Colorado CSCI 5454: Algorithms October 1 and 3, 2019

Lecture 11-12

Lecturer: Bo Waggoner *Scribe: Bo Waggoner*

Randomized Algorithms

1 Probability review

Probability space. Sample space: a finite (for now) set Ω of possible *outcomes* (things that might happen). Distribution p assigning a number $p(\omega) \in [0,1]$ for each $\omega \in \Omega$ with $\sum_{\omega \in \Omega} p(\omega) = 1$.

Example: roll two die, possible outcomes are $\Omega = \{(1,1),(1,2),\cdots,(6,6)\}$. Distribution (if both fair independent die) is $p(\omega) = \frac{1}{36}$ for all ω.

An *event* is a subset of outcomes, for example, the event that the sum of the die is 11 or higher is $A = \{(5,6),(6,5),(6,6)\}$.

Because we have a finite discrete sample space, $\Pr[A] = \sum_{\omega \in A} \Pr[\omega]$. In this example $\Pr[A] = \frac{3}{36}$.

Joint probability. Event that A and B both occur: $A \cap B$.

Example: event that first roll is a 6 is $B = \{(6,1),(6,2),\ldots,(6,6)\}$. Event that sum is 11 or higher and first roll is a 6: $A \cap B = \{(6,5),(6,6)\}$.

Probability of both A and B: $\Pr[A \cap B] = \frac{2}{36}$ (this is also written $\Pr[A \text{ and } B]$).

Probability that either A occurs, or B occurs, or both: $A \cup B$. Event that sum is 11 or higher *or* first roll is a 6: $A \cup B = \{(6,1),(6,2),(6,3),(6,4),(6,5),(6,6),(5,6)\}$. Probability of this event: $\Pr[A \cup B] = \Pr[A \text{ or } B] = \frac{7}{36}$.

Conditional probability. Probability of event B given that A occurs: $\Pr[B \mid A] := \frac{\Pr[A \cap B]}{\Pr[A]}$.

Example: probability first roll is six given sum is at least 11 is

$$\begin{aligned}\Pr[B \mid A] &= \frac{\Pr[A \cap B]}{\Pr[A]} \\ &= \frac{2/36}{3/36} \\ &= \frac{2}{3}.\end{aligned}$$

Independence. Events A and B are **independent** if $\Pr[A] = \Pr[A \mid B]$. Verbally, conditioning on whether B has happened does not impact the probability of A. Check this is equivalent to the conditions $\Pr[A \cap B] = \Pr[A]\Pr[B]$ and $\Pr[B] = \Pr[B \mid A]$. For example, the event C that the first die exceeds 5 and the event D that the second die exceeds 4 are independent.

Random variables. Formally, a R.V. is a function $X : \Omega \to \mathbb{R}$. (Doesn't have to be the real numbers, but usually it is.)

Example: X = sum of the two die. $X(a,b) = a + b$.

We can now write events as e.g. $X \geq 11$, which technically means the event $\{\omega : X(\omega) \geq 11\}$. This is the same as event A above. So we can write $\Pr[X \geq 11]$.

Example: Y = value of first die. $Y(a,b) = a$. Example: $\Pr[Y = 6 \mid X \geq 11] = \frac{2}{3}$.

If we take a function of random variables, e.g. $5X^2 + Y + 2$, this is another random variable, and we can talk about probabilities of events like $\Pr[5X^2 + Y + 2 \geq 10]$.

Random variables X, Y are **independent** if for all values x, y, $\Pr[X = x \text{ and } Y = y] = \Pr[X = x]\Pr[Y = y]$. For example, if Y is the value of the first die and Z the second die, then Y and Z are independent.

Expectation. The expectation of a random variable is $\mathbb{E}[X] := \sum_x x \Pr[X = x]$. The sum ranges over all possible values x of X. We can also write this very formally as $\mathbb{E}[X] = \sum_{\omega \in \Omega} p(\omega) X(\omega)$.

Examples above:

$$\begin{aligned}\mathbb{E}[Y] &= \sum_{y=1}^{6} \Pr[Y = y] \cdot y \\ &= \frac{1}{6}(1) + \frac{1}{6}(2) + \frac{1}{6}(3) + \frac{1}{6}(4) + \frac{1}{6}(5) + \frac{1}{6}(6) \\ &= 3.5.\end{aligned}$$

Similarly, $\mathbb{E}[X] = 7$.

Again, we can consider random variables that are functions of others, e.g. $\mathbb{E}[5X^2 + Y + 2]$.

