15-750 Graduate Algorithms
Notes for 04/28/04 [Avrim Blum]

```
* Approximation algorithms for MAX-SAT
  - with the probabilistic method
  - and with randomized rounding
* randomized rou{t,nd}ing   [probably won't get to this]
```

---

1 MAX-SAT

A CNF formula is an AND of clauses. A k-CNF is a CNF where each clause has size at most k. Let's define an "exactly-k CNF" to be a CNF where each clause is of size exactly k (a given variable is not allowed to appear several times in the same clause to inflate its size).

MAX-SAT Given a CNF formula, find a solution that satisfies as many clauses as possible.

MAX-kSAT Given a k-CNF formula, find a solution that satisfies as many clauses as possible.

MAX-EkSAT Given an exactly-k CNF formula, find a solution that satisfies as many clauses as possible.

These are all NP-hard ($k \geq 2$). Natural goal: try to approximate the optimal solution.

Claim: For any "exactly-3 CNF", there exists a solution that satisfies at least 7/8 of the clauses.
Proof: consider a random assignment. Expected number of clauses satisfied is $(7/8)m$. (m =number of clauses). Therefore, an assignment with this property exists. ■

Question: Can you give me an efficient algorithm to *find* an assignment that satisfies at least $7m/8$ clauses?
Simple randomized algorithm: just keep trying random assignments. Need to be a little careful since sometimes you can have random variables that are almost always below their expectation (e.g., if they have a large range or if they are almost always just barely below their expectation). But, since max value is m, and the closest smaller integer to $7m/8$ is at least 1/8 away, there has to be at least a $1/(m+1)$ chance of being $\geq$ the expectation.

So, this algorithm trivially gives us a 7/8 approximation for MAX-E3SAT, since we surely satisfy at least 7/8 of the maximum possible. Interesting fact: approximating MAX-E3SAT to $7/8 + \epsilon$ is NP-hard [Hastad].

Question: How about a deterministic algorithm? For this, we can use something called the "conditional expectation method". Let's first generalize the above analysis to general CNF. If there are m_k clauses of size k, a random assignment will satisfy an expected $\sum_k m_k(1-1/2^k)$ of them. Now, we can do the following. Look at x_1: for each of the two possible settings (0 or 1) we can calculate the expected number of clauses satisfied if we were to go with that setting, and then set the rest of the variables randomly. It is just (number of clauses satisfied by setting $x_1 = 0$) + $\sum_k m'_k(1-1/2^k)$, or (number of clauses satisfied by setting $x_1 = 1$) + $\sum_k m''_k(1-1/2^k)$, where m'_k and m''_k are the number of clauses of size k in the "formula to go" in each case. Fix x_1 to the setting that gives us a larger expectation. Now go on to x_2 and do the same thing, setting it to the value with the highest expectation-to-go, and then x_3 and so on. The point is that the expected number of clauses satisfied by a random assignment is the *average* of the expected number given $x_i = 0$ and the expected number given $x_i = 1$, so our "expectation to go" never decreases.

Note that this algorithm is completely deterministic — in fact we could rewrite it to get rid of any hint of randomization by just viewing $\sum_k m_k(1-1/2^k)$ as a way of weighting the clauses to favor the small ones, but our motivation was based on thinking probabilistically.

What about MAX-SAT in general? Actually, for the bigger clauses, we do even better. The problem is the *smaller* clauses. For MAX-E2SAT, we satisfy at least 3/4 of the maximum possible. For the MAX-SAT problem in general, with this algorithm the best we can say is we satisfy at least $m/2$ clauses because of clauses of size 1. We'll now show a different approach, called "randomized rounding" we can combine with the previous algorithm to get at least 3/4 of the maximum possible even in the general case.

(Current best for MAX-SAT: 0.785 [Asano-Williamson], and possibly 0.8331 based on conjecture of Zwick. 0.931 for MAX 2SAT [Feige-Goemans]; 0.801 for MAX 3-SAT [Trevisan, Sorkin, Sudan, Williamson], and if formula is satisfiable then can get 7/8 [Karloff, Zwick].)

Here is the randomized rounding algorithm:

- Solve a fractional version of the problem. Instead of requiring variable $x_i \in \{0, 1\}$, we allow $x_i \in [0, 1]$. Define $\neg x_i$ to be $1 - x_i$. Allow clauses to be "partially satsified:" for clause $(x_1 \vee x_2 \vee \neg x_3)$, let "satisfiedness" be: $\min(1, x_1 + x_2 + (1 - x_3))$. I.e., if the sum is less than 1, then it represents how satisfied the clause is, and if greater than 1, then we say the clause is satisfied. Now we find the solution that maximizes total satisfaction using Linear Programming.

  To set up as an LP: $x_i \in [0, 1]$. Variable z_j for clause j: $z_j \leq 1$ and $z_j \leq$ sum-of-literals-in-clause-j. Then, maximize $\sum z_j$.

- Now, let's do randomized rounding: set variable i to 1 with probability x_i.

