CSC2421 Topics in Algorithms: Online and Other Myopic Algorithms Fall 2019

Allan Borodin

October 23, 2019

Announcements amd agenda

Reminder: Seminar Friday, 11AM, Oct 25
Spyros Angelopolous will speak on "Online computationn with untrusted advice

Todays agenda: The primal dual framework

The primal dual framework

We will temporarily skip over chapter 7 and move on to chapter 8. In Chapter 8, we describe a framework for designing and analyzing online algorithms based on linear programming (LP) duality. The design and analysis of algorithms based on the primal-dual correspondence is a fundamental idea that goes well beyond online algorithms.

We begin by recalling some basic definitions from the theory of linear programming in the offline setting. Then we describe how linear programming can be stated in the online setting. The key idea is for the algorithm to maintain two online solutions: one to the primal formulation and another to the dual formulation. The goal of the algorithm is to minimize the gap between the objectives achieved by each of the two online solutions. The competitive ratio can be readily derived from the gap by using approximate complementary slackness. This is called the *primal-dual approach*.

Linear programming

Linear programs can arise naturally for some problems, but our interest will be with respect to linear programs as a relaxation of integer programs.

For example, consider the following (conseptually simple) way to derive an offline approximation for the weighted vertex cover problem. Let the input be a graph $G = (V, E)$ with a weight function $w : V \to \Re^{\geq 0}$. To simplify notation let the vertices be $\{1, 2, \ldots .n\}$. Then we want to solve the following "natural IP representation" of the problem:

- Minimize $\mathbf{w} \cdot \mathbf{x}$
- subject to $x_i + x_j \geq 1$ for every edge $(v_i, v_j) \in E$
- $x_j \in \{0, 1\}$ for all j.

The *intended meaning* is that $x_j = 1$ iff vertex v_j is in the chosen cover. The constraint forces every edge to be covered by at least one vertex.

Solving an IP is an *NP*-hard problem. Instead we relax the integral $x_j \in \{0, 1\}$ constraints and instead allow fractional solutions $x_j \in [0, 1]$. Here the rounding is quite obvious: round LP solution x_j^* to integral $\bar{x}_j$ iff $x_j^* \geq 1/2$. The "rounding step" can sometime be non-trivial.

The integrality gap

- For LP relaxations of an IP we can define the integrality gap (for a minimization problem) as $\max_{\mathcal{I}} \frac{IP-OPT}{LP-OPT}$; that is, we take the worst case ratio over all input instances $\mathcal{I}$ of the IP optimum to the LP optimum. (For maximization problems we take the inverse ratio.)
- Note that the integrality gap refers to a particular IP/LP relaxation.
- The same concept of the integrality gap can be applied to other relaxations such as in semi definite programming (SDP).
- It should be clear that the simple IP/LP rounding we just used for the vertex cover problem shows that the integrality gap for the previously given IP/LP formulation is at most 2.
- By considering the complete graph K_n on n nodes, it is also easy to see that this integrality gap is at least $\frac{n-1}{n/2} = 2 - \frac{1}{n}$.

Linear programming continued

A linear program is a maximization or minimization problem with a linear objective function subject to linear non-strict inequality constraints. We will first consider minimization problems. Every linear program has an equivalent formulation in the standard form which is the following program P for a minimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} c_i x_i & \\ \text{subj. to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i & i \in [m] \\ & x_i \geq 0 & i \in [n] \end{aligned}$$

In the above linear program we have n variables $x_1, \ldots, x_n$ and m constraints involving a_{ij} and b_i constants, as well as n nonnegativity constraints $x_i \geq 0$.

Linear programming continued

We can write down the above program in matrix form as follows:

$$\begin{aligned} \text{minimize} \quad & c^t x \\ \text{subj. to} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

In the above, $A = (a_{ij})$ is the matrix of coefficients, $x = (x_1, \ldots, x_n)$, $b = (b_1, \ldots, b_m)$ and $c = (c_1, \ldots, c_n)$ are vectors[1] of variables, right-hand side constant terms, and coefficients of the objective function, respectively. For two vectors v and u of the same dimension, the notation $v \geq u$ is interpreted component-wise, i.e., $v_i \geq u_i$ for all i.

[1] All vectors are column vectors unless stated otherwise. Note that we represent contents of all vectors as row vectors inside text for typographical reasons.

And more linear programming

Any vector $x \in \mathbb{R}^n$ satisfying the constraints of a given LP, i.e., $Ax \geq b$, is called **feasible**. A feasible vector x^* that optimizes the value of the objective is called an **optimal solution**, or simply, an **optimum**.

A linear program might not admit any feasible vectors: a simple example is when there are clearly contradictory constraints such as $x_1 - x_2 \geq 1$, $x_2 - x_1 \geq 1$, and $x_1, x_2 \geq 0$. An LP that does not admit any feasible vectors is called **infeasible**.

When a linear program does have feasible vectors it can be either **bounded**, i.e., has a finite optimum, or **unbounded**, i.e., has an infinite optimum. A simple example of an unbounded LP is to minimize $-x_1$ subject to $x_1 \geq 0$. We shall denote the value of an optimal solution of a linear program P by $OPT(P)$.

Duality

One way of proving lower bounds on the value of $OPT(P)$ is to take non-negative combinations of constraints of P. Let A_i denote the i^{th} row of matrix A. Then the i^{th} constraint of P is $A_i x \geq b_i$. Consider multiplying constraint i by $y_i \geq 0$ and adding up all the resulting inequalities. Then you get $\sum_{i=1}^{m} y_i A_i x \geq \sum_{i=1}^{m} b_i y_i$. In matrix form we can write it as $(A^t y)^t x \geq b^t y$. If we choose y so that $(A^t y) \leq c$ then we get $c^t x \geq (A^t y)^t x \geq b^t y$. Thus, such a choice of y demonstrates a lower bound $b^t y$ on the value of P for *any* feasible x. What is the best possible lower bound that can be derived this way? This is given by another linear program, the **dual** program D:

$$\begin{aligned} \text{maximize} \quad & b^t y \\ \text{subj. to} \quad & A^t y \leq c \\ & y \geq 0 \end{aligned}$$

Relating the primal and the dual

The original linear program P is referred to as **primal**, and (as we just said) the derived linear program D is referred to as **dual**. We have just derived the basic fact about the relationship between the primal and the dual:

Theorem (Weak Duality)

$$OPT(D) \leq OPT(P).$$

We also state the stronger version of this theorem without proof. The proof of the following Strong Duality Theorem relies on more advanced machinery from polytope theory and can be found in any standard text on linear programming.

Theorem (Strong Duality)

If the primal P is feasible and bounded then so is its dual D and, moreover, we have

$$OPT(D) = OPT(P).$$

Approximate complementary slackness

Using linear programming formulations, one of the main tools for proving offline approximation ratios and online competitive ratios is the following approximate complementary slackness theorem.

Theorem (Approximate Complementary Slackness)

Suppose that x is a feasible solution to the primal P and y is a feasible solution to the dual D. If there exist $\alpha, \beta > 1$ such that

- *if $x_i > 0$ then $(c_i/\alpha) \leq (A^t)_i y \leq c_i$; and*
- *if $y_i > 0$ then $b_i \leq A_i x \leq \beta b_i$*

then $c^t x \leq (\alpha\beta) b^t y$.

Proof of complememtary slackness theorem

Proof.

The proof is rather straightforward:

$$\begin{aligned} c^t x &= \sum_{i:x_i>0} c_i x_i \le \sum_{i:x_i>0} (\alpha(A^t)_i y) x_i \\ &= \sum_{i:x_i>0} \alpha \sum_{j:y_j>0} A_{ji} y_j x_i = \sum_{j:y_j>0} (\alpha y_j) A_j x \\ &\le \sum_{j:y_j>0} (\alpha y_j)(\beta b_j) = (\alpha\beta) b^t y. \end{aligned}$$

□

Applying approximate complementary slackness

Approximate complementary slackness can be used to design competitive online algorithms. Suppose that an algorithm maintains a primal solution x and a dual solution y, such that they satisfy the approximate complementary slackness conditions with parameters α and β. Then the solution x is immediately $(\alpha\beta)$ competitive, since

$$OPT(P) = c^t x^* \leq c^t x \leq (\alpha\beta) b^t y \leq (\alpha\beta) b^t y^* = (\alpha\beta) OPT(D),$$

where P is the primal, D is the dual, x^* is an integral optimum for P, and y^* is an LP optimum for D.

There is often a very elegant and general way to uitlize the primal dual framework. We use the primal dual approach to construct a fractional slolution to a problem. We then use the fractional solution to derive a randomized (or maybe even deterministiic) algorithm for the (integral) problem. We will first demonstrate the framework for the online set cover problem.

The offline and online set cover problem

In the offline and online Set Cover problems, a universe of elements X and a family $\mathcal{S} \subseteq 2^X$ of subsets of X are given to an online algorithm in advance. Moreover, each set $S \in \mathcal{S}$ is associated with a cost $c_S \in \mathbb{R}$, which is also known in advance. We assume that $c_S \geq 1$ for all S. The goal is to select sets from the collection $\mathcal{S}$ to cover X while minimizing the total cost.

