

Lecture 11: Randomized Algorithms

1. Randomized algorithm for finding a perfect matching

Today we extend the randomized algorithm from last lecture that determined the existence of a perfect matching in a graph $G(V, E)$, to actually finding a perfect matching in G . The key computational step will be a matrix inversion.

First we establish a key (and surprising) property of subsets of random numbers:

Definition: A *set system* (S, F) consists of a finite set S of elements, $S = \{x_1, x_2, \dots, x_n\}$ and a family F of subsets of S , $F = \{S_1, \dots, S_k\}$, $S_j \subseteq S$.

For each element of S , assign weight w_i to x_i , where w_i is chosen uniformly and independently from $\{1, \dots, 2n\}$. Denote the weight of a set S_j to be $\sum_{x_i \in S_j} w_i$.

Lemma 1 (Isolating lemma) *The probability that there is a unique minimum weight set is at least $1/2$.*

Proof: Fix the weight of all elements except x_i . Define the *threshold* for element x_i to be the number α_i such that if $w_i > \alpha_i$ then x_i is in no set with minimum weight, and if $w_i \leq \alpha_i$ the x_i is in some set with minimum weight.

Clearly, if $w_i < \alpha_i$, then x_i is in *every* set with minimum weight. The only ambiguity occurs when $w_i = \alpha_i$. In this case we say that x_i is *singular*.

A key observation is that the weight of element x_i is *independent* of the threshold value α_i . Since w_i is chosen uniformly at random from $\{1, 2, \dots, 2n\}$, the probability that an element is singular is at most $1/2n$.

Observation 1 *If no element is singular, then the subset with minimum weight is unique.*

Since S contains n elements, the probability that S contains a singular element is at most $n \times \frac{1}{2n} = \frac{1}{2}$. Thus, with probability at least $1/2$, there exists no singular element, implying the lemma. ■

2. Matchings in bipartite graphs

Given a bipartite graph $G(U, V, E)$, to each edge $(i, j) \in E$ assign a weight $w_{i,j}$ chosen

uniformly and independently from $[1, 2m]$, where $m = |E|$. By the isolating lemma, the minimum weight perfect matching in G will be unique with probability at least $1/2$.

As in the previous lecture, we define the matrix B such that

$$b_{i,j} = \begin{cases} x_{i,j} & \text{if } i \sim j, i \in U, j \in V \\ 0 & \text{otherwise.} \end{cases}$$

Recall that the determinant of this matrix is equivalent to zero iff G has no perfect matchings. Set $x_{i,j} = 2^{w_{i,j}}$.

Lemma 2 *Suppose that the minimum weight perfect matching M is unique with weight W . Then $|B| \neq 0$ and the highest power of 2 that divides $|B|$ is 2^W .*

Proof: A permutation of length n corresponds to a perfect matching in M . Recall that

$$|B| = \sum_{\pi} \text{sign}(\pi) \text{value}(\pi),$$

where $\text{value}(\pi) = \prod_{i=1}^n b_{i,\pi(i)}$. Let π_M be the permutation corresponding to M , with $\text{value}(\pi_M) = 2^W$. Then the value of every other permutation is either 0 or a greater power of 2 than W . This implies the lemma. ■

Thus, we can recover W by calculating the determinant of B . The following lemma gives us a way of recovering which edges belong to M . Denote $B_{i,j}$ as the matrix derived from B by removing the i th row and j th column.

Lemma 3 *The edge (u_i, v_i) belongs to M iff $\frac{|B_{i,j}|2^{w_{i,j}}}{2^W}$ is odd.*

Proof:

Note that

$$|B_{i,j}|2^{w_{i,j}} = \sum_{\pi: \pi(i)=j} \text{sign}(\pi) \text{value}(\pi).$$

Let π_M be the permutation corresponding to M . If (u_i, v_j) is an edge in M , then one permutation, π_M , will have weight 2^W . All other permutations will either have weight 0 or a higher power of 2 than W . Therefore, 2^W will be the highest power of 2 which divides the righthand side iff $(u_i, v_j) \in M$. ■

$|B_{i,j}|$ is the (i, j) entry of the cofactor matrix, which we can compute from the following relationship:

$$T^{-1} = \frac{1}{\det(T)} (\text{cofactor}(T))^T$$

We can therefore find the perfect matching with minimum weight M with a single matrix inversion. Since M will be unique with probability at least $1/2$, we only need to run this algorithm a constant number of times to have an arbitrarily high probability of success.

3. Matchings in general graphs

Recall from last lecture the definition of the Tutte matrix T for a graph $G(V, E)$:

$$t_{i,j} = \begin{cases} x_{i,j} & \text{if } i \sim j, i > j \\ -x_{i,j} & \text{if } i \sim j, i < j \\ 0 & \text{otherwise.} \end{cases}$$

Recall that we showed that the determinant of T is equivalent to 0 iff T has a perfect matching.

To find this perfect matching, for each i, j set $x_{i,j}$ to $2^{w_{i,j}}$, where $w_{i,j}$ is chosen uniformly and independently from $\{1, \dots, 2m\}$ with $m = |E|$.

