CSC2420: Algorithm Design, Analysis and Theory Fall 2017

Allan Borodin and Nisarg Shah

October 18, 2017

Lecture 6

Announcements:

- The first assignment was due today. Any questions?
- The first two questions for assignment 2 have been posted.
- This week and next week are my last two lectures and then my colleague Nisarg Shah will do the remaining lectures.

Todays agenda and continuing on to next week

- Go over application of min cut to the metric labelling problem with two labels.
- Return to IP/LP. Makespan problem for the unrelated machines model
- Introduction to duality, primal dual algorithms, dual fitting.
- Introduction to randomized algorithms
  1. Randomization and relation to complexity theory
  2. Naive exact max-k-sat algorithm
  3. De-randomization by the method of conditional expectation
  4. Yannakakis randomized LP rounding for max-sat
  5. The KVV algorithm for online unweighted bipartite matching
  6. The Buchbinder et al two sided online greedy algorithm and application to max-sat.
  7. Online with advice and relation to randomized online algorithms
  8. De-randomization using two and multi pass algorithms
- ROM, i.i.d. online models
- The landscape for vertex and edge weighted online bipartite matching.

The $\{0,1\}$ metric labelling problem.

We consider one more application of max flow-min cut, the $\{0,1\}$ metric labelliing problem, discussed in 7.10 and 12.6 of the Kleinberg-Tardos text. This problem is a special case of the following more general metric labelling problem defined as follows:

- The input is an edge weighted graph $G = (V, E)$, a set of labels $L = \{a_1, \ldots, a_r\}$ in a metric space with distance metric d, and functions $w : E \to \mathbb{R}^{\geq 0}$ and $\beta : V \times L \to \mathbb{R}^{\geq 0}$.
- $\beta(u, a_j)$ is the benefit of giving label a_j to node u.
- Goal: Find a labelling $\lambda : V \to L$ of the nodes so as to maximize

$$\sum_u \beta(u, \lambda(u)) - \sum_{(u,v)\in E} w(u, v) \cdot d((\lambda(u), \lambda(v))$$

The $\{0,1\}$ metric labelling problem.

We consider one more application of max flow-min cut, the $\{0,1\}$ metric labelliing problem, discussed in 7.10 and 12.6 of the Kleinberg-Tardos text. This problem is a special case of the following more general metric labelling problem defined as follows:

- The input is an edge weighted graph $G = (V, E)$, a set of labels $L = \{a_1, \ldots, a_r\}$ in a metric space with distance metric d, and functions $w : E \to \mathbb{R}^{\geq 0}$ and $\beta : V \times L \to \mathbb{R}^{\geq 0}$.
- $\beta(u, a_j)$ is the benefit of giving label a_j to node u.
- Goal: Find a labelling $\lambda : V \to L$ of the nodes so as to maximize

$$\sum_u \beta(u, \lambda(u)) - \sum_{(u,v)\in E} w(u, v) \cdot d((\lambda(u), \lambda(v))$$

- For example, the nodes might represent documents, the labels are topics, and the edges are links between documents weighted by the importance of the link.
- When there are 3 or more labels, the problem is NP-hard even for the case of the $\{0,1\}$ metric d where $d(a_i, a_j) = 1$ for $a_i \neq a_j$.

The labelling problem with 2 labels

- When there are only 2 labels, the only metric is the $\{0,1\}$ metric.
- While the labelling problem is NP-hard for 3 or more labels (even for the $\{0,1\}$ metric), it is solvable in polynomial time for 2 labels by reducing the problem to the min cut problem. This is what is being done in Section 7.10 of the KT text for a special graph relating to pixels in an image.
- In section 12.6 of the KT text, there an approximation algorithm for the $\{0,1\}$ metric with 3 or more labels that uses local search while using a min cut so as to do the local search of a neighbourhood.
- Informally, the idea is that we will reframe the problem as a minimization problem. We then construct a flow network such that the nodes on the side of the source node s will correspond to (say) nodes labled a and the nodes on the side of the terminal node t will correspond to the nodes labeled b.
- We will place capacities between the source s and other nodes to reflect the cost of a "mislabel" and similarly for the terminal t.
- The min cut will then correspond to a min cost labelling.