It is known that the "natural greedy algorithm" achieves an H_n approximation for the offline set cover problem and it is NP-hard to achieve an approximation better than $\Omega(\log n)$ where $n = |X|$. Using a stronger complexity assumption it is hard to approximate better than H_n.

In the online set cover problem, a target set $X' \subseteq X$ arrives online one element at a time in an adversarial order. The goal is to select sets from the collection $\mathcal{S}$ to cover X' while minimizing the total cost. That is the goal is to find $\mathcal{S}' \subseteq \mathcal{S}$ such that $X' \subseteq \bigcup_{S \in \mathcal{S}'} S$ and $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c_S$ is as small as possible.

The online set cover problem continued

An online algorithm maintains a solution $\mathcal{S}'$ that is initially empty. When an element $e \in X'$ is revealed if it is already "covered" by one of the sets in the partial solution $\mathcal{S}'$ then the online algorithm doesn't have to do anything and the next element from X' can be revealed. If e is not yet covered by one of the sets in $\mathcal{S}'$ then a deterministic (resp. randomized) algorithm has to pick some new sets $S \in \mathcal{S}'$ to add to $\mathcal{S}'$ such that e is now in at least one set partial solution $\mathcal{S}'$. This way, the algorithm maintains the invariant that each of the revealed elements is covered by at least one set from $\mathcal{S}'$ at all times. We shall denote the size of X by n and (as stated) the size of $\mathcal{S}$ by m.

Primal dual statement for set cover

To state the primal-dual formulation for Online Set Cover, we introduce primal variables x_S for $S \in \mathcal{S}$ with the intended meaning "$x_S = 1$" indicating that S is part of the solution, and "$x_S = 0$" indicating that S is not part of the solution. In the relaxed LP formulation, the variables x_S can take on fractional values. We denote the dual variables by y_e for $e \in X$. The primal and dual LPs are stated below.

$$\begin{array}{llll}
 & \textit{Primal} & & \\
\text{minimize} & \displaystyle\sum_{S\in\mathcal{S}} c_S x_S & & \\
\text{subject to} & \displaystyle\sum_{S\in\mathcal{S}: e\in S} x_S \geq 1 & e \in X' \\
 & x_S \in [0,1] & S \in \mathcal{S}
\end{array}
\qquad
\begin{array}{llll}
 & \textit{Dual} & & \\
\text{maximize} & \displaystyle\sum_{e\in X'} y_e & & \\
\text{subject to} & \displaystyle\sum_{e\in S} y_e \leq c_S & S \in \mathcal{S} \\
 & y_e \geq 0 & e \in X'
\end{array}$$

The fractional primal dual online set cover problem

The online version of Set Cover translates into the following online version of the above primal-dual formulation as follows:

Primal online input: constraints $\sum_{S \in \mathcal{S} : e \in S} x_S \geq 1$ arrive one at a time in the primal; and

Dual online input: variables y_e arriving one at a time in the dual; and

Online decisions: an online algorithm for primal-dual is allowed to update values of variables x_S, y_e only in a monotonically increasing fashion.

A deterministic algorithm for the set cover primal dual formulation

The primal dual formulation just given will lead to a deterministic algorithm that computes a fractional set cover solution with a "good" competitive ratio. Then, as previously foretold, we will be able to transform the fractional algorithm into an integral randomized algorithm with a somewhat worse competitive ratio. And finally we will be able to transform the fractional algorithm into a deterministic online algorithm maintaining the competitive ratio of the randomized algorithm.

We need some additional notation. Let F_e denote the frequency of element e among sets in $\mathcal{S}$, i.e., $F_e := |\{S \in \mathcal{S} : e \in S\}|$. Let F denote the maximum frequency among input elements, i.e., $F = \max_{e \in X'} F_e$. Clearly $F \leq m$.

The fractional "water-level" algorithm

```
All primal variables are initially set to 0
While new elements $e \in X'$ arrive
    A new constraint $\sum_{S \in \mathcal{S}: e \in S} x_S \geq 1$ for the primal arrives.
    A new variable $y_e$ for the dual is introduced.
    While $\sum_{S \in \mathcal{S}: e \in S} x_S < 1$
      For $S \in \mathcal{S}$ such that $e \in S$
        $x_S \leftarrow x_S(1 + 1/c_S) + 1/(c_S F_e)$
      EndFor
      $y_e \leftarrow 1$
    EndWhile
EndWhile
```

Theorem

The Algorithm produces a feasible primal fractional solution. The algorithm yields an integral dual solution that can be scaled down by a factor $O(\log F) = O(\log m)$ to obtain a feasible dual fractional solution. This then establishes that the primal fractional is $O(\log F) = O(\log m)$ competitive.

