Lecture 8

Online bipartite matching (cont.)

Online Bipartite Matching

- Bipartite graph $G = (U \cup V, E)$, $E \subseteq U \times V$
- $|U| = |V| = n$
- V is fixed
- Nodes in U arrive online, adversarially
  - Say the arrival order is $u_1, \dots, u_n$
  - With arrival of each $u_i \in U$, you discover its edges to V
  - Must irrevocably match it to one of its neighbors in V that is yet unmatched (if possible and desired)
- Compete with the maximum offline matching

Online Bipartite Matching

- Algorithm INFANT
  - For every u_i, if it has unmatched neighbors in V, match it to one of the unmatched neighbors selected *arbitrarily*.

- Produces a *maximal* matching
  - Worst case ½ approximation of the maximum matching
  - WHY?

- Can we do better?

Online Bipartite Matching

- Algorithm RANKING [KVV90]
  - Before the arrival starts, fix a random permutation σ of vertices in V. This acts as random priorities.
  - For every u_i, match it to its unmatched neighbor that is highest in σ (if one exists).
- Claim: RANKING gives a $1 - {}^{1}/_{e}$ approximation.
- Question: If the priorities are random anyway, how is this different from matching to a random unmatched neighbor (Algorithm INFANT)?

Proofs

- The original 1990 paper had a bug in the proof, which was corrected by Krohn and Varadarajan in 2007 (17 years later!)
- Simple combinatorial proof by Birnbaum and Mathieu [08]
- A different IP/LP duality proof by Devanur, Jain and Kleinberg [13]

An Incorrect Proof

- Note: For the worst-case, we can assume the offline optimal m^* is a perfect matching.
- Suppose RANKING produces matching m_σ.
- Claim 1: For $u \in U$, if $v = m^*(u)$ is unmatched in m_σ, then $m_\sigma(u) = v'$ such that $\sigma(v') < \sigma(v)$.
  - If v is unmatched at the end, v was unmatched when u arrived
  - u must have been matched to a higher priority vertex in V

An Incorrect Proof

- Claim 2: Let p_t = probability (over σ) that priority t vertex is matched. Then $1 - p_t \leq \frac{1}{n} \sum_{1 \leq s \leq t} p_s$
- Incorrect Proof:
  - Let $u \in U$ be matched to priority t vertex ($v = \sigma(t)$) in m^*
    - That is, $m^*(u) = v$. Note that both u and v are random variables.
  - Let $U_t \subseteq U$ be matched to vertices with priority < t in m_σ
  - By Claim 1, if v is not matched, then u must be matched to a vertex with priority < t. Thus, $1 - p_t \leq \Pr[u \in U_t]$.
  - u is independent of U_t, so $\Pr[u \in U_t] = \frac{|U_t|}{n} = \frac{1}{n} \sum_{1 \leq s \leq t} p_s$
  - What's wrong in this argument?

Sketch of the Correct Proof

- u and U_t are dependent on each other due to t
  - u is matched to vertex with priority t under m^*
  - U_t has vertices matched to priority $< t$ under m_σ

- The correct (but less intuitive, and more complex) proof demonstrates that …
  - We can choose u independent of v ($\Rightarrow$ independent of R_t)
  - And yet achieve "v unmatched in $m_\sigma \Rightarrow u \in R_t$"

The rest of the proof

- Claim 2: Let p_t = probability (over σ) that priority t vertex is matched. Then $1 - p_t \le \frac{1}{n} \sum_{1 \le s \le t} p_s$
  - How does this help derive $1 - {}^{1}/_{e}$ approximation?
  - $S_t = \sum_{1 \le s \le t} p_s$.
  - Then, $1 - (S_t - S_{t-1}) \le (1/n) S_t$
    - This simplifies to $1 + S_{t-1} \le \left(\frac{n+1}{n}\right) S_t$ (1)
  - Approximation ratio = $\frac{|m_\sigma|}{n} = \frac{S_n}{n}$
    - Smallest when all inequalities in (1) are equalities.
    - Solve the recurrence to get $\frac{S_n}{n} \ge 1 - \left(\frac{n}{n+1}\right)^n \ge 1 - {}^{1}/_{e}$

Devanur et al. Proof

- Proof using LP relaxation + duality
  - Hope is that this will help in analyzing the unsolved adwords problem

Primal

max $\sum_{e \in E} x_e$
s.t.