The reduction for the two label case

For the two label case (labels {a,b}), we can let $a_u = \beta(u, a)$ and $b_u = \beta(u, b)$.
The goal is to maximize $\sum_{u \in A} a_u + \sum_{v \in B} b_v - \sum_{(u,v) \in A \times B} w(u, v)$
Leting $Q = \sum_{u \in V} a_u + b_u$, the goal is then equivalent to maximizing
$Q - \sum_{u \in A} b_u - \sum_{v \in B} a_v - \sum_{(u,v) \in A \times B} w(u, v)$
Equivalently, to minimizing
$\sum_{u \in A} b_u + \sum_{v \in B} a_v + \sum_{(u,v) \in A \times B} w(u, v)$

- We transform this problem to a min cut problem as follows: construct the flow network $\mathcal{F} = (G', s, t, c)$ such that $G' = (V', E')$
- $V' = V \cup \{s, t\}$
- $E' = \{(u, v) \in E\} \cup \{(s, u) | u \in V\} \cup \{(u, t) | u \in V\}$
- $c(u, v) = c(v, u) = w(u, v); c(s, u) = a_u; c(u, t) = b_u$

Claim:

For any partition $V = A \cup B$, the capacity of the cut $c(A, B) = \sum_{u \in A} b_u + \sum_{v \in B} a_v + \sum_{(u,v) \in A \times B} w(u, v)$.

Makespan for the unrelated and restricted machine models: a more sophisticated rounding

In the vertex cover example I used the terms "(input) independent rounding" and "oblivious" rounding.)

- We now return to the makespan problem with respect to the unrelated machines model and the special case of the restricted machine model.
- Recall the unrelated machines model where a job j is represented by a tuple $(p_{j,1}, \ldots, p_{j,m})$ where $p_{j,i}$ is the time that job j uses if scheduled on machine i.
- An important scheduling result is the Lenstra, Shmoys, Tardos (LST) [1990] IP/LP 2-approximation algorithm for the makespan problem in the unrelated machine model (when m is part of the input). They also obtain a PTAS for fixed m.

The natural IP and the LP relaxation

The IP/LP for unrelated machines makespan

- Minimize T
- Subject to
  1. $\sum_i x_{j,i} = 1$ for every job j % schedule every job
  2. $\sum_j x_{j,i} p_{j,i} \leq T$ for every machine i % do not exceed makespan
  3. $x_{j,i} \in \{0, 1\}$ % $x_{j,i} = 1$ iff job j scheduled on machine i

- The immdiate LP relaxation is to just have $x_{j,i} \geq 0$
- Even for identical machines (where $p_{j,i} = p_j$ for all i), the integrality gap IG is unbounded since the input could be just one large job with say size T leading to an LP-OPT of T/m and IP-OPT $=$ OPT $= T$ so that the IG $= m$.

Adapting the natural IP

- As in the PTAS for the identical machine makespan PTAS, we use binary search to find an appropriate approximation T for the optimal makespan.
- Given a candidate T, we remove all x_{ji} such that $p_{j,i} > T$ and obtain a "search problem" (i.e. constant or no objective function) for finding $x_{j,i}$ satisfying the IP constraints.
- Once we have found the optimal T for the search problem, the LST algorithm then shows how to use a non-independent rounding to obtain an integral solution yielding a 2-approximation.
- Note: We use the term "rounding" in a very general sense to mean any efficient way to convert the LP solution into an intergral solution.

Sketch of LST rounding for makespan problem

- Using slack form, LP theory can be used to show that if $\mathcal{L}$ is a feasible bounded LP with $m + n$ constraints (not counting the non-negativity constraints for the variables) then $\mathcal{L}$ has an optimal basic solution such that at most $n + m$ of the variables are non-zero.
- It follows (how?) that at most m of the n jobs have fractional non-integral solutions (i.e. are not assigned to a single machine).
- Jobs assigned to a single machine do not need to be rounded; i.e. if $x_{j,i} = 1$ then schedule job j on machine i.
- Construct a bipartite graph between the $y \leq m$ fractionally assigned jobs and the m machines.

The rounding continued

- The goal is then to construct a matching of size y; that, is, the matching dictates how to schedule these fractionally assigned jobs. So it "only" remains to show that this bipartite graph has a maximum matching of size y. Note, of course, this is what makes the "rounding" non-independent .
- The existence of this matching requires more LP theory whereby it can be shown (LST credit Dantzig [1963]) that the connected components of the bipartite graph are either trees or trees with one added edge (and therefore causing a unique cycle).
- Given the structure of the biprtite graph, one can construct a matching of all the jobs. How?
- The resulting schedule then has makespan at most $2T$ since each fractional job has $p_{j,i} \leq T$ and the LP has guaranteed a makespan at most T before assigning the fractional jobs.