Some motivation for the fractional set cover algorithm

While the framework for the primal dual algorithm is pretty well prescribed, the creative aspect of the algorithm is in how the primal variable is being increased. (Here we want to increase the dual to reflect that the element is covered so lets mainly focus on the primal update.)

This is analogous to the use of priority functions in online algorthms that we previously considered. There it was the specification of an appropriate priority function that was the creative step. In hindsight, knowing what a possible proof might look like, we can possibly see how one can discover an appropriate priority function.

The same is true here. It is reasonable to want to charge the cost of covering an element to the sets that cover it. Since in the online problem, the elements X' that need to be covered is unknown and only being revealed online, it seems reasonable to want to gradually fractionally charge all the possible F_e sets for an element e (in proportion to the costs of the sets) and this is what the primal update step is doing.

Motivation for the fractional set cover algorithm continued

Note that we could also do the while loop in the primal update for an element in one step by incrementing the sets by some μ determined by the values c_S and the values x_S just before e arrives.

Note that while we are updating the X_e covering the current element e, we are also indirectly amortizing that charge againt possible future elements just as the charge for the current element was possibly already partially charged to previous elements.

Then considering how the analysis would proceed (i.e. relating the change in the primal and the dual after processing an element), we can then hope to come up with an appropriate way to define the primal update step.

The analysis for the fractional set cover algorithm

The analysis of the above primal-dual algorithm consists of the following three steps:

1. The primal solution $(x_S)_{S \in \mathcal{S}}$ is feasible. The inner while loop insures that the primal constraint is satisfied. Future increases in x_S cannot make the constraint infeasible after it has already been satisfied.
2. Let ΔP_e (ΔD_e) denote the change in the objective of the primal (dual) after processing constraint (variable) corresponding to the arrival of element $e \in X'$. We show that $\Delta P_e \leq 2\Delta D_e$.
3. We need to show that the dual solution $(y_e)_{e \in X'}$ can violate dual constraints by a factor of at most $O(\log F)$ (per each constraint), i.e., $\sum_{e \in S} y_e = O(c_S \log F)$ for all $S \in \mathcal{S}$.
   Observe that as soon as x_S reaches 1, the variable y_e stops being updated for $e \in S$. Moreover, each update of x_S in the innermost for loop corresponds to increasing exactly one y_e by 1. Thus, the maximum value $\sum_{e \in S} y_e$ can get is bounded by the number of updates required to increase x_S from 0 to 1. (See text.)

Finishing the analysis for the fractional set cover

From part 3 of the proof above it follows that we can scale the dual solution y_e down by a factor $\ell = O(\log F) = O(\log m)$ and obtain a fractional solution to the dual that satisfies all the constraints.

Let $\widehat{y_e} = y_e/\ell$ denote such a fractional solution. Let $\Delta\widehat{D_e}$ denote the change in the dual according to $\widehat{y_e}$ when element e is processed by the online algorithm. It follows that $\Delta\widehat{D_e} = \Delta D_e/\ell$.

Thus, we have

$$OPT \leq ALG = \sum_e \Delta P_e \leq 2\sum_e \Delta D_e = 2\ell \sum_e \Delta\widehat{D_e} \leq 2\ell OPT,$$

where the first inequality follows since primal solution x_S is feasible, the second inequality follows from part (2) above, and the last inequality follows from the feasibility of the dual solution $\widehat{y_e}$.

This completes the analysis

Converting the fractional solution to a randomized online algorithm for the integral set cover problem

We note that in the offline (integral) set cover problem, there is a greedy algorithm that achieves approximation $H_k \leq H_n \leq \log n + 1$ where $k = \max_{|S|:S \in \mathcal{S}}$. Furthermore under well accepted complexity assumptions, this is the optimal approximation ratio. It is not suprising then that in what follows an additional $O(\log n)$ factor will arise when we try to transform the fractional solution to an integral solution.

We can design a randomized online algorithm that with high probability achieves competitive ratio $O(\log n \log m)$ for the *integral* Online Set Cover problem. Algorithm 24 on the next slide presents the pseudocode. (Algorithm 23 is the fractional set cover algorithm.) In words, the algorithm can be described as follows: for each $S \in \mathcal{S}$ choose $2 \ln n$ independent random variables $Z_{S,i}$, if x_S exceeds the minimum of the corresponding $Z_{S,i}$ then include S in the integral solution.

