CS 590R: Min-Cut Algorithm and Random Variables

Chun-Kong Cheng

1 Min-Cut Algorithm

Suppose each link connected to nodes in a network has a certain failure probability p, what is the probability that the network is disconnected? We can estimate the probability by simulating the network, but when p is small e.g., 10^{-6}, it is very difficult for us to collect useful data. Let us look at how we can look at this question using a theoretical approach.

This question is related to the problem of finding the *min-cut* of a connected graph. A min-cut is the minimum number of edges that have to be removed from a connected graph in order to create two disjoint graphs, and a *min-cut set* is the set of edges that contribute to min-cut. For example, in a tree, the min-cut value is one. Knowing the min-cut provides us more information on how a network can be disconnected. The min-cut algorithm provided here is a randomized algorithm – its correctness depends on the number of times it is executed.

1.1 Randomized Min-Cut Algorithm

Input: A n node graph
Output: A minimal set of edges that disconnects the graph.

1. **repeat** $n-2$ **times**:
   (a) Pick an edge uniformly at random.
   (b) Contract the edge.
endrepeat

2. **output** the set of edges connecting the two remaining vertices.

In the above algorithm, at each iteration, an edge is picked at random, and an operation called *contraction* is performed. Contraction of an edge means merging two vertices adjacent to that edge together, resulting in the reduction of one node at each iteration. Hence, after $n-2$ contractions, only two vertices are remained. The set of edges that connect these two edges are output in step 2.

1.2 Correctness of the Algorithm

In step 2, the set of edges returned must be the cut set of the graph connecting these two vertices. First, we want to show that the cut set of the final graph in step 2 is a cut set of the original graph. We obtain this result if the following lemma is true.

Lemma 1.1 *Every cut set in the new graph after contraction is a cut set in the original graph.*

Proof: Consider the two sets of disjoint vertices, V_1 and V_2 which are obtained after removing the edges in the cut set from the new graph after contraction. We can obtain the same cut-set in the graph before contraction occurs because contraction occurs in either one of the nodes of V_1 or V_2, but not in the cut-set. □

Based on the above proof, we can claim the cut set of the final graph is the same as the original graph without any contraction. That is, any cut set in the final graph must also be a cut set of the original graph, and we obtain the following lemma:

Lemma 1.2 *Contraction operation (step 1(b)) does not reduce the size of the min-cut set.*

The drawback of this algorithm is that the cut set may not be minimal. We now derive the probability that the cut set is minimal.

1.3 Probability that the Min-Cut Set is Found

Assume that the graph has a min-cut set of k edges. The probability of finding one such set C is given by the following theorem:

Theorem 1.1 *The algorithm outputs a min-cut set of edges with probability* $\geq \frac{2}{n(n-1)}$.

To prove this theorem, let us state the following lemma:

Lemma 1.3 *Let C be a min-cut set of G. If the run of the algorithm did not contract any edge of C, it also did not eliminate any edge of C.*

Proof: This lemma is true because any edge that are not eliminated by the contraction operation will remain and is returned in step 2. □

Since the minimum cut-set has k edges, all vertices have degree $\geq k$ (otherwise, we can find a cut-set of $k-1$ edges or less by simply disconnecting the vertex with $\leq k-1$ edges from the graph). Also, since each edge connecting two vertices contributes two degrees, and the total degree of the vertices is $\geq nk$, the graph must have $\geq nk/2$ edges.

Let $E_i =$ "the edge contracted in iteration i is not in C". The probability that a min-cut set is obtained is the probability that in all $n-2$ iterations, no edges in C are contracted, i.e., $\Pr(\cap_{i=1}^{n-2} E_i)$.

Note that $\Pr(\cap_{i=1}^{n-2} E_i) = \Pr(E_1)\Pr(E_2|E_1)\Pr(E_3|E_2 \cap E_1)\ldots\Pr(E_{n-2}|\cap_{i=1}^{n-3} E_i)$ (because $Pr(A|B) = \frac{Pr(A \cap B)}{Pr(B)}$).

$\Pr(E_1) \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$ (k edges out of at least of $nk/2$ edges are in C, so the probability of choosing an edge in C is at most $\frac{k}{nk/2}$.)

$\Pr(E_2|E_1) \geq 1 - \frac{k}{(n-1)k/2} = 1 - \frac{2}{n-1}$ (by lemma 2, and given E_1, none of the edge in C is eliminated. Since one node is less, the total number of edges now is at least $(n-1)k/2$.)

Using similar arguments, $\Pr(E_i|\cap_{j=1}^{i-1} E_j) \geq 1 - \frac{2}{n-i+1}$.

Thus, $\Pr(\cap_{i=1}^{n-2} E_i) \geq \Pi_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \Pi_{i=1}^{n-2}\left(\frac{n-i-1}{n-i+1}\right) = \frac{2}{n(n-1)}$, and the theorem is proved.

1.4 High Probability Min-Cut Algorithm

The probability of achieving a min-cut set by the previous algorithm can be increased by running it $n^2 \ln n$ times, as shown below:

Input: A n node graph
Output: A minimal set of edges that disconnects the graph.

