictive sense.[2] An upper bound of $O(k^2)$ and a lower bound of $\Omega(k)$ on the competitive factor for deterministic online algorithms in this model was given in [15].

Distributed models for solving linear programs are considered in [18, 17], where the complete matrix is input to the system at the start of execution, but it is distributed among different agents. In [4] new variables and constraints can be introduced over time, and the system will stabilize to an approximately optimal solution, but the variables may be both increased and decreased by the algorithm, so the algorithm is not online in the sense considered in the current paper.

When $\sigma = 2$, the offline problem becomes the independent set problem in graphs. Our algorithm matches a weighted extension of Turán's theorem, due to Sakai, Togasaki and Yamazaki [20], that was achieved with a greedy offline algorithm. In the unweighted case, our algorithm then matches the probabilistic version of Turán's theorem, due to Alon and Spencer [3]. It was generalized to the weak independent set problem in hypergraphs by Shachnai and Srinivasan [22] (whereas, our set packing problem is known as the *strong* independent set problem in hypergraphs). That algorithm was recently treated in a similar setting to ours, but with a focus on the space requirements as a streaming algorithm [10].

Paper organization. The remainder of this paper is organized as follows. In Section 2 we formalize the problem and state our main results. In Section 3 we describe and analyze our algorithm. In Section 4 we prove our lower bounds. We conclude in Section 5 with some open problems.

2 Notation, Problem Statement and Results

Problem statement. A weighted set system consists of a set U of n elements, a family $\mathcal{C} = \{S_1, S_2, \ldots, S_m\}$ of m subsets of U, and a *weight function* assigning a non-negative weight $w(S)$ to each set $S \in \mathcal{C}$. We also assume that some *capacity* $b(u) \in \mathbb{N}$ is associated with each element $u \in U$.

We concentrate on the *online set packing* problem, defined as follows. Initially, for each set we know only its weight and size (but not its members). In each step j, a new element u_j arrives along with its capacity $b(u_j)$ and with $\mathcal{C}(u_j) \stackrel{\text{def}}{=} \{S \in \mathcal{C} : u_j \in S\}$, i.e., the names of all sets containing u_j. The algorithm must output in step j a collection of set names $\mathcal{A}(j) \subseteq \mathcal{C}(u_j)$ such that $|\mathcal{A}(j)| \leq b(u_j)$. The algorithm is said to *complete* a set S if $S \in \mathcal{A}(j)$ for each of its elements $u_j \in S$. The output of an algorithm for an instance $\mathcal{I}$, denoted $\textsc{alg}(\mathcal{I})$ (or simply $\textsc{alg}$), is the collection of sets completed by the algorithm, and the benefit to the algorithm is in that case $w(\textsc{alg}(\mathcal{I}))$, where the weight of a collection of sets is the sum of the set weights. If the algorithm is randomized, the benefit for a given instance is a random variable, and we shall use its expected value. We say that an online OSP algorithm is distributed if the decisions it makes upon arrival of element u_j depend only on the identities of the sets in $\mathcal{C}(u_j)$.

The performance of OSP algorithms is measured using the standard notion of competitive analysis: the *competitive ratio* of an algorithm is the supremum, over all instances $\mathcal{I}$, of $w(\textsc{opt}(\mathcal{I}))/w(\textsc{alg}(\mathcal{I}))$,

[2] The sets are said to be *well ordered* if for any $1 \leq \ell, \ell' \leq k$, the ℓth element of set A arrives before the ℓth element of set B if and only if the ℓ'th element of set A arrives before the ℓ'th element of set B.

where $\textsc{opt}(\mathcal{I})$ denotes the collection of subsets in $\mathcal{C}$ that maximizes the target function in (1).

Special interesting classes of instances are the *unweighted* instances where $w(S) = 1$ for all $S \in \mathcal{C}$, and the *unit capacity* instances where $b(u) = 1$ for all $u \in U$.

Notation. For a family of sets $\mathcal{D} \subseteq \mathcal{C}$ and an element u of the universe, let $\mathcal{D}(u) = \{S \in \mathcal{D} : u \in S\}$. For example, we will write $\textsc{opt}(u)$. We also denote

$$\begin{aligned}
k(S) &= |S|; \\
\sigma(u) &= |\mathcal{C}(u)| = \text{degree of the element } u; \\
\sigma_{\mathcal{D}}(u) &= |\mathcal{D}(u)|; \\
w(u) &= w(\mathcal{C}(u)); \\
b(u) &= \text{the capacity associated with the element } u, \text{ assumed to be at most } \sigma(u); \\
\nu(u) &= \frac{\sigma(u)}{b(u)}.
\end{aligned}$$

For functions g and h defined on a set V, we write $\langle g, h\rangle_V = \sum_{v\in V} g(v)h(v)$. When the set V is not explicitly indicated, we will assume V to be the common domain of g and h. We will use $\mathbb{1}$ to denote the function that maps all elements of the domain to 1.

Main results. We present a randomized algorithm ALG and analyze its performance. Our main results are the following.

Theorem 1. *For unit capacity instances,*

$$\mathbb{E}[w(\textsc{alg})] \;\geq\; \max\left\{\frac{w(\mathcal{C})^2}{\langle\sigma, w\rangle_U}, \frac{w(\textsc{opt})^2}{\langle k, w\rangle_{\mathcal{C}}}\right\} \;\geq\; \frac{w(\mathcal{C})w(\textsc{opt})}{\sqrt{\langle\sigma, w\rangle_U \langle k, w\rangle_{\mathcal{C}}}}\ .$$

This theorem immediately gives the following bound on the competitive ratio.

Corollary 1. *There is an* OSP *algorithm with competitive ratio at most* $k_{\max}\sqrt{\sigma_{\max}}$*, where* $k_{\max} = \max_{S\in\mathcal{C}} k(S)$ *and* $\sigma_{\max} = \max_{u\in U}\sigma(u)$*.*

Note, however, that the competitive ratio can be much smaller than this bound: for example, if all sets have the same size k and all elements have the same degree, then the competitive ratio improves to k in the unweighted case. We discuss some of these special cases in Section 3.4.

For the variable capacity case, the same bound holds, up to a factor of 2, after replacing the degree $\sigma(u)$ of the element with its *adjusted degree* $\nu(u)$.

Theorem 2. *For general instances,*

$$\mathbb{E}[w(\textsc{alg})] \;\geq\; \max\left\{\frac{w(\mathcal{C})^2}{2\langle\nu, w\rangle_U}, \frac{w(\textsc{opt})^2}{2\langle k, w\rangle_{\mathcal{C}}}\right\} \;\geq\; \frac{w(\mathcal{C})w(\textsc{opt})}{2\sqrt{\langle\nu, w\rangle_U \langle k, w\rangle_{\mathcal{C}}}}\ .$$

No algorithm can make a baseline improvement over our algorithm, or, more precisely:

Theorem 3. *For any randomized online algorithm, there exists an infinite family of unweighted, unit-capacity instances of* OSP *for which the competitive ratio is*

$$\Omega\left(\left(\frac{\log\log k_{\max}}{\log k_{\max}}\right)^2 k_{\max}\sqrt{\sigma_{\max}}\right) .$$

Our lower bound argument uses Yao's principle: we build a distribution of the inputs for which the optimal value is large, while the expected value for all deterministic algorithms is small. Also note that we use two different lower bound constructions: in the first $k_{\max}$ and $\sigma_{\max}$ are linearly related; in the second $k_{\max}$ is constant.

