New Bounds for Online Packing LPs*

Matthias Englert[1], Nicolaos Matsakis[1], and Marcin Mucha[2]

[1] DIMAP and Dept. of Computer Science, University of Warwick,
{M.Englert,N.Matsakis}@warwick.ac.uk.
[2] Institute of Informatics, University of Warsaw, mucha@mimuw.edu.pl

Abstract. Solving linear programs online has been an active area of research in recent years and was used with great success to develop new online algorithms for a variety of problems. We study the setting introduced by Ochel et al. as an abstraction of lifetime optimization of wireless sensor networks.
In this setting, the online algorithm is given a packing LP and has to monotonically increase LP variables in order to maximize the objective function. However, at any point in time, the adversary only provides an α-approximation of the remaining slack for each constraint. This is designed to model scenarios in which only estimates of remaining capacities (e.g. of batteries) are known, and they get more and more accurate as the remaining capacities approach 0.
Ochel et al. (ICALP'12) gave a $\Theta(\ln \alpha/\alpha)$-competitive online algorithm for this online packing LP problem and showed an upper bound on the competitive ratio of any online algorithm, even randomized, of $O(1/\sqrt{\alpha})$. We significantly improve the upper bound and show that any deterministic online algorithm for LPs with d variables is at most $O(d^2\alpha^{1/d}/\alpha)$-competitive. For randomized online algorithms we show an upper bound of $O(m^2\alpha^{1/m}/\alpha)$ for LPs with $m^{m!\ln\alpha}$ variables. For LPs with sufficiently many variables, these bounds are $O(\ln^2\alpha/\alpha)$, nearly matching the known lower bound.
On the other hand, we also show that the known lower bound can be significantly improved if the number of variables in the LP is small. Specifically, we give a deterministic $\Theta(1/\sqrt{\alpha})$-competitive online algorithm for packing LPs with two variables. This is tight, since the previously known upper bound of $O(1/\sqrt{\alpha})$ still holds for 2-dimensional LPs.

1 Introduction

In recent years, there has been great interest in methods for solving linear programs online, mainly to facilitate the development of new online algorithms with improved competitive guarantees. Buchbinder and Naor [3] give a general online primal-dual approach to (approximately) solve the following type of packing (and the dual covering version) linear program online.

* The first and second author are supported by the Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, EPSRC award EP/D063191/1. The third author is supported by NCN grant N N206 567940.

The underlying packing LP is of the form

$$\begin{aligned} \max \quad & b_1 x_1 + \ldots + b_d x_d \\ \text{subject to} \quad & A \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \leq \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} \\ & x_1, \ldots, x_d \geq 0 \ , \end{aligned}$$

where A is matrix with non-negative entries, and all b_i and c_i are positive.

In the online version, an online algorithm plays against an adversary which only reveals entries of b and A online. More specifically, the online algorithm is only allowed to increase variables, not decrease them. The algorithm also must guarantee that all LP constraints are satisfied. The vector c is given to the algorithm upfront, but b and A are initially hidden. The adversary then successively reveals all coefficients of a variable x_j at a time of its choosing, i.e., in round j, b_j and, for all i, A_{ij} are revealed to the online algorithm. The goal is to maximize the objective function $b^T x$.

The primal-dual technique by Buchbinder and Naor and their extensions have been applied with great success to develop new improved online algorithms for a variety of online problems, among them the k-server problem [2] and generalized caching [1].

Ochel, Radke, and Vöcking [5] introduce a related model in which b and A are initially given to the online algorithm and instead c is only gradually revealed. At each point in time, the adversary reveals a vector ℓ^t, which can be seen as the current right hand side values of the LP, and the online algorithm responds by increasing variables x_i. The algorithm may never decrease a variable and has to ensure that the constraint $Ax \leq \ell^t$ is satisfied. This is not possible without imposing further restrictions on ℓ^t. Therefore we require that

1. The revealed ℓ^t are (component wise) lower bounds on c, i.e., $\ell^t \leq c$.
2. If x is the current online solution, the next revealed vector ℓ^t has to satisfy $(c - Ax) \leq \alpha(\ell^t - Ax)$.

