15-854 Approximation and Online Algorithms
Notes for 01/26/00 and some of 01/31/00

```
* MAX SAT (kSAT, Ek-SAT)
* Routing to minimize congestion
* Techniques: conditional expectation and randomized rounding
```

Recap

Last time we looked at the set-cover and p-center (firehouse location) problems. In the center problem, we have a directed graph and want to pick a small number of center vertices (firehouses) so that no vertex in the graph is too far from its nearest center. If we fix the radius and try to minimize the number of centers, we get the set-cover problem, which we can approximate to ratio of $O(\log n)$. But, if we fix the number and try to minimize the radius, we were able to get $O(\log^* n)$ [PV]. We did this by repeated set-covers, and we needed the more detailed bound of $1 + \ln(n/p)$ for set cover, since we're using the fact that as n gets closer to p then our approximation gets better and better. [In class we only went through the weaker result that gets $O(\log^* n)$ using $2p$ centers. Rough idea of how to get down to p: if there exists a node that covers everything that covers it, then that is a good case – like the undirected problem. If no such nodes exist, then there must exist a "pretty good" set of $p/2$ centers that covers everything in radius $5R$. See end of notes from last time.]

MAX-SAT

Today we'll look at a classic problem called the MAX-SAT problem.
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.

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. ∎

How about a deterministic algorithm to find a solution that satisfies at least $(7/8)m$? Let's first generalize the above 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, here's how we can use this to get a deterministic algorithm. 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 the number of clauses already satisfied plus $\sum_k m_k(1-1/2^k)$, where m_k is the number of clauses of size k in the "formula to go".) 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.
This is called the "conditional expectation" method. The algorithm itself 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 the randomized method.

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. 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. But, there is a different idea that we can use to get at least 3/4 of the maximum possible even in the general case. It's surprising there's no really obvious way to do it — to get 3/4 we're going to use randomized rounding.

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. Then we find the solution that maximizes total satisfaction.

  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 literals and $z_j = 1$, what is the worst case in terms of the probability that clause j is satisfied?
Equivalently, flipping k coins whose biases sum to 1, what can we say about the probability of at least one heads? Best case is one of them has bias equal to 1. Worst case is they are all equal to $1/k$, and so the probability is $1-(1-1/k)^k$. Why is this the worst case? Easiest

to think in terms of maximizing the probability of all tails: given k numbers that all sum to a fixed value, the maximum possible product is when they are all the same.

What if $z_j \neq 1$? Then the worst case is they are all equal to z_j/k, so $\mathbf{Pr}$(clause j is satisfied) $\geq 1 - (1 - z_j/k)^k$. 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.]

Now, here's the clincher. 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.

Notes: Approximating MAX-E3SAT to $7/8 + \epsilon$ is NP-hard [Hastad]. Current best approxs: 0.931 for MAX 2SAT [Feige-Goemans], 0.801 for MAX 3-SAT [Trevisan, Sorkin, Sudan, Williamson], (if formula is satisfiable then can get 7/8 [Karloff, Zwick]), 0.770 for MAX-SAT [Asano] building on [Goemans-Williamson].

Open problem: can you get 3/4 for MAX-SAT without solving an LP?

Randomized routing

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 in a VLSI chip. 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 $0 \leq \delta \leq 1$, the following inequalities hold:

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

Additive bounds:

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

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

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