Lecture - 13, 14

Wednesday, 24 August 2016 (14:25 - 15:15)
Friday, 26 August 2016 (16:30 - 17:15)

Randomized Min-Cut Algorithm

This class mainly focused on defining terms and making observations that are required for the analysis of the randomized min-cut algorithm, known as Karger's algorithm. These notes are written with the intention to capture the same flow as discussed in class.

We began by making note of the *Handshaking Lemma* which states that sum of the degrees of all nodes in an undirected graph is equal to twice the number of edges.

Lemma 1. *Given an undirected graph $G(V, E)$ such that $|V| = n$ and d_i is the degree of node v_i, $\sum_{i=1}^{n} d_i = 2|E|$.*

Proof. It is clearly observable that each edge $e \in E$ contributes to the degree of precisely two nodes, the nodes that the edge is incident on. The lemma hence follows.

Then, we saw what is meant by a partition of vertex set, cut, cut-set followed by min-cut.

Definition 01 *The disjoint union of the vertex set, denoted by $V = A \dot{\cup} B$, is a* ***partition*** *of the vertex set into two sets A and B such that $A \cap B = \Phi$ and $A \cup B = V$.*

Definition 02 *A* ***cut*** *$C = (A, B)$ is a partition of V, given a graph $G(V, E)$. The* ***cut-set*** *of the cut $C = (A, B)$ is the set of all edges $= \{(u, v) \in E | u \in A, v \in B\}$.*

Definition 03 *Given a graph $G(V, E)$, a cut is said to be a* ***min-cut****, if the cut has minimum cardinality cut-set in G.*

Based on the lemma and definitions stated above, we can make a few simple observations as stated below:

Observation 1:
Given a complete graph K_n, the size of the min-cut $= n - 1$

Observation 2:
There need not be a unique min-cut in a given graph

Observation 3:
Size of the min-cut in a given graph is always less than or equal to the minimum degree. That is, given a graph $G(V, E)$, if $|C_{min}|$ is the cardinality of the cut-set of a min-cut denoted by C_{min} and k denotes the minimum degree of a node in G,

$$|C_{min}| \leq k \tag{1}$$

Observation 4:
Given a graph $G(V, E)$, if k denotes the minimum degree of a node in G and $|V| = n$, then

$$|E| \geq \frac{kn}{2} \tag{2}$$

With all the necessary artillery in place, we are ready to take a look at the randomized algorithm to determine the min-cut of a given graph. But, before we dwell into the details, let us discuss the working principle of the strategy that the algorithm adopts.

A weird computing strategy: A single run of the algorithm generates an 'answer'. You might get lucky and this turn out to be the 'best answer'. Repeat the algorithm in case you are not satisfied with the answer.

We often see such strategies in play even in our real lives. Assume you are standing at a bus stop in city A and wish to travel to city B, both cities are in Punjab. Given that all the cities in Punjab are well connected through a fleet of buses, you are guaranteed that you can travel from A to B using just the bus service. However, assume that the buses do not follow a routine and come in a random fashion. Ideally you would like a direct bus from A to B that takes the minimum (in fact zero) number of switch of buses. Or you could catch a bus and get off at a convenient location and continue doing this until you reach the final destination. This would however cost you more switches from bus to bus, and still get you to the destination. What we can observe is that, if you wait sufficiently at bus stop A, you are most likely to get the direct bus to B.

Following a similar strategy of repeating trials, we solved the following exercise problem:

Ex:1) There is a sack containing 1000 fruits of which only 10 of them are apples. Assume that you randomly pick a fruit each time and drop it back into the sack. You repeat this 5000 times. What is the probability that you fail seeing an apple at all ??

Solution :
The probability that you see an apple in an attempt $= \frac{10}{1000}$
The probability that you do not see an apple in an attempt hence $= (1 - \frac{10}{1000})$

The probability that you do not see an apple in 5000 attempts $= (1 - \frac{10}{1000})^{5000} \approx (\frac{1}{e})^{50}$

The probability that the above event does not occur, i.e the probability that you see at least one apple in the 5000 attempts $= 1 - (1 - \frac{10}{1000})^{5000} \approx 1 - (\frac{1}{e})^{50}$

——————————x-x-x——————————

Question to ponder on :
We observed that every partition of the vertex set V, given a graph $G(V, E)$, induces a cut C on G. Choose a subset A uniformly at random from all possible subsets and let the set $B = \overline{A}$. Let $C = (A, B)$ be the induced cut. Repeat the process n times to obtain cuts $c_1, c_2, \ldots c_n$. If $c_{min} = Min(c_1, c_2, \ldots c_n)$, how good is c_{min} compared to the actual min-cut of the given graph G?

Questioning about the min-cut of a graph, we saw that there is always an $F \subseteq E$ such that F is a global min-cut. The BIG question is to devise a method to find $|F|$.

Observation 5:

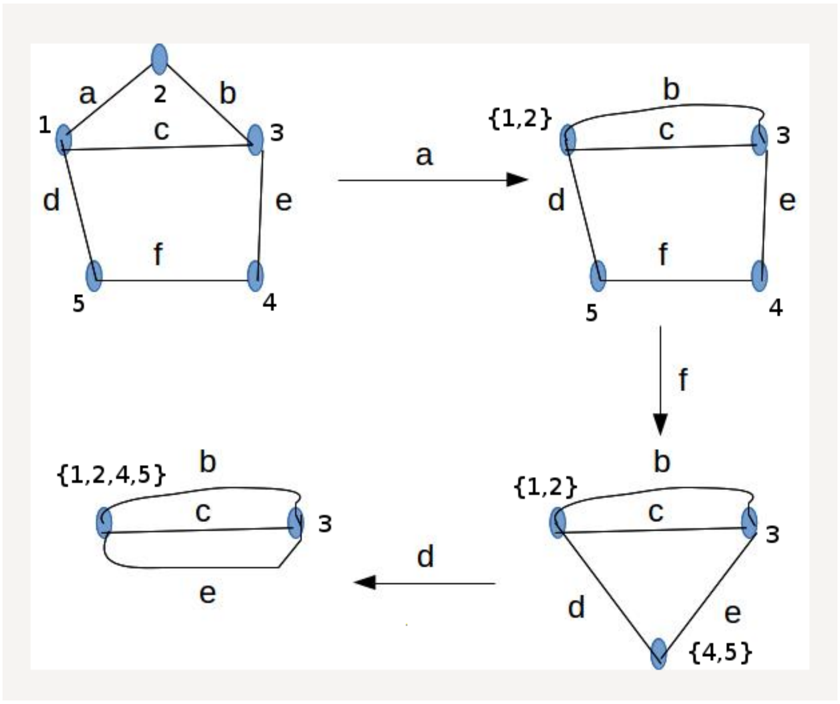
Combining observation 3 and 4, it is easy to observe that :

$$|E| \geq \frac{kn}{2} \geq \frac{n|F|}{2} \tag{3}$$

Definition 04 ***Edge Coalescing***

Coalescing of an edge (u, v) results in merging of the two vertices u and v into a single vertex labeled ***uv****, such that the graph now has one lesser vertex. Every edge in the original graph of the form (w,u) or (w,v), where $w \neq u, v$, is now replaced with the edge (w,**uv**).*

Example :



Randomized Min-Cut Algorithm - Karger's Algorithm

We will now see a randomized approach to obtaining a cut of a given graph. More interesting, we will show that the approach in fact gives the min-cut on repeated trials. The technique discussed below was given by Karger in 1993, and hence named after him. The algorithm uses the operation of edge contraction/ coalescing in order to determine the cut.

Let us say we are to determine the min-cut on a given graph $G_1(V_1, E_1)$. The algorithm begins by randomly choosing an edge $e_1 \in E_1$ and coalesces it, to obtain a new graph $G_2(V_2, E_2)$. The algorithm proceeds by randomly picking an edge $e_2 \in E_2$ and coalescing it. This process of picking and edge and coalescing it continues until we obtain a graph $G_k(V_k, E_k)$ such that $|V_k| = 2$. That is, the process stops when we are left with just two nodes in the graph. We will see how the remaining edges E_k in the final graph G_k is in fact a cut of the original graph G_1. Prior to that, there are a few observations that can be made :

(a) When we coalesce an edge, two nodes are merged and hence the resulting graph has precisely 1 lesser node than before. Please note that the same cannot, however, be said about the edges. More than one edge might be lost due to the merging. Thus, the algorithm would take precisely $|V| - 2$ steps before it halts.

(b) If the given graph if G_1, we obtain several intermediary multi-graphs, $G_2, G_3, \ldots G_{|V|-1}$. An important fact to note is that a cut of any of the graphs G_i is a cut of G_1 as well.

(c) The two nodes that remain at the end of the algorithm in fact represent a partition of the vertex set. Consider the example shown above, where the nodes were numbered 1 through 5. The final graph had $\{1, 2, 4, 5\}$ as one node and $\{3\}$ as the other, which qualifies as a partition of the vertex set.

(d) The edges that remain in the end are those edges of the original graph that lie across the partition. That is, assume the vertex set is partitioned into sets S, T such that $S \dot{\cup} T = V$. Then, every edge that remains is of the form (u, v), where node $u \in S$ and node $v \in T$ and edge $(u, v) \in E_1$. This is easy to observe as, every edge in the original graph that lies between nodes belonging to the same partition would be lost when the corresponding nodes are merged, to belong to the same partition. Thus, this is a result of the coalescing operation itself.

(e) Every edge $(u, v) \in E_1$ such that $u \in S$ and $v \in T$ is present in the end of the algorithm. Let us assume the contrary, that is, $\exists$ an edge $(u, v) \in E_1$ but in not present in the $E_{|V|-1}$ (final set of edges). Then, we can say that this edge was lost when u, v were merged. This is contradictory to the fact that u, v belong to different partitions and hence never merged.

Lemma 2. *Given a graph $G_1(V_1, E_1)$, the edges $e_1, e_2, \ldots e_f$ that remain form a cut of the graph G_1.*

Proof. *This follows directly from the observations 2, 4, and 5 that were made above.*

Theorem 1. *The success probability of the algorithm is bounded below by $\binom{n}{2}^{-1}$*

Proof. *Given $G(V, E)$, let $F \subseteq E$ be the global min-cut such that $|F| = f$ and $|V| = n$.*

$$\begin{aligned}
\textit{The probability that an edge } e \in F \textit{is coalesced in the first attempt } &= \frac{f}{|E|} \\
\textit{The probability that the edge is not coalesced} &= \left(1 - \frac{f}{|E|}\right) \\
&\geq \left(1 - \frac{2}{n}\right)
\end{aligned}$$

We know that the algorithm runs through $|V| - 2$ steps. After step 1 we would have $n - 1$ nodes, and similarly after step i we would have $n - i$ nodes. Thus, after each step the remaining number of edges E_i would be : $|E_i| \ \geq \frac{f(n-i)}{2}$

Thus, the probability that the algorithm does not coalesce any of the edges belonging to the cut F is given by probability p, where:

$$\begin{aligned}
p &\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\cdots\left(1 - \frac{2}{3}\right) \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{1}{3}\right) \\
&= \frac{2}{n(n-1)} \\
&= \binom{n}{2}^{-1}
\end{aligned}$$

Thus, if the probability of success is p, then the experiment must be repeated $\frac{1}{p}$ times for the event to occur at least once. Hence proved.