In other words, the remaining slacks of constraints, if ℓ^t is taken as the right hand side, are an α-approximation of the remaining slacks with respect to the *true* right hand side c. We call $\ell^t - Ax$ the *revealed remaining slacks* and $c - Ax$ the *true remaining slacks* and study the performance of online algorithms in dependence of the problem parameter α.

Ochel et al. [5] give the problem of lifetime optimization in wireless sensor networks as a motivating application (see, e.g., [4]). There, the right hand sides of the constraints correspond to battery lifetimes of sensors. We only know the lower bounds on the remaining lifetimes, but the true values are always within a fixed factor of the revealed values. Given this information, we need to choose among a set of broadcasting scenarios in a way that maximizes the number of broadcasts performed before empty batteries prevent any further broadcasts.

1.1 Our results

Ochel et al. [5] give a $\Theta(\ln \alpha/\alpha)$-competitive online algorithm for their online packing LP problem and show an upper bound on the competitive ratio of any online algorithm, even randomized, of $O(1/\sqrt{\alpha})$.

We significantly improve the upper bound. For LPs involving d or more variables we show an upper bound of $O(d^2\alpha^{1/d}/\alpha)$ on the competitive ratio of any deterministic algorithm. At the cost of increasing the number of variables, we obtain a similar bound on the competitive ratio of any randomized algorithm against an oblivious adversary. With $m^{\lceil m! \ln \alpha \rceil}$ or more variables we construct an upper bound of $O(m^2\alpha^{1/m}/\alpha)$.

For $d = \Omega(\ln \alpha)$ in the case of deterministic algorithms and $d = \Omega(\alpha^{(\ln \alpha)!})$ in the case of randomized algorithms, this results in an upper bound of $O(\ln^2 \alpha/\alpha)$, which nearly matches the known lower bound by Ochel et al.

However, we also demonstrate that the achievable competitive ratio crucially depends on the number of variables d in the LP. We give a simple $\Theta(1/\sqrt{\alpha})$-competitive deterministic online algorithm that beats the general upper bound of $O(\ln^2 \alpha/\alpha)$ for LPs only involving two variables. This is tight, since the previously known general upper bound of $O(1/\sqrt{\alpha})$ still holds for 2-dimensional LPs.

The paper is organized as follows. In Section 2 we give the upper-bound on the competitive ratio achievable by deterministic algorithms. The techniques developed in the process are then extended in Section 3 to handle randomized algorithms. In Section 4 we describe and analyze an $O(1/\sqrt{\alpha})$-competitive algorithm for 2-dimensional LPs. We end with conclusions and open problems in Section 5.

2 Deterministic upper bound

In this section, we describe our upper bound construction and prove the following theorem.

Theorem 1. *The competitive ratio of any deterministic online algorithm is at most $O(d^2\alpha^{1/d}/\alpha)$, for LPs involving d or more variables.*

Note that the bound in the claim above is minimized for $d = \Theta(\ln \alpha)$. Using this value, gives us the following corollary.

Corollary 1. *The competitive ratio of any deterministic online algorithm is at most $O(\ln^2 \alpha/\alpha)$.*

We proceed with the construction of the adversary to prove Theorem 1. The construction will use exactly d variables. The theorem follows since any additional variables $x_{d+1}, x_{d+2}, \ldots$ can be made irrelevant by adding constraints of the form $x_{d+1} \leq 0, x_{d+2} \leq 0, \ldots$

The basic idea behind our construction is to present an LP to the online algorithm that is completely symmetric. Once the online algorithm has increased

some variable x_i so much that the revealed remaining slacks of some constraints become small, the adversary decides that these are exactly the constraints for which the revealed right hand sides were already quite close to the true right hand sides. As a consequence, x_i cannot be increased much further in the future and the online algorithm is, more or less, left with a similarly constructed input for the remaining $d-1$ variables.