The randomized online set cover algorithm

Algorithm 24 A randomized algorithm for the Online Set Cover.

```
procedure RandomizedSetCover
    $R \leftarrow \emptyset$                                  ▷ stores integral solution
    for $S \in \mathcal{S}$ do
        $Z_S \leftarrow \infty$
        for $i \leftarrow 1$ to $2 \ln n$ do
            $Z_{S,i} \leftarrow$ an independent random variable distributed uniformly on interval $[0, 1]$.
            $Z_S \leftarrow \min(Z_S, Z_{S,i})$
    Run Algorithm 23 and include $S$ into $R$ as soon as $x_S$ exceeds $Z_S$.
    return $R$
```

Theorem

The randomized set cover algorithm produces an integral solution such that

- *the expected competitive ratio is at most* $O(\log n \log F) = O(\log n \log m)$, *and*
- *the solution is feasible with probability at least* $1 - 1/n$.

Proof of analysis

- The expected cost is at most $O(\log n)$ times the cost of the fractional solution $(x_S)_{S\in\mathcal{S}}$. The claim then follows by fractional set cover competitive bound. Let $A_{S,i}$ denote the event that $Z_{S,i} \leq x_S$, which happens with probability exactly x_S. The probability that S is included in R is the probability that one of the $A_{S,i}$ happens for $i \in [2\ln n]$, i.e., $\Pr\left(\bigcup_{i=1}^{2\ln n} A_{S,i}\right) \leq \sum_{i=1}^{2\ln n} \Pr(A_{S,i}) = (2\ln n)x_S$. By the linearity of expectation, the expected value of the objective function achieved by the randomized algorithm is $\sum_{S\in\mathcal{S}}(2\ln n)c_S x_S$
- The probability of a given S being not included in the solution is $\Pr\left(\bigcap_{i=1}^{2\ln n}\overline{A_{S,i}}\right) = (1-x_S)^{2\ln n}$. The probability that the constraint corresponding to element $e\in X'$, i.e., $\sum_{S\in\mathcal{S}:e\in S} x_S$, is violated is the probability that none of the sets from $S\in\mathcal{S} : e\in S$ are included in the solution. This probability is $\prod_{S\in\mathcal{S}:e\in S}(1-x_S)^{2\ln n} \leq \prod_{S\in\mathcal{S}:e\in S}\exp(-x_S(2\ln n)) =$ $\exp\left(-(2\ln n)\sum_{S\in\mathcal{S}:e\in S} x_S\right) \leq \exp(-2\ln n) \leq 1/n^2$

Some comments on fractional and randomized algorithms

- The statement of the competitive bound for the fractional solution (and then for the randomized integral solution) is in terms of $\log F \leq \log m$. If we are willing to just state the bounds in terms of m, then the update step in the fractional algorithm can be simplified to $x_S \leftarrow x_S(1 + 1/c_S)$.
  The algorithm as stated obtains a good competitive ratio but has a small probability of not being feasible (ie.. one or more elements may not be covered). This can happen even though there is only a small probability that none of the sets we have probabilistically chosen to cover an element doesn't actually get chosen. This can likley be avoided at a small cost in the expected ratio.
- What will be done instead is to provide a potential function argument that will tell us when to include a set leading to a deterministic online set cover algorithms.

The integral set cover algorithm

In what follows we will indicate how the fractional algorithm can be directly converted into a deterministic integral algorithm for set cover. This will be done so as to achieve the same $O(\log n \log m)$ competitive ratio and will also insure that the solution obtained will aways be feasible and will obtain the competitive ratio with certainty.

We employ one familiar idea, namely assume the existence of an algorithm $SETCOVER_\alpha$ that will compute the desired approximation if $\alpha \geq OPT$.

$SETCOVER_\alpha$ also has to be able to report failure if we have not chosen a sufficiently large α.

