CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
Lecture 7

Allan Borodin

March 3, 2016

Announcements and todays agenda

- Announcements
  1. Start of Assignment 2 was first posted last Thursday and then changed and reposted on Friday and then again last night. There are now 6 questions plus a bonus question.
- Todays agenda
  1. Derandomizing Max-Sat by method of conditional expectations; Max-Sat input models
  2. Johnson's deterministic algorithm for Max-Sat
  3. The Yanakakis LP and randomized rounding for Max-Sat
  4. Vector programs and randomized rounding for Max-2-Sat
  5. Submodular set functions
  6. Greedy algorithms for monotone submodular functions subject to cardinality, matroid and knapsack constraints
  7. The Buchbinder et al "double-sided" greedy algorithm for unconstrained non-monotone submodular maximization.
  8. The "It can't be derandomized but it can be derandomized" results.

Derandomizing the naive algorithm: redux

We derandomize the naive algorithm by what is called the method of conditional expectations. Let $F[x_1, \ldots, x_n]$ be an exact k CNF formula over n propositional variables $\{x_i\}$. For notational simplicity let $true = 1$ and $false = 0$ and let $w(F)|\tau$ denote the weighted sum of satisfied clauses given truth assignment τ.

- Let x_j be any variable. We express $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}]$ as $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 1] \cdot (1/2) + \mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 0] \cdot (1/2)$
- This implies that one of the choices for x_j will yield an expectation at least as large as the overall expectation.
- It is easy to determine how to set x_j since we can calculate the expectation clause by clause.
- We can continue to do this for each variable and thus obtain a deterministic solution whose weight is at least the overall expected value of the naive randomized algorithm.
- NOTE: The derandomization can be done so as to achieve an online algorithm. Here the (online) input items are the propostional variables. What input representation is needed so that it fits (say) the priority formulation for an online algorithm?

Input models for Max-Sat

- For online and priority models we need to specify the input model. For Max-Sat, the natural formulation is that the input items are the propositional variables. In increasing order of providing more information (and possibly better approximations), the following input models can be considered:
  [**Model 0**] : Each propositional variable x is represented by the names of the positive and negative clauses in which it appears.
  [**Model 1**] : Each propositional variable x is represented by the length of each clause C_i in which x appears positively, and for each clause C_j in which it appears negatively.
  [**Model 2**] : In addition, for each C_i and C_j, a list of the other variables in that clause is specified.
  [**Model 3**] : The variable x is represented by a complete specification of each clause it which it appears.
- The naive randomized algorithm can be implemented in "model 0" where we don't even specify the lengths of the clauses and the derandomized 's algorithm can be implemented using input model 1 for arbitrary Max-Sat (resp. model 0 for Exact-k-SAT).

(Exact) Max-k-Sat

- For exact Max-2-Sat (resp. exact Max-3-Sat), the approximation (and totality) ratio of the derandomized naive algorithm is $\frac{3}{4}$ (resp. $\frac{7}{8}$).
- For $k \geq 3$, using PCPs (probabilistically checkable proofs), Hastad proves that it is NP-hard to improve upon the $\frac{2^k-1}{2^k}$ approximation ratio for exact Max-k-Sat and hence cannot improve upon $\frac{7}{8}$ for (arbitrary) Max-Sat.
- For arbitrary Max-Sat (resp. arbitrary Max-2-Sat), the current best approximation ratio is .797 (resp. .931) using semi-definite programming and randomized rounding.
- The analysis for exact Max-k-Sat clearly needed the fact that all clauses have at least k literals. What bound does the naive online randomized algorithm or its derandomization obtain for (not exact) Max-2-Sat or arbitrary Max-Sat (where there can be unit clauses)?