Lemma 4 *Suppose that the minimum weight perfect matching in $G(V, E)$ is unique. Let this matching be M and its weight be W . Then $|T| \neq 0$, and the highest power of 2 which divides $|T|$ is 2^{2W} .*

Proof: For a permutation π , we define $value(\pi)$ as in the bipartite case. In addition, for a cycle decomposition $\{c_1, c_2, \dots, c_k\} = \pi$,

$$value(\pi) = \prod_{i=1}^k value(c_i).$$

Observation 2 *Terms corresponding to permutations with odd cycles cancel each other.*

Proof: To see this let $\pi = \{c_1, c_2, \dots, c_k\}$ be a permutation with an odd cycle. WLOG assume that this cycle is c_1 . Note that if we reverse the direction of all the edges to get cycle c'_1 , $value(c_1) = -value(c'_1)$. Let $\pi' = \{c'_1, c_2, c_3, \dots, c_k\}$. Then $value(\pi) = -value(\pi')$. Since reversing the edges of a cycle does not change the sign of a permutation, π and π' cancel each other in the determinant calculation, which implies the observation. ■

Therefore, we only need to consider even cycles.

Each perfect matching corresponds to a permutation with $n/2$ cycles of length 2. For a given even cycle x_1, x_2, \dots, x_{2k} for some $1 \leq k \leq n/2$, note that there are two corresponding perfect matchings of these vertices.

Observation 3 *The value of an even cycle of c length greater than 2 is at least as much as the average value of its component perfect matchings.*

Proof: The contribution from the even cycle is just the product of the weights of its edges. Denote the value of the (x_{2i}, x_{2i+1}) perfect matching as a , and denote the value of the $(x_{2i+1}, x_{2(i+1)})$ matching as b . Then the value cycle of cycle c is ab , and the weight of the two perfect matchings are a^2 and b^2 respectively. Therefore,

$$(a - b)^2 \geq 0 \quad (1)$$

$$a^2 + b^2 \geq 2ab \quad (2)$$

$$\frac{a^2 + b^2}{2} \geq ab \quad (3)$$

■

Since M is unique, it follows that the weight of any permutation with even cycles of length greater than 2 will be strictly greater than 2^{2W} . This means that all of the terms in the determinant will be composed of sums of powers of 2 with exponents greater than $2W$ plus 2^{2W} , which implies the lemma. ■

Lemma 5 *If M is the unique minimum weight matching in G with weight W , then $\frac{|T_{i,j}|2^{2w_{i,j}}}{2^{2W}}$ is odd iff the edge $(i, j) \in M$.*

The proof of this lemma for general graphs is identical to the corresponding lemma for bipartite graphs. So by the same algorithm as for the bipartite case, we can determine the minimum weight perfect matching of a general graph G with a single matrix inversion.

4. Randomized algorithm for global min-cut

In the *global min-cut* problem, we wish to find a minimum set of edges such that a graph becomes disconnected. Formally,

Definition: Given an undirected graph $G(V, E)$, a **global min-cut** is a partition of V into two subsets (A, B) such that the number of edges between A and B is minimized.

To derive an algorithm for this, note that the global min-cut is the minimum over all possible $s - t$ cuts.

Solution 1: *Compute the min $s - t$ cut for all pairs $s, t \in V$.*

This algorithm requires $O(n^2)$ calls to a min s-t cut solver. We can reduce this by noting that any node s must appear in one of A or B , meaning we can reduce the number of min s-t cut solves by a factor of n .

Solution 2: Fix s and find min $s - t$ cut for all $t \in V$.

For a long time, the intuition was that global min-cut was harder than $s - t$ cut. David Karger was able to show that this is not the case with a randomized algorithm.

Algorithm 1 Karger's randomized global min-cut

repeat

 Choose an edge $\{u, v\}$ uniformly at random from E .

 Contract the vertices u and v to a super-vector w .

 Keep parallel edges but remove self-loops.

until G has only 2 vertices.

 Report the corresponding cut.

Theorem 1 The probability that the algorithm finds the minimum cut in G is at least $2/n^2$.

Proof: Let F be the set of edges in a global min-cut and suppose $|F| = k$. Let E be the even that the algorithm does not contract an edge from F in step 1. Since the minimum cut has value k , the degree of each vertex must be at least k , which implies the following claim:

Claim 1 $|E| \geq \frac{k|V|}{2} = \frac{kn}{2}$.

Denote E_i be the event that the i th edge picked belongs to the min-cut. The probability that the first edge picked belongs to the min-cut is therefore:

$$\begin{aligned} Pr(E_1) &= 1 - \frac{k}{|E|} \\ &\geq 1 - \frac{2}{n} \end{aligned}$$

Similarly,

$$\begin{aligned} Pr(E_2|E_1) &\geq 1 - \frac{2}{n-1} \\ Pr(E_i|E_1 \cap E_2 \cap \dots) &\geq 1 - \frac{2}{n-i+1} \end{aligned}$$

Combining these to get the total probability of success gives

$$\begin{aligned} Pr(\text{success}) &= Pr(E_1 \cap E_2 \cap \dots \cap E_{n-2}) \\ &\geq Pr(E_1)Pr(E_2|E_1) \dots Pr(E_{n-2}|E_1 \cap \dots \cap E_{n-3}) \\ &\geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \dots \left(1 - \frac{2}{3}\right) \\ &\geq 2 \frac{(n-2)!}{n!} \geq \frac{2}{n^2}, \end{aligned}$$

So the algorithm will succeed at with probability at least $2/n^2$. ■

The probability of finding a min-cut seems rather low, and goes to zero as $n \rightarrow \infty$. However, what happens to the probability of success if we run the algorithm t times? The probability of failure will be at most

$$\left(1 - \frac{2}{n^2}\right)^t.$$

So if we set $t = cn^2$ for some constant c , the probability of failure will be at most $\left(1 - \frac{2}{n^2}\right)^{cn^2} \leq e^{-2c}$. To make the probability of failure as small as, say e^{-14} , it is only necessary to run the algorithm $7n^2$ times.