The deterministic *SETCOVER*$_\alpha$

The subroutine $SetCover_\alpha$ relies on the fractional solution x_S produced by the fractional set cover algorithm. Define $x_e := \sum_{S \in \mathcal{S}: e \in S} x_S$. The subroutine keeps track of an integral solution R and makes use of the following potential function:
$\Phi = \sum_{e \notin \cup_{S \in R} S} n^{2x_e} + n \cdot$
$\exp\left(\frac{1}{2\alpha}\left(\sum_{S \in R}(c_S - 3x_S c_S \log n) - \sum_{S \notin R} 3x_S c_S \log n\right)\right)$. The pseudocode of the subroutine $SetCover_\alpha$ is as follows:

```
procedure $SetCover_\alpha$
    $R \leftarrow \emptyset$
    Run Algorithm 23 and each time $x_S$ is updated perform the following:
    if $S \notin R$ then
        $\Phi \leftarrow$ value of the potential function before increasing $x_S$
        $\Phi' \leftarrow$ value of the potential function after increasing $x_S$
        $\Phi'' \leftarrow$ value of the potential function after increasing $x_S$ and including $S$ into $R$
        if $\Phi'' \leq \Phi$ then
            $R \leftarrow R \cup \{S\}$
        if $\Phi > \max(\Phi', \Phi'')$ then
            return fail.
```

What needs to be proved about $SETCOVER_\alpha$

The following lemma (so far without proof) summarizes the properties of the subroutine $SetCover_\alpha$.

Lemma

During the execution of the Algorithm the following hold:

1. *if $\alpha \geq OPT$, the algorithm doesn't fail;*
2. *if $x_e > 1$ then e is covered by one of the sets in R;*
3. $\sum_{S \in R} c_S = \alpha O(\log n \log m)$.

The full algorithm then just proceeds as in the standard doubling strategy; that is, double α whenever $SETCOVER\alpha$ reports failure.

A nearly matching inapproximation

Finally, we just state (see the text for the proof) that there is a "nearly" matching inapproximation even for the unweighted problem (i.e., $c_S = 1$ for all sets S. .

Theorem

Let ALG be an online deterministic algorithm for the Unweighted Online Set Cover problem. Fix $\delta > 0$. Then for values of n and m satisfying $\log n \leq m \leq e^{\sqrt{n}-\delta}$ we have[a]

$$\rho(ALG) = \Omega\left(\frac{\log m \log n}{\log\log n + \log\log m}\right).$$

[a]We note that for the exceptional values of m outside of the stated bounds, there are better competitie ratios that can be obtained.

The makespan problem in the unrelated machine model.

Recall in this most general machine model, every job J_j is represented by a vector $(p_{1,j}, p_{2,j}, \ldots, p_{m,j})$ where $p_{i,j}$ is the load or processing time incurred if the j^{th} job is scheduled on machine i. The objective is as in other makespan problems to minimize the maxiumum (over all machines) load on any machine where the load on a machine is the sum of the loads of jobs assigned to that machine.

Since makespan for identical machine is NP-hard, it follows that this is also an NP-hard problem. In fact it is hard to approximate within a factor $\frac{3}{2} + \epsilon$. It is also known that there is an IP/LP rounding algorithm that solves the problem with a 2-approximation.

Closing this gap has attracted some recent attention but the gap remains.

Setting up the LP and dual

We again want to use the doubling strategy but this time applied to the dual. That is, if we knew OPT, then we would want schedule all jobs while insuring that every machine does not exceed load OPT. So we now ask how many job can be scheduled within some estimate α of OPT. If $\alpha \geq OPT$, then we will insure that all jobs are scheduled. If we learn that α is not sufficient then we will double α.

We make one more observation so as to simplify the primal and dual LPs. Namely, if we can schedule all jobs within makespan α, then we can schedule all "normalized jobs" $\widehat{p_{i,j}} = p_{i,j}/\alpha$ within makespan 1.

For each job j we define the set of compatible machines, denoted by $M(\alpha, j)$, as those on which normalized processing time is at most 1. We are applying the doubling strategy to the dual problem D, a maximization problem, so in keeping with the framework as stated for minimnization problems, we take the dual of the dual D to obtain the primal P.

The LPs for the normalized makespan problem

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{j\in[n]} \sum_{i\in M(\alpha,j)} x_{i,j} & \\
\text{subj. to} & \displaystyle\sum_{i\in M(\alpha,j)} x_{i,j} \le 1 & j \in [n] \\
 & \displaystyle\sum_{j\in[n]} \widehat{p_{i,j}} x_{i,j} \le 1 & i \in [m] \\
 & x_{i,j} \ge 0 & i \in [m], j \in [n]
\end{array}
$$

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{j\in[n]} y_j + \sum_{i\in[m]} z_i & \\
\text{subj. to} & y_j + \widehat{p_{i,j}} z_i \ge 1 & j \in [n], i \in M(\alpha, j) \\
 & y_j, z_i \ge 0 & i \in [m], j \in [n]
\end{array}
$$

A primal dual analysis of the Ranking algorithm for bipartite matching

Recall that nodes $V = \{v_j\}$ are offline and known in advance, while nodes $U = \{u_i\}$ appear online one at a time. When node u_i arrives, we also learn its neighbors $N(u_i) \subseteq V$ and we must declare which node u_i is matched to or that u_i is not matched. We also write $N_c(u_i)$ for the "current neighborhood" of u_i, i.e., the set of neighbors of u_i that are currently unmatched.