Fact 1 (Linearity of expectation)**.** *For any random variables X, Y, we have $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.*

Proof.

$$\begin{aligned}\mathbb{E}[X + Y] &= \sum_{\omega \in \Omega} p(\omega) \left(X(\omega) + Y(\omega)\right) \\ &= \left(\sum_{\omega \in \Omega} p(\omega) X(\omega)\right) + \left(\sum_{\omega \in \Omega} p(\omega) Y(\omega)\right) \\ &= \mathbb{E}[X] + \mathbb{E}[Y].\end{aligned}$$

□

This is great because it's true no matter what, even if X and Y are correlated.

Also: convince yourself that if $\beta \in \mathbb{R}$ is any real number and X is any random variable, then $\mathbb{E}[\beta \cdot X] = \beta \cdot \mathbb{E}[X]$.

1.1 Conditional probability

Consider an event such as $Y = y$. We define the conditional expectation of X, conditioned on this event, as:

$$E[X \mid Y = y] := \sum_x \Pr[X = x \mid Y = y] x.$$

We may use subscripts to denote an expectation taken only over a particular random variable:

$$\mathop{\mathbb{E}}_{Y} \mathop{\mathbb{E}}_{X} [X \mid Y] := \sum_y \Pr[Y = y] \left(\sum_x \Pr[X = x \mid Y = y] x \right).$$

Another important probability fact:

Fact 2 (Law of total expectation)**.** *Let X, Y be random variables. Then $\mathbb{E}[X] = \mathbb{E}_Y \left(\mathbb{E}_X [X \mid Y]\right)$.*

Proof.

$$\begin{aligned}\mathop{\mathbb{E}}_{Y} \left(\mathop{\mathbb{E}}_{X} [X \mid Y] \right) &= \sum_{x,y} \Pr[X = x \text{ and } Y = y] x \\ &= \sum_x x \Pr[X = x] \left(\sum_y \Pr[Y = y \mid X = x] \right) \\ &= \sum_x x \Pr[X = x]\end{aligned}$$

□

2 Randomized Algorithms

We will formalize randomized algorithms via the word RAM model, where we add an "oracle", i.e. a function or API the algorithm can query. We suppose that the algorithm can, in constant time, ask for and receive an independent random number. For our purposes, this can be a bit, uniformly random integer in a finite range, or even random real number in $[0, 1]$. (In some contexts, one may rightly worry about whether this is realistic).

2.1 Randomized approximation algorithms

When an algorithm is randomized, we modify the definition of approximation to consider the *expected* value obtained by the algorithm. Formally, a randomized algorithm for optimization problem f has performance that is a random variable for any given instance, call that random variable ALG.

Then the algorithm has an approximation ratio α if $\mathbb{E}[ALG] \geq \alpha \cdot OPT$ (maximization problem); or it is a C-approximation if $\mathbb{E}[ALG] \leq C \cdot OPT$ (minimization problem).

One can also ask that the algorithm is exactly correct with high probability, or acheives a certain approximation with high probability, or so on.

3 Max-3SAT

A 3-CNF is a boolean formula in conjunctive normal form (CNF) with 3 literals per clause. Let us unpack what this means.

- We have n variables $x_1, \ldots, x_n$ which are boolean (can be set to true or false).
- A *literal* is either a variable x_i or its negation $\bar{x_i}$, meaning NOT x_i.
- A *clause* in this case is an OR of three literals, e.g. x_1 OR $\bar{x_2}$ OR x_3, where a variable cannot appear twice. Note the clause is true if any of the three literals evaluates to true, otherwise it is false.
- A 3-CNF formula is an AND of a sequence of m clauses, e.g.

$$(x_i \text{ OR } x_j \text{ OR } \bar{x_k}) \text{ AND } \cdots (x_a \text{ OR } \bar{x_b} \text{ OR } x_c)$$

  This evaluates to TRUE if *all* of the m clauses evaluate to TRUE.

Determining if a given 3-CNF is satisfiable, i.e. if there exists a setting of $x_1, \ldots, x_n$ such that it is TRUE, is NP-complete.

The Max-3SAT problem: Input a 3SAT on n variables, m clauses. Output: a setting of $x_1, \ldots, x_n$ to maximize the number of TRUE clauses.

Theorem 1. *There always exists a choice of variables such that $\geq \frac{7}{8}$ of the clauses are TRUE.*

Proof. Consider the following algorithm: For each x_i, independently set it to be TRUE with half probability, FALSE otherwise.

Consider any clause $j \in \{1, \ldots, m\}$. Let the random variable $Z_j = 1$ if it is satisfied, 0 otherwise.

Lemma 1. *For any j, $\Pr[Z_j = 1] = \frac{7}{8}$.*

Proof: homework!