Johnson's Max-Sat Algorithm

Johnson's [1974] algorithm

For all clauses C_i, $w_i' := w_i/(2^{|C_i|})$
Let L be the set of clauses in formula F and X the set of variables
For $x \in X$ (or until L empty)
  Let $P = \{C_i \in L$ such that x occurs positively$\}$
  Let $N = \{C_j \in L$ such that x occurs negatively$\}$
  If $\sum_{C_i \in P} w_i' \geq \sum_{C_j \in N} w_j'$
    $x := true; L := L \setminus P$
    For all $C_r \in N$, $w_r' := 2w_r'$ **End For**
  Else
    $x := false; L := L \setminus N$
    For all $C_r \in P$, $w_r' := 2w_r'$ **End For**
  End If
  Delete x from X
End For

Aside: This reminds me of the Freund and Shapire [1997] and related boosting algorithms from ML

Johnson's algorithm is the derandomized algorithm

- Twenty years after Johnson's algorithm, Yannakakis [1994] presented the naive algorithm and showed that Johnson's algorithm is the derandomized naive algorithm.
- Yannakakis also observed that for arbitrary Max-Sat, the approximation of Johnson's algorithm is at best $\frac{2}{3}$. For example, consider the 2-CNF $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge \bar{y}$ when variable x is first set to true.
- Chen, Friesen, Zheng [1999] showed that Johnson's algorithm achieves approximation ratio $\frac{2}{3}$ for arbitrary weighted Max-Sat.
- A somewhat simpler analysis is later obtained by Engebretsen [2004] and the approximation bound is better stated as $W_{Johnson} \geq \frac{W_{Total}+W_{OPT}}{3}$.
- In proving the $(2/3)$ approximation ratio for Johnson's Max-Sat algorithm,, Chen et al asked whether or not the ratio could be improved by using a random ordering of the propositional variables (i.e. the input items). This is an example of the random order model (ROM), a randomized variant of online algorithms.

Improving on Johnson's algorithm

- The question asked by Chen et al was answered by Costello, Shapira and Tetali [2011] who showed that in the ROM model, Johnson's algorithm achieves approximation $(2/3 + \epsilon)$ for $\epsilon \approx .003653$
- Poloczek and Schnitger [same SODA 2011 conference] show that the approximation ratio for Johnsons algorithm in the ROM model is at most $2\sqrt{157} \approx .746 < 3/4$ noting that $3/4$ is the ratio first obtained by Yannakakis' IP/LP approximation.
- Poloczek and Schnitger first consider a "canonical randomization" of Johnson's algorithm"; namely, the canonical randomization sets a variable $x_i = \mathit{true}$ with probability $\frac{w_i'(P}{w_i'(P)+w_i'(N)}$ where $w_i'(P)$ (resp. $w_i'(N)$) is the current combined weight of clauses in which x_i occurs positively (resp. negatively). Their substantial additional idea is to adjust the random setting so as to better account for the weight of unit clauses in which a variable occurs.

A few comments on the Poloczek and Schnitger algorithm

- The Poloczek and Schnitger algorithm is called Slack and has approximation ratio $= 3/4$.
- In terms of priority algorithms this is a randomized online algorithm (i.e. adversary chooses the ordering) where the variables are represented in the Model 2 input model.
- This approximation ratio is in contrast to Azar et al [2011] who prove that no randomized online algorithm can achieve approximation better than $2/3$ when the input model is the weakest of the input models.
- Finally (in this regard), Poloczek [2011] shows that no deterministic priority algorithm (and hence online algorithm) can achieve a $3/4$ approximation within input Model 2. This provides a sense in which to claim the that Poloczek and Schnitger Slack algorithm "cannot be derandomized".
- The best deterministic priority algorithm within input Model 3 remains an open problem as does the best randomized priority algorithm and the best ROM algorithm.

Revisiting the "cannot be derandomized comment"

Spoiler alert: we will soon be discussing how algorithms that cannot be derandomized in one sense can be derandomized in another sense.

Here is a preview of what is to come:

- The Buchbinder et al [2012] online randomized 1/2 approximation algorithm for Unconstrained Submodular Maximization (USM) cannot be derandomized into a "similar" deterministic algorithm by a result of Huang and Borodin [2014].
- However, Buchbinder and Feldman [2016] show how to derandomize the Buchbinder et al algorithm into a polynomial time "online" deterministic algortihm.
- The Buchbinder et al USM algorithm is the basis for a randomized 3/4 approximation "online" MaxSat (even Submodular Max Sat) algorithm.
- Pena shows how to derandomize this 3/4 approximation algorithm following the approach of Buchbinder and Feldman.