Question: if clause j has k_j literals, and the fractional assignments sum up to z_j, what can we say about the probability our procedure will satisfy the clause?

Equivalently, flipping k_j coins whose biases sum to z_j. What is the least possible probability we get at least one heads? Turns out, the worst case is when they are all equal to z_j/k_j. For intuition, think of $k_j = 2$ and $z_j = 1$. Mathematically, it's easiest to think in terms of *maximizing* the probability we get all tails: given k_j numbers that all sum up to a fixed value (in this case $k_j - z_j$), the maximum *product* occurs when they are all equal. So **Pr**(clause j is satisfied) $\geq 1 - (1 - z_j/k_j)^{k_j}$. This looks like a complete mess, but lets plot things as a function of k. Prob(clause j is satisfied) as a function of k is at least:

	strategy 1	strategy 2
$k = 1$	1/2	z_j
$k = 2$	3/4	$1 - (1 - z_j/2)^2 \geq (3/4)z_j$
$k \geq 3$	7/8 or more	$1 - (1 - z_j/k)^k \geq 1 - (1/e)^{z_j} \geq (5/8)z_j$

[You can check that the very last inequality holds for $z_j \in [0,1]$ by looking at the endpoints ($1 - 1/e > 5/8$) and checking that the 2nd derivative is negative.]

Notice that strategy 1 does well when there are lots of big clauses. Strategy 2 does well when there are lots of little clauses. So, without even looking carefully at the CNF formula, we will just try both and output whichever answer is best. Here's the cute part of the analysis: our global strategy clearly does at least as well as one that says: flip a coin and with probability 1/2 use strategy 1, and with probability 1/2 use strategy 2. For this algorithm, the chance that clause j is satisfied is just the average of the two values from the above table, and this average is always bigger than $(3/4)z_j$. So our expected number of clauses satisfied is at least $(3/4)\sum_j z_j$, which is at least 3/4 of the optimal fractional solution, which is at least 3/4 of the optimal integer solution.

1.1 Randomized routing

Consider the following problem. Given an undirected graph and a set of pairs $\{(s_i, t_i)\}$ we want to route these pairs to minimize the maximum congestion. E.g., think of routing paths on the internet. This problem is NP-hard. Can we find an approximate solution?

Idea: (Raghavan & Thompson)

1. Solve fractionally. Think of as multi-commodity flow (e.g., allow s_i to route to t_i by sending 1/2 down one path, 1/4 down another path, and 1/4 down another). Can solve with linear programming: for each (directed) edge e, and each commodity i, have variable X_{ei}. Constraints for inflow = outflow. Constraints $\forall e, \sum_i X_{ie} \leq C$, and minimize C.

2. For each pair (s_i, t_i) we have a flow. Now what we do is view these fractional values as probabilities and select a path such that the probability we pick edge e is equal to the flow of this commodity on e. How can we do this algorithmically? (Give proof that greedy approach works.)

Analysis: fix some edge. Let f_i be the flow of commodity i on this edge. This also means that f_i is the probability that we picked this edge for routing (s_i, t_i). So, for a given edge, can think of $\{0,1\}$ random variables X_i corresponding to event that we picked this edge for commodity i, where $\mathbf{Pr}[X_i = 1] = f_i$. For a given edge, these X_i are all INDEPENDENT. (Not independent for the same i across different edges, but that's OK). Expected value of sum is at most C. Now apply Chernoff bounds (see below).

$$\mathbf{Pr}[total > (1+\epsilon)C] < e^{-\epsilon^2 C/3}$$

The point now is if this is small enough (e.g., $o(1/n^2)$) then the probability that *there exists* an edge whose congestion exceeds this bound is also small ($o(1)$).

So, if $C \gg \log(n)$, then w.h.p., maximum is only $1+\epsilon$ times larger than the expectation, and we have a $(1+\epsilon)$ approximation. For small C we can apply the bound:

$$Pr[total > kC] < (e^{k-1}/k^k)^C$$

This is small enough for $k = O(\log(n)/\log\log(n))$. Open problem: can you get a constant-factor approximation?

Chernoff bounds

Let $X_1, \ldots, X_n$ be a sequence of n independent $\{0,1\}$ random variables with $\mathbf{Pr}[X_i = 1] = p_i$ not necessarily the same. Let S be the sum of the r.v., and $\mu = \mathbf{E}[S]$. Then, for $0 \le \delta \le 1$, the following inequalities hold:

- $\mathbf{Pr}[S > (1+\delta)\mu] \le e^{-\delta^2\mu/3}$,
- $\mathbf{Pr}[S < (1-\delta)\mu] \le e^{-\delta^2\mu/2}$.

Additive bounds:

- $\mathbf{Pr}[S - \mu > \delta n] \le e^{-2n\delta^2}$.
- $\mathbf{Pr}[S - \mu < -\delta n] \le e^{-2n\delta^2}$.

Also, for any $k > 1$, we get:

- $\mathbf{Pr}[S > k\mu] < \left(\frac{e^{k-1}}{k^k}\right)^{\mu}$.