Let Y be the number of satisfied clauses, i.e. $Y = \sum_{j=1}^{m} Z_j$. Then the expected number of satisfied clauses is

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{j=1}^{m} Z_j\right] = \sum_{j=1}^{m} \mathbb{E}[Z_j] = m \cdot \frac{7}{8}.$$

We used linearity of expectation. Then we used that

$$\mathbb{E}[Z_j] = \Pr[Z_j = 1](1) + \Pr[Z_j = 0](0) = \Pr[Z_j = 1].$$

Now we use another important and common fact from the course: If $\mathbb{E}[Y] \geq c$, then there exists a value of Y that is at least c. (Otherwise, if all values were strictly less than c, the expectation would have to be less than c.)

So there must exist at least one outcome in the sample space where $Y \geq \frac{7}{8}m$. In other words, there is some choice of $x_1, \ldots, x_n$ such that at least $\frac{7}{8}$ of the clauses are satisfied. □

This is an example of the *probabilistic method*: prove something exists by randomly constructing it, and showing this random construction sometimes succeeds.

In fact, Theorem 1 yields an approximation result:

Theorem 2. *There is a randomized $\frac{7}{8}$-approximation-ratio algorithm for MAX-3SAT.*

Proof. Our naive randomized algorithm achieves $\mathbb{E}[ALG] \geq \frac{7}{8}m$. Since $OPT \leq m$ always, we have $\mathbb{E}[ALG] \geq \frac{7}{8}OPT$. □

Amazingly, the following theorem is known:

Theorem 3 (Håsted)**.** *If $P \neq NP$, then no polynomial-time algorithm guarantees better than a $\frac{7}{8}$ approximation ratio for Max-3SAT.*

4 Min Weighted Vertex Cover

In the *weighted vertex cover problem*, the input is an undirected graph $G = (V, E)$ and a list of positive vertex weights: w_v for each $v \in V$.

The goal is to output a vertex cover with smallest total weight. Recall that a *vertex cover* is a set of vertices, $S \subseteq V$, such that for all edges $(u, v) \in E$, at least one of $\{u, v\}$ is in S. The *total weight* of a vertex cover is $w(S) := \sum_{v \in S} w_v$.

Algorithm: Initialize $S = \emptyset$. For each edge $e = (u, v)$:

- if u or v are already in S, continue.
- with probability $\frac{w_v}{w_u + w_v}$, add u to S. Otherwise, add v. (Note the probability of adding v is $\frac{w_u}{w_u + w_v}$.)

Note the idea is to bias toward adding the *smaller*-weight vertex: If w_v is much larger than w_u, then we are much more likely to add u.

First, note this does give a vertex cover, because we iterate through all edges and, if not yet covered, always add one of its endpoints to S. We will skip the running time and space analysis. The key idea for approximation will be that most of the "weight" in S comes from OPT vertices, so S cannot be much larger than OPT.

Theorem 4. *This randomized algorithm gives a 2-approximation.*

Proof. Let $S_0, \ldots, S_m$ be the sets of the algorithm at each round, with $S_0 = \emptyset$. Define $A_t = S_t \cap OPT$. In other words, it is all of the vertices added so far that are also in the optimal smallest-weight vertex cover. Meanwhile, define $B_t = S_t \setminus A_t$. This is all of the non-OPT vertices that are added.[1]

We will prove in Lemma 2 that $\mathbb{E}[w(B_m)] \leq \mathbb{E}[w(A_m)]$. In other words, S contains more weight from OPT than weight from non-OPT vertices. This implies

$$\begin{aligned} \mathbb{E}[w(S)] &= \mathbb{E}[w(A_m)] + \mathbb{E}[w(B_m)] \\ &\leq 2\,\mathbb{E}[w(A_m)] \\ &\leq 2\,\mathbb{E}[w(OPT)]. \end{aligned}$$

The last line follows because $A_m \subseteq OPT$, so $w(A_m) \leq w(OPT)$. □

[1] Notice the algorithm doesn't know these things because it doesn't know OPT. We are only using them in the analysis to prove performance.

Now we need to prove Lemma 2.

Lemma 2. $\mathbb{E}[w(B_m)] \leq \mathbb{E}[w(A_m)]$.

Proof. Let $X_t = w(A_t) - w(A_{t-1})$. Let $Y_t = w(B_t) - w(B_{t-1})$. These are random variables, the increases in the weights of the sets at each round. Now, we will need one more fact, Lemma 3: at each round t, if we fix the outcomes of the previous rounds $S_{t-1} = s$, then $\mathbb{E}[X_t \mid S_{t-1} = s] \geq \mathbb{E}[Y_t \mid S_{t-1} = s]$.