Yannakakis' IP/LP *randomized rounding* algorithm for Max-Sat

- We will formulate the weighted Max-Sat problem as a $\{0,1\}$ IP.
- Relaxing the variables to be in $[0,1]$, we will treat some of these variables as probabilities and then round these variables to 1 with that probability.
- Let F be a CNF formula with n variables $\{x_i\}$ and m clauses $\{C_j\}$. The Max-Sat formulation is :
  maximize $\sum_j w_j z_j$
  subject to $\sum_{\{x_i \text{ is in } c_j\}} y_i + \sum_{\{\bar{x}_i \text{ is in } c_j\}} (1 - y_i) \geq z_j$
  $y_i \in \{0,1\}$; $z_j \in \{0,1\}$
- The y_i variables correspond to the propositional variables and the z_j correspond to clauses.
- The relaxation to an LP is $y_i \geq 0$; $z_j \in [0,1]$. Note that here we cannot simply say $z_j \geq 0$.

Randomized rounding of the y_i variables

- Let $\{y_i^*\}, \{z_j^*\}$ be the optimal LP solution,
- Set $\tilde{y}_i = 1$ with probability y_i^*.

Theorem

Let C_j be a clause with k literals and let $b_k = 1 - (1 - \frac{1}{k})^k$. Then $Prob[C_j$ is satisifed $]$ is at least $b_k z_j^*$.

- The theorem shows that the contribution of the j^{th} clause C_j to the expected value of the rounded solution is at least $b_k w_j$.
- Note that b_k converges to (and is always greater than) $1 - \frac{1}{e}$ as k increases. It follows that the expected value of the rounded solution is at least $(1 - \frac{1}{e})$ LP-OPT $\approx .632$ LP-OPT.
- Taking the max of this IP/LP and the naive randomized algorithm results in a $\frac{3}{4}$ approximation algorithm that can be derandomized.

The SDP/vector program approach: Max-2-Sat

- We briefly consider an important extension of the IP/LP approach, namely representing a problem as a strict quadratic program and then relaxing such a program to a vector program. Vector programs are known to be equivalent to semidefinite programs.
- For our purposes of just introducing the idea of this approach we will not discuss SDP concepts but rather just note that such programs (and hence vector programs) can be solved to arbitrary precision within polynomial time. This framework provides one of the most powerful optimization methods.
- We illustrate the approach in terms of the Max-2-Sat problem. A very similar algorithm and analysis produces the same approximation ratio for the Max-Cut problem.

The quadratic program for Max-2-Sat

- We introduce {-1,1} variables y_i corresponding to the propositional variables. We also introduce a homogenizing variable y_0 which will correspond to a constant truth value. That is, when $y_i = y_0$, the intended meaning is that x_i is set *true* and *false* otherwise.
- We want to express the $\{-1, 1\}$ truth value $val(C)$ of each clause C in terms of these $\{-1, 1\}$ variables.
  1. $val(x_i) = (1 + y_i y_0)/2$
     $val(\bar{x}_i) = (1 - y_i y_0)/2$
  2. If $C = (x_i \vee x_j)$, then $val(C) = 1 - val(\bar{x}_i \wedge \bar{x}_j) = 1 - (\frac{1-y_i y_0}{2})(\frac{1-y_j y_0}{2}) = (3 + y_i y_0 + y_j y_0 - y_i y_j)/4 = \frac{1+y_0 y_i}{4} + \frac{1+y_0 y_j}{4} + \frac{1-y_i y_j}{4}$
  3. If $C = (\bar{x}_i \vee x_j)$ then $val(C) = (3 - y_i y_0 + y_j y_0 + y_i y_j)/4$
  4. If $C = (\bar{x}_i \vee \bar{x}_j)$ then $val(C) = (3 - y_i y_0 - y_j y_0 - y_i y_j)/4$

