CSC2420: Algorithm Design, Analysis and Theory Spring (or Winter for pessimists) 2017

Allan Borodin

February 13, 2017

Lecture 6

Announcements:

- Assignment 1 is due today. Any questions?
- I am reserving Thursdays 3-4 for an office hour but since there is little interest in the office hour, it might be best to simply email me and let me know when you would like to see me. I welcome discussing the course whenever I am free.
- I wanted to try to arrange a 1 or 2 hour lecture outside of the usual time but received little feedback. So I am now working with Lalla Mouatadid and she will, give the lecture on February 27. We are planning a lecture on "fine grained complexity".
- I plan to assign some questions for assignment 2 this week.

Todays agenda:

- Continue randomized algorithms.

The naive randomized algorithm for exact Max-k-Sat

We continue our discussion of randomized algorthms by considering the use of randomization for improving approximation algorithms. In this context, randomization can be (and is) combined with any type of algorithm.
For the following discussion of Max-Sat, we will follow the prevailing convention by stating approximation ratios as fractions $c < 1$.

- Recall the exact Max-k-Sat problem where we are given a CNF propositional formula in which every clause has exactly k literals. We consider the weighted case in which clauses have weights. The goal is to find a satisfying assignment that maximizes the size (or weight) of clauses that are satisfied. We considered this problem when discussing local search.
- Since exact Max-k-Sat generalizes the exact k- SAT decision problem, it is an NP hard problem for $k \geq 3$ and moreoever, while 2-SAT is polynomial time computable, Max-2-Sat is still NP hard.
- The naive randomized (online) algorithm for Max-k-Sat is to randomly set each variable to *true* or *false* with equal probability.

Analysis of naive Max-k-Sat algorithm continued

- Since the expectation of a sum is the sum of the expectations, we just have to consider the probability that a clause is satisfied to determine the expected weight of a clause.
- Since each clause C_i has k literals, the probability that a random assignment of the variables occuring in C_i will set the clause to be satisfied is exactly $\frac{2^k-1}{2^k}$. Hence $\mathbf{E}$ [weight of satisfied clauses] $= \frac{2^k-1}{2^k} \sum_i w_i$
- Of course, this probability only improves if some clauses have more than k literals. It is the small clauses that are the limiting factor in this analysis.
- Similatr to the local seacrh analysis, this ratio is not only an approximation ratio but is also a "totality ratio"; the algorithms expected value is a factor $\frac{2^k-1}{2^k}$ of the sum of all clause weights whether satisfied or not.
- We can hope that when measuring against an optimal solution (and not the sum of all clause weights), small clauses might not be as problematic as they are in the above analysis of the naive algorithm.

Derandomizing the naive algorithm

We can 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 change in 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/sufficient?

(Exact) Max-k-Sat

- For exact Max-2-Sat (resp. exact Max-3-Sat), the approximation (and totality) ratio 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.
- For Max-2-Sat, the $\frac{3}{4}$ ratio can be improved (as we will see) by the use of semi-definite programming (SDP).
- 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 derandomztion obtain for (not exact) Max-2-Sat or arbitrary Max-Sat (when there can be (say) unit clauses)?
- Note that the naive randomized algorithm computes an assignment for any propositional formula but the question is how good is the approximation.

Johnson's (1974) deterministic 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 $C_r \in N, \quad w_r' := 2w_r'$ **End For**
  Else
    $x := false; L := L \setminus N$
    For all $C_r \in P, \quad w_r' := 2w_r'$ **End For**
  End If
  Delete x from X
End For

Aside: This reminds me of boosting (Freund and Shapire [1997])

Johnson's algorithm is the derandomization of the naive randomized 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.
- For arbitrary (weighted) Max-Sat (resp. Max-2-Sat), the current best approximation ratio is .797 (resp. .931) using semi-definite programming and "randomized rounding".

Modifying Johnson's algorithm for Max-Sat

- 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), that was in one of the questions in Assignment 1.
- To precisely model the Max-Sat problem within the online of priority frameworks, we need to specify the input model.
- In increasing order of providing more information (and possibly better approximation ratios), 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.

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 $2\sqrt{157} \approx .746 < 3/4$, the ratio first obtained by Yannakakis' IP/LP approximation that we will soon present.
- 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 within input modeli 1.
- 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 input model 0.
- Poloczek [2011] shows that no deterministic priority algorithm can achieve a $3/4$ approximation within input model 2. This provides "a sense" in which to claim that the Poloczek and Schnitger Slack algorithm "cannot be derandomized".
- The best deterministic online or priority algorithm within the most powerful (and natural) input model 3 remains an open problem as does the best randomized priority algorithm and the best deterministic or randomized ROM algorithm.