The restricted machine makespan problem

- The restricted machines model is a special case of the unrelated machines problem where for every job j, $p_{j,i} \in \{p_j, \infty\}$. Hence the LST 2-approximation applies.
- LST show that it is NP hard to do better than a 1.5 approximation for the restricted machines (and hence unrelated machines) problem.
- There is a somewhat strange result due to Svensson [2011]. He shows how to approximate the *value* of the optimum makespan to within a factor of $33/17 \approx 1.9413 < 2$. This is proven constructively by a local search algorithm satisfying the approximation. However, the local search is not shown to terminate in polynomial time.
- Note that if we could determine the optimal makespan value in polynomial time, then we can also find an optimal solution in polynomial time. However, the same cannot be said when we are only approximating the makespan value.

The special case of graph orientation

- Consider the special case when there are (at most) two allowable machines for each job. This is called the graph orientation problem.
- It turns out easier to reason about the LP rounding applied to the graph orientation problem for the given IP/LP but still the integrality gap is 2.
- A more refined IP/LP by Eveblendr, Krcal and Sgall [2008] achieves a 1.75 approximation for the graph orientation problem.
- Even for the case when each job can only be scheduled on at most 3 machines, beating the 2-approximation remains an open problem.

Some concluding remarks (for now) about LP rounding

- We will return later to more LP applications. There are some nice notes by Allan Jepson providing some of the geometric concepts underlying LP solutions. His CSC373 course can be accessed here: http://www.cs.toronto.edu/ jepson/csc373/
- There can be, of course, many different IP/LP formulations for a given problem. In particular, one often adds additional constraints so that the polytope of the LP solutions is smaller.
- For example, in the vertex cover LP, one could simply add constraints $x_i + x_j + x_k \geq 2$ for every triangle in the graph and more generally, constraints for every odd length cycle. (These inequalities do not essentially change the integrality gap.)
- Adding such constraints corresponds to one round of what is called the LS lift and project method.
- There are a number of lift and project methods. If you are interested, then consult our local expert Toni Pitassi.

Duality: See Vazirani and Shmoys/Williamson texts, and Williamson article

- For a primal maximization (resp. minimization) LP in standard form, the dual LP is a minimization (resp. maximization) LP in standard form.
- Specifically, if the primal $\mathcal{P}$ is:
  - Minimize $\mathbf{c} \cdot \mathbf{x}$
  - subject to $A_{m \times n} \cdot \mathbf{x} \geq \mathbf{b}$
  - $\mathbf{x} \geq 0$
- then the dual LP $\mathcal{D}$ with dual variables $\mathbf{y}$ is:
  - Maximize $\mathbf{b} \cdot \mathbf{y}$
  - subject to $A^{tr}_{n \times m} \cdot \mathbf{y} \leq \mathbf{c}$
  - $\mathbf{y} \geq 0$
- Note that the dual (resp. primal) variables are in correspondence to primal (resp. dual) constraints.
- If we consider the dual $\mathcal{D}$ as the primal then its dual is the original primal $\mathcal{P}$. That is, the dual of the dual is the primal.

An example: set cover

As already noted, the vertex cover problem is a special case of the set cover problem in which the elements are the edges and the vertices are the sets, each set (ie vertex v) consisting of the edges adjacent to v.

The set cover problem as an IP/LP

minimize $\sum_j w_j x_j$
subject to $\sum_{j:e_i \in S_j} x_j \geq 1$ for all i; that is, $e_i \in U$
$x_j \in \{0,1\}$ (resp. $x_j \geq 0$)

The dual LP

maximize $\sum_i y_i$
subject to $\sum_{i:e_i \in S_j} y_i \leq w_j$ for all j
$y_i \geq 0$

If all the parameters in a standard form minimization (resp. maximization) problem are non negative, then the problem is called a covering (resp. packing) problem. Note that the set cover problem is a covering problem and its dual is a packing problem.

Duality Theory Overview

- An essential aspect of duality is that a finite optimal value to either the primal or the dual determines an optimal value to both.
- The relation between these two can sometimes be easy to interpret. However, the interpretation of the dual may not always be intuitively meaningful.
- Still, duality is very useful because the duality principle states that optimization problems may be viewed from either of two perspectives and this might be useful as the solution of the dual might be much easier to calculate than the solution of the primal.
- In some cases, the dual might provide additional insight as to how to round the LP solution to an integral solution.
- Moreover, the relation between the primal $\mathcal{P}$ and the dual $\mathcal{D}$ will lead to primal-Dual algorithms and to the so-called dual fiiting analysis.
- In what follows we will assume the primal is a minimization problem to simplify the exposition.