The quadratic program for Max-2-Sat continued

- The Max-2-Sat problem is then to maximize $\sum w_k val(C_k)$ subject to $(y_i)^2 = 1$ for all i
- By collecting terms of the form $(1 + y_i y_j)$ and $(1 - y_i y_j)$ the max-2-sat objective can be represented as the strict quadratic objective: $\max \sum_{0 \leq i < j \leq n} a_{ij}(1 + y_i y_j) + \sum b_{ij}(1 - y_i y_j)$ for some appropriate a_{ij}, b_{ij}.
- Like an IP this integer quadratic program cannot be solved efficiently.

The vector program relaxation for Max-2-Sat

- We now relax the quadratic program to a vector program where each y_i is now a unit length vector $\mathbf{v}_i$ in $\Re^{n+1}$ and scalar multiplication is replaced by vector dot product. This vector program can be (approximately) efficiently solved (i.e. in polynomial time).
- The randomized rounding (from $\mathbf{v}_i^*$ to y_i) proceeds by choosing a random hyperplane in $\Re^{n+1}$ and then setting $y_i = 1$ iff $\mathbf{v}_i^*$ is on the same side of the hyperplane as $\mathbf{v}_0^*$. That is, if $\mathbf{r}$ is a uniformly random vector in $\Re^{n+1}$, then set $y_i = 1$ iff $\mathbf{r} \cdot \mathbf{v}_i^* \geq 0$.
- The rounded solution then has expected value $2\sum a_{ij} Prob[y_i = y_j] + \sum b_{ij} Prob[y_i \neq y_j]$; $Prob[y_i \neq y_j] = \frac{\theta_{ij}}{\pi}$ where θ_{ij} is the angle between $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$.

The approximation ratio (in expectation) of the rounded solution

Let $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta)} \approx .87856$ and let OPT_{VP} be the value obtained by an optimal vector program solution.
Then $\mathbf{E}$[rounded solution] $\geq \alpha \cdot (OPT_{VP})$.

Submodular maximization problems; A small diversion before returning to MaxSat

- A set function $f : 2^U \to \Re$ is submodular if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq U$.
- Equivalently, f is submodular if it satisfies decreasing marginal gains; that is, $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$ for all $S \subseteq T \subseteq U$ and $x \in U$
- We will always assume that f is *normalized* in that $f(\varnothing) = 0$ and non-negative.
- Submodular functions arise naturally in many applications and has been a topic of much recent activity.
- Probably the most frequent application of (and papers about) submodular functions is when the function is also monotone (non-decreasing) in that $f(S) \leq f(T)$ for $S \subseteq T$.
- Note that linear functions (also called modular) functions are a special case of monotone submodular functions.

Submodular maximization continued

In the submodular maximization problem, we want to compute S so as to maximize $f(S)$.

- For monotone functions, we are maximizing $f(S)$ subject to some constraint (otherwise just choose $S = U$).
- For the non monotone case, the problem is already interesting in the unconstrained case. Perhaps the most prominent example of such a problem is Max-Cut (and Max-Di-Cut).
- Max-Cut is an NP-hard problem. Using an SDP approach just as we will see for the Max-2-Sat problem yields the approximation ratio $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta)} \approx .87856$. Assuming UGC, this is optimal.
- For a submodular function, we may be given an explicit representation (when a succinct representation is possible as in Max-Cut) or we access the function by an oracle such as the *value oracle* which given S, outputs the value $f(S)$ and such an oracle call is considered to have $O(1)$ cost. Other oracles are possible (e.g. given S, output the element x of U that maximizes $f(S \cup \{x\}) - f(S)$).