The situation is much worse for deterministic algorithms:

Theorem 4. *The competitive ratio of any deterministic* OSP *algorithm is at least* ${\sigma_{\max}}^{k_{\max}-1}$*, even for unweighted unit-capacity instances.*

3 Randomized Upper Bounds

In this section we describe our randomized algorithm for OSP and analyze it. We start the section with a description of the algorithm. Then, in Section 3.2 we analyze the algorithm in the unit-capacity model, in Section 3.3 we extend the analysis to the general capacity model, and finally a few sharper results for some special cases are given in Section 3.4.

3.1 The Randomized Algorithm

Random variables. Recall that the (cumulative) distribution $F_X : \mathbb{R} \to [0,1]$ of a random variable X is defined by

$$F_X(x) = \Pr[X \le x].$$

For $w > 0$, let $D_w : \mathbb{R} \to [0,1]$ be defined by

$$D_w(x) \;=\; \begin{cases} 0 & \text{if } x < 0; \\ x^w & \text{if } 0 \le x < 1; \\ 1 & \text{if } 1 \le x. \end{cases} \tag{2}$$

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 randPr. We first describe our algorithm for instances with unit capacities. For each set $S \in \mathcal{C}$, we independently choose a random priority $r(S) \in [0,1]$ with distribution $D_{w(S)}$. When the element u arrives with a list $\mathcal{C}(u)$ of parent sets, we assign u to the highest priority set in $\mathcal{C}(u)$.

We next show how general instances can be reduced to unit capacity instances. Given a general instance $\mathcal{I}$, the unit capacity instance $\mathcal{I}'$ is constructed as follows. The element u with capacity $b(u)$ is replaced by $b(u)$ new elements: $u_1, u_2, \ldots, u_{b(u)}$, each with unit capacity. The sets are modified by substituting u with one of the u_i's. This substitution is done randomly as follows. First partition $C(u)$ randomly into $b(u)$ blocks of size $\left\lceil \frac{\sigma(u)}{b(u)} \right\rceil$ or $\left\lfloor \frac{\sigma(u)}{b(u)} \right\rfloor$ (each such partition being equally likely):

$C_1(u), C_2(u), \ldots, C_{b(u)}(u)$; replace u by u_i in all sets in $C_i(u)$. This completes the description of the unit capacity instance $\mathcal{I}'$. On this instance $\mathcal{I}'$, run the algorithm above: if u_i is assigned to the set S in $\mathcal{I}'$, declare that u is assigned to the corresponding set S in $\mathcal{I}$ (note that u is assigned to exactly $b(u)$ sets). (The proofs in this version do not require this kind of partitioning: it would work equally well if we placed each set independently into a randomly chosen block (each with probability $\frac{1}{b(u)}$).)

Remarks about implementation. Algorithm RANDPR can be implemented by a distributed algorithm. We assume that whenever the element u is to be assigned to some sets, the list of identifiers of all the sets that u belongs to is provided explicitly. There are two types of random choices made in the above algorithm. The first involves the choice of priorities for the various sets. These can be made by a randomly chosen shared hash function with range $[0, 1]$.[3] By computing $h(S)$ we obtain a uniformly generated number in $[0, 1]$; we set $r(S) = h(s)^{\frac{1}{w(S)}}$. The second type of random choices are made while partitioning the lists that arrive with the elements. Note that these choices do not require global coordination. Even though we state our algorithm using the unit capacity instance $\mathcal{I}'$ derived from $\mathcal{I}$, the random choices are not required to be made right in the beginning, but can be made as the elements arrive with their lists of parent sets.

3.2 Unit Capacity Instances

Clearly the algorithm completes a set S if and if and only if it assigns S a priority higher than that of all other sets with whom it competes for some element.

Lemma 2. *Let $N \subseteq \mathcal{C} \setminus \{S\}$. Then,* $\Pr[r(S) > \max\{r(S') : S' \in N\}] = \dfrac{w(S)}{w(N) + w(S)}$.

Proof. If the weights are all integral, then we may analyze the event in question by viewing $r(S')$ as the maximum of $w(S')$ independent random variables, each uniformly distributed in $[0, 1]$. For all the sets in $N \cup \{S\}$ put together, there are $w(N) + w(S)$ such variables, and, by symmetry, one of the $w(S)$ variables associated with S will be the maximum with probability $\frac{w(S)}{w(N)+w(S)}$.

When the weights are not necessarily integral, we have the following calculation. Let $r_{\max} = \max\{r(S') : S' \in N\}$. Then, for $x \in [0, 1]$ we have

$$\Pr[r_{\max} < x] \;=\; \prod_{S' \in N} \Pr[r(S') < x] \;=\; \prod_{S' \in N} x^{w(S')} \;=\; x^{\sum_{S' \in N} w(S')} \;=\; x^{w(N)} \;,$$

namely $r_{\max}$ has distribution $D_{w(N)}$. Hence,

$$\Pr[r(S) > r_{\max}] \;=\; \int_0^1 \Pr[r_{\max} < x] \cdot f_{r(S)}(x)dx \;=\; \int_0^1 x^{w(N)} \cdot w(S)x^{w(S)-1}dx \;=\; \frac{w(S)}{w(N) + w(S)}$$

as required. □

For a set $S \in \mathcal{C}$, let $N[S] = \{S' \in \mathcal{C} : S \cap S' \neq \emptyset\}$.

Lemma 3. *For any collection of sets $\mathcal{D} \subseteq \mathcal{C}$,* $\mathbb{E}[w(\text{ALG})] \geq \dfrac{w(\mathcal{D})^2}{\sum_{S \in \mathcal{D}} w(N[S])}$.

[3] Practically, any off-the-shelf hash function would do. And even theoretically, it suffices for the hash function to have $k_{\max}\sigma_{\max}$-wise independence, say using universal hashing.

Proof. By Lemma 2, $\Pr[S \in \text{ALG}] = \frac{w(S)}{w(N[S])}$. Thus, by linearity of expectation, we obtain

$$\mathbb{E}[w(\text{ALG})] \;=\; \sum_{S\in\mathcal{D}} w(S)\cdot\frac{w(S)}{w(N[S])} \;\geq\; \frac{\left(\sum_{S\in\mathcal{D}} w(S)\right)^2}{\sum_{S\in\mathcal{D}} w(N[S])} \;=\; \frac{w(\mathcal{D})^2}{\sum_{S\in\mathcal{D}} w(N[S])}\;,$$

where, to justify the second inequality, we used the following consequence (see [13]) of the Cauchy-Schwarz inequality (with $w(S)$ for a_i and $w(N[S])$ for b_i): For positive reals $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$: $\sum_i \frac{a_i^2}{b_i} \geq \frac{\left(\sum_i a_i\right)^2}{\sum_i b_i}$. □

We will apply the above lemma for two choices of $\mathcal{D}$, the optimal solution OPT and the collection $\mathcal{C}$ of all the sets in the instance. In both cases, we will need to derive appropriate upper bounds on the denominator, $\sum_{S\in\mathcal{D}} w(N[S])$.

