HW07

Jiaxin Li, Sunjun Gu

November 1, 2020

1 Question 1

Input: a flow network N=(V,E,c) in which every edge has capacity c=1; a given integer k.
Output: k edges deleted in N, making the maximum value of a flow in the remaining network is as small as possible.
Intuition: We want to reