15-750 Graduate Algorithms
Notes for 04/03/01

```
* The probabilistic method
  - Spencer's graduation game, method of conditional expectations
* Approximation algorithm for MAX-SAT
  (using above idea + randomized rounding -- GW1)
```

---

1 The probabilistic method

Spencer's "graduation/tenure game":

- Begin with some set of people at various positions (number of years to go to graduate).
- 2 players: Paul (partitioner) splits people into two groups. Carole (chooser) selects one group to be removed, other advances.
- Start: people at various years until graduation. Paul wins if somebody graduates, Carole wins if nobody graduates.

Question: given a starting configuration, who wins? Note: this is a full information finite game. So, for a given starting configuration, somebody has a winning strategy. Let's say that at the start, there are a_k people at distance k from graduation (call this "level k").

Theorem 1: If $\sum_k a_k/2^k < 1$, then Carole has a winning strategy.
Theorem 2: If $\sum_k a_k/2^k \geq 1$, then Paul has a winning strategy.

Proof of Theorem 1. Suppose Carole plays randomly. What is the expected number of people that reach the goal? For instance, for fixed person who is initially at level k, what is the probability that this person reaches the goal? Ans: $1/2^k$. Therefore, no matter what Paul's strategy is, a random Carole has a non-zero chance of winning. Therefore, Paul cannot have a winning strategy and so Carole does. ■

> **"The Probabilistic Method"**: We want to show that something exists. Do this by setting up some probability distribution and showing that what we want happens with nonzero probability.

Actually, we can figure out Carole's winning strategy. Ideas? Just use Theorem 1: pick subset where sum is less than 1/2, so when they advance, the sum is less than 1, and we still have a winning strategy.

"Method of conditional expectations": General idea: say we've shown that making random choices results in a good expected value. Now we consider our first choice, and calculate conditional expectations given this first choice, and we pick one where conditional expectation is still good (which we know must occur, for appropriate definitions of "good").

Proof of Theorem 2: Suppose Paul can split so that both groups have sum $\geq 1/2$. Then the preconditions of the theorem are maintained and Paul wins. How can Paul do this? (Go through the elements from smallest k to largest k.) This is like the simple (dyadic) case for Huffman codes. $\blacksquare$

[Note: similar game: 20 questions with k lies allowed. questioner splits the set, and the chooser gives answer. Those that do not agree with answer move up one. position of element corresponds to the number of lies if element is the correct one. questioner's goal is to reduce the number of elements below the k-boundary to 1 as quickly as possible.]

2 MAX-SAT

We now turn to the classic 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. $\blacksquare$

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, we'll use the conditional expectation method to get a deterministic algorithm as follows. 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.

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.

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.]

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?