Online Linear Programming

Online Algorithms

Summer 2014

Linear Programs

Online Covering Example

Online Set Cover

Online Revenue

Online Load Balancing

Linear Programming

We consider primal/dual pairs of covering/packing linear programs with $c_i, b_j, a_{ij} \geq 0$:

Primal

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{n} c_i x_i & & & \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_{ij} x_i & \geq & b_j & \forall j \\ & x_i & \geq & 0 & \forall i \end{aligned}$$

Dual

$$\begin{aligned} \text{Max} \quad & \sum_{j=1}^{m} b_i y_j & & & \\ \text{s.t.} \quad & \sum_{j=1}^{m} a_{ij} y_j & \leq & c_i & \forall i \\ & y_j & \geq & 0 & \forall j \end{aligned}$$

We will see below some prominent examples that fall into this class of problems. In some cases, we need to find integral solutions that are in $\{0, 1\}$. In this case, the linear programs above are relaxations of the problem, and we relate the quality of the output of our algorithm to the fractional optima.

Toy Example

Ski Rental as a Covering Program:

$$\begin{aligned} \text{Min} \quad & Bz + \sum_{i=1}^{n} x_i & & & \\ \text{s.t.} \quad & x_i + z & \geq & 1 & \forall i \\ & x_i, z & \in & \{0,1\} & \forall i \end{aligned}$$

- For each day i, variable x_i encodes decision to rent on day i with cost 1.
- A global variable z encodes decision to buy with cost B.
- For each day i a constraint requires to rent or buy.
- We have $x_i, z \in \{0,1\}$, and our algorithm will produce such a solution.
- As competitive ratio we will measure the ratio w.r.t. cost of the fractional optimum with $x_i, z \geq 0$.

Another Example

Set Cover as a Covering Program:

$$\begin{aligned} \text{Min} \quad & \sum_{S \subseteq E} c_S x_S & & & \\ \text{s.t.} \quad & \sum_{S: e \in S} x_S & \geq & 1 & \forall e \in E \\ & x_S & \in & \{0,1\} & \forall S \subseteq E \end{aligned}$$

- There is a set of elements E, each subset $S \subseteq E$ has a cost $c_S \geq 0$.
- Pick a subset of sets that has minimum total cost.
- The constraints require that every $e \in E$ is in at least one chosen set.
- In the fractional version, we assume $x_S \geq 0$.
- When we must obtain a binary solution, we relate its cost to the fractional optimum.

A Packing Problem

Generalized Matching (Ad-Auction Revenue) as a Packing Program:

$$
\begin{aligned}
\text{Max} \quad & \sum_{ij} b_{ij} y_{ij} & & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ij} & \leq & 1 & \forall j \\
& \sum_{j=1}^{m} b_{ij} y_{ij} & \leq & B_i & \forall i \\
& y_{ij} & \in & \{0,1\} & \forall i,j
\end{aligned}
$$

- n buyers, each buyer i has daily budget B_i
- Set M of m items (possible ads). Buyer i offers to pay b_{ij} for item $j \in M$.
- Each buyer is willing to pay in total at most B_i for all items he gets.
- Assign items to buyers to maximize revenue.

Some LP Basics

Reconsider the primal/dual pairs of covering/packing LPs from the beginning. Some useful facts about LPs are as follows.

Theorem (Weak Duality)

Let x and y be feasible solutions to a pair of primal (minimization) and dual (maximization) linear programs, respectively. Then

$$\sum_{i=1}^{n} c_i x_i \quad \geq \quad \sum_{j=1}^{m} b_j y_j \quad .$$

Proof: As both solutions are feasible:

$$\sum_{i=1}^{n} c_i x_i \quad \geq \quad \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij} y_j \right) x_i \quad = \quad \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ij} x_i \right) y_j \quad \geq \quad \sum_{j=1}^{m} b_j y_j \quad .$$

□

Some LP Basics

Theorem (Strong Duality)

The primal has a solution if and only if the dual has a finite optimal solution. Let x^* and y^* be optimal solutions if they exist. Then

$$\sum_{i=1}^{n} c_i x_i^* \quad = \quad \sum_{j=1}^{m} b_j y_j^* \quad .$$

The proof of this statement is much more involved. Instead, for us a different condition will be useful.