Lemma 4. *For all $\mathcal{D} \subseteq \mathcal{C}$, $\sum_{S\in\mathcal{D}} w(N[S]) \;\leq\; \langle\sigma_{\mathcal{D}}, w\rangle_U$. In particular,*

$$\sum_{S\in\mathcal{C}} w(N[S]) \;\leq\; \langle\sigma, w\rangle_U; \tag{3}$$

$$\sum_{S\in\text{OPT}} w(N[S]) \;\leq\; \langle k, w\rangle_{\mathcal{C}}. \tag{4}$$

Proof. Observe that

$$\sum_{S'\in\mathcal{D}} w(N[S']) \;=\; \sum_{S'\in\mathcal{D}}\sum_{S\in N[S']} w(S) \;\leq\; \sum_{u\in U}\sum_{S'\in\mathcal{D}(u)} w(u) \;=\; \sum_{u\in U}\sigma_{\mathcal{D}}(u)w(u) \;=\; \langle\sigma_{\mathcal{D}}, w\rangle_U\;.$$

Inequality (3) follows from this by taking $\mathcal{D} = \mathcal{C}$. For Inequality (4), take $\mathcal{D} = \text{OPT}$ and note that $\sigma_{\text{OPT}}(u) \leq 1$ for all u. It follows that $\langle\sigma_{\text{OPT}}, w\rangle_U \leq \langle\mathbb{1}, w\rangle_U = \langle k, w\rangle_{\mathcal{C}}$. □

Proof of Theorem 1. Lemmas 3 and 4 imply that

$$\mathbb{E}[w(\text{ALG})] \;\geq\; \max\left\{\frac{w(\mathcal{C})^2}{\langle\sigma, w\rangle_U}, \frac{w(\text{OPT})^2}{\langle k, w\rangle_{\mathcal{C}}}\right\} \;\geq\; \frac{w(\mathcal{C})w(\text{OPT})}{\sqrt{\langle\sigma, w\rangle_U \langle k, w\rangle_{\mathcal{C}}}}\;,$$

where the last inequality holds because the maximum of two quantities is always at least their geometric mean. □

Corollary 1 follows since $\langle\sigma, w\rangle_U \leq \sigma_{\max}k_{\max}\cdot w(\mathcal{C})$ and $\langle k, w\rangle_{\mathcal{C}} \leq k_{\max}\cdot w(\mathcal{C})$.

3.3 Instances with Arbitrary Capacities

Recall that our algorithm on a general instance $\mathcal{I}$ derives a unit capacity instance $\mathcal{I}'$, and runs the unit capacity algorithm on this instance. Note that every set in the instance $\mathcal{I}$ corresponds to a set in $\mathcal{I}'$ with the same number of elements and the same weight, so we will continue to use the $\mathcal{C}$, S, $k(S)$ and $w(S)$ while referring to these sets. However, we will use special notation while referring to the elements of the new instance: we use U' to refer to the new universe of elements, and $d'(u')$ and $w'(u')$ to refer to the degree and weighted degree of the element $u' \in U'$.

Lemma 5. *Let $\mathcal{D} \subseteq \mathcal{C}$. Then,* $\mathbb{E}[w(\text{ALG})] \geq \dfrac{w(\mathcal{D})^2}{E[\langle d'_{\mathcal{D}}, w' \rangle_{U'}]}$.

Proof. This follows from Lemmas 3 and 4 and Jensen's inequality

$$\mathbb{E}[w(\text{ALG})] \;\geq\; \mathbb{E}\left[\frac{w(\mathcal{D})^2}{\langle d'_{\mathcal{D}}, w' \rangle_{U'}}\right] \;\geq\; \frac{w(\mathcal{D})^2}{\mathbb{E}[\langle d'_{\mathcal{D}}, w' \rangle_{U'}]} \;.$$

□

To complete our analysis we will find appropriate upper bounds for the denominator for two choices of the subfamily $\mathcal{D}$. Recall that $\nu(u) = \frac{\sigma(u)}{b(u)}$.

Lemma 6.

$$\mathbb{E}[\langle d', w' \rangle_{U'}] \;\leq\; 2\langle \nu, w \rangle_U; \tag{5}$$

$$\mathbb{E}[\langle d'_{\text{OPT}}, w' \rangle_{U'}] \;\leq\; \langle k, w \rangle_{\mathcal{C}} + \langle k, w \rangle_{\text{OPT}} \;\leq\; 2\langle k, w \rangle_{\mathcal{C}}. \tag{6}$$

Proof. We begin by opening out $\langle d'_{\mathcal{D}}, w' \rangle_{U'}$. Fix $u \in U$ and consider its versions $u_1, u_2, \ldots, u_{b(u)}$ in the instance $\mathcal{I}'$. Now,

$$d'_{\mathcal{D}}(u_i) w'(u_i) \;=\; \sum_{S' \in \mathcal{D}(u)} \sum_{S \in \mathcal{C}(u)} w(S) \eta_{S',S}(u_i) \;,$$

where $\eta_{S',S}(u_i) = 1$ if $S' \in \mathcal{D}(u_i)$ and $S \in \mathcal{C}(u_i)$, and $\eta_{S',S}(u_i) = 0$ otherwise (that is, $\eta_{S',S}(u_i)$ is the indicator function for the event *S' and S both use the same copy u_i of u when the instance $\mathcal{I}'$ is constructed from $\mathcal{I}$*). Summing over $i = 1, 2, \ldots, b(u)$, we have

$$\sum_{i=1}^{b(u)} d'_{\mathcal{D}}(u_i) w'(u_i) \;=\; \sum_{S' \in \mathcal{D}(u)} \sum_{S \in \mathcal{C}(u)} \sum_{i=1}^{b(u)} w(S) \eta_{S',S}(u_i) \;=\; \sum_{S \in \mathcal{C}(u)} w(S) \sum_{S' \in \mathcal{D}(u)} \sum_{i=1}^{b(u)} \eta_{S',S}(u_i) \;.$$

Let us denote $\sum_{i=1}^{b(u)} \eta_{S',S}(u_i)$ by $\chi_{S',S}(u)$, the indicator random variable for the event *S' and S fall in the same block in the random partition of $\mathcal{C}(u)$ used for constructing $\mathcal{I}'$ from $\mathcal{I}$*; thus, $\Pr[\chi_{S',S}(u) = 1] \leq \frac{1}{b(u)}$, if $S' \neq S$, and $\Pr[\chi_{S',S}(u) = 1] = 1$ if $S = S' \in \mathcal{D}(u)$. We then have

$$\sum_{i=1}^{b(u)} d'_{\mathcal{D}}(u_i) w'(u_i) \;=\; \sum_{S \in \mathcal{C}(u)} w(S) \sum_{S' \in \mathcal{D}(u)} \chi_{S',S}(u) \;.$$

We will split this sum depending on whether or not $S \in \mathcal{D}(u)$, and whether or not $S = S'$.