1.**for** $i = 1$ **to** $n^2 \ln n$
    1.1. **repeat** $n - 2$ **times**:
        (a) Pick an edge uniformly at random.
        (b) Contract the edge.
    endrepeat
    1.2 Let C_i be the set of edges connecting the
    two remaining vertices.
endfor
2. **output** the set with the minimum size among the C_i's.

Theorem 1.2 *The high probability algorithm outputs a min-cut set with probability at least* $1 - 1/n^2$.

Proof: Let S_1 and F_1 be the events that the first algorithm succeeds and fails to find the min-cut set respectively, and S_2 and F_2 be the events that the new algorithm succeeds and fails to find the min-cut set respectively.

$$\begin{aligned} Pr(F_1) &= 1 - Pr(S_1) \\ &\leq 1 - \frac{2}{n(n-1)} \\ Pr(F_2) &\leq (1 - \frac{2}{n(n-1)})^{n^2 \ln n} \end{aligned}$$

By the fact that $1 + x \leq e^x$ (from Taylor's expansion), we have

$$\begin{aligned} Pr(F_2) &\leq e^{(\frac{-2}{n(n-1)}) \cdot n^2 \ln n} \\ &\leq e^{-2 \ln n} \\ &= e^{\ln n^{-2}} \\ &= \frac{1}{n^2} \end{aligned}$$

Hence, $Pr(S_2) = 1 - Pr(F_2) \geq 1 - \frac{1}{n^2}$. $\square$

1.5 Running Time

Let m be the maximum degree of a vertex. Then each contraction requires $O(m)$ operations. The first algorithm will need $(n-2) \cdot O(m) = O(mn)$ running time.

Since the second algorithm runs the first algorithm $n^2 \ln n$ times, its running cost is $O(mn^3 \ln n)$ time.

The second algorithm is called a *whp* algorithm with correctness probability greater than $1 - \frac{1}{n^c}$ where $c > 1$.

2 Random Variables

Let $(\mathcal{S}, Pr)$ be a discrete probability space, and V be a set of values. A random variable X defined on $(\mathcal{S}, Pr)$ is a *function*

$$X : \mathcal{S} \to V$$

Let $\mathcal{E}(r) = \{s \in \mathcal{S} \mid X(s) = r\}$. Then,

$$Pr(X = r) \;=\; Pr(\mathcal{E}(r)) = \sum_{s \in \mathcal{E}(r)} Pr(s).$$

A random variable allows us to handle probability problems in any domains (usually real value domain).

Two random variables X and Y (defined on the same sample space) are called independent if for all x and y,

$$\Pr\{X = x \text{ and } Y = y\} = \Pr\{X = x\} \Pr\{Y = y\}$$

2.1 Expectation

The **expectation** of a discrete random variable X:

$$E[X] \;=\; \sum_{i \in range((X))} i \cdot Pr(X = i).$$

Example. Consider throwing a fair dice of 6 faces. The sample space is 6 faces. Define a random variable X which represents the face value of each outcome. Then X can be any of the integer in the range $[1, 6]$, and

$$\begin{aligned} E(X) &= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} \\ &= \frac{1}{6} \cdot \frac{6 \cdot 7}{2} \\ &= 3.5 \end{aligned}$$

Notice that 3.5 is not in the range of X.

2.2 Linearity of Expectation

Theorem 2.1 *For any two random variables X and Y*

$$E[X + Y] \;=\; E[X] \;+\; E[Y].$$

Proof:

$$\begin{aligned} E[X+Y] &= \sum_{i \in range(X)} \sum_{j \in range(Y)} (i+j) Pr((X=i) \cap (Y=j)) \\ &= \sum_i \sum_j i Pr((X=i) \cap (Y=j)) + \sum_j \sum_i j Pr((X=i) \cap (Y=j)) \end{aligned}$$

Using Lemma 2.1, we sum over all possible choices of i (j) and the following is obtained:

$$\begin{aligned} E[X+Y] &= \sum_i i Pr(X=i) + \sum_j j Pr(Y=j) \\ &= E[X] + E[Y] \end{aligned}$$

□

As shown by the proof, we do not make assumption about the nature of X and Y. In particular, X and Y need not be independent. We can generalize the theorem to three or more random variables, e.g.,

$$E[X + Y + Z] = E[X] + E[Y] + E[Z].$$

Linearity For any real values a and b and random variables X and Y, the following holds:

$$E[aX + bY] = aE[X] + bE[Y]$$

Lemma 2.1 *If $E_1, E_2, \ldots, E_k$ are disjoint events such that $\sum_{i=1}^{k} Pr(E_i) = 1$ then for any event B,*

$$\sum_{i=1}^{k} Pr(B \cap E_i) = Pr(B).$$

Proof: This lemma can be understood by recognizing the fact that the union of $E_1, E_2, \ldots, E_k$ forms the whole sample space. The event B is then "partitioned" into the regions of $E_1, E_2, \ldots, E_k$, or mathematically, $B = \bigcup_{i=1}^{k}(B \cap E_i)$. Thus,

$$\begin{aligned} Pr(B) &= Pr(\bigcup_{i=1}^{k}(B \cap E_i)) \\ &= \sum_{i=1}^{k} Pr(B \cap E_i) \end{aligned}$$

□