Complementary Slackness

Theorem (Approximate Complementary Slackness)

Let x and y be feasible solutions to primal and dual programs and $\alpha, \beta \geq 1$ that satisfy the following conditions:

- *Primal complementary slackness:*
  For each i, if $x_i > 0$ then $c_i/\alpha \leq \sum_j a_{ij} y_j \leq c_i$.
- *Dual complementrary slackness:*
  For each j, if $y_j > 0$ then $b_j \leq \sum_j a_{ij} x_i \leq \beta b_j$

Then

$$\sum_{i=1}^{n} c_i x_i \quad \leq \quad \alpha \cdot \beta \cdot \sum_{j=1}^{m} b_j y_j \quad .$$

Proof: We upper bound the cost of x and lower bound the value of y:

$$\sum_{i=1}^{n} c_i x_i \quad \leq \quad \alpha \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij} y_j \right) x_i \quad = \quad \alpha \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ij} x_i \right) y_j \quad \leq \quad \alpha \cdot \beta \sum_{j=1}^{m} b_j y_j \quad .$$

Proving Bounds

The latter theorem gives an elegant way to prove approximation bounds:

- Suppose you build a feasible solution x for your primal problem.
- Simultaneously you maintain a feasible solution y for the dual problem.
- If they satisfy approximate complementary slackness, then the cost of x is at most $\alpha \cdot \beta$ the value of y.
- Value of any feasible y is lower bound for the optimum cost of the primal.
- Hence, the cost of x is at most $\alpha \cdot \beta$ the cost of the optimum.

Linear Programs

Online Covering Example

Online Set Cover

Online Revenue

Online Load Balancing

Online Ski Rental

Online Covering Problems:

- Costs c_i are known, constraint structure (a_{ij} and b_j) unknown.
- Constraints for the covering problem arrive one-by-one over time
- When constraint j arrives, coefficients a_{ij} and b_j are revealed.
- We may raise any x_i at any point but never decrease them.
- Throughout, the solution must be feasible w.r.t. arrived constraints

Primal and Dual LPs for Ski-Rental:

$$\begin{aligned} \text{Min} \quad & Bz + \sum_{i=1}^{k} x_i & & & \\ \text{s.t.} \quad & x_i + z & \geq & 1 & \forall i \\ & x_i & \geq & 0 & \forall i \end{aligned}$$

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{k} y_i & & & \\ \text{s.t.} \quad & \sum_{i=1}^{k} y_i & \leq & B & \\ & y_i & \in & [0,1] & \forall i \end{aligned}$$

The Classic Algorithm

The classic algorithm rents skis for the first $B-1$ days and buys on day B. We interpret it as a primal-dual algorithm to show how it maintains primal and dual solutions:

1. On day i when constraint $x_i + z \geq 1$ arrives do the following.
2. If constraint is satisfied, do nothing.
3. Otherwise, increase y_i continuously until some dual constraint goes tight. Set the corresponding primal variable (x_i or z) to 1.

Analysis:

- Primal CS: If $z > 0$, then $\sum_{i=1}^{k} y_i = B$. If $x_i > 0$, then $y_i = 1$.
- Dual CS: If $y_i > 0$, then $1 \leq z + x_i \leq 2$.
- We satisfy approximate complementary slackness with $\alpha = 1$ and $\beta = 2$.
- Thus, the algorithm is 2-competitive. □

An Improved Randomized Algorithm

To build a better algorithm, we solve the problem fractionally.

1. Initially set $z = 0$.
2. For each day j, if $z < 1$ do the following
3. Set $x_j = 1 - z$
4. Set $z = z(1 + 1/B) + 1/(cB)$ (c is given below)
5. Set $y_j = 1$.

Using randomized rounding, the fractional algorithm can be turned into a randomized algorithm for the binary problem.

Theorem

The fractional/randomized algorithm for ski rental is $(1 + 1/c)$-competitive, and with $c = (1 + 1/B)^B$, the ratio approaches $e/(e - 1)$ for $B \gg 1$.

Proof: Exercise.

Linear Programs

Online Covering Example

Online Set Cover

Online Revenue

Online Load Balancing

Online Set Cover

Online Set Covering:

- Costs of sets are known, structure of sets unknown
- A subset of elements arrive one-by-one.
- Upon arrival, element e reveals in which sets it is contained
- Element must be covered by at least one of the sets at time of arrival

Primal and Dual LPs for Set Cover:

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{n} c_S x_S & & & \\ \text{s.t.} \quad & \sum_{S:e\in S} x_S & \geq & 1 & \forall e \\ & x_S & \geq & 0 & \forall S \end{aligned}$$

$$\begin{aligned} \text{Max} \quad & \sum_{e\in E} y_e & & & \\ \text{s.t.} \quad & \sum_{e\in S} y_e & \leq & c_S & \forall S \\ & y_e & \geq & 0 & \forall e \end{aligned}$$

Basic Discrete Algorithm

The basic discrete algorithm does the following:

- Upon arrival of a new element with constraint $\sum_{S:e\in S} x_S \geq 1$ and the corresponding dual variable y_e do:
- While $\sum_{S:e\in S} x_S < 1$:
- Let $f_e = |\{S \mid e \in S\}|$.
- For each S with $e \in S$ set $x_S = x_S(1 + 1/c_S) + 1/(f_e c_S)$.
- Set $y_e = y_e + 1$.

Wlog we assume $c_S \geq 1$ for all $S \subseteq E$ and obtain the following theorem.

Theorem

The algorithm yields a fractional solution that is $O(\log d)$-competitive, where $d = \max_e f_e \leq m$.

Proof of $O(\log d)$-competitiveness

Proof:

- Denote P and D the objective values of the primal and dual solutions of the algorithm. Initially, $P = D = 0$.
- Denote ΔP and ΔD the changes in the primal and dual cost for one iteration of the inner loop.