We previously proved that the Ranking algorithm achieves competitive ratio $1 - 1/e$ in expectation and that no other algorithm can do better. In this section, we present an analysis of Ranking (for the unweighted BMM problem) via a primal-dual approach and then we extend this analysis to work for the offline vertex-weighted version of bipartite matching where weights are on vertices in V.

Rather than ranking the offline vertices by a random permutation, we can equivalently choose random $Y_i \in [0,1]$ for each $v_i \in V$ and prioritize the $\{v_i\}$ in order of increasing values of Y_i.

The fractional LPs for the BMM problem

Once again, we will present the maximization BMM problem as the dual. We introduce variables $x_{i,j}$ with the intended meaning "$x_{i,j} = 1$" indicates that $u_i \in U$ is matched with $v_j \in V$. The fractional version of the Online BMM can be stated as a linear program as follows.

$$\begin{array}{llll}
 & \text{Primal} & & \\
\text{minimize} & \sum_{u_i \in U} \beta_i + \sum_{v_j \in V} \alpha_j & & \\
\text{subj. to} & \beta_i + \alpha_j \geq 1 & & (u_i, v_j) \in E \\
 & \beta_i \geq 0 & & u_i \in U \\
 & \alpha_j \geq 0 & & v_j \in V
\end{array}$$

$$\begin{array}{llll}
 & \text{Dual} & & \\
\text{maximize} & \sum_{(u_i, v_j) \in E} x_{i,j} & & \\
\text{subj. to} & \sum_{j : (u_i, v_j) \in E} x_{i,j} \leq 1 & & u_i \in U \\
 & \sum_{i : (u_i, v_j) \in E} x_{i,j} \leq 1 & & v_j \in V \\
 & x_{i,j} \geq 0 & & (u_i, v_j) \in E
\end{array}$$

The primnal dual version of the Ranking algorithm

```
procedure $RankingPD$
    $x_{i,j} \leftarrow 0$ for $(u_i, v_j) \in E$
    $\beta_i \leftarrow 0$ for $u_i \in U$
    $\alpha_j \leftarrow 0$ for $v_j \in V$
    $M \leftarrow \emptyset$
    for $j \in V$ do
        $Y_j \leftarrow$ uniformly random number in $[0, 1]$
    while new online vertex $u_i$ with neighborhood $N(u_i)$ arrives do
        if $N_c(u_i) \neq \emptyset$ then
            $v_j \leftarrow \arg\min\{Y_j \mid v_j \in N_c(u_i)\}$
            $M \leftarrow M \cup \{(u_i, v_j)\}$
            $x_{i,j} \leftarrow 1$
            $\beta_i \leftarrow (1 - \exp(Y_j - 1))/(1 - 1/e)$
            $\alpha_j \leftarrow \exp(Y_j - 1)/(1 - 1/e)$
    return $M$
```

The primal dual analysis of Ranking

Aside: The authors Devanur, Jain and R. Kleinberg say “we provide what we believe to be the simplest analysis yet of the Ranking algorithm”. Of course, simplicity is in the eye of the beholder. Or rather, what individuals are most familiar with. I am not personally convinced that this is a simpler analysis even assuming that the assignments to β_i and α_i are what is needed.

Theorem

The primal solution $\{x_{i,j}\}$ constructed by the primal dual Ranking algorithm is feasible. The dual solution $\{\beta_i, \alpha_j\}$ is feasible in expectation. The expected competitive ratio of Ranking is $1 - 1/e$.

It is easy to see that the primal solution is always feasible, since M forms a matching and indicator variables $x_{i,j}$ encode this matching. It is harder to see that the dual solution is feasible in expectation.

The dual is feasible in expectation

We need to establish that for every i and j it holds that $\mathbb{E}_{Y_1,\ldots,Y_n}(\beta_i + \alpha_j) \geq 1$.
I'll repeat the proof in the text just to get an indication that this is not a straightforward proof.