$$\sum_{i=1}^{b(u)} d'_{\mathcal{D}}(u_i) w'(u_i) \;=\; \sum_{S \in \mathcal{D}(u)} w(S) \sum_{S' \in \mathcal{D}(u) \setminus \{S\}} \chi_{S',S}(u) \;+\; \sum_{S' \in \mathcal{D}(u)} w(S) \chi_{S',S'}(u) \;+$$
$$\sum_{S \in \mathcal{C}(u) \setminus \mathcal{D}(u)} w(S) \sum_{S' \in \mathcal{D}(u)} \chi_{S',S}(u).$$

Taking expectations we obtain,

$$\mathbb{E}\left[\sum_{i=1}^{b(u)} d'_{\mathcal{D}}(u_i) w'(u_i)\right] \;\leq\; \sum_{S \in \mathcal{D}(u)} w(S) \frac{\sigma_{\mathcal{D}}(u) - 1}{b(u)} \;+\; \sum_{S' \in \mathcal{D}(u)} w(S) \;+\; \sum_{S \in \mathcal{C}(u) \setminus \mathcal{D}(u)} w(S) \frac{\sigma_{\mathcal{D}}(u)}{b(u)} \;. \tag{7}$$

We are now ready to make specific choices for $\mathcal{D}$ to justify the inequalities claimed above. First, we take $\mathcal{D} = \mathcal{C}$, so that the last sum vanishes, and the first two can be combined to obtain

$$\mathbb{E}\left[\sum_{i=1}^{b(u)} d'_{\mathcal{D}}(u_i)w'(u_i)\right] \;\leq\; \sum_{S\in C(u)} w(S)\left(\frac{\sigma(u)-1}{b(u)}+1\right) .$$

Summing over all u, we obtain

$$\mathbb{E}[\langle d', w'\rangle_{U'}] \;=\; \sum_u w(u)\left(\frac{\sigma(u)+b(u)-1}{b(u)}\right) \;\leq\; 2\langle \nu, w\rangle_U ,$$

where, for the last inequality, we used $\frac{\sigma(u)+b(u)-1}{b(u)} \leq 2\frac{\sigma(u)}{b(u)} = 2\nu(u)$. This justifies (5).

Next, to derive (6), we take $\mathcal{D} = \text{OPT}$. Note that $\sigma_{\mathcal{D}}(u) \leq b(u)$. Then (by combining the last two sums in (7)) we obtain

$$\mathbb{E}\left[\sum_{i=1}^{b(u)} d'_{\text{OPT}}(u_i)w'(u_i)\right] \;\leq\; \sum_{S\in \text{OPT}(u)} w(S)\frac{b(u)-1}{b(u)} + \sum_{S\in\mathcal{C}(u)} w(S) .$$

Finally, summing over all $u \in U$, we get

$$\mathbb{E}[\langle d'_{\text{OPT}}, w'\rangle_{U'}] \;\leq\; \sum_{u\in U} w(\text{OPT}(u))\frac{b(u)-1}{b(u)} + \sum_{u\in U}\sum_{S\in\mathcal{C}(u)} w(S) .$$

The first term is zero if $b(u) = 1$ for all $u \in U$; in any case, it is at most $\langle k, w\rangle_{\text{OPT}}$. The second term is precisely $\sum_{S\in\mathcal{C}} k(S)w(S) = \langle k, w\rangle_{\mathcal{C}}$. Returning to (6), we now have:

$$\mathbb{E}[\langle d'_{\text{OPT}}, w'\rangle_{U'}] \;\leq\; \langle k, w\rangle_{\mathcal{C}} + \langle k, w\rangle_{\text{OPT}} \;\leq\; 2\langle k, w\rangle_{\mathcal{C}} .$$

□

Proof of Theorem 2. We combine Lemmas 5 and 6:

$$\mathbb{E}[w(\text{ALG})] \;\geq\; \max\left\{\frac{w(\mathcal{C})^2}{2\langle \nu, w\rangle_U}, \frac{w(\text{OPT})^2}{2\langle k, w\rangle_{\mathcal{C}}}\right\} \;\geq\; \frac{w(\mathcal{C})w(\text{OPT})}{2\sqrt{\langle \nu, w\rangle_U \langle k, w\rangle_{\mathcal{C}}}} .$$

□

3.4 Some special cases

In this section we analyze algorithm RANDPR for special cases of the unit capacity model.

Graphs. When the degrees of all elements is 2, the OSP becomes an independent set problem in graphs. The sets correspond to vertices of the graph, and the elements to its edges. The input is presented to the algorithm as a stream of edges. Our online algorithm then achieves the following weighted extension of Turán's bound, previously established by Sakai, Togasaki and Yamazaki [20] by analyzing an offline greedy algorithm. Let $G = (V, E)$ be the input graph, let $\deg(v)$ denote the

degree of v (i.e., $k(S)$), let w be a non-negative weight function on the vertices, and let $N(v)$ be the set of neighbors of v. Following [13], we define the *average weighted degree* of G by

$$\overline{\deg}_w(G) = \frac{1}{w(V)} \sum_{v \in V} w(v) \cdot \sigma(v).$$

Theorem 5. *If the input is viewed as a graph G, then* $\mathbb{E}[w(\text{ALG})] \geq \dfrac{w(V)}{\overline{\deg}_w(G) + 1}$.

Proof. Lemma 3 gives

$$\mathbb{E}[w(\text{ALG})] \;\geq\; \frac{w(V)^2}{\sum_{v \in V}(w(v) + w(N(v)))} \;=\; \frac{w(V)^2}{w(V) + \sum_{v \in V} w(v) \deg(v)} \ .$$

The lemma follows. □

Instances with unit weights. We have the following results when the sizes of the sets or the degrees of the elements (or both) are assumed to be uniform. Let $\bar{\sigma} = \frac{1}{n} \sum_{u \in U} \sigma(u)$; let $\bar{\sigma^2} = \frac{1}{n} \sum_{u \in U} \sigma(u)^2$.

Theorem 6. *If all sets have the same size k, then* $\mathbb{E}[w(\text{ALG})] \geq |\text{OPT}| \cdot \dfrac{\bar{\sigma}^2}{k \cdot \bar{\sigma^2}}$.

Proof. From Theorem 1, we have

$$\mathbb{E}[|\text{ALG}|] \;\geq\; \frac{m^2}{\langle \sigma, w \rangle_U} \;=\; \frac{m^2}{n\bar{\sigma}^2} \;=\; \frac{n^2\bar{\sigma}^2}{k^2 n \bar{\sigma^2}} \;\geq\; \frac{\bar{\sigma}^2}{k\bar{\sigma^2}} \cdot w(\text{OPT}) \ ,$$

where the equality is due to the fact that $mk = n\bar{\sigma}$ always holds, and the last inequality is due to the fact that $|\text{OPT}| \leq n/k$ in the unit capacity, fixed k case. □

The following corollary of Theorem 6 is our only upper bound that is independent of σ.

Corollary 7. *If all sets have the same size k and all elements have the same degree, then*

$$\mathbb{E}[|\text{ALG}|] \geq \frac{|\text{OPT}|}{k} \ .$$

Let $\bar{k} = \frac{1}{m} \sum_{S \in \mathcal{C}} k(S)$.