The initial linear program presented to the online algorithm is

$$\begin{aligned} & \max && \sum_{i=0}^{d-1} x_i \\ \forall \text{ permutations } \pi : & && \sum_{i=0}^{d-1} \alpha^{i/d} \cdot x_{\pi(i)} \leq \alpha \\ & && x_0, x_1, \ldots, x_{d-1} \geq 0 \ . \end{aligned}$$

The adversary maintains a set of *active* constraints and a set of *active* variables. Initially all $d!$ constraints and all d variables are active.

The adversary proceeds in d rounds numbered $d-1, d-2, \ldots, 0$. Round r ends in the first step in which there exists an active constraint with a revealed remaining slack of at most $\alpha^{r/d}$. At the end of a round, the adversary does the following:

- Determine the index k_r of an active variable of maximum value among the active variables.
- Increase the right hand side of all active constraints that do not correspond to permutations with $\pi(r) = k_r$ to $\alpha \cdot \alpha^{r/d}$. Note that this is possible, i.e., this does not violate the condition that revealed remaining slacks always have to be an α-approximation of the true remaining slacks, since before this point in time, all constraints have a revealed remaining slack of at least $\alpha^{r/d}$.
- Remove all these constraints from the set of active constraints.
- Remove the variable with index k_r from the set of active variables.

Remark 1. It might happen that the online algorithm ends its execution before all rounds are completed. In this case, the adversary still executes the steps above (without waiting for slacks of constraints to trigger the next round).

We start our proof with two easy observations. At the end of round r, we remove all constraints with $\pi(r) \neq k_r$ from the set of active constraints. Since round numbers are decreasing, we have the first observation.

Observation 2 *A permutation π corresponds to a constraint active in round r iff $\pi(s) = k_s$ for all $s > r$.*

Let x_i^r be the value of the variable x_i at the end of round r, for $i, r \in \{0, \ldots, d-1\}$. Since for non-negative numbers $a_1 \geq a_2 \geq \ldots$ and non-negative numbers b_i, $\sum_i a_i \cdot b_{\pi(i)}$ is maximized if $b_{\pi(1)} \geq b_{\pi(2)} \geq \ldots$ we get the second observation.

Observation 3 *Let $r = 0, \dots, d-1$ be a round. Also, let π be a permutation corresponding to a constraint active in round r and such that*

$$x^r_{\pi(r)} \geq x^r_{\pi(r-1)} \geq \dots \geq x^r_{\pi(0)}.$$

Note that such a permutation exists due to Observation 2. Then, the constraint corresponding to π has the smallest revealed remaining slack among all permutations active in round r.

For any $r = 0, \dots, d-1$, let π_r be the permutation corresponding to the constraint that causes round r to end. Due to Observation 3,

$$x^r_{\pi_r(r)} \geq x^r_{\pi_r(r-1)} \geq \dots \geq x^r_{\pi_r(0)}$$

and therefore, we may assume without loss of generality that $\pi_r(r) = k_r$.

Lemma 1. *For any round $r < d-1$ we have*

$$x^r_{\pi_r(i)} \geq x^{r+1}_{\pi_{r+1}(i)}$$

for any $i = 0, \dots, d-1$

Proof. Due to Observation 2, we have $\pi_r(i) = \pi_{r+1}(i) = k_i$ for $i > r+1$ since the constraints corresponding to π_r and π_{r+1} are both active in rounds $d-1, \dots, r+1$. Since $\pi_r(i)$ is also active in round r and $\pi_{r+1}(r+1) = k_{r+1}$, we also have $\pi_r(i) = \pi_{r+1}(i) = k_i$ for $i = r+1$. Variables can only increase, this implies for $i \geq r+1$, $x^r_{\pi_{r+1}(i)} = x^r_{k_i} \geq x^{r+1}_{k_i} = x^{r+1}_{\pi_{r+1}(i)}$.