Unconstrained (non monotone) submodular maximization

- Feige, Mirrokni and Vondrak [2007] began the study of approximation algorithms for the unconstrained non monotone submodular maximization (USM) problem establishing several results:
  1. Choosing S uniformly at random provides a $1/4$ approximation.
  2. An oblivious local search algorithm results in a $1/3$ approximation.
  3. A non-oblivious local search algorithm results in a $2/5$ approximation.
  4. Any algorithm using only value oracle calls, must use an exponential number of calls to achieve an approximation $(1/2 + \epsilon)$ for any $\epsilon > 0$.
- The Feige et al paper was followed up by improved local search algorithms by Gharan and Vondrak [2011] and Feldman et al [2012] yielding (respectively) approximation ratios of .41 and .42.
- The $(1/2 + \epsilon)$ inapproximation was augmented by Dobzinski and Vondrak showing the same bound for an explicitly given instance under the assumption that $RP \neq NP$.

The Buchbinder et al (1/3) and (1/2) approximations for USM

In the FOCS [2012] conference, Buchbinder et al gave an elegant linear time deterministic 1/3 approximation and then extend that to a randomized 1/2 approximization. The conceptually simple form of the algorithm is (to me) as interesting as the optimality (subject to the proven inapproximation results) of the result. Let $U = u_1, \ldots u_n$ be the elements of U *in any order*.

The deterministic 1/3 approximation for USM

$X_0 := \varnothing$; $Y_0 := U$
For $i := 1 \ldots n$
  $a_i := f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$; $b_i := f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$
  If $a_i \geq b_i$
    then $X_i := X_{i-1} \cup \{u_i\}$; $Y_i := Y_{i-1}$
    else $X_i := X_{i-1}$; $Y_i := Y_{i-1} \setminus \{u_i\}$
  End If
End For

The randomized 1/2 approximation for USM

- Buchbinder et al show that the "natural randomization" of the previous deterministic algorithm achieves approximation ratio $1/2$.
- That is, the algorithm chooses to either add $\{u_i\}$ to X_{i-1} with probability $\frac{a'_i}{a'_i+b'_i}$ or to delete $\{u_i\}$ from Y_{i-1} with probability $\frac{b'_i}{a'_i+b'_i}$ where $a'_i = \max\{a_i, 0\}$ and $b'_i = \max\{b_i, 0\}$.
- If $a_i = b_i = 0$ then add $\{u_i\}$ to X_{i-1}.
- Note: Part of the proof for both the deterministic and randomized algorithms is the fact that $a_i + b_i \geq 0$.
- This fact leads to the main lemma for the deterministic case:

$$f(OPT_{i-1} - f(OPT_i) \leq [f(X_i - f(X_{i-1}] + [f(Y_i) - f(Y_{i-1}]$$

Here $OPT_i = (OPT \cup \{X_i\}) \cap Y_i$ so that OPT_i coincides with X_i and Y_i for elements $1, \ldots i$ and coincides with OPT on elements $i+1, \ldots, n$. Note that $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$. That is, the loss in OPTs value is bounded by the total value increase in the algorithm's solutions.

Applying the algorithmic idea to Max-Sat

Buchbinder et al are able to adapt their randomized algorithm to the Max-Sat problem (and even to the Submodular Max-Sat problem). So assume we have a monotone normalized submodular function f (or just a linear function as in the usual Max-Sat). The adaption to Submodular Max-Sat is as follows:

- Let $\phi : X \to \{0\} \cup \{1\} \cup \varnothing$ be a standard partial truth assignment. That is, each variable is assigned exactly one of two truth values or not assigned.
- Let $\mathcal{C}$ be the set of clauses in formula Ψ. Then the goal is to maximize $f(\mathcal{C}(\phi))$ where $\mathcal{C}(\phi)$ is the sat of formulas satisfied by ϕ.
- An extended assignment is a function $\phi' : X \to 2^{\{0,1\}}$. That is, each variable can be given one, two or no values. (Equivalently $\phi' \subseteq X \times \{0,1\}$ is a relation.) A clause can then be satisfied if it contains a positive literal (resp. negative literal) and the corresponding variable has value $\{1\}$ or $\{0,1\}$ (resp. has value $\{0\}$ or $\{0,1\}$.
- $g(\phi') = f(\mathcal{C}(\phi'))$ is a monotone normalized submodular function. ‘

Buchbinder et al Submodular Max-Sat

Now starting with $X_0 = X \times \varnothing$ and $Y_0 = Y \times \{0, 1\}$, each variable is considered and set to either 0 or to 1 (i.e. a standard assignment of precisely one truth value) depending on the marginals as in USM problem.