Theorem 7. *If all elements have the same degree σ, then* $\mathbb{E}[|\text{ALG}|] \geq \dfrac{|\text{OPT}|}{\bar{k} \cdot \sqrt{\sigma}}$.

Proof. Our assumptions imply that $\langle k, w \rangle_{\mathcal{C}} = \bar{k}m$ and $\langle \sigma, w \rangle_U = \sigma^2 n$. Thus, it follows from Theorem 1 that

$$\mathbb{E}[|\text{ALG}|] \;\geq\; \frac{|\text{OPT}| \cdot m}{\sqrt{\bar{k} m \sigma^2 n}} \ .$$

Our claim follows from this by noting that $dn = \bar{k}m$. □

4 Lower Bounds

In this section we prove lower bounds on the competitive ratio of online OSP algorithms. The bad examples in our online lower bounds are all unweighted, and further, all sets have a common size k and all elements have unit capacity. In view of Corollary 7, however, element degrees necessarily vary. The deterministic lower bound is rather simple; the randomized lower bound is more involved, and uses a construction based on combinatorial designs.

4.1 Deterministic Online Algorithms

In this section we establish a lower bound on the competitive ratio of any deterministic OSP algorithm.

Proof of Theorem 4. Fix a deterministic OSP algorithm. We construct an unweighted OSP instance $\mathcal{C}$ containing σ^k sets, each of size exactly k. The construction ensures that $|\text{ALG}| \leq 1$ while $|\text{OPT}| \geq \sigma^{k-1}$.

We describe the construction by building the sets incrementally, as a function of the algorithm at hand. Call a set *active* at a given time if the algorithm assigned to it all its elements up to that point. Initially, all σ^k sets are active. After each phase $i = 1, \ldots, k$ there will be at most σ^{k-i} active sets. This is ensured by partitioning the sets that are active before phase i into σ^{k-i} collections of σ sets each; for each such collection of σ sets we introduce a new element, which is a member of these σ sets. Clearly, at most one set from each collection remains active when the phase ends, and therefore $|\text{ALG}| \leq 1$ after k phases.

Note that at this point most sets have less than k elements defined. We now introduce new elements to complete all sets to size k. All these elements have degree one (i.e., each belongs to a single set).

Observe that in an optimal solution, it is possible to complete σ^{k-1} sets by assigning the first phase elements to sets that were not chosen by the algorithm. These sets survive, since they do not participate in the following $k-1$ phases. The theorem follows. □

4.2 Randomized Algorithms

We now turn to the main technical contribution of Section 4: developing lower bounds for the competitive ratio of randomized OSP algorithms and establishing Theorem 3. We show that for several settings of the parameters, the upper bound on the competitive ratio guaranteed by our randomized algorithm is nearly optimal. For this, we will describe suitable probability distributions on instances such that the expected size of the solution returned by any deterministic algorithm is much smaller than the optimal offline solution. It will follow from Yao's min-max principle that every randomized algorithm, for its worst-case input instance, suffers the same limitation.

4.2.1 Gadgets

Before we formally prove our lower bounds, let us consider an example.

Example 1. Fix a positive integer ℓ. Consider the (deterministic) OSP instance I_0 consisting of one element and ℓ identical singleton sets $A_1, A_2, \ldots, A_\ell$. There are ℓ optimal solutions $\mathcal{S}_i = \{A_i\}$. Any algorithm, when presented with this instance, will have to assign the element to one of the sets immediately. We will exploit this early commitment to design harder input instances.

Now, suppose ℓ independent instances I_0 are presented to the algorithm (that is ℓ^2 sets on ℓ elements) one after the other. Call these instances $I_0^1, I_0^2, \ldots, I_0^\ell$. Recall that each instance I_0^j has ℓ optimal solutions; we call them $\mathcal{S}_i(I_0^j)$, for $i = 1, 2, \ldots, \ell$. Any deterministic algorithm must commit to one solution for each I_0^j. Now, pick an $s \in [\ell]$ uniformly at random, and introduce a new element e and place it in all sets other than those in $\mathcal{S}_s(I_0^j)$ for $j = 1, 2, \ldots, \ell$. The optimal solution, $\bigcup_j \mathcal{S}_s(I_0^j)$, has ℓ sets, but the expected size of the solution produced by any deterministic algorithm is at most 2.

We now build on the ideas contained in the above examples. The instances we produce below will depend on a parameter ℓ. We will assume that ℓ is a large prime power.

Definition 1 (Gadget)**.** A (β, ϵ)-gadget is a random OSP instance I with the following properties.

(a) There exist ℓ disjoint solutions to I, denoted $\mathcal{S}_1(I), \mathcal{S}_2(I), \ldots, \mathcal{S}_\ell(I)$, each of size exactly $|\mathcal{C}(I)|/\ell$. These solutions will be referred to as the standard solutions.

(b) For any solution $\mathcal{S}$ of I, $\mathcal{S} \cap \mathcal{S}_i(I) \neq 0$ for at most one $i \in [\ell]$.

(c) For any deterministic algorithm ALG, $\Pr\left[\max_{i \in [\ell]} |\text{ALG}(I) \cap \mathcal{S}_i(I)| > \beta\right] \leq \epsilon$.

Example 2 (Trivial gadget)**.** The instance I_0 defined above is a $(1, 0)$-gadget.

Our goal is to produce a (β, ϵ)-gadget I for small ϵ (not 0 though) and β much smaller than $\mathcal{C}(I)/\ell$. The trivial gadget in Example 2 does not achieve this (in fact, $\beta = |\mathcal{C}(I)|/\ell$). Nevertheless, it forms the basis for our later nearly optimal gadgets.

4.2.2 Powering the gadget

We will consider a method for obtaining larger gadgets from smaller ones by taking their *powers*. (Refer to Figure 1 for illustration.)

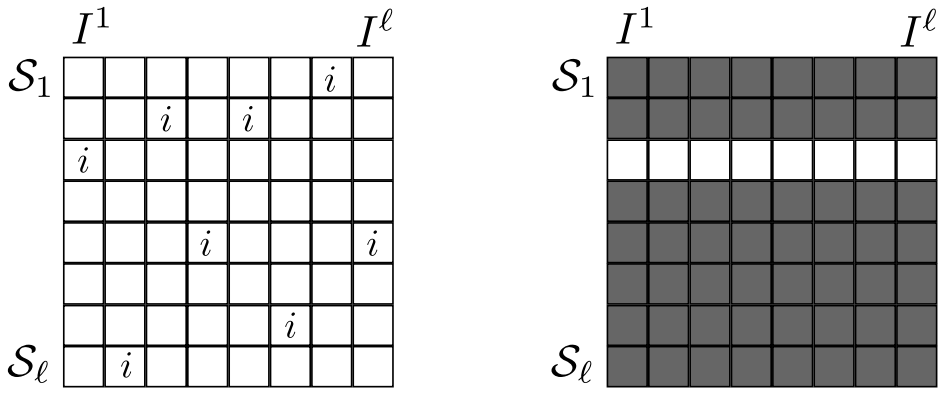
Product construction: Consider a gadget I with $|\mathcal{C}(I)|$ a (positive) power of ℓ. The *ℓ-th power* of I is obtained from I as follows. First, ℓ independently generated instances of I are presented one after the other; we refer to them as $I^1, I^2, \ldots, I^\ell$. (For example, if I is the gadget I_0 of Example 1, then this amounts to presenting ℓ^2 singleton sets supported on a universe of ℓ elements.) Recall that each I^j comes with ℓ standard solutions: $\mathcal{S}_1(I^j), \mathcal{S}_2(I^j), \ldots, \mathcal{S}_\ell(I^j)$. The *weak product* $I^{(\ell)}$ is obtained by performing the following *shuffling* and *weak binding* operations on these instances.