It remains to prove the lemma for $i \leq r$. Again because $\pi_r(i) = \pi_{r+1}(i) = k_i$ for $i > r+1$, the sequence $\pi_{r+1}(r+1), \pi_{r+1}(r), \dots, \pi_{r+1}(0)$ is a permutation of the sequence $\pi_r(r+1), \pi_r(r), \dots, \pi_r(0)$. Therefore, the sets of variables $\{x_{\pi_{r+1}(r+1)}, \dots, x_{\pi_{r+1}(0)}\}$ and $\{x_{\pi_r(r+1)}, \dots, x_{\pi_r(0)}\}$ are identical. Hence, since variables can only increase, the sequence

$$x^r_{\pi_r(r+1)} \geq x^r_{\pi_r(r)} \geq \dots \geq x^r_{\pi_r(0)}$$

is obtained from

$$x^{r+1}_{\pi_{r+1}(r+1)} \geq x^{r+1}_{\pi_{r+1}(r)} \geq \dots \geq x^{r+1}_{\pi_{r+1}(0)}$$

by increasing the values of some variables and rearranging the sequence so that it is sorted. The claim follows. □

We can now show the key lemma.

Lemma 2. *For any $r \in \{0, \dots, d-1\}$ and $i \leq r$, we have $x^r_{\pi_r(i)} \leq (d-r)\alpha^{1/d}$.*

Proof. We use downward induction on r. The claim is clear for $r = d-1$, since during round $d-1$ all constraints still have right hand sides equal to α and for any variable x_i there is a constraint that contains this variable with a coefficient of $\alpha^{1-1/d}$.

Consider now any round $r < d-1$. We have

$$\sum_{i=0}^{d-1} \alpha^{i/d} \cdot x^r_{\pi_r(i)} = \sum_{i=0}^{d-1} \alpha^{i/d} \cdot x^{r+1}_{\pi_{r+1}(i)} + \sum_{i=0}^{d-1} \alpha^{i/d} \cdot \left(x^r_{\pi_r(i)} - x^{r+1}_{\pi_{r+1}(i)} \right).$$

The first sum is lower bounded by $\alpha - \alpha^{(r+1)/d}$ by the definition of π_{r+1}. Moreover, by Lemma 1 we have $x^r_{\pi_r(i)} \geq x^{r+1}_{\pi_{r+1}(i)}$ for any $i = 0, \ldots, d-1$. Therefore

$$\sum_{i=0}^{d-1} \alpha^{i/d} \cdot x^r_{\pi_r(i)} \geq \alpha - \alpha^{(r+1)/d} + \sum_{i=0}^{d-1} \alpha^{i/d} \cdot \left(x^r_{\pi_r(i)} - x^{r+1}_{\pi_{r+1}(i)} \right).$$

with all the terms in the right sum being non-negative. Since the constraint corresponding to π_r is active and feasible in round r, the left hand side is upper-bounded by α. By ignoring all terms except the one corresponding to $i = r$ in the sum of the right hand side we obtain

$$x^r_{\pi_r(r)} \leq x^{r+1}_{\pi_{r+1}(r)} + \frac{\alpha^{(r+1)/d}}{\alpha^{r/d}} = x^{r+1}_{\pi_{r+1}(r)} + \alpha^{1/d} \leq (d-r)\alpha^{1/d},$$

where the last inequality follows from induction.

The claim for $x^r_{\pi_r(i)}$ with $i < r$ follows as well, since $x^r_{\pi_r(0)} \leq \cdots \leq^r_{\pi_r(r)} \leq (d-r)\alpha^{1/d}$. □

Lemma 3. *After the online algorithm terminates we have $x_i = O(d\alpha^{1/d})$ for all $i = 0, \ldots, d-1$.*

Proof. Let r be the last round that is fully performed. Consider any permutation π corresponding to a constraint that is active when the algorithm ends. From Observation 2 we know that these are exactly the constraints satisfying $\pi(t) = k_t$ for all $t \geq r$. Choose one such π so that it also satisfies

$$x_{\pi(r-1)} \geq x_{\pi(r-2)} \geq \ldots \geq x_{\pi(0)},$$

where x_i is the final value of a variable.