We show:

1. The algorithm produces a feasible primal (covering) solution.
2. In each iteration, $\Delta P \leq 2\Delta D$.
3. Each packing constraint in the dual program is violated by at most $O(\log d)$.

This implies:

- By dividing each y_e in the dual solution by factor of $O(\log d)$, we get a feasible dual solution. Hence, D is at most $O(\log d)$ times the value of a feasible dual solution.
- As $P \leq 2D$, a competitive ratio of $O(\log d)$ follows by weak duality.

Proof of 1. and 2.

1. Feasbilibity:
Algorithm increases variables until constraint for arriving element is satisfied.
Later iterations only increase variables, constraint never becomes violated.

2. $\Delta P \leq 2\Delta D$:
For every iteration in the inner loop, the increase in dual value is $\Delta D = 1$.
The change in the primal is

$$\sum_{S:e\in S} c_S \Delta x_S \quad = \quad \sum_{S:e\in S} c_S \left(\frac{x_S}{c_S} + \frac{1}{f_e c_S}\right) \quad = \quad \sum_{S:e\in S} \left(x_S + \frac{1}{f_e}\right) \quad \leq \quad 2 \ ,$$

as the covering constraint is infeasible during the inner loop.

The Main Argument

3. Violating dual constraints by $O(\log d)$:
   - Consider a dual constraint $\sum_{e \in S} y_e \leq c_S$.
   - Whenever we increase some y_e with $e \in S$ by 1, we increase the variable x_S by (at least) some factor.
   - The value of x_S behaves similar to a geometric sequence with $a = 1/(dc_S)$ and $q = (1 + 1/c_S)$. More formally, we show by induction

$$x_S \geq \frac{1}{d}\left(\left(1 + \frac{1}{c_S}\right)^{\sum_{e \in S} y_e} - 1\right) . \tag{1}$$

   - Initially $x_i = 0$, so the initial case holds.

The Main Argument

Next, consider an iteration where a variable y_e with $e \in S$ is incremented by 1. Let $x_S(start)$ and $x_S(end)$ be the values of x_S before and after the update. Then,

$$\begin{aligned} x_S(end) &= x_S(start)\left(1+\frac{1}{c_S}\right)+\frac{1}{f_e c_S} \\ &\geq x_S(start)\left(1+\frac{1}{c_S}\right)+\frac{1}{dc_S} \\ &\geq \frac{1}{d}\left(\left(1+\frac{1}{c_S}\right)^{-1+\sum_{e\in S} y_e}-1\right)\left(1+\frac{1}{c_S}\right)+\frac{1}{dc_S} \\ &= \frac{1}{d}\left(\left(1+\frac{1}{c_S}\right)^{\sum_{e\in S} y_e}-1\right) \quad . \end{aligned}$$

This proves (1) by induction.

The Main Argument

- We never update any $x_S \geq 1$ (constraint satisfied for every $e \in S$).
- Since each $c_S \geq 1$ and $d \geq 1$, we have $x_S = x_S(1 + 1/c_S) + 1/(f_e c_S) < 3$.
- Together with the lower bound in (1) this implies

$$3 \quad \geq \quad x_S \quad \geq \quad \frac{1}{d}\left(\left(1+\frac{1}{c_S}\right)^{\sum_{e\in S} y_e} - 1\right)$$

and with $c_S \geq 1$ we can simplify and get

$$\sum_{e\in S} y_e \quad \leq \quad c_S \log_2(3d+1) = c_S \cdot O(\log d) \ .$$

□

A Lower Bound

Lemma

There is an instance of online fractional set cover with m sets such that any online algorithm is $\Omega(\log m)$-competitive on this instance.

Proof:

- $m = 2^k$ sets, cost for every set $i = 1, \ldots, m$ is $c_i = 1$.
- First arriving element is contained in all sets. If algorithm sets $\sum_{i=1}^{m/2} x_i \leq \sum_{i=m/2+1}^{m} x_i$, second element is contained in sets $1, \ldots, m/2$. Otherwise, in the other half of sets.
- Recurse by halving and continuing with the half of smaller sum until a single set remains. Denote this set by i^*.
- Optimum solution pics only i^*, covers all elements, achieves cost 1.
- For any online algorithm, the total value on "useless" sets that we do not recurse on is at least $1/2$.
- $k + 1$ iterations, cost of any online algorithm at least $1 + k/2$. □

A Randomized Algorithm

One can turn the fractional algorithm into a randomized algorithm as follows.

- Initially, choose random threshold $T(S)$ uniformly in $[0, 1]$ for each S.
- Add S to the cover when the fractional algorithm raises $x_S \geq T(S)$.
- Constraints could be satisfied by many small x_S. Hence, we must adapt the thresholds to the number of arrived elements to cover all elements with high probability.
- Whenever the number of arrived elements doubles, we draw a new random variable $T'(S) \in [0, 1]$ for each set and update $T(S) = \min(T(S), T'(S))$.
- This only decreases the thresholds and allows for more sets to be chosen.
- The final threshold becomes the minimum of $O(\log n)$ random draws.
- The solution covers all elements with high probability, the competitive ratio increases by at most a $O(\log n)$-factor.