Shuffling: The shuffling step combines the standard solutions of the copies to produce the standard solutions of the ℓ-th power. Let $\pi_1, \pi_2, \ldots, \pi_\ell$ be random permutations of $[\ell]$, each independently and uniformly chosen from the set of $\ell!$ possibilities. The standard solutions of the new instance $I^{(\ell)}$ are defined by setting

$$\mathcal{S}_i(I^{(\ell)}) = \bigcup_{j=1}^{\ell} \mathcal{S}_{\pi_j(i)}(I^j)$$

for $i = 1, 2, \ldots, \ell$.

Weak binding: In the weak binding step, we introduce new elements (whose exact structure will be determined later on) to bind the solutions such that the following two conditions hold.



(a) weak product (b) strong product

Figure 1: A matrix view of the standard solutions $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\ell$ (rows) and the instances $I^1, I^2, \ldots, I^\ell$ (columns). In the shuffling step of the weak product construction (depicted by (a)), each permutation π_j shuffles the standard solutions $\mathcal{S}_1(I^j), \mathcal{S}_2(I^j), \ldots, \mathcal{S}_\ell(I^j)$ of I^j so that the new standard solution $\mathcal{S}_i(I^{(\ell)})$ gets one cell from each column j (the one corresponding to $\mathcal{S}_{\pi_j(i)}(I^j)$). The collating step of the strong product construction uses the identity permutation for all $j \in [\ell]$. A single row of the matrix (that corresponding to $\mathcal{S}_i(I^{[\ell]})$) is then "kept alive" by the strong binding step (depicted by (b)).

(i) If $S \in \mathcal{S}_i(I^{(\ell)})$ and $T \in \mathcal{S}_j(I^{(\ell)})$, $i \neq j$, then $S \cap T \neq \emptyset$.

(ii) For all $j \in [\ell]$ and for all $S, T \in \mathcal{S}_j$, $S \cap T = \emptyset$.

This ensures that any solution to the new instance $I^{(\ell)}$ may use sets from only one of the standard solutions $\mathcal{S}_i(I^{(\ell)})$.

We also need a slightly different product construction. The *strong product* $I^{[\ell]}$ is is obtained by performing the following *collating* and *strong binding* (instead of shuffling and weak binding) operations on the instances $I^1, I^2, \ldots, I^\ell$.

Collating: The collating is similar to the shuffling step in the construction of the weak product, only that here we use the identity permutation for $i = 1, 2, \ldots, \ell$; we then obtain ℓ families $\mathcal{S}_1(I^{[\ell]}), \mathcal{S}_2(I^{[\ell]}), \ldots, \mathcal{S}_\ell(I^{[\ell]})$ just as we did while constructing $I^{(\ell)}$ above.

Strong binding: Of the ℓ potential standard solutions, we choose one uniformly at random; denote its index by $i \in [\ell]$. We then introduce new elements (once again, their structure will be determined later on) such that the following condition holds.

(iii) $S \cap T = \emptyset$ if and only if both sets S and T are in the standard solution $\mathcal{S}_i(I^{[\ell]})$.

This ensures that any solution can save at most one set from outside $\mathcal{S}_i(I^{[\ell]})$. Note that in contrast to the weak product $I^{(\ell)}$, the strong product $I^{[\ell]}$ is not a gadget; in particular, the families $\mathcal{S}_1(I^{[\ell]}), \ldots, \mathcal{S}_{i-1}(I^{[\ell]}), \mathcal{S}_{i+1}(I^{[\ell]}), \ldots, \mathcal{S}_\ell(I^{[\ell]})$ are not valid solutions anymore.

We now turn to analyze how well a deterministic algorithm might perform on our powering products. This analysis will not depend on the implementation of the binding steps, as long as conditions

(i), (ii), and (iii) are respected (the first two for weak binding; the third one for strong binding). However, the implementation of the binding steps affects other parameters of our lower bound such as the sizes of sets, the number of elements, and their degrees. To show that the performance of the randomized algorithm established in Section 3 is nearly optimal, careful binding schemes will be designed in Sections 4.2.3 and 4.2.4.

Lemma 8. *If I is a (β, ϵ)-gadget, then the weak product $I^{(\ell)}$ is a $(\beta \cdot \frac{10 \log \ell}{\log \log \ell}, \ell^{-8} + \epsilon\ell)$-gadget.*

Proof. After $I^1, I^2, \ldots, I^\ell$ are presented to a deterministic algorithm, the sets it saves from I^j reside within just one of its standard solutions, say $\mathcal{S}_{i_j}(I^j)$. Since I is a (β, ϵ)-gadget, by the union bound, the probability that any of these solutions exceeds β is at most $\epsilon\ell$. We now present a routine *balls and bins* argument to show that our shuffling ensures that only a small number of these saved sets can fall inside any one standard solution of $I^{(\ell)}$.

When the permutations are chosen at random, the probability that t of the $\mathcal{S}_{i_j}(I^j)$'s (where the sets saved by the deterministic algorithm are confined) are mapped to any fixed standard solution of $I^{(\ell)}$ is at most

$$\binom{\ell}{t}\left(\frac{1}{\ell}\right)^t \;\leq\; \left(\frac{e\ell}{t}\right)^t\left(\frac{1}{\ell}\right)^t \;\leq\; \left(\frac{e}{t}\right)^t .$$

For $t = \left\lceil \left(\frac{10 \log \ell}{\log \log \ell}\right) \right\rceil$, the RHS of the last inequality is at most ℓ^{-9}. Finally, using the union bound (to account for all ℓ standard solutions of the new instance), we conclude that

$$\Pr\left[\max_j |\textsc{alg}(I^{(\ell)}) \cap \mathcal{S}_j(I^j)| > \beta \cdot \left\lfloor \left(\frac{10 \log \ell}{\log \log \ell}\right) \right\rfloor\right] \leq \ell \cdot \ell^{-9} + \epsilon\ell \leq \ell^{-8} + \epsilon\ell \ .$$

The assertion follows. □

Example 3. Let $I_1 = I_0^{(\ell)}$ and $I_2 = I_1^{(\ell)}$. Since I_0 is a $(1, 0)$-gadget, I_1 is an $(\frac{10 \log \ell}{\log \log \ell}, \ell^{-8})$-gadget. Consequently, I_2 is an $\left(\left(\frac{10 \log \ell}{\log \log \ell}\right)^2, 2\ell^{-7}\right)$-gadget.