$\sum_{e \in \delta(v)} x_e \leq 1 \quad \forall v \in \mathrm{V}$
$\sum_{e \in \delta(u)} x_e \leq 1 \quad \forall u \in U$
$x_e \geq 0 \quad \forall e \in E$

Dual

min $\sum_{v \in V} \alpha_v + \sum_{u \in U} \beta_u$
s.t.

$\alpha_v + \beta_u \geq 1 \quad \forall (u, v) \in E$
$\alpha_v, \beta_u \geq 0 \quad \forall v \in V, u \in U$

Devanur et al. Proof

- Standard technique
  - Start constructing a dual solution (e.g., using water-filling)
  - This may be a fractional solution
    - Thus not a feasible solution for the integral problem
  - Use this as a guide to set integral values of variables in the primal problem to generate a feasible solution that is not too far from the dual value
- But we already have a solution given by RANKING
  - We will simply see what it does in the dual formulation

Devanur et al. Proof

- Outline
  - Take the primal solution given by RANKING
    - Primal objective value P = size of matching
  - Construct the corresponding fractional dual solution
    - Let the dual objective value be D
  - Show that the dual solution is feasible
    - So $D \geq OPT \geq P$
    - $OPT =$ size of maximum matching
  - Show that the primal value is not too far
    - $P \geq \left(1 - \frac{1}{e}\right) D \geq \left(1 - \frac{1}{e}\right) OPT$

Devanur et al. Proof

- Outline
  - Take the primal solution given by RANKING
    - Primal objective value P = size of matching
  - Construct the corresponding fractional dual solution
    - Let the dual objective value be D
  - Show that the dual solution is feasible
    - A technical note: Since m_σ is a random variable, the dual solution constructed is also random.
    - It suffices to show that the *expected dual solution* (i.e., one obtained by taking expected value of each variable) is feasible.

Devanur et al. Proof

- Another side note
  - For simplicity, we will analyze the following algorithm equivalent to RANKING.
  - Instead of creating a priority ordering σ, we will assign a random number $Y_v \sim U[0,1]$ to each node $v \in V$
    - Lower number means higher priority.

Devanur et al. Proof

- Step 1: Construct dual solution from primal
  - Take a function $g: [0,1] \rightarrow [0,1]$ such that $g(1) = 1$.
  - Let F be the approximation factor we want to prove.
    - For us, $F = 1 - 1/e$
  - For every (u, v) matched by RANKING, set

$$\alpha_v = \frac{g(Y_v)}{F}, \qquad \beta_u = \frac{1 - g(Y_v)}{F}$$

  - For all other u and v, set α_v and β_u to 0.

Devanur et al. Proof

- A couple of observations about RANKING
  - Take any edge (u, v) in the graph
  - Let y^c denote the priority of the vertex to which u would be matched if v was absent
    - If u would have been unmatched, set $y^c = 1$
  - Claim 1: If $Y_v < y^c$, then v must get matched.
    - v may get matched before u arrives. But if not, it is surely matched to u.
  - Claim 2: u cannot be matched to a worse priority vertex due to presence of v
    - WHY?
    - Thus, $\beta_u \geq \beta_u^c$ (which is β_u when v is absent)

Devanur et al. Proof

- Step 2: Show that the expected dual is feasible.
  - We want to show that for any edge (u, v) in the graph, $E[\alpha_v + \beta_u] \geq 1$
  - Recall: $\alpha_v = g(Y_v)/F$ if v is matched by ranking.
  - Recall: v is matched if $Y_v < y^c$
    - $E[\alpha_v] \geq \int_0^{y^c} g(y)dy/F$
  - Recall: $\beta_u \geq \beta_u^c = (1 - g(y^c))/F$
  - Thus, $E[\alpha_v + \beta_u] = \left(\frac{1}{F}\right) E\left[\int_0^{y^c} g(y)dy + 1 - g(y^c)\right]$
    - Result follows if $\int_0^{\theta} g(y)dy + 1 - g(\theta) \geq F$, for all $\theta \in [0,1]$

Devanur et al. Proof

- Now it's simple calculus.
  - We can show that the optimal g is $g(y) = e^{y-1}$
  - And the corresponding highest value of F (the highest approximation this method can prove) is $1 - e^{-1}$
    - $\int_0^\theta g(y)dy + 1 - g(\theta) = e^{\theta-1} - e^{-1} + 1 - e^{\theta-1} = 1 - e^{-1}$

- We already know that RANKING does no better than $1 - e^{-1}$.

Devanur et al. Proof

- Step 4: Show that the integral primal solution is not too far from the fractional dual solution: $P \geq F \cdot D$
  - Recall that in our construction, for every edge (u, v) in the primal, we set α_v and β_u such that $\alpha_v + \beta_u = 1/F$.
  - Crucially, for all other vertices, we set them 0.
  - So $D = \sum_v \alpha_v + \sum_u \beta_u = P/F$
  - QED!

What's Cookin'?

- Better approximations in other models
  - CR(adv) $\le$ CR(ROM) $\le$ CR(Unknown-IID) $\le$ CR(Known-IID)

- Q: Why is CR(ROM) $\le$ CR(Unknown-IID)?
  - Take an algorithm with α approximation for ROM, and apply it for Known-IID model.
  - Take sequences generated by known-IID model.
  - Partition them such that in each part, all sequences have same multiset of items.
  - In each part, ROM approximation applies.

What’s Cookin’?

- Better approximations in other models
  - CR(adv) $\leq$ CR(ROM) $\leq$ CR(Unknown-IID) $\leq$ CR(Known-IID)

- ROM/Unknown-IID: RANKING gives 0.696. It’s not clear if we can do better.
- Known-IID: Can do at least 0.708, but not better than 0.823.

What's Cookin'?

- Adwords Problem
  - Left = advertisers, right (online) = ads
  - Advertisers bid on incoming ads (weighted edges)
  - Advertisers have budget
    - Cannot always assign every ad to highest-bid advertiser

- Adversarial model: Greedy gives (1/2)-approximation, but it's not clear if we can do better
  - If we assume bids << budget, then $1 - 1/e$ approximation is possible.

Randomization Continued

- In previous examples, we used randomization to achieve approximation because OPT is
  - either unknowable (online case)
  - or incomputable (NP-hard)

- Randomization can also be used to reduce the expected running time of an algorithm
  - We still want *the* optimal solution, but we want to compute it in time that is polynomial in expectation

Revisiting 2-SAT

- CNF formula with two literals in every clause
  - E.g., $(x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3) \wedge (x_1 \vee x_2)$

- Bad example because
  - MAX-2-SAT is NP-hard, but 2-SAT (find a satisfying assignment if it exists, return FALSE if it doesn't) is in P.
  - We want to solve 2-SAT, which can be solved in polytime deterministically.
  - We'll use randomization anyway. Just because.

Revisiting 2-SAT

- First, let's do deterministic polytime 2-SAT.
- Algorithm:
  - Eliminate all unit clauses, set the corresponding literals.
  - Create a graph with $2n$ literals as vertices.
  - For every clause $(x \vee y)$, add two edges: $\bar{x} \rightarrow y$ and $\bar{y} \rightarrow x$.
    - If the source is true, then the destination must be true.
  - Formula is satisfiable iff there are no paths from x to $\bar{x}$ or $\bar{x}$ to x for any x
  - Just solve $s - t$ connectivity problem in polynomial time

Random Walk + 2-SAT

- Here's a cute randomized algorithm by Papadimitriou [1991]

- Start with an arbitrary assignment.
- While there is an unsatisfied clause $C = (x \vee y)$
  - Pick one of the two literals with equal probability.
  - Flip the variable value so that C is satisfied.

- But, but, this can hurt other clauses?