Theorem

There is a randomized algorithm for online set cover that covers all elements with high probability and is $O(\log n \log m)$-competitive.

Linear Programs

Online Covering Example

Online Set Cover

Online Revenue

Online Load Balancing

Online Packing

Online Packing Problems:

- Right-hand sides known, coefficients in constraints and objective unknown.
- Variables for the packing problem arrive one-by-one over time
- When a variable arrives, all coefficients for this variable are revealed.
- We may change a variable only in the iteration it arrives.
- Throughout, the solution must be feasible.

Online Revenue Maximization

- Budgets B_i of buyers are known, each item given to at most one buyer
- Items arrive online one-by-one over time
- When item j arrives, it reveals all corresponding variables y_{ij} and parameters b_{ij} for all buyers i
- Assign all variables for item j before next item arrives, keep solution feasible.

Linear Programs

Primal (Covering) and Dual (Packing) LPs for Revenue Maximization:

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{n} B_i x_i + \sum_{j=1}^{m} z_j \\ \text{s.t.} \quad & b_{ij} x_i + z_j \geq b_{ij} \quad \forall i,j \\ & x_i, z_j \geq 0 \quad \forall i,j \end{aligned}$$

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_{ij} \leq 1 \quad \forall j \\ & \sum_{j=1}^{k} b_{ij} y_{ij} \leq B_i \quad \forall i \\ & y_{ij} \geq 0 \quad \forall i,j \end{aligned}$$

For consistency reasons, we refer to the packing problem as the dual throughout. Hence, in this case we are solving the dual program in an online fashion.

Algorithm for Online Revenue Maximization

1. Initially, for each buyer i set $x_i = 0$.
2. Upon arrival of a new item j with variables y_{ij} do:
3. Allocate j to buyer i' maximizing $b_{i'j}(1 - x_{i'})$
4. If $x_{i'} \geq 1$, do nothing. Otherwise:
5. Charge buyer minimum of $b_{i'j}$ and his remaining budget, set $y_{i'j} = 1$.
6. Set $z_j = b_{i'j}(1 - x_{i'})$
7. Set $x_{i'} = x_{i'}(1 + b_{i'j}/B_{i'}) + b_{i'j}/((c-1) \cdot B_{i'})$.

Let $R_{max} = \max_{ij} b_{ij}/B_i$ be the maximum ratio of the offer for a item and the budget of the buyer.

Theorem

The algorithm returns an integral solution and has competitive ratio $(1 - 1/c)(1 - R_{\max})$, *where* $c = (1 + R_{\max})^{1/R_{\max}}$. *For* $R_{\max} \to 0$, *the ratio tends to* $(1 - 1/e)$.

Proof of the Competitive Ratio

Proof:

- Denote P and D the objective values of the primal and dual solutions of the algorithm. Initially, $P = D = 0$.
- Denote ΔP and ΔD the changes in the primal and dual cost for one iteration.

We show:

1. The algorithm produces a feasible primal (covering) solution.
2. In each iteration, $\Delta P \leq c/(c-1)\Delta D$.
3. The packing constraint with B_i in the dual is violated by at most $\max_j b_{ij}$.

To prove the theorem, we can assemble these properties very similarly as for set cover above. We come back to this issue below after showing the three properties.

Properties 1. and 2.

1. Feasbilibity:
Algorithm increases x_i and adjusts z_j so that all constraints for item j are satisfied. If $x_i \geq 1$, constraint is satisfied. Otherwise, algorithm assigns j to buyer i' with maximum $b_{i'j}(1 - x_{i'})$. Setting $z_j = b_{i'j}(1 - x_{i'})$ then satisfies constraint for item j and each buyer i. Later iterations only increase x_i, constraints never become violated.

2. $\Delta P \leq c/(c-1)\Delta D$:
For every iteration, the increase in dual value is $\Delta D = b_{i'j}$. Even if this exceeds the budget $B_{i'}$, we increase $y_{i'j}$ to 1. Hence, we might violate the dual budget constraint (see 3. below).
The change ΔP in the primal is

$$\begin{aligned} B_{i'}\Delta x_{i'} + z_j &= \left(b_{i'j}x_{i'} + \frac{b_{i'j}}{c-1}\right) + b_{i'j}(1 - x_{i'}) \\ &= b_{i'j}\left(1 + \frac{1}{c-1}\right) = b_{i'j}c/(c-1) \ . \end{aligned}$$

The Main Argument

3. Violating dual constraints with B_i by $\max_j b_{ij}$:
   - Consider a dual constraint $\sum_j b_{ij} y_{ij} \leq B_i$. Whenever we increase some y_{ij} by 1, we increase the variable x_i by (at least) some factor.
   - The value of x_i behaves similar to a geometric sequence. More formally

$$x_i \geq \frac{1}{c-1}\left(c^{\frac{\sum_j b_{ij} y_{ij}}{B_i}} - 1\right) . \tag{2}$$

   - Observe that $\sum_j b_{ij} y_{ij} \geq B_i$ implies $x_i \geq 1$, in which case we (satisfy all primal constraints for buyer i and) stop updating x_i in the algorithm.
   - As before, we show (2) by induction on the (relevant) iterations of the algorithm. Initially $x_i = 0$, so the initial case holds.