We will first prove the claim for variables that are not active when the algorithm ends. Any such variable has index $k_t = \pi_t(t) = \pi(t)$ for some $t \geq r$. We have

$$\sum_{i=0}^{d-1} \alpha^{i/d} \cdot x^t_{\pi_t(i)} \geq \alpha - \alpha^{t/d},$$

and similarly to the reasoning in the the proof of Lemma 2 we can argue that

$$x_{k_t} \leq x^t_{k_t} + \frac{\alpha^{t/d}}{\alpha^{t/d}} \leq (d-t)\alpha^{1/d} + 1.$$

Consider now the variables that are still active when the algorithm ends, i.e. $x_{\pi(i)}$ for $i < r$. It is enough to prove the claim for $x_{\pi(r-1)}$, since it is the largest one. We have

$$\sum_{i=0}^{d-1} \alpha^{i/d} \cdot x^r_{\pi_r(i)} \geq \alpha - \alpha^{r/d},$$

and again using the reasoning from the proof of Lemma 2 we get that

$$x_{\pi(r-1)} \leq x^r_{\pi_r(r-1)} + \frac{\alpha^{r/d}}{\alpha^{(r-1)/d}} \leq (d-r)\alpha^{1/d} + \alpha^{1/d} = (d-r+1)\alpha^{1/d}. \quad \square$$

It remains to show a lower bound on the profit an optimal offline strategy can achieve.

Lemma 4. *An optimal offline algorithm can obtain a profit of α.*

Proof. An offline algorithm can set x_{k_0} to α and all other variables to 0. This solution is feasible.

For any $i \geq 1$, consider any of the $(d-1)!$ constraints in which x_{k_0} has a coefficient of $\alpha^{i/d}$. In other words, a constraint corresponding to a permutation with $\pi(i) = k_0$. Such a constraint becomes inactive by the end of round i the latest, since $\pi(i) = k_0 \neq k_i$. Due to the adversary's strategy this means that, once the constraint becomes inactive, the right hand side increases to at least $\alpha \cdot \alpha^{i/d}$ and therefore the constraint is satisfied.

Constraints in which the coefficient of x_{k_0} is 1 are clearly satisfied as well, since the right hand side of all constraints is at least α. $\square$

By combining Lemma 3 and Lemma 4 we obtain Theorem 1.

3 Randomized upper bound

The upper bound on the competitive ratio of randomized online algorithms against oblivious adversaries is based on the construction from the previous section and uses a technique that is, at least implicitly, also used by Ochel et al. [5]. Recall that each round ends when the slack of at least one active constraint drops below a certain threshold value. The adversary then identifies the offending (i.e. largest) active variable and renders all constraints, except for those in which this variable appears with a certain coefficient, irrelevant. This is done by increasing the right hand side of these constraints sufficiently. The offending variable becomes inactive and cannot be further increased by much anymore.

To obtain our upper bound on randomized algorithms, we use the standard approach based on Yao's min-max principle; instead of proving bounds for randomized algorithms, we construct a distribution on the inputs that (in expectation) foils any deterministic algorithm.

Technical issues. Our previous problem description involves a constant interaction between the online algorithm and an adversary; each increase of LP variables by the algorithm is followed by an update of the right hand sides of the LP constraints by the adversary.

In order to define oblivious adversaries, we need to remove this interaction and allow the adversary to specify its complete behavior upfront. For this, the input consists, as before, of a packing LP whose constraints are given by $Ax \leq c$. Additionally, for each constraint i, an adversary specifies a monotonically

increasing function $\ell_i(\lambda_i)$ of the left hand side of the constraint $\lambda_i := (Ax)_i$. This function models the right hand side of the constraint in dependence of the current value of the left hand side. The function has to satisfy $\ell_i(\lambda_i) = c_i$ for $\lambda_i \geq c_i$ and $c_i - \lambda_i \leq \alpha(\ell_i(\lambda_i) - \lambda_i)$ for $\lambda_i < c_i$.

If λ_i is the current value of the left hand side of the i-th constraint, the online algorithm does know all values of $\ell_i(z)$ for $z \leq \lambda_i$ but does not know any values for $z > \lambda_i$.