Algorithm 3: RandomizedSSAT(f, Ψ)

$X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N} \times \{0, 1\}$.
for i = 1 *to* n **do**
  $a_{i,0} \leftarrow g(X_{i-1} \cup \{u_i, 0\}) - g(X_{i-1})$.
  $a_{i,1} \leftarrow g(X_{i-1} \cup \{u_i, 1\}) - g(X_{i-1})$.
  $b_{i,0} \leftarrow g(Y_{i-1} \setminus \{u_i, 0\}) - g(Y_{i-1})$.
  $b_{i,1} \leftarrow g(Y_{i-1} \setminus \{u_i, 1\}) - g(Y_{i-1})$.
  $s_{i,0} \leftarrow \max\{a_{i,0} + b_{i,1}, 0\}$.
  $s_{i,1} \leftarrow \max\{a_{i,1} + b_{i,0}, 0\}$.
  with probability $s_{i,0}/(s_{i,0} + s_{i,1})^*$ **do**:
  $X_i \leftarrow X_{i-1} \cup \{u_i, 0\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 1\}$.
  else (with the compliment probability
  $s_{i,1}/(s_{i,0} + s_{i,1})$) **do**:
  $X_i \leftarrow X_{i-1} \cup \{u_i, 1\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 0\}$.
return X_n (or equivalently Y_n).

* If $s_{i,0} = s_{i,1} = 0$, we assume $s_{i,0}/(s_{i,0} + s_{i,1}) = 1$.

Further discussion of Submodular Max-Sat

- The algorithm is shown to have a $\frac{3}{4}$ approximation ratio for Monotone Submodular Max-Sat.
- In the paper, they claim that for the standard Max-Sat (i.e. when the function f is a linear function), that the algorithm can be made to run in linear time.
- Poloczek et al show that the algorithm turns out to be equivalent to a previous Max-Sat algorithm by van Zuylen.
- As mentioned at the start of thre lecture, Huang and B. provide an abstract priority model for the USM problem that provides strong evidence that a deterministic algorithm (in the style of Buchbinder et ali but allowing some ordering of the input elements being accessed by a value oracle) cannot achieve a $\frac{1}{2}$ approximation.

Derandomizing the algorithms for USM and Submodular Max-Sat

- Contrary to the Poloczek, (resp. Huang and B.) priority inapproximations for Max-Sat (resp. UFL), there is a sense in which these algorithms can be derandomized.
- In fact the derandomization becomes an "online algorithm" in the sense that an adversary is choosing the order of the input variables. However rather than creating a single solution, the algorithm is creating a tree of solutions and then takng the best of these.
- The idea is as follows. The analysis of the randomized USM approximation bound shows that a certain linear inequality holds at each iteration of the algorithm. Namely,

$$E[f(OPT_{i-1} - f(OPT_i)] \leq \frac{1}{2} E[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1}]$$

That is, the expected change in restricting OPT in an iteration (by setting the i^{th} variable) is bounded by the average change in the two values being maintained by the algorithm.

Continuing the Buchbinder and Feldman derandomization idea

- These inequalities induce two additional inequalties per iteration on the distributions of solutions that can exist at each iteration.
- This then gets used to describe an LP corresponding to these $2i$ constraints we have for the distributions that hold at each iteration of the algorithm.
- But then using LP theory again (i.e. the number of non-zero variables in a basic solution). It follows that we only need distributions with support $2i$ at each iteration rather than the nave 2^i that would follow from just considering the randomized tree.
- Finally, since there must be at least one distrubution (amongst the final $2n$ distributions) for which the corresponding solution is at least as good as the expected value. Thus if suffices to take the max over a "small" number of solutions.

Randomized online bipartite matching and the adwords problem.