Random Walk + 2-SAT

- Theorem: If there exists a satisfying assignment τ^*, then the expected time taken by the algorithm to reach a satisfying assignment is at most $2n^2$.
- Proof:
  - Fix τ^*. Let τ_0 be the starting assignment. Let τ_i be the assignment after i iterations.
  - Consider the "hamming distance" d_i between τ_i and τ^*
  - $d_i \in \{0,1,\dots,n\}$.
  - We want to show that in expectation, we will hit $d_i = 0$ in $2n^2$ iterations, unless the algorithm stops before that.

Random Walk + 2-SAT

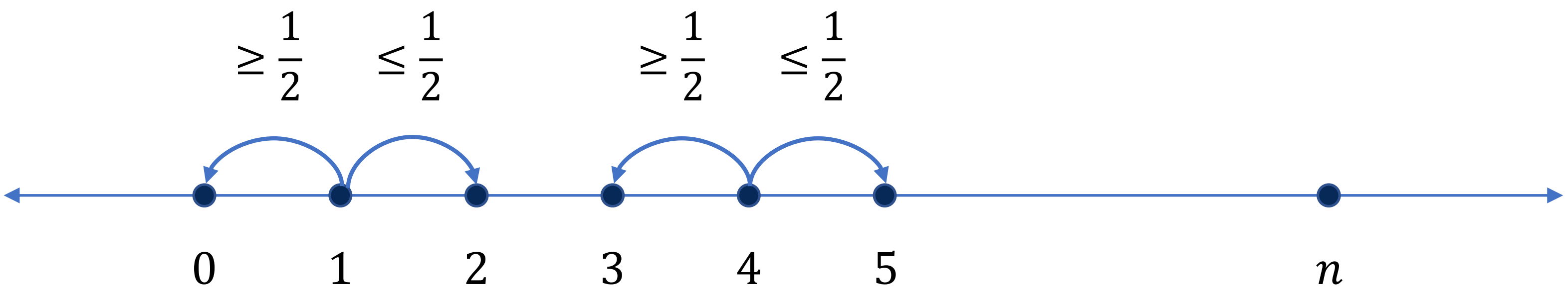
- Observation: $d_{i+1} = d_i - 1$ or $d_{i+1} = d_i + 1$
  - Because we change one variable in each iteration.
- Claim: $\Pr[d_{i+1} = d_i - 1] \geq 1/2$
- Proof:
  - Iteration i considers an unsatisfied clause $C = (x \vee y)$
  - τ^* satisfies at least one of x or y, while τ_i satisfies neither
  - Because we pick a literal randomly, w.p. at least ½ we pick one where τ_i and τ^* differ, and decrease distance.
  - Q: Why did we need an unsatisfied clause? What if we pick one of n variables randomly, and flip it?

Random Walk 2-SAT

- A: We want the distance to decrease with probability at least $\frac{1}{2}$ no matter how close or far we are from τ^*.
- If we are already close, choosing a variable at random will likely choose one where τ and τ^* already match.
  - Flipping this variable will increase the distance with high probability.
- An unsatisfied clause narrows it down to two variables s.t. τ and τ^* differ on at least one of them