Strong and Weak Duality

Strong Duality

If x^* and y^* are (finite) optimal primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}^*) = \mathcal{P}(\mathbf{x}^*)$.

Note: Before it was known that solving LPs was in polynomial time, it was observed that strong duality proves that LP (as a decision problem) is in $\mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$ which strongly suggested that LP was not NP-complete.

Weak Duality for a Minimization Problem

If $\mathbf{x}$ and $\mathbf{y}$ are primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}) \leq \mathcal{P}(\mathbf{x})$.

- Duality can be motivated by asking how one can verify that the minimum in the primal is at least some value z. To get witnesses, one can explore non-negative scaling factors (i.e. the dual variables) that can be used as multipliers in the constraints. The multipliers, however, must not violate the objective (i.e cause any multiplies of a primal variable to exceed the coefficient in the objective) we are trying to bound.

Motivating duality

Consider the motivating example in V. Vazirani's text:

Primal

minimize $7x_1 + x_2 + 5x_3$

subject to

- (1) $x_1 - x_2 + 3x_3 \geq 10$
- (2) $5x_1 + 2x_2 - x_3 \geq 6$
- $x_1, x_2, x_3 \geq 0$

Dual

maximize $10y_1 + 6y_2$

subject to

$$y_1 + 5y_2 \leq 7$$
$$-2y_1 + 2y_2 \leq 1$$
$$3y_1 - y_2 \leq 5$$
$$y_1, y_2 \geq 0$$

Adding (1) and (2) and comparing the coefficient for each x_i, we have:

$7x_1 + x_2 + 5x_3 \geq (x_1 - x_2 + 3x_3) + (5x_1 + 2x_2 - x_3) \geq 10 + 6 = 16$

Better yet,

$7x_1 + x_2 + 5x_3 \geq 2(x_1 - x_2 + 3x_3) + (5x_1 + 2x_2 - x_3) \geq 26$

For an upper bound, setting $(x_1, x_2, x_3) = (7/4, 0, 11/4)$

$7x_1 + x_2 + 5x_3 = 7 \cdot (7/4) + 1 \cdot 0 + 5 \cdot (11/4) = 26$

This proves that the optimal value for the primal and dual (with solution $(y_1, y_2) = (2, 1)$ must be 26.

Easy to prove weak duality

The proof for weak duality

$$\begin{aligned}
&\mathbf{b} \cdot \mathbf{y} = \sum_{j=1}^{m} b_j y_j \\
&\leq \sum_{j=1}^{m} (\sum_{i=1}^{n} A_{ji} x_i) y_j \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ji} y_j) x_i \\
&\leq \sum_{i=1}^{n} c_i x_i = \mathbf{c} \cdot \mathbf{x}
\end{aligned}$$

Solving the f-frequency set cover by a primal dual algorithm

- In the f-frequency set cover problem, each element is contained in at most f sets.
- Clearly, the vertex cover problem is an instance of the 2-frequency set cover.
- As in the vertex cover LP rounding, we can similarly solve the f-frequency cover problem by obtaining an optimal solution $\{x_j^*\}$ to the (primal) LP and then rounding to obtain $\bar{x}_j = 1$ iff $x_j^* \geq \frac{1}{f}$. This is, as noted before, a conceptually simple method but requires solving the LP.
- We know that for a minimization problem, any dual solution is a lower bound on any primal solution. One possible goal in a primal dual method for a minimization problem will be to maintain a fractional feasible dual solution and continue to try improve the dual solution. As dual constraints become tight we then set the corresponding primal variables.

Primal dual for f-frequency set cover continued

Suggestive lemma for following primal dual algorithm

Claim: Let $\{y_i^*\}$ be an optimal solution to the dual LP and let $\mathcal{C}' = \{S_j | \sum_{e_i \in S_j} y_i^* = w_j\}$. Then $\mathcal{C}'$ is a cover.