The Main Argument

Next, consider an iteration where a variable y_{ik} is incremented by 1 – i.e., item k is sold to buyer i. Let $x_i(start)$ and $x_i(end)$ be the values of x_i before and after the update. Then,

$$\begin{aligned}
x_i(end) &= x_i(start)\left(1+\frac{b_{ik}}{B_i}\right)+\frac{b_{ik}}{(c-1)B_i} \\
&\geq \frac{1}{c-1}\left[c^{\frac{\sum_{j\neq k} b_{ij}y_{ij}}{B_i}}-1\right]\left(1+\frac{b_{ik}}{B_i}\right)+\frac{b_{ik}}{(c-1)B_i} \\
&= \frac{1}{c-1}\left[c^{\frac{\sum_{j\neq k} b_{ij}y_{ij}}{B_i}}\left(1+\frac{b_{ik}}{B_i}\right)-1\right] \\
&\geq \frac{1}{c-1}\left[c^{\frac{\sum_{j\neq k} b_{ij}y_{ij}}{B_i}}c^{\frac{b_{ik}}{B_i}}-1\right] \\
&= \frac{1}{c-1}\left[c^{\frac{\sum_{j} b_{ij}y_{ij}}{B_i}}-1\right] .
\end{aligned}$$

This proves (2) by induction.

The Main Argument

- The second inequality follows by hypothesis. The forth inequality uses the fact that for any $0 < x \leq y \leq 1$ we have $\ln(1+x)/x \geq \ln(1+y)/y$.
- Note that when $b_{ik}/B_i = R_{\max}$ and $c = (1+R_{\max})^{1/R_{\max}}$, then

$$\left(1+\frac{b_{ik}}{B_i}\right) \quad = \quad (1+R_{\max}) \quad = \quad \left((1+R_{\max})^{1/R_{\max}}\right)^{R_{\max}} \quad = \quad c^{\frac{b_{ik}}{B_i}}$$

  and the forth inequality holds with equality. This is why we have to choose $c \leq (1+R_{\max})^{1/R_{\max}}$ for the forth inequality to hold.

- When the dual constraint for buyer i becomes violated, we stop updating x_i. Hence, there can be at most one iteration, in which a buyer pays less than b_{ij}.
- Thus, for each buyer i

$$\sum_{j=1}^{m} b_{ij} y_{ij} \leq B_i + \max_j b_{ij}.$$

  This proves property 3.

Proving the Bound

We assemble the properties very similarly as for set cover. In contrast, here we need to lower bound the value of the dual by the primal, which inverts some of the bounds.

Due to (3.), the additional profit obtained by violating the dual constraints is at most $\max_j b_{ij}$ for buyer i. Thus, the real profit from buyer i is at least

$$\left[\sum_{j=1}^{m} b_{ij} y_{ij}\right] \cdot \frac{B_i}{B_i + \max_j b_{ij}} \quad \geq \quad \left[\sum_{j=1}^{m} b_{ij} y_{ij}\right] (1 - R_{\max}) \ .$$

Thus, the real profit of the final allocation is at least $D \cdot (1 - R_{\max})$.

By (2.), $P \leq (c/(c-1)) \cdot D$, or in turn

$$D \quad \geq \quad ((c-1)/c)P \quad = \quad (1 - 1/c) \cdot P \ .$$

Hence, the real profit is at least $(1 - 1/c)(1 - R_{\max}) \cdot P$. By (1.), our primal solution is feasible, and hence, by weak duality, the real profit is at least $(1 - 1/c)(1 - R_{\max})$ of the fractional optimum for the dual. □

Extension to Search Pages with Multiple Slots

- n buyers, each buyer i has daily budget B_i.
- Set M of m search result pages, each page j has ℓ slots for ads.
- Buyer i offers to pay b_{ijk} for slot k on page j.
- Each buyer is willing to pay in total at most B_i for all slots he gets.
- Assign slots to buyers to maximize revenue.

$$\begin{aligned}
\text{Max} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{\ell} b_{ijk}\, y_{ijk} & & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} y_{ijk} & \leq & 1 & \forall j,k \\
& \sum_{j=1}^{m}\sum_{k=1}^{\ell} b_{ijk}\, y_{ijk} & \leq & B_i & \forall i \\
& \sum_{k=1}^{\ell} y_{ijk} & \leq & 1 & \forall i,j \\
& y_{ijk} & \geq & 0 & \forall i,j,k
\end{aligned}$$

Primal Covering LP

Primal Covering LP for Slot-Allocation:

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{n} B_i x_i + \sum_{j=1}^{m} \sum_{k=1}^{\ell} z_{jk} + \sum_{i=1}^{n} \sum_{j=1}^{m} s_{ij} \\ \text{s.t.} \quad & b_{ijk} x_i + z_{jk} + s_{ij} \geq b_{ijk} \qquad \forall i, j, k \\ & x_i, z_{jk}, s_{ij} \geq 0 \qquad \forall i, j, k \end{aligned}$$

Our algorithm will not update variables s and z explicitly. They are only assigned and used within our analysis. The main property, as before, is the correct relation between multiplicative updates of x and additive updates of y.