Note that we only need basic threshold functions to construct the deterministic upper bound from the previous section. Define functions $f_j(\lambda)$, for $j \in [0, d-1]$, as

$$f_j(\lambda) := \begin{cases} \alpha & \lambda < \alpha - \alpha^{j/d} \\ \alpha^{1+j/d} & \lambda \geq \alpha - \alpha^{j/d} \end{cases}.$$

The LP is the same as the one in the previous section. The monotonically increasing function assigned to a constraint that corresponds to permutation π, is f_r if r is the largest integer such that $\pi(r) \neq k_r$. (Recall that k_r is the index of the largest active variable at the end of round r.) If no such integer exist, f_0 is assigned to the constraint.

This way we can achieve the same upper bound as in the previous section, but with an explicit, predefined behavior of the adversary. Two things need to be noted here. First of all, in order to actually foil an online algorithm with a fixed adversary one needs to know the k_r values. Therefore, we might need a different fixed adversary for different algorithms. Second, even if we do know the k_r values, the resulting adversary is not identical to the one from the previous section. Adversaries from the previous section always increase the right-hand sides of constraints as soon as they know what their final values should be. The adversaries defined here delay the increase (almost) as long as possible. However, it can be easily verified that the bound of Theorem 1 still holds, as long as the adversary is built with the correct k_r values.

Parallel adversaries. In the construction of the previous section, the order in which the variables become inactive defines a permutation π_o of $\{0, \ldots, d-1\}$. In other words, if the variable with index k_r becomes inactive in round r we have $\pi_o(r) = k_r$.

Consider an adversary that acts in exactly the same way as before, but it guesses the permutation π_o and proceeds as if the variables would actually become inactive in this order. If the adversary guesses correctly, the construction works as intended and the algorithm can only obtain a value of $O(d^2\alpha^{1/d})$, while the optimum value is α. If the adversary chooses the permutation uniformly at random, the success probability is $1/(d!)$, i.e., with this probability we achieve the same upper bound as in the previous section. However, if the adversary guesses incorrectly the algorithm can perform better than $O(d^2\alpha^{1/d})$, perhaps even obtain a value of α, while the optimal value remains α. We are not going to attempt to analyze the average performance of an algorithm in this setting. Instead, we will increase the success probability for the adversary from $1/(d!)$ to about $1 - 1/\alpha$.

The idea is to have K adversaries with random π_o sequences working in parallel and require the algorithm to beat all of them, for some sufficiently large K. This is done as follows. Suppose we want to have K d-dimensional adversaries. We construct a packing program with d^K variables $\{x_{i_1,\ldots,i_K} | 0 \leq i_1, \ldots, i_K \leq d-1\}$. The objective function is the sum of all variables.

For the k-th adversary, we add $d!$ constraints to the LP. The constraints have the same form as the ones in the previous section but instead of variables x_j the variables are

$$x_j^k := \sum_{(i_1,\ldots,i_K):i_k=j} x_{i_1,\ldots,i_K} \ .$$

For each adversary k, a permutation π^k of $\{0,\ldots,d-1\}$ is chosen independently and uniformly at random and the functions f_r are randomly assigned to constraints based on this permutation as described earlier.

Note that, for any adversary k, the objective function is

$$\sum_{0\leq i_1,\ldots,i_K\leq d-1} x_{i_1,\ldots,i_K} = \sum_{j=0}^{d-1} x_j^k \ .$$

Therefore the objective function is also equal to $\min_k \sum_{j=0}^{d-1} x_j^k$.

We now allow the online algorithm to increase x_j^k directly instead of increasing the underlying $x_{i_1,\ldots,i_K}$ variables. Note that, in reality, the algorithm cannot increase the x_j^k variables completely independently of each other, since increasing one of the underlying $x_{i_1,\ldots,i_K}$ variables always affects multiple x_j^k variables. However, allowing the algorithm to directly and individually increase x_j^k variables can only give the online algorithm more power.