Primal dual algorithm for set cover

Set $y_i = 0$ for all i

$\mathcal{C}' := \varnothing$

While there exists an e_i not covered by $\mathcal{C}'$

Increase the dual variables y_i until there is some j : $\sum_{\{k:e_i \in S_j\}} y_i = w_j$

$\mathcal{C}' := \mathcal{C}' \cup \{S_j\}$

Freeze the y_i associated with the newly covered e_i

End While

Theorem: Approximation bound for primal dual algorithm

The cover formed by tight constraints in the dual solution provides an f approximation for the f-frequency set cover problem.

Comments on the primal dual algorithm

- What is being shown is that the integral primal solution is within a factor of f of the dual solution which implies that the primal dual algorithm is an f-approximation algorithm for the f-frequency set cover problem.
- Additionally, what is being shown is that the integraility gap of this IP/LP formulation for f-frequency set cover problem is at most f.
- In terms of implementation we would calculate the minimum ϵ needed to make some constraint tight so as to chose which primal variable to set. This ϵ could be 0 if a previous iteration had more than one constraint that becomes tight simultaneously. This ϵ would then be subtracted from w_j for j such that $e_i \in S_j$.

More comments on primal dual algorithms

- We have just seen an example of a basic form of the primal dual method for a minimization problem. Namely, we start with an infeasible integral primal solution and feasible (fractional) dual. (For a covering primal problem and dual packing problem, the initial dual solution can be the all zero solution.) Unsatisfied primal constraints suggest which dual constraints might be tightened and when one or more dual constraints become tight this determines which primal variable(s) to set.
- Some primal dual algorithms extend this basic form by using a second (reverse delete) stage to achieve minimality. Bar-Yehuda and Rawitz argue that this is equivalent to *local ratio* algorithms. Our priority stack algorithms are a basic example of such algorithms.
- **NOTE** In the primal dual method we are not solving any LPs. Primal dual algorithms are viewed as "combinatorial algorithms" and in some cases they might even suggest an explicit greedy algorithm.

Using dual fitting to prove the approximation ratio of the greedy set cover algorithm

Early in the term, we mentioned the following natural greedy algorithm for the weighted set cover problem:

The greedy set cover algorithm

$\mathcal{C}' := \varnothing$

While there are uncovered elements

Choose S_j such that $\frac{w_j}{|\tilde{S}_j|}$ is a minimum where

$\tilde{S}_j$ is the subset of S_j containing the currently uncovered elements

$\mathcal{C}' := \mathcal{C}' \cup S_j$

We wish to prove the following theorem (Lovasz[1975], Chvatal [1979]):

Approximation ratio for greedy set cover

The approximation algorithm for the greedy algorithm is H_d where d is the maximum size of any set S_j.

The dual fitting analysis

The greedy set cover algorithm setting prices for each element

$\mathcal{C}' := \varnothing$

While there are uncovered elements

Choose S_j such that $\frac{w_j}{|\tilde{S}_j|}$ is a minimum where

$\tilde{S}_j$ is the subset of S_j containing the currently uncovered elements

%Charge each element e in $\tilde{S}_j$ the average cost $price(e) = \frac{w_j}{|\tilde{S}_j|}$

% This charging is just for the purpose of analysis

$\mathcal{C}' := \mathcal{C}' \cup S_j$

End While

- We can account for the cost of the solution by the costs imposed on the elements; namely, $\{price(e)\}$. That is, the cost of the greedy solution is $\sum_e price(e)$.

Dual fitting analysis continued

- The goal of the dual fitting analysis is to show that $y_e = price(e)/H_d$ is a feasible dual and hence any primal solution must have cost at least $\sum_e price(e)/H_d$.
- Consider any set $S = S_j$ in $\mathcal{C}$ having say $k \leq d$ elements. Let $e_1, \ldots, e_k$ be the elements of S in the order covered by the greedy algorithm (breaking ties arbitrarily). Consider the iteration is which e_i is first covered. At this iteration $\tilde{S}$ must have at least $k - i + 1$ uncovered elements and hence S could cover cover e_i at the average cost of $\frac{w_j}{k-i+1}$. Since the greedy algorithm chooses the most cost efficient set, $price(e_i) \leq \frac{w_j}{k-i+1}$.
- Summing over all elements in S_j, we have $\sum_{e_i \in S_j} y_{e_i} = \sum_{e_i \in S_j} price(e_i)/H_d \leq \sum_{e_i \in S_j} \frac{w_j}{k-i+1} \frac{1}{H_d} = w_j \frac{H_k}{H_d} \leq w_j$. Hence $\{y_e\}$ is a feasible dual.