Algorithm for Slot-Allocation

1. Initially, for each buyer i set $x_i = 0$.
2. Upon arrival of a new item j with variables y_{ijk} do:
3. Generate a bipartite graph H: n buyers on one side, ℓ slots of j on the other side. Edge $(i,k) \in H$ has weight $b_{ijk}(1 - x_i)$.
4. Find a maximum weight (integral) matching in H.
5. Assign the variables y_{ijk} as in the matching in H.
6. Charge buyer i minimum of $\sum_{k=1}^{\ell} b_{ijk} y_{ijk}$ and his remaining budget.
7. For each buyer i, if there is a slot k for which $y_{ijk} > 0$, then set

$$x_i = x_i \left(1 + \frac{b_{ijk} y_{ijk}}{B_i}\right) + \frac{b_{ijk}}{(c-1) \cdot B_i} \quad .$$

Theorem

The algorithm returns an integral solution and has competitive ratio $(1 - 1/c)(1 - R_{\max})$, *where* $c = (1 + R_{\max})^{1/R_{\max}}$ *tends to* e *for* $R_{\max} \to 0$.

Proof of the Competitive Ratio

Proof:
Again, we show:

1. The algorithm produces a feasible primal (covering) solution.
2. In each iteration, $\Delta P \leq c/(c-1)\Delta D$.
3. The packing constraint with B_i is violated by at most $\max_{j,k} b_{ijk}$.

The only difference to the previous case is that we now can assign a single item j to several buyers by giving them slots on the page. We tackle this adjustment using the bipartite graph H and a maximum matching for slot assignment on page j. In this step, we crucially rely on strong duality:

The value of a maximum weight (integral) matching in H equals the value of an optimal primal solution for the (sub-)problem of optimal slot allocation for a single item.

Slot Allocation on a Single Page via Maximum Matching

The following programs describe the bipartite matching problem in graph H. Primal (Covering) and Dual (Packing) LPs for slot allocation via H for a single item j:

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{n} s_{ij} + \sum_{k=1}^{\ell} z_{jk} & & & \\ \text{s.t.} \quad & s_{ij} + z_{jk} & \geq & b_{ijk}(1 - x_i) & \forall i, k \\ & s_{ij}, z_{jk} & \geq & 0 & \forall i, k \end{aligned}$$

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{n} \sum_{k=1}^{\ell} b_{ijk}(1 - x_i) y_{ijk} & & & \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_{ijk} & \leq & 1 & \forall k \\ & \sum_{k=1}^{\ell} y_{ijk} & \leq & 1 & \forall i \\ & y_{ijk} & \geq & 0 & \forall i, k \end{aligned}$$

Observe x_i's are constant here, since we strive to assign only the y_{ijk} for fixed j. The dual y_{ijk} and primal s_{ij} and z_{jk} are the ones from the global LPs.

Strong Duality

By strong duality, we know that the maximum matching in H yields an (integral) optimal dual solution y and a (fractional) optimal primal solution (s, z) such that

$$\sum_{i=1}^{n}\sum_{k=1}^{\ell} b_{ijk}(1-x_i)y_{ijk} \quad = \quad \sum_{i=1}^{n} s_{ij} + \sum_{k=1}^{\ell} z_{jk} \ . \tag{3}$$

Using this equality, we now imitate the previous proof and show the three properties.

1. Feasibility:
Each primal constraint reads $b_{ijk}x_i + z_{jk} + s_{ij} \geq b_{ijk}$. Recall the primal program for matching in H. Since our assignments for s and z are feasible (and optimal) for this program, we have $z_{jk} + s_{ij} \geq b_{ijk}(1 - x_i)$. This implies feasibility also for the global primal constraints.