This completes the construction of an input that combines K adversaries from the previous section, each of them independently guessing a random order in which variables becomes inactive. The profit of the online algorithm against one of the adversaries is bounded by $O(d^2\alpha^{1/d})$ with probability $1/(d!)$ (namely if the adversary guessed the correct permutation) and by α otherwise.

The total profit of the online algorithm is bounded by the minimum profit the algorithm achieves against any of the K adversaries. By choosing $K = \lceil d! \ln \alpha \rceil$ we get that the expected overall profit of the online algorithm is bounded by

$$\left(1-\frac{1}{d!}\right)^K \alpha + \left(1-\left(1-\frac{1}{d!}\right)^K\right) O(d^2\alpha^{1/d}) = O(d^2\alpha^{1/d}) \ .$$

The optimal profit is always α. To see this set $x_{\pi_1(0),\ldots,\pi_K(0)} = \alpha$ and all other variables to 0. Consider the constraints that belong to the k-th adversary. Then x_j^k is equal to α for $j = \pi^k(0)$ and 0 otherwise. This is exactly the feasible solution from Lemma 4, applied to the constraints of the k-th adversary. Altogether, this gives the desired theorem.

Theorem 4. *The competitive ratio of any randomized online algorithm against an oblivious adversary is at most $O(m^2\alpha^{1/m}/\alpha)$ for LPs involving $m^{\lceil m! \ln \alpha \rceil}$ variables.*

4 Tight lower bound for two dimensions

In this section we give a simple deterministic $\Theta(1/\sqrt{\alpha})$-competitive algorithm for packing linear programs involving two variables x_1 and x_2.

For convenience, in a first step, the algorithm normalizes variables such that the objective function is $x_1 + x_2$ instead of the more general form $b_1x_1 + b_2x_2$, with positive b_1 and b_2. For this we divide all entries a_{ij} of the constraint matrix by b_j. This does not change the profit of an optimal solution. In fact, an increase of x_i by ε in the normalized LP exactly corresponds to an increase of x_i by ε/b_i in the old LP. Both of these increases would increase the objective function of the respective LP by the same amount, namely ε.

Now, let $x^*(z) = (x_1^*(z), x_2^*(z))$ denote an optimal solution of the linear program where the capacities, that is, the right hand sides of constraints, are given by the vector z. Let $x = (x_1, x_2)$ be the current online solution.

At time t, our algorithm ALG, which uses a parameter γ, does the following:

1. If any constraint is tight, STOP.
2. Else if $x_1^*(\ell^t) > x_1\gamma$, increase x_1 infinitesimally.
3. Else if $x_2^*(\ell^t) > x_2\gamma$, increase x_2 infinitesimally.
4. Else, STOP.

Theorem 5. *For $\gamma = 1 + 1/\sqrt{\alpha}$, the optimum profit is at most $(\sqrt{\alpha} + 1)$ times the profit obtained by ALG.*

Proof. Let $x' = (x_1', x_2')$ indicate the point where *ALG* stops. Since *ALG* will either stop due to Step 1 or Step 4, we distinguish these two cases:

ALG stops due to Step 1. There is a tight constraint k. Let us write this constraint as $a_1x_1 + a_2x_2 \leq \ell_k^t$. Since $(c - Ax) \leq \alpha(\ell^t - Ax)$, this implies $a_1x_1' + a_2x_2' = c_k$.

Assume, without loss of generality, that $a_1 \geq a_2$. Then, the optimum profit can be at most c_k/a_2.

For every point in time t, $x^*(\ell^t)$ has to satisfy $a_1x_1^*(\ell^t) + a_2x_2^*(\ell^t) \leq \ell_k^t \leq c_k$. Therefore, $x_1^*(\ell^t) \leq c_k/a_1$. Additionally, since ALG increases variable x_1 only if $x_1\gamma < x_1^*(\ell^t) \leq c_k/a_1$, it holds $x_1' \leq c_k/(a_1\gamma)$. But since $a_1x_1' + a_2x_2' = c_k$, we obtain $a_2x_2' = c_k - a_1x_1' \geq (1 - 1/\gamma)c_k$.