Lemma 9. *If I is a (β, ϵ)-gadget, then the strong product $I^{[\ell]}$ satisfies $\mathbb{E}[|\textsc{alg}(I^{[\ell]})|] \;\leq\; 1 + \beta + \frac{\epsilon}{\ell}|\mathcal{C}(I)|$.*

Proof. Consider the sets saved by the algorithm just before the new elements are presented to it (in the strong binding step). We know that all the sets saved from I^j reside in a single standard solution $\mathcal{S}_{i_j}(I^j)$, and the expected number of such sets is at most $\beta + \epsilon|\mathcal{S}_{i_j}(I^j)| = \beta + (\frac{\epsilon}{\ell})|\mathcal{C}(I)|$. It follows that the expected total number of sets saved by the algorithm before the new elements are presented to it is at most $\ell(\beta + (\frac{\epsilon}{\ell})|\mathcal{C}(I)|)$. Now, since i is chosen uniformly at random out of $[\ell]$, only a fraction $\frac{1}{\ell}$ of these sets are mapped to $\mathcal{S}_i(I^{[\ell]})$. Once the strong binding is applied, the algorithm can save at most one set outside $\mathcal{S}_i(I^{[\ell]})$. The assertion follows. □

Example 4. I_0 is a $(1, 0)$-gadget. Thus, the expected number of sets saved by any deterministic algorithm when presented with $I_0^{[\ell]}$ is at most 2.

Example 5. I_2 is a $((\frac{10 \log \ell}{\log \log \ell})^2, 2\ell^{-7})$-gadget with ℓ^3 sets. Thus, the expected number of sets saved by any deterministic algorithm when presented with $I_2^{[\ell]}$ is at most

$$1 + \left(\frac{10 \log \ell}{\log \log \ell}\right)^2 + 2\ell^{-7}\left(\frac{\ell^3}{\ell}\right) \leq \left(\frac{10 \log \ell}{\log \log \ell}\right)^2 + 2 \ .$$

We will soon show that the bounds stated in Examples 4 and 5 are all we need. But first, we turn to bound other parameters, namely, the degrees of the elements and the sizes of the sets.

4.2.3 Implementation of the binding steps

In this section, we formally describe some constructions based on affine planes.

Lemma 10. *Let t be a prime power and let $r \leq t$.*

(A) Let $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r$ be disjoint families with t sets each. It is possible to assign t^2 new elements to the sets in these families such that (1) each element is assigned to exactly one set in each $\mathcal{F}_i$; (2) for every $i, j \in [r]$, $i \neq j$, $S \in \mathcal{F}_i$, and $T \in \mathcal{F}_j$, there is a unique new element assigned to $S \cap T$; and (3) each set receives exactly t new elements.

(B) Let $\mathcal{F}$ be a family of tr sets. It is possible to assign $t^2 + r$ new elements to the sets in this family such that (1) for every $S, T \in \mathcal{F}$, $S \neq T$, there exists a unique new element in $S \cap T$; (2) of the new elements, t^2 have degree r and the remaining r have degree t; and (3) each set is assigned exactly t degree-r elements and 1 degree-t element.

Proof. We begin with the proof of part (A). Identify $[t]$ with the field $\mathbb{F}_t$ and $[r]$ with a subset of $\mathbb{F}_t$ of cardinality r. Index the elements in $\mathcal{F}_j$ using elements of $[t]$, referring to its i-th element as $\mathcal{F}_j(i)$. The t^2 elements we introduce will be called e_{ab}, where a, b range over $\mathbb{F}_t$. The element e_{ab} is assigned to the sets $\mathcal{F}_j(aj + b)$, $j \in [r]$. (Recall that we view $[r]$ as a subset of $\mathbb{F}_t$, and so we may perform field arithmetic using the index $j \in [r]$.) It is easy to verify that this assignment has the required properties; in particular, the unique element assigned to $\mathcal{F}_j(i) \cap \mathcal{F}_{j'}(i')$ is e_{ab}, where $a = (i' - i)/(j' - j)$ and $b = -(i'j - ij')/(j' - j)$.

For part (B), partition $\mathcal{F}$ into r equal parts $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r$ and introduce t^2 new elements of the form e_{ab}, as in part (A). Further, add r new elements, $f_1, f_2, \ldots, f_r$, and assign f_j to the t sets in $\mathcal{F}_j$. The assertion follows trivially. □

Now, consider the weak binding step used in constructing the gadget $I^{(\ell)}$ from I. Let s denote the number of sets in I. Recall that s is a (positive) power of ℓ, hence s is also a prime power. New elements are introduced as suggested by Lemma 10(A) with $t \leftarrow s$, $r \leftarrow \ell$, and the families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r$ replaced by the standard solutions $\mathcal{S}_1(I^{(\ell)}), \mathcal{S}_2(I^{(\ell)}), \ldots, \mathcal{S}_\ell(I^{(\ell)})$. This immediately yields the following.

Corollary 11. *Let I be a gadget with $|\mathcal{C}(I)| = s$ sets. We can implement the weak binding step needed for constructing $I^{(\ell)}$ by introducing s^2 new degree-ℓ elements, in such a way that each set gets s new elements.*

For the strong binding step, we use two different implementations that will be presented in the next section.

4.2.4 Optimality

We now revisit the constructions described in the aforementioned examples. Corollary 11 yields the following account of the weak products.

Gadget	Sets	Elements	OPT	$\mathbb{E}[\text{ALG}]$
$I_2^{[\ell]}$	$m = \ell^4$, $k = 2\ell^2 + \ell + 2$	$n = 2\ell^5(1 + o(1))$ $\sigma_{\max} = \ell^2 - \ell$, $\bar{\sigma} = \theta(\ell)$	ℓ^3	$O\left(\left(\frac{\log \ell}{\log \log \ell}\right)^2\right)$
$I_0^{[\ell]}$	$m = \ell^2$, $k = 2$	$n = \ell + 1$ $\sigma_{\max} = \ell^2 - \ell$, $\bar{\sigma} \leq 2\ell + 1$	ℓ	≤ 2

Table 1: The parameters of the random OSP instances corresponding to the gadgets $I_2^{[\ell]}$ and $I_0^{[\ell]}$. The fourth column presents the size of an optimal solution (of any OSP instance in the support), while the fifth column presents the expected size of the solution returned by any deterministic algorithm ALG.

- The gadget I_0 consists of ℓ sets of size 1 and a single element of degree ℓ.
- The gadget $I_1 = I_0^{(\ell)}$ consists of ℓ^2 sets. In addition to the ℓ original elements, it also contains ℓ^2 new elements. The size of each set is $\ell + 1$; the degree of each new element is ℓ.
- The gadget $I_2 = I_1^{(\ell)}$ consists of ℓ^3 sets. In addition to the $\ell \cdot (\ell^2 + \ell) = \ell^3 + \ell^2$ original elements, it also contains ℓ^4 new elements. The size of each set is $\ell^2 + \ell + 1$; the degree of each new element is ℓ.

