Advanced Algorithms (67824) – Exercise 3

Edan Orzech 322362849, Mike Greenbaum 211747639

November 2021

1. Define ϕ as the maximum concurrent flow. Define m as the sparsity of a sparsest cut. In class we proved that $\boxed{\phi \leq m}$. Define $N(S)$ to be the number of sources and sinks that S separates. Define $(X, \overline{X})$ to be all the edges in the cut X. Define $\alpha(S, \overline{S})$ to be the sparsity function. Define f_1^* to be a max-flow through $s_1 \to t_1$. Due to the max-flow min-cut theorem, there exists a cut S_1 s.t. $c(S_1, \overline{S_1}) = |f_1^*|$.

   Notice that $m \leq \alpha(S_1, \overline{S_1}) = \frac{c(S_1, \overline{S_1})}{N(S_1)} \leq \frac{c(S_1, \overline{S_1})}{1} = c(S_1, \overline{S_1}) = f_1^*$. Therefore, exists a flow f_1 through $s_1 \to t_1$ s.t. $|f_1| = m$.

   We use the following result from the literature[1]. In the paper, the authors show an algorithm (algorithm TWO-COMMODITY FLOW ALGORITHM in page 3), that finds a flow $s_1 \to t_1$ and then finds a maximal flow through $s_2 \to t_2$ without changing the value of the flow through $s_1 \to t_1$.

   The proofs of correctness of the algorithm don't depend on the optimality of the flow from $s_1 \to t_1$ and therefore, we can use the algorithm with a non-optimal flow.

   Therefore, instead of choosing the flow $s_1 \to t_1$ to be maximal (i.e. f_1^*), we will choose it to be thef_1 that we defined previously (s.t. $|f_1| = m$).

   Denote the output of the algorithm as (F_1, F_2) (where F_i is a flow from $s_i \to t_i$). we know that $|F_1| = m$ by the algorithm's definition.

   We will use lemma 2.3 (stated in the end of page 4) that states that there exists a cut S that separates both sources and sinks s.t. $|F_1| + |F_2| = c(S, \overline{S})$. Define the cut X to be the cut induced by the algorithm as stated in lemma 2.3.

   The problem with the lemma is that in its proof there exists a case which assumes the optimality of the flow F_1, therefore, we will tackle this case independently. The only case that assumes the optimal flow for F_1 happens if and only if $\forall (u, v) \in (X, \overline{X}),\ F_1(u, v) = 0$.

   (a) If $\forall (u, v) \in (X, \overline{X}),\ F_1(u, v) = 0$. In this case, as stated in the paper, due to the max-flow min cut theorem for s_2, t_2, it holds that $|F_2| = c(X, \overline{X})$. Therefore

   $$m \leq \alpha(X, \overline{X}) = \frac{c(X, \overline{X})}{N(X)} \leq \frac{c(X, \overline{X})}{1} = c(X, \overline{X}) = |F_2|$$

   (b) Else, lemma 2.3 holds, in this case exists a cut S that separates both sources and sinks s.t. $F_1 + F_2 = c(S, \overline{S})$. Therefore,

   $$m \leq \alpha(S, \overline{S}) = \frac{c(S, \overline{S})}{N(S)} = \frac{c(S, \overline{S})}{2} = \frac{|F_1| + |F_2|}{2} = \frac{m + |F_2|}{2}$$

   Therefore, $m \leq \frac{m + |F_2|}{2} \implies |F_2| \geq m$.

[1] Itai, A., 1978. Two-commodity flow. Journal of the ACM (JACM), 25(4), pp.596-611. We use Algorithm TWO-COMMODITY FLOW ALGORITHM and Lemma 2.3.

We showed that in both cases it holds that $|F_2| \geq m$ and we know that $|F_1| = m$.

We showed feasible flows F_1, F_2 from s_1, s_2 to t_1, t_2 s.t. $F_1, F_2 \geq m$, therefore due to the definition of maximum concurrent flow we get that $\boxed{\phi \geq m}$.

We showed $m \leq \phi$ and $\phi \leq m$ and therefore $\boxed{\phi = m}$, as required.

2. We use the following claim[2]: in the same settings, there are a flow f and a cut C where the size of the cut (denote as $|C|$) is at most twice the flow ($|f|$). As a result, for a minimum cut C^* and a maximum flow f^* it holds that $|C^*| \leq |C| \leq 2|f| \leq 2|f^*|$, as required. All the flows and cuts here are multi-commodity.

3. Let the set of all elements be denoted as U. Denote the value of the optimal set cover solution with OPT (namely the ILP version of the following LP). Denote $|U| = n$.

   Define the following LP problem:

$$\begin{aligned} &\min \sum_{s \in S} x_s \\ &s.t. \forall u \in U, \sum_{s \in S \wedge u \in s} x_s \geq 1 \\ &\forall s \in S, x_s \geq 0 \end{aligned}$$

   Denote the optimal solution to the LP problem as x^*. Given a solution of set cover denoted as C, we can define $x_s = \mathbb{1}_{s \in C}$. Notice that x is a feasible solution to the LP problem, and therefore $\sum_{s \in S} x_s^* = \sum_{s \in S} x_s = |C|$. Therefore, we get that $\sum_{s \in S} x_s^* \leq OPT$.

   Due to the optimality of x^*, it holds that $\forall s \in S,\ x_s^* \leq 1$ (else we can find a better feasible solution). Therefore, $\forall s \in S,\ 0 \leq x_s^* \leq 1$, therefore we can sample with probability x_s^*.