Proofs of Properties

2.$\Delta P \leq (c/(c-1)) \cdot \Delta D$:
Upon arrival of item j,

$$\begin{aligned}\Delta P &= \sum_{i=1}^{n} s_{ij} + \sum_{k=1}^{\ell} z_{jk} + \sum_{i=1}^{n} B_i \Delta x_i \\ &= \sum_{i=1}^{n}\sum_{k=1}^{\ell} b_{ijk}(1-x_i)y_{ijk} + \sum_{i=1}^{n}\sum_{k=1}^{\ell} B_i \left(\frac{b_{ijk} x_i y_{ijk}}{B_i} + \frac{b_{ijk} y_{ijk}}{(c-1)B_i} \right) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{\ell} b_{ijk} y_{ijk} \left(\frac{c}{c-1} \right),\end{aligned}$$

where we use (3) and the definition of the update of x_i. Now the claim follows since

$$\Delta D = \sum_{i=1}^{n}\sum_{k=1}^{\ell} b_{ijk} y_{ijk} \quad .$$

The Main Argument

3. Violating dual constraints with B_i by $\max_{j,k} b_{ijk}$:
   - The remaining proof follows almost literally as in the previous proof. We just incorporate the slots into the notation.
   - Consider a dual constraint $\sum_{j,k} b_{ijk} y_{ijk} \leq B_i$. Whenever we increase some y_{ijk} by 1, we increase the variable x_i by (at least) some factor.
   - The value of x_i behaves similar to a geometric sequence. More formally

$$x_i \geq \frac{1}{c-1}\left(c^{\frac{\sum_{j,k} b_{ijk} y_{ijk}}{B_i}} - 1\right) . \tag{4}$$

   - Observe that $\sum_{j,k} b_{ijk} y_{ijk} \geq B_i$ implies $x_i \geq 1$, in which case we (satisfy all primal constraints for buyer i and) stop updating x_i in the algorithm.
   - We can show (4) by induction on the (relevant) iterations of the algorithm. The calculations are exactly as in the previous proof and omitted :)

Finishing the Proof

- As before, only in one iteration per buyer we account for more profit in D than we really receive (the iteration where we violate the buyer's dual constraint).
- Hence, the dual constraint is violated by at most:

$$\sum_{j=1}^{m}\sum_{k=1}^{\ell} b_{ijk}y_{ijk} \quad \leq \quad B_i + \max_{j,k} b_{ijk} \ .$$

This proves property 3.

Finally, we assemble the three properties exactly as in the previous proof:

3.: Real profit at least $(1 - R_{\max}) \cdot D$ by bounded violation of dual constraints.

2.: $P \leq (c/(c-1))D$, so $D \geq (1 - 1/c)P$.

1.: Primal solution feasible, so P upper bound for dual optimum. □

Linear Programs

Online Covering Example

Online Set Cover

Online Revenue

Online Load Balancing

Load Balancing on Unrelated Machines

Load Balancing

- n jobs, m machines, jobs arrive one-by-one over time
- Job i reveals a load or processing time $p_{ij} \geq 0$ for each machine j
- Assign each job to some machine
- Goal: Minimize the makespan, i.e., the maximum total load of the machines:

$$\max_j \sum_{i \text{ on } j} p_{ij}$$

Suppose we know an upper bound $\gamma \geq \Lambda^*$ on the value Λ^* of the optimal makespan. For a given upper bound γ, we can formulate finding a feasible assignment with makespan at most γ as a packing problem.

Linear Programs

Let $\tilde{p}_{ij} = p_{ij}/\gamma$ and $S_i = \{j \mid \tilde{p}_{ij} \leq 1\}$. We consider the following primal (Covering) and dual (Packing) LPs:

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{m} x_j + \sum_{i=1}^{n} z_i & & & \\ \text{s.t.} \quad & \tilde{p}_{ij} x_j + z_i & \geq & 1 & \forall i, j \in S_i \\ & x_j, z_i & \geq & 0 & \forall i, j \end{aligned}$$

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{n} \sum_{j \in S_i} y_{ij} & & & \\ \text{s.t.} \quad & \sum_{j \in S_i} y_{ij} & \leq & 1 & \forall i \\ & \sum_{i=1}^{n} \sum_{j \in S_i} \tilde{p}_{ij} y_{ij} & \leq & 1 & \forall j \\ & y_{ij} & \geq & 0 & \forall i, j \end{aligned}$$

If $\gamma \geq \Lambda^*$, there is a feasible integral packing of all jobs and the dual optimum has objective value n.

Algorithm

1. Inititally, set $x_j = 1/(2m)$ for all j
2. When a new job i arrives:
3. In case of the very first job, initialize $\gamma = \min_j p_{ij}$
4. If there is no j with $\tilde{p}_{ij} \leq 1$, or if there exists j with $x_j > 1$ then
5. Set $\gamma = 2\gamma$ and $x_j = 1/(2m)$ for all j.
6. Otherwise:
7. Let ℓ be a machine minimizing $\tilde{p}_{i\ell} x_\ell$
8. Assign i to machine ℓ
9. Set $z_i = 1 - \tilde{p}_{i\ell} x_\ell$ and $y_{i\ell} = 1$.
10. Set $x_\ell = x_\ell \cdot (1 + \tilde{p}_{i\ell}/2)$.

The algorithm maintains a "guess" γ of the optimal makespan and tries to pack jobs onto machines to maintain a makespan of γ. If this fails, then in lines 4-5 it increases the guess by a factor of 2 and restarts on the remaining jobs by re-setting (the relevant part x of) the primal solution.