Randomized algorithms

Our next theme will be randomized algorithms. For the main part, our previous themes have been on algorithmic paradigms. Randomization is not per se an algorithmic paradigm (in the same sense as greedy algorithms, DP, local search, LP rounding, primal dual algorithms).

Randomized algorithms

Our next theme will be randomized algorithms. For the main part, our previous themes have been on algorithmic paradigms. Randomization is not per se an algorithmic paradigm (in the same sense as greedy algorithms, DP, local search, LP rounding, primal dual algorithms).

Rather, randomization can be thought of as a tool that can be used in conjuction with any algorithmic paradigm. However, its use is so prominent and varied in algorithm design and analysis, that it takes on the sense of an algorithmic way of thinking.

The why of randomized algorithms

- There are some problem settings (e.g. simulation, cryptography, interactive proofs, sublinear time algorithms) where randomization is necessary.
- We can use randomization to improve approximation ratios.
- Even when a given algorithm can be efficiently derandomized, there is often conceptual insight to be gained from the initial randomized algorithm.
- In complexity theory a fundamental question is how much can randomization lower the time complexity of a problem. For decision problems, there are three polynomial time randomized classes ZPP (zero-sided), RP (1-sided) and BPP (2-sided) error. The big question (and conjecture?) is BPP $=$ P?
- One important aspect of randomized algorithms is that the probability of success can be amplified by repreated independent trials of the algorithm.

Some problems in randomized polynomial time not known to be in polynomial time

1. The symbolic determinant problem.
2. Given n, find a prime in $[2^n, 2^{n+1}]$
3. Estimating volume of a convex body given by a set of linear inequalitiies.
4. Solving a quadratic equation in $Z_p[x]$ for a large prime p.

Polynomial identity testing

- The general problem concerning polynomial identities is that we are implicitly given two multivariate polynomials and wish to determine if they are identical. One way we could be implicitly given these polynomials is by an arithmetic circuit. A specific case of interest is the following symbolic determinant problem.
- Consider an $n \times n$ matrix $A = (a_{i,j})$ whose entries are polynomials of total degree (at most) d in m variables, say with integer coeficients. The determinant $det(A) = \sum_{\pi \in S_n} (-1)^{sgn(\pi)} \prod_{i=1}^{n} a_{i,\pi(i)}$, is a polynomial of degree nd. The symbolic determinant problem is to determine whether $det(A) \equiv \mathbf{0}$, the zero polynomial.

Schwartz Zipple Lemma

Let $P \in \mathbf{F}[x_1, \ldots, x_m]$ be a non zero polynomial over a field $\mathbf{F}$ of total degree at most d. Let S be a finite subset of $\mathbf{F}$. Then $Prob_{r_i \in_u S}[P(r_1, \ldots . r_m) = 0] \leq \frac{d}{|S|}$

Schwartz Zipple is clearly a multivariate generalization of the fact that a univariate polynomial of degree d can have at most d zeros.

Polynomial identity testing and symbolic determinant continued

- Returning to the symbolic determinant problem, suppose then we choose a suffciently large set of integers S (for definiteness say $|S| \geq 2nd$). Randomly choosing $r_i \in S$, we evaluate each of the polynomial entries at the values $x_i = r_i$. We then have a matrix A' with (not so large) integer entries.
- We know how to compute the determinant of any such integer matrix $A'_{n\times n}$ in $O(n^3)$ arithmetic operations. (Using the currently fastest, but not necessarily practical, matrix multiplication algorithm the determinant can be computed in $O(n^{2.38})$ arithmetic operations.)
- That is, we are computing the $det(A)$ at random $r_i \in S$ which is a degree nd polynomial. Since $|S| \geq 2nd$, then $Prob[det(A') = 0] \leq \frac{1}{2}$ assuming $det(A) \not\equiv \mathbf{0}$. The probability of correctness con be amplifed by choosing a bigger S or by repeated trials.
- In complexity theory terms, the problem (is $det(A) \equiv \mathbf{0}$) is in co-RP.

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.
Warning: For the following discussion of Max-Sat, we will follow the prevailing convention by stating approximation ratios as fractions $c < 1$.

- Consider 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.
- Since exact Max-k-Sat generalizes the exact k- SAT decision problem, it is clearly an NP hard problem for $k \geq 3$. It is interesting to note that 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 variables, the probability that a random assignment of the literals 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.
- This is not only an approxination ratio but moreover a "totality ratio" in that 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 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 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 clauses. 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 unit clauses)?