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 derandomized algorithmi will still be solving LPs.)

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 corresponding quadratic program for the weighted Max-Cut problem is to have $\{-1, 1\}$ variables $\{y_i\}$ to express which side of the cut a vertex is on. Then the Max-Cut problem is to maximize $\frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - y_i y_j)$. The Max-Cut problem is an example of the *unconstrained non-monotone submodular maximization problem (USM)* which we will soon consider.

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 **E**[rounded solution] $\geq \alpha \cdot (OPT_{VP})$.

Submodular maximization problems; An important 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

$X_0 := \emptyset$; $Y_0 := U$
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 the Unconstrained Submodular Maximiation and Submodular Max-Sat algorithms

- The Buchbinder et al [2012] online randomized 1/2 approximation algorithm for Unconstrained Submodular Maximization (USM) cannot be derandomized into a "similar" deterministic online or priority style algorithm by a result of Huang and Borodin [2014]. Like the Poloczek result, we claime this was "in some sense" evidence that this algorithm cannot be derandomized.
- Their algorithm is shown to have a $\frac{3}{4}$ approximation ratio for Monotone Submodular Max-Sat.
- Poloczek et al (to appear in SICOMP) show that the Buchbinder et al algorithm turns out to be equivalent to a previous Max-Sat algorithm by van Zuylen.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses not yet unsatisfied.
Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., unsatsifed clauses) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses not yet unsatisfied.
Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., unsatsifed clauses) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The algorithm's plan is to randomly set variable x_i so as to increase $\mathbb{E}[B_i - B_{i-1}]$.

The randomized (weighted) max-sat $\frac{3}{4}$ approximation algorithm

The idea of the algorithm is that in setting the variables, we want to balance the weight of clauses satisfied with that of the weight of clauses not yet unsatisfied.
Let S_i be the assignment to the first i variables and let SAT_i (resp. $UNSAT_i$) be the weight of satisfied clauses (resp., unsatsifed clauses) with respect to S_i. Let $B_i = \frac{1}{2}(SAT_i + W - UNSAT_i)$ where W is the total weight of all clauses.

The algorithm's plan is to randomly set variable x_i so as to increase $\mathbb{E}[B_i - B_{i-1}]$.

To that end, let t_i (resp. f_i) be the value of $w(B_i) - w(B_{i-1})$ when x_i is set to true (resp. false).

The randomized max-sat approximation algorithm continued

For $i = 1 \ldots n$
  If $f_i \leq 0$, then set $x_i = \text{true}$
  Else if $t_i \leq 0$,
    then set $x_i = \text{false}$
  Else set x_i true with probability $\frac{t_i}{t_i + f_i}$.
End For

The randomized max-sat approximation algorithm continued

For $i = 1 \ldots n$
  If $f_i \leq 0$, then set $x_i = \text{true}$
  Else if $t_i \leq 0$,
    then set $x_i = \text{false}$
  Else set x_i true with probability $\frac{t_i}{t_i + f_i}$.
End For

Consider an optimal solution (even an LP optimal) $\mathbf{x}^*$ and let OPT_i be the assignment in which the first i variables are as in S_i and the remiaing $n - i$ variables are set as in $\mathbf{x}^*$. (Note: x^* is not calculated.)

The analysis follows as in Poloczek and Schnitger, Poloczek, and explicitly in Buchbinder et al. One shows the following:

- $t_i + f_i \geq 0$
- $\mathbb{E}[w(OPT_{i-1}) - w(OPT_i)] \leq \mathbb{E}[w(B_i) - w(B_{i-1})]$

Derandomizing the algorithms for USM and Submodular Max-Sat: Buxchbinder and Feldman[2016]

- Contrary to the Poloczek, (resp. Huang and B.) priority inapproximations for Max-Sat (resp. USM), 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 analysis of the randomized USM approximation bound shows that at each iteration of the algorithm the following linear inequality holds:

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

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 naive 2^i that would follow from just considering the randomized tree.
- Finally, since there must be at least one distribution (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 USM 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.

And some more recent progress

- Manshadi et al give a .823 inapproximation for biparitie matching in the known distribution model. This implies the same inapproximation in the ROM model inproving the $\frac{5}{6}$ inapproximation of Goel and Mehta.
- In Lecture 9 we will mention some extensions (budgetted bidders as in the adwords problem) of the basic unweighted bipartite matching problem in adversarial, stochastic and ROM mdoels.
- A reasonably up to date survey for bipartite macthing and related problems is given by Mehta [2011]