- We return to online algorithms and algorithms in the random order model (ROM). We have already seen evidence of the power of randomization in the context of the UFL and MaxSat problems.
- Another nice sequence of results begins with a randomized online algorithm for bipartite matching due to Karp, Vazirani and Vazirani [1990]. We quickly overview some results in this area as it represents a topic of continuing interest. (The FOCS 2012 conference had a session of three papers related to this topic.)
- In the online bipartite matching problem, we have a bipartite graph G with nodes $U \cup V$. Nodes in U enter online revealing all their edges. A deterministic greedy matching produces a maximal matching and hence a $\frac{1}{2}$ approximation.
- It is easy to see that any deterministic online algorithm cannot be better than a $\frac{1}{2}$ approximation even when the degree of every $u \in U$ is at most (equal) 2

The randomized ranking algorithm

- The algorithm chooses a random permutation of the nodes in V and then when a node $u \in U$ appears, it matches u to the highest ranked unmatched $v \in V$ such that (u, v) is an edge (if such a v exists).
- Aside: making a random choice for each u is still only a $\frac{1}{2}$ approx.
- Equivalently, this algorithm can be viewed as a deterministic greedy (i.e. always matching when possible and breaking ties consistently) algorithm in the ROM model.
- That is, let $\{v_1, \ldots, v_n\}$ be any fixed ordering of the vertices and let the nodes in U enter randomly, then match each u to the first unmatched $v \in V$ according to the fixed order.
- To argue this, consider fixed orderings of U and V; the claim is that the matching will be the same whether U or V is entering online.

The KVV result and recent progress

KVV Theorem

Ranking provides a $(1 - 1/e)$ approximation.

- Original analysis is not rigorous.
- There is an alternative proof (and extension) by Goel and Mehta [2008], and then another proof in Birnbaum and Mathieu [2008].
- Recall that this positive result can be stated either as the bound for a particular deterministic algorithm in the stochastic ROM model, or as the randomized Ranking algorithm in the (adversarial) online model.
- KVV show that the $(1 - 1/e)$ bound is essentially tight for any randomized online (i.e. adversarial input) algorithm. In the ROM model, Goel and Mehta state inapproximation bounds of $\frac{3}{4}$ (for deterministic) and $\frac{5}{6}$ (for randomized) algorithms.
- In the ROM model, Karande, Mehta, Tripathi [2011] show that Ranking achieves approximation at least .653 (beating $1 - 1/e$) and no better than .727.

Some comments on the Birnbaum and Mathieu proof

- The worst case example a (n, n) graph with a perfect matching.
- In particular, for $n = 2$, the precise expected competitive (i.e. approximation) ratiois $\frac{3}{4}$. The inapproximation can be seen by using the Yao principle for obtaining bounds on randomized algorithms.

The main lemma in the analysis

Let x_t be the probability (over the random permutations of the vertices in V) that the vertex of rank t is matched. Then $1 - x_t \leq \frac{1}{n} \sum_{s=1}^{t} x_s$

- Letting $S_t = \sum_{s=1}^{t} x_s$ the lemma can be restated as $S_t(1 + 1/n) \geq 1 + S_{t-1}$ fo all t. Given that the graph has a perfect matching, the expected competitive ratio is S_n/n. It is shown that $\frac{1}{n} S_n \geq 1 - (1 - \frac{1}{n+1})^n \to 1 - 1/e$.

Getting past the $(1 - 1/e)$ bound

- The ROM model can be considered as an example of what is called stochastic optimization in the OR literature. There are other stochastic optimization models that are perhaps more natural, namely i.i.d sampling from known and unknown distributions.
- Feldman et al [2009] study the known distribution case and show a randomized algorithm that first computes an optimal offline solution (in terms of expectation) and uses that to guide an online allocation.
- They achieve a .67 approximation (improved to .699 by Bahmani and Kapralov [2010] and also show that no online algorithm can achieve better than i $26/27 \approx .99$ (improved to .902).
- Karande, et al [2011] how that an approximation in the ROM model implies the same approximation in the unknown distribution model. As previously stated, they show that the KVV Ranking algorithm achieves approximation .653 in the ROM model and is no better than .727.