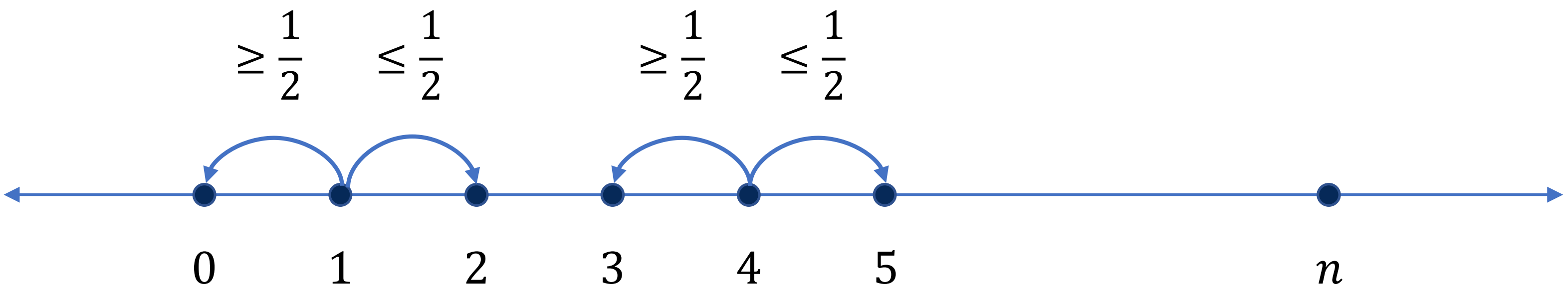
Random Walk + 2-SAT

- Observation: $d_{i+1} = d_i - 1$ or $d_{i+1} = d_i + 1$
- Claim: $\Pr[d_{i+1} = d_i - 1] \geq 1/2$



- How does this help?

Random Walk + 2-SAT



- How does this help?
  - Can view this as Markov chain and use hitting time results
  - But let's prove it with elementary methods.
  - $T_{i+1,i}$ = expected time to go from $i+1$ to i
    - $T_{i+1,i} \leq \left(\frac{1}{2}\right) * 1 + \left(\frac{1}{2}\right) * T_{i+2,i} \leq \frac{1}{2} + \left(\frac{1}{2}\right) * \left(T_{i+2,i+1} + T_{i+1,i}\right)$
    - Thus, $T_{i+1,i} \leq 1 + T_{i+2,i+1} \rightarrow T_{i+1,i} = O(n)$
    - $T_{n,0} \leq T_{n,n-1} + \cdots + T_{1,0} = O(n^2)$

Random Walk + 2-SAT

- Can view this algorithm as a "drunken local search"
  - We are searching the local neighborhood
  - But we don't ensure that we necessarily improve.
  - We just ensure that in expectation, we aren't hurt.
  - Hope to reach a feasible solution in polynomial time
- Schöning extended this technique to k-SAT
  - Schöning's algorithm no longer runs in polynomial time, but this is okay because k-SAT is NP-hard
  - It still improves upon the naïve 2^n
  - Later derandomized by Moser and Scheder [2011]

Schöning's Algorithm

- Choose a random assignment τ.
- Repeat $3n$ times (n = #variables)
  - If τ satisfies the CNF, stop.
  - Else, pick an arbitrary unsatisfied clause, and flip a random literal in the clause.

Schöning's Algorithm

- Randomized algorithm with one-sided error
  - If the CNF is satisfiable, it finds an assignment with probability at least $\left(\frac{1}{2}\right)\left(\frac{k}{k-1}\right)^n$
  - If the CNF is unsatisfiable, it surely does not find an assignment.

- Expected # times we need to repeat = $\left(2\left(1-\frac{1}{k}\right)\right)^n$
  - For $k = 3$, this gives $O(1.3333^n)$
  - For $k = 4$, this gives $O(1.5^n)$

Best Known Results

- 3-SAT

- Deterministic
  - Derandomized Schöning's algorithm: $O(1.3333^n)$
  - Best known: $O(1.3303^n)$ [HSSW]
    - If there is a unique satisfying assignment: $O(1.3071^n)$ [PPSZ]

- Randomized
  - Nothing better known without one-sided error
  - With one-sided error, best known is $O(1.30704^n)$ [Modified PPSZ]

Random Walk + 2-SAT

- Random walks are not only of theoretical interest
  - WalkSAT is a practical SAT algorithm
  - At each iteration, pick an unsatisfied clause *at random*
  - Pick a a variable in the unsatisfied clause to flip:
    - With some probability, pick at random.
    - With the remaining probability, pick one that will make the fewest previously satisfied clauses unsatisfied.
  - Restart a few times (avoids being stuck in local minima)
- Faster than "intelligent local search" (GSAT)
  - Flip the variable that satisfies most clauses

Random Walks on Graphs

- Aleliunas et al. [1979]
  - Let G be a connected undirected graph. Then a random walk starting from any vertex will cover the entire graph (visit each vertex at least once) in $O(mn)$ steps.
- Also care about limiting probability distribution
  - In the limit, the random walk with spend $\frac{d_i}{2m}$ fraction of the time on vertex with degree d_i
- Markov chains
  - Generalize to directed (possibly infinite) graphs with unequal edge probabilities