The strong binding step is used twice in our examples: once in the construction of $I_0^{[\ell]}$ and once in the construction of $I_2^{[\ell]}$. We now turn to present how these constructions are implemented, starting with the former. $I_0^{[\ell]}$ consists of ℓ independent instances of I_0. After the collating step, we have ℓ^2 sets, arranged in ℓ disjoint families $\mathcal{S}_1(I_0^{[\ell]}), \mathcal{S}_2(I_0^{[\ell]}), \ldots, \mathcal{S}_\ell(I_0^{[\ell]})$ so that the sets within the same family $\mathcal{S}_j(I_0^{[\ell]})$ are pairwise disjoint. All we have to do now in order to implement the strong binding step is to pick some $i \in [\ell]$ uniformly at random and to introduce a single new element appearing in every set $S \in \mathcal{S}_j(I_0^{[\ell]})$ such that $j \neq i$. In fact, the second OSP instance in our introductory Example 1 does exactly that, hence it can be described as $I_0^{[\ell]}$. Note that the sets in $\mathcal{S}_j(I_0^{[\ell]})$, $j \neq i$, are of size 2 while the sets in $\mathcal{S}_i(I_0^{[\ell]})$ are of size 1. To keep the set sizes uniform, we assign one additional new element to each set in $\mathcal{S}_i(I_0^{[\ell]})$ (a total of ℓ additional degree-1 elements).

Now, consider the random OSP instance $I_2^{[\ell]}$. Recall that it consists of ℓ^4 sets, of which ℓ^3 belong to $\mathcal{S}_i$. We apply Lemma 10(B) with $t \leftarrow \ell^2$, $r \leftarrow \ell^2 - \ell$, and the family $\mathcal{F}$ replaced by the union $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i-1} \cup \mathcal{S}_{i+1} \cup \cdots \cup \mathcal{S}_\ell$ (containing $(\ell - 1) \cdot \ell^3 = \ell^2 \cdot (\ell^2 - \ell)$ sets). The lemma guarantees that $S \cap T \neq \emptyset$ for every two sets S, T such that $S, T \notin \mathcal{S}_i$. Since the new elements do not involve the sets of $\mathcal{S}_i$, the design of the collating step implies that $S \cap T = \emptyset$ for every two sets $S, T \in \mathcal{S}_i$. In addition to the $\ell \cdot (\ell^4 + \ell^3 + \ell^2) = \ell^5 + \ell^4 + \ell^3$ original elements (each of degree ℓ), $O(\ell^4)$ new elements of degree $O(\ell^2)$ are introduced so that each set in $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i-1} \cup \mathcal{S}_{i+1} \cup \cdots \cup \mathcal{S}_\ell$ gets $\ell^2 + 1$ new elements. To keep the set sizes uniform, we introduce $\ell^2 + 1$ additional new elements, each of degree 1, for every set in $\mathcal{S}_i$ (a total of $\ell^3 \cdot (\ell^2 + 1) = \ell^5 + \ell^3$ additional new elements).

Combined with Lemmas 8 and 9, we obtain the parameters depicted by Table 1. Theorem 3 is now established by Yao's principle as cast in the following two corollaries.

Corollary. *For every randomized* OSP *algorithm* ALG *and for every ℓ_0, there exist some $\ell \geq \ell_0$ and some instance I in the support of $I_2^{[\ell]}$ with sets of size $k = \Theta(\ell^2)$ and maximum degree $\sigma_{\max} = \Theta(\ell^2)$,*

such that $\text{OPT}(I) = \ell^3 = \Omega(k_{\max}\sqrt{\sigma_{\max}})$ *and* $\mathbb{E}[\text{ALG}(I)] = O\left(\left(\frac{\log \ell}{\log\log \ell}\right)^2\right) = \tilde{O}(1)$.

Corollary. *For every randomized* OSP *algorithm* ALG *and for every* ℓ_0*, there exist some* $\ell \geq \ell_0$ *and some instance* I *in the support of* $I_0^{[\ell]}$ *with sets of size* $k = 2$ *and maximum degree* $\sigma_{\max} = \Theta(\ell^2)$*, such that* $\text{OPT}(I) = \ell = \Omega(k_{\max}\sqrt{\sigma_{\max}})$ *and* $\mathbb{E}[\text{ALG}(I)] = O(1)$.

5 Conclusions and Open Problems

We have introduced a new variant the online set packing problem and presented a competitive algorithm that solves it. Many questions remain open in this area. We mention a few major ones.

- It seems interesting to generalize the problem to arbitrary packing problems, where the entries in the matrix are arbitrary non-negative integers.
- Recalling the networking motivation, it is interesting to understand the effect of buffers on the problem.
- We studied the scenario where the benefit of a set is collected by the algorithm only if all its elements were assigned to it. In some cases, the benefit may be collected even if some elements are missing. What is the effect on the upper and lower bounds in this case? (cf. [15]).

Acknowledgments

We thank Geir Agnarsson and Bjarni V. Halldórsson for helpful discussions.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proc. 35th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 100–105. ACM, 2003.

[2] N. Alon, U. Arad, and Y. Azar. Independent sets in hypergraphs with applications to routing via fixed paths. In *Proc. 2nd Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pp. 16–27, 1999.

[3] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 1992.

[4] B. Awerbuch and R. Khandekar. Stateless distributed gradient descent for positive linear programs. *SIAM J. Comput.*, 38(6):2468–2486, 2009.

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

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

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

[8] M. M. Halldórsson, K. Iwama, S. Miyazaki, and S. Taketomi. Online independent sets. *Theoretical Computer Science*, 289(2):953–962, 2002.

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

[10] B. V. Halldórsson, M. M. Halldórsson, E. Losievskaja, and M. Szegedy. Streaming algorithms for independent sets. In *Proc. 37th Intl. Colloq. on Algorithms, Languages and Programming (ICALP)*, pages, 641–652, 2010.

[11] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-dimensional matching. In *Proc. 6th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 83–97, 2003.

[12] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Disc. Math.*, 2(1):68–72, 1989.

[13] A. Kako, T. Ono, T. Hirata, and M. M. Halldórsson. Approximation algorithms for the weighted independent set problem in sparse graphs. *Discrete Applied Mathematics*, 157(4):617–626, 2009.

[14] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 352–358. ACM, 1990.

[15] A. Kesselman, B. Patt-Shamir, and G. Scalosub. Competitive buffer management with packet dependencies. In *Proc. 23rd IEEE Int. Symp. on Parallel and Distributed Processing (IPDPS) 2009, Rome, Italy, May 23-29, 2009*, pages 1–12. IEEE, 2009.

[16] V. F. Kolchin, B. A. Sevastyanov, and V. P. Chistyakov. *Random Allocations*. Winston & Sons, Washington, 1978.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 980–989, 2006.

[18] C. H. Papadimitriou and M. Yannakakis. Linear programming without the matrix. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 121–129, 1993.

[19] M. Raab and A. Steger. "Balls into Bins" - A Simple and Tight Analysis. In *Proc. 2nd Intl. Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 159–170, 1998.

[20] S. Sakai, M. Togasaki, K. Yamazaki. A note on greedy algorithms for maximum weighted independent set problem. *Discrete Applied Mathematics*, 126:313–322, 2003.

[21] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[22] H. Shachnai and A. Srinivasan. Finding Large Independent Sets of Hypergraphs in Parallel. *SIAM J. Discrete Math.*, 18(3):488–500, 2005.

[23] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 222–227, 1977.