Fix an arbitrary edge $\{u_i, v_j\} \in E$. Consider

$$\mathbb{E}_{Y_1,\ldots,Y_n}(\beta_i + \alpha_j) = \mathbb{E}_{Y_1,\ldots,\widehat{Y}_j,\ldots,Y_n}\left(\mathbb{E}_{Y_j}(\beta_i + \alpha_j \mid Y_1, \ldots, \widehat{Y}_j, \ldots, Y_n)\right),$$

where the hat indicates that the corresponding entry is missing from the vector. We will show that $\mathbb{E}_{Y_j}(\beta_i + \alpha_j \mid Y_1, \ldots, \widehat{Y}_j, \ldots, Y_n) \geq 1$ for any values of $Y_{j'}$ for $j \neq j'$. As such, fix values of $Y_{j'}$ for $j \neq j'$. Let $V' = V \setminus \{v_j\}$. Consider running *RankingPD* on the instance G' where the offline side is V' with the fixed values of $Y_{j'}$. If *RankingPD* leaves online node u_i unmatched, then set $y^* = 1$, otherwise set y^* to the value of $Y_{j'}$ for $j' \neq j$ such that $v_{j'}$ is matched with u_i. Let β_i^* the value of β_i in this run, i.e., $\beta_i^* = (1 - \exp(y^* - 1))/(1 - 1/e)$. Now return to the run of *RankingPD* on the original instance G with offline side V and consider the same values of $Y_{j'}$ for $j' \neq j$ and random Y_j.

The dual is feasible in expectation continued

We have the following observations:

- If $Y_j < y^*$ then node v_j gets matched. If v_j is matched prior to arrival of u_i we are done. Otherwise suppose v_j is not yet matched when u_i arrives. Then this run is so far identical to the run on G'. Now, v_j has the corresponding value Y_j which is smaller than the value y^* by assumption. Therefore v_j will be selected in the argmin step to be matched with u_i. This implies that

$$\mathbb{E}_{Y_j}(\alpha_j \mid Y_1, \ldots, \widehat{Y}_j, \ldots, Y_n) \geq \int_0^{y^*} \exp(z-1)/(1-1/e)dz. \qquad (1)$$

- We have $\beta_i \geq \beta_i^*$ regardless of the value of Y_j. Consider executing *RankingPD* on G' and G in parallel. Then at any point during the execution it can be easily seen by induction that the set of unmatched V vertices contains all unmatched V' vertices. Thus, when vertex u_i arrives in the execution of *RankingPD* on G, its current neighborhood contains its current neighborhood from the execution of *RankingPD* on G'. Therefore, the Y-value of a node matched with u_i can only decrease compared to the y^*.

The dual is feasible in expectation continued

- Since $\exp(z-1)$ is increasing in z, it follows that $(1-\exp(z-1))/(1-1/e)$ is decreasing in z. Therefore, the value of β_i cannot decrease. This implies that
$$\mathbb{E}_{Y_j}(\beta_i \mid Y_1, \ldots, \widehat{Y}_j, \ldots, Y_n) \geq \beta^* = (1-\exp(y^*-1))/(1-1/e). \quad (2)$$

Combining Equations (1) and (2) we obtain:

$$\mathbb{E}_{Y_j}(\beta_i + \alpha_j \mid Y_1, \ldots, \widehat{Y}_j, \ldots, Y_n) \geq \frac{1-\exp(y^*-1)}{1-1/e} + \int_0^{y^*} \frac{\exp(z-1)}{1-1/e} dz$$
$$= \frac{1-\exp(y^*-1)}{1-1/e} + \left(\frac{\exp(z-1)}{1-1/e}\right)\Big|_0^{y^*}$$
$$= \frac{1-\exp(y^*-1)}{1-1/e} + \frac{\exp(y^*-1)}{1-1/e} - \frac{1/e}{1-1/e} = 1$$

The extension to the offline vertex weighted casse

Aggarwal et al [2011] extend the $1 - \frac{1}{e}$ competitive ratio to the case where each offline $v_j \in V$ has a weight w_j. The goal then is to match the online verteics so as to maximize the sum of weights of matched offline vertices.

If $N_c(u_i)$ are the currently available neighbors in V, then if $N_c(u_i) \neq \emptyset$, then we match u_i to $v_j \in N_c(u_i)$ where $v_j = argmax\{v_j(1 - e^{Y_i - 1})\}$.

Note that when $w_j = 1$ for all j, this is the ranking algorithm.

In the primal dual formulation, the primal is unchanged and now the dual is to minimize $\sum \alpha_i + \sum \beta_j$ subject to $\alpha_i + \beta_j \geq w_j$

The α_i, β_j variables are now set to be :
$\beta_j \leftarrow w_j(1 - \exp(Y_j - 1))/(1 - 1/e)$
$\alpha_i \leftarrow w_j \exp(Y_j - 1)/(1 - 1/e)$

Note: If the online vertices are weighted then we cannot achieve any competitive ratio that does not depend on these values.