Analysis

Instead of doubling our guess of the optimal makespan iteratively, we first assume that in line 3 we can initialize a correct upper bound $\gamma \geq \Lambda^*$. We prove the following claim.

Claim

If in line 3 we initialize a correct guess $\gamma \geq \Lambda^*$, the algorithm assigns all jobs and yields a makespan at most $\gamma \cdot O(\log m)$.

Proof of Claim:
We show the following two properties:

1. The load on each machine is at most $\gamma \cdot O(\log m)$.
2. If the algorithm restarts in line 4-5, then there is a feasible primal solution of value strictly smaller than n.

Proof of 1.

1. Makespan is $\gamma \cdot O(\log m)$
The algorithm never assigns i to j with $x_j > 1$. As $\tilde{p}_{ij} \leq 1$, the update in line 10 increases x_j to at most $3/2$. Initially, $x_j = 1/(2m)$. Let J_j be the jobs assigned to machine j. Then, the observations imply

$$\frac{3}{2} \geq x_j \geq \frac{1}{2m} \prod_{i \in J_j} \left(1 + \frac{\tilde{p}_{ij}}{2}\right) \geq \frac{1}{2m} \prod_{i \in J_j} \left(\frac{3}{2}\right)^{\tilde{p}_{ij}}$$

$$= \frac{1}{2m} \exp\left(\ln\left(\frac{3}{2}\right) \sum_{i \in J_j} \tilde{p}_{ij}\right) .$$

Rearranging yields

$$\sum_{i \in J_j} \tilde{p}_{ij} \leq \frac{\ln(3m)}{\ln(3/2)} = O(\log m) .$$

This bound also holds in case the algorithm would attempt a restart.

Proof of 2.

2. If algorithm restarts, primal optimum value $< n$.
Observe that our choice of z_i is sufficient to make all primal covering constraints true. The x_j are non-decreasing and never violate constraints later on. Hence, (x, z) computed by the algorithm is a feasible primal solution.

If job i gets assigned to machine ℓ, the change in primal objective function value is

$$1 - \tilde{p}_{i\ell} x_\ell + \frac{\tilde{p}_{i\ell} x_\ell}{2} \quad = \quad 1 - \frac{\tilde{p}_{i\ell} x_\ell}{2} \ .$$

On the other hand, when we update x_ℓ, it also increases exactly by $\tilde{p}_{i\ell} x_\ell / 2$. Thus, the negative term above represents exactly the increase in x_ℓ. Hence, when a job arrives and get assigned to machine ℓ, the increase in the primal is 1 minus the increase of x_ℓ.

Proof of 2.

Combining this insight over all arrivals of n jobs yields the following expression for the final value P of the primal solution, where $x_j^{init} = 1/(2m)$ from line 1:

$$\begin{aligned} P &= \sum_{j=1}^{m} x_j^{init} + n - \sum_{j=1}^{m} (x_j - x_j^{init}) \\ &= 2\sum_{j=1}^{m} x_j^{init} + n - \sum_{j=1}^{m} x_j = 1 + n - \sum_{j=1}^{m} x_j \ . \end{aligned}$$

Obviously, if $x_j > 1$ for some machine j, the primal value is strictly smaller than n. By weak duality, the dual optimum value must be strictly smaller than n. Recall that this is impossible if $\gamma \geq \Lambda^*$. Hence, the algorithm never restarts when $\gamma \geq \Lambda^*$. □ (Claim)

Main Result

Theorem

The algorithm is $O(\log m)$-competitive.

Proof:

- If $\gamma < \Lambda^*$, algorithm might fail and execute a restart in lines 4-5.
- γ doubles with every restart and values compose a geometric sequence.
- The claim implies that for jobs arriving in between two restarts, the makespan constructed is at most $\gamma \cdot O(\log m)$ for the current γ.
- Thus, in terms of the final γ, the overall makespan becomes at most

$$\sum_{k=1}^{\infty} \gamma/2^k \cdot O(\log m) = 2\gamma \cdot O(\log m) \ .$$

- By the claim, γ gets doubled only if $\gamma < \Lambda^*$.
- Hence, in the end, $\Lambda^* \geq \gamma/2$ and makespan is at most $4\Lambda^* \cdot O(\log m)$.

Recommended Literature

An overview of techniques and results in online linear programming:

- N. Buchbinder, J. Naor. The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science 3(23):93-263, 2009.

A classic introduction to linear programming:

- V. Chvatal. Linear Programming. W. H. Freeman and Co., New York, 1983.

Recommended Literature

Original publications where the results appeared:

- N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, J. Naor. The online set cover problem. SIAM J. Comput. 39(2):361-370, 2009.
- J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. J. ACM 44(3):486-504, 1997.
- N. Buchbinder, K. Jain, J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. ESA 2007.
- N. Buchbinder, J. Naor. Fair online load balancing. J. Scheduling 16(1):117-127, 2013.
- A. Mehta, A. Saberi, U. Vazirani, V. Vazirani. Adwords and generalized online matching. J. ACM 54(5):22, 2007.