Therefore, the profit of ALG is at least $(1 - 1/\gamma)c_k/a_2$. Since the optimum profit is at most c_k/a_2 and with $\gamma = 1 + 1/\sqrt{\alpha}$, the profit of ALG is at least a a $1/(\sqrt{\alpha} + 1)$ fraction of the optimum profit.

ALG stops due to Step 4. In this case, by the choice of the stopping condition, $x_1^*(\ell^t) \leq x_1'\gamma$ and $x_2^*(\ell^t) \leq x_2'\gamma$. Adding the two inequalities, we get $\|x'\gamma\|_1 \geq \|x^*(\ell^t)\|_1$.

Since $(c - Ax) \leq \alpha(\ell^t - Ax)$, we have $\ell^t \geq ((\alpha - 1)Ax + c)/\alpha$. Hence,

$$\begin{aligned}\|x'\gamma\|_1 &\geq \|x^*(\ell^t)\|_1 \\ &\geq \left\|x^*\Big(\frac{(\alpha-1)Ax' + c}{\alpha}\Big)\right\|_1 \\ &\geq \Big(1 - \frac{1}{\alpha}\Big)\|x^*(Ax')\|_1 + \frac{\|x^*(c)\|_1}{\alpha} \\ &\geq \Big(1 - \frac{1}{\alpha}\Big)\|x'\|_1 + \frac{\|x^*(c)\|_1}{\alpha} \ .\end{aligned}$$

The second inequality follows from the fact that $\ell^t \geq ((\alpha - 1)Ax + c)/\alpha$; therefore the polytope defined by setting the capacity vector to $((\alpha-1)Ax+c)/\alpha$ is enclosed by the polytope defined by setting the capacity vector to ℓ^t. The last inequality follows from the fact that point x' is a feasible solution if the capacities are set to Ax'.

Solving for x' gives $\|x'\|_1 \geq \|x^*(c)\|_1/(\alpha\gamma - \alpha + 1) = \|x^*(c)\|_1/(\sqrt{\alpha} + 1)$, showing again that the optimum profit cannot be greater than $(\sqrt{\alpha} + 1)$ times the profit of ALG. □

5 Conclusions

Although we significantly narrow the gap between lower and upper bound for this online LP problem the obvious open question whether it is possible to beat the competitive ratio of $\Omega(\ln^2 \alpha/\alpha)$ in general, either with a randomized or even with a deterministic algorithm, remains.

Our new bounds also suggest that it is interesting to study the influence of the number of variables d in the LP on the achievable competitive ratio. We would be interested in bounds that are tight for any fixed number of variables, not just when the number of variables is very large.

This is further emphasized by the fact that, if d is very large, there seems little difference in the power of randomized and deterministic online algorithms. However, taking d into account and considering the particular interesting case of moderately large values of d, randomization has the potential to greatly improve performance. It is, for instance, easy to see that the algorithm that picks one of the d variables uniformly at random and increases that variable until a constraint becomes tight is $1/d$-competitive. Take for example $d = 2$, where this is a significant improvement over the $\Theta(1/\sqrt{\alpha})$ competitive ratio deterministic algorithms can achieve. In fact, this trivial randomized algorithm beats the general deterministic upper bound as long as, say, $d \leq \sqrt[3]{\alpha}/2$.

Another interesting question is whether our upper bound construction can be realized by some natural combinatorial packing problem.

References

1. Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An $O(\log k)$-competitive algorithm for generalized caching. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1681–1689, 2012.

2. Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 267–276, 2011.
3. Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.
4. Gruia Călinescu, Sanjiv Kapoor, Alexander Olshevsky, and Alexander Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In *Proceedings of the 11th European Symposium on Algorithms (ESA)*, pages 114–126, 2003.
5. Marcel Ochel, Klaus Radke, and Berthold Vöcking. Online packing with gradually improving capacity estimations and applications to network lifetime maximization. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 648–659, 2012.