   We define the following algorithm:

Algorithm 1: approximation set cover

```
Solve the LP problem ans store the solution as x*.
Define C = ∅
for 1 ≤ i ≤ ⌈ln 4n⌉ do
    C_i = ∅
    for each s ∈ S, add s to C_i with probability x*_s
    C = C ∪ C_i
end
output C
```

   Let $u \in U$, denote $S_u = \{s \in S \mid u \in s\}$. Notice that $\sum_{s \in S_u} x_s^* \geq 1$ because x^* is a feasible solution.

$$\mathbb{P}(u \notin C_i) = \prod_{s \in S_u} (1 - x_s^*) \overset{1+x \leq e^x}{\leq} \prod_{s \in S_u} e^{-x_s^*} = e^{\sum_{s \in S_u} -x_s^*} \leq e^{-1} = \frac{1}{e}$$

   Therefore,

$$\mathbb{P}(u \notin C) = \mathbb{P}(\forall 1 \leq i \leq \lceil \ln 4n \rceil, u \notin C_i) \overset{\text{independent events}}{=} \prod_{i=1}^{\lceil \ln 4n \rceil} \mathbb{P}(u \notin C_i) \leq \prod_{i=1}^{\lceil \ln 4n \rceil} \frac{1}{e} = e^{-\lceil \ln 4n \rceil} \leq \frac{1}{4n}$$

[2] Garg, N., Vazirani, V.V. and Yannakakis, M., 1997. Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica, 18(1), pp.3-20. The claim is proved in section 5, using the algorithm in figure 3.

Therefore,

$$\mathbb{P}\left(\exists u \in U \text{ s.t. } u \notin C\right) \overset{\text{union bound}}{\leq} \sum_{u \in U} \mathbb{P}\left(u \notin C\right) \leq \sum_{u \in U} \frac{1}{4n} = \frac{n}{4n} = \frac{1}{4}$$

And notice that

$$\mathbb{E}\left[|C|\right] = \mathbb{E}\left[\left|\bigcup_{i=1}^{\lceil \ln 4n \rceil} C_i\right|\right] \leq \mathbb{E}\left[\sum_{i=1}^{\lceil \ln 4n \rceil} |C_i|\right] = \sum_{i=1}^{\lceil \ln 4n \rceil} \mathbb{E}\left[|C_i|\right] = \sum_{i=1}^{\lceil \ln 4n \rceil} \left(\sum_{s \in S} x_s^*\right) \leq \sum_{i=1}^{\lceil \ln 4n \rceil} OPT = \lceil \ln 4n \rceil \cdot OPT$$

Therefore,

$$\mathbb{P}\left(|C| \geq 4 \lceil \ln 4n \rceil \cdot OPT\right) \leq \mathbb{P}\left(|C| \geq 4 \cdot \mathbb{E}\left[|C|\right]\right) \overset{\text{Markov inequality}}{\leq} \frac{\mathbb{E}\left[|C|\right]}{4 \cdot \mathbb{E}\left[|C|\right]} = \frac{1}{4}$$

Therefore,

$$\begin{aligned}
&\mathbb{P}\left(|C| \geq 4 \lceil \ln 4n \rceil \cdot OPT \vee \exists u \in U \text{ s.t. } u \notin C\right) \\
&\overset{\text{union bound}}{\leq} \mathbb{P}\left(|C| \geq 4 \lceil \ln 4n \rceil \cdot OPT\right) + \mathbb{P}\left(\exists u \in U \text{ s.t. } u \notin C\right) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}
\end{aligned}$$

Notice that the algorithm noted above is polynomial in the input, because LP can be solved in polynomial time and the update takes $O(|U| \cdot |S|)$ time. So, we showed a polynomial probabilistic algorithm that finds a set cover with cardinality less than $4 \lceil \ln 4n \rceil \cdot OPT$ with probability at least $\frac{1}{2}$.

We can repeat the algorithm a constant amount of time and choose the best set cover found and get a polynomial probabilistic algorithm that finds a set cover with value less than $4 \lceil \ln 4n \rceil \cdot OPT$ with probability at least $1 - \frac{1}{e}$.

Therefore, we showed a polynomial probabilistic algorithm that approximates set cover with approximation $4 \lceil \ln 4n \rceil$ with probability $1 - \frac{1}{e}$, as required.

4. Let $S =: \{v \in V \mid d(v) > k\}$, $T := \{v \mid d(v) < k\}$. Then by the given $S \sqcup T = V$. Also, by the push-relabel algorithm $s \in S$, $t \in T$, so (S, T) is an s-t cut. Assume for contradiction that it is not a minimum cut. Let G_f be the residual network at that stage of the algorithm. By the assumption, there is an edge $(u, v) \in E(G_f)$ such that $u \in S$, $v \in T$. By the labeling's definition, $d(u) \leq d(v)+1$. $d(u) > k \neq d(v)$, so $d(v) > k$ as well, in contradiction to $v \in T$. Hence there are no S-T edges, and equivalently for any S-T edge (u, v) in G, $c_f(u, v) = 0$. By the algorithm's definition, it follows that the residual capacity of these edges will remain 0 until the algorithm stops. Therefore, for the resulting flow g we will still have that there are no S-T edges in G_g, which means that (S, T) is a minimum cut by the min-cut max-flow theorem.

Did you know? There are no Fields medalists with Erdos number of less than 2.