Given this fact, we know that $\mathbb{E}[Y_t] \leq \mathbb{E}[X_t]$ by the law of total expectation, because $\mathbb{E}[X_t] = \mathbb{E}_{S_{t-1}} \mathbb{E}[X_t \mid S_{t-1}]$ and similarly for Y_t. So

$$\begin{aligned}
\mathbb{E}[w(B_m)] &= \mathbb{E}\left[\sum_{t=1}^{m} Y_t\right] \\
&= \sum_{t=1}^{m} \mathbb{E}[Y_t] \\
&\leq \sum_{t=1}^{m} \mathbb{E}[X_t] \\
&= \mathbb{E}[w(A_m)].
\end{aligned}$$

□

Lemma 3. *For each round t and fixed $S_{t-1} = s$, $\mathbb{E}[X_t \mid S_{t-1} = s] \geq \mathbb{E}[Y_t \mid S_{t-1} = s]$.*

Proof. At round t, if the edge is already covered by some vertex in S_{t-1}, the sets remain the same, i.e. $\mathbb{E}[X_t \mid S_{t-1} = s] = \mathbb{E}[Y_t \mid S_{t-1} = s] = 0$. If the edge is not yet covered, then OPT must contain either u, or v, or both. If it contains u only, then

$$\begin{aligned}
\mathbb{E}[X_t \mid S_{t-1} = s] &= \Pr[\text{add } u] w_u \\
&= \frac{w_u w_v}{w_u + w_v} \\
&= \Pr[\text{add } v] w_v \\
&= \mathbb{E}[Y_t \mid S_{t-1} = s].
\end{aligned}$$

If OPT contains v only, then the expected increases are exactly the same. (Check.) And if OPT contains both u and v, then $\mathbb{E}[Y_t \mid S_{t-1} = s] = 0$ while $\mathbb{E}[X_t \mid S_{t-1} = s] > 0$. This proves that for any outcome of S_{t-1}, the inequality holds. □

5 Min-cut contraction algorithm

In the *global min cut problem*, we are given an undirected graph $G = (V, E)$. Given any cut U_1, U_2, the *value* of the cut is the number of edges crossing it.

The goal is to find any cut, with both U_1 and U_2 nonempty, minimizing the value of the cut.

Note one approach could be to consider all possible pairs of vertices s, t, convert the graph to a directed graph, and find the min s-to-t cut. The smallest of these would be the global min cut.

Here, we will consider a randomized algorithm that finds the *exact* correct solution with some probability. We will assume the graph is connected, otherwise there is a cut with value 0 that can be easily found in linear time (how?).

Define a *multigraph*: graph that may have multiple edges between any pair of vertices.

First, define "contraction". When contracting a graph $G = (V, E)$ by edge $e = (u, v)$, we create a copy of the graph G' without u, v or their incident edges. Then we add a new vertex a_{uv} that represents the "joining" of these two. For every edge (u', u) or (u', v) in the original graph, we add an edge (u', a_{uv}) in the new graph. (Note edges are still undirected and it may be a multigraph.)

Algorithm:

- Pick an edge uniformly at random.
- Contract the graph along that edge.
- Repeat until only two vertices u, v remain.
- Return the cut corresponding to these two vertices (i.e. U_1 = all vertices that were contracted into u, U_2 = all that were contracted into v).

Theorem 5. *The probability of correctness is at least* $\frac{2}{n(n-1)}$.

(Note: this is not very good, but in HW we will see that repeating the algorithm polynomially many times gives a good success probability. What would be the success chance if we picked a cut uniformly at random from the space of all cuts?)

Proof. Let k be the size of the min cut (pick a particular min cut if there are more than one). Let E_t be the event that that, in round t, the edge selected does *not* belong to the min cut. There are $n-2$ rounds, because in each round we decrease the number of vertices by one until we have two left. The algorithm outputs the min cut if it never contracts any edge in the min cut (agreed?), so the goal is to prove that $\Pr[E_1 \cap \cdots \cap E_{n-2}] \geq \frac{2}{n(n-1)}$.

Note each vertex has degree at least k, otherwise putting it in U_1 alone would be a smaller cut. So the total number of edge endpoints is at least nk where n is the number of vertices. So the number of edges is at least $\frac{nk}{2}$.

We pick an edge uniformly at random, and there are at least $\frac{nk}{2}$ edges, so $\Pr[E_1] \geq 1 - \frac{k}{|E|} \geq 1 - \frac{2}{n}$. If we succeeded in the first round, we have an $n-1$ vertex graph with min cut k, so by the same reasoning, $\Pr[E_2 \mid E_1] \geq 1 - \frac{2}{n-1}$. Repeat until n drops to 2: we have

$$\begin{aligned}
\Pr[E_1 \cap \cdots \cap E_{n-2}] &= \Pr[E_1] \Pr[E_2 \mid E_1] \cdots \Pr[E_{n-2} \mid E_1 \cap \cdots \cap E_{n-3}] \\
&\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{3}\right) \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right) \cdots \left(\frac{1}{3}\right) \\
&= \frac{(n-2)!(2)}{n!} \\
&= \frac{2}{n(n-1)}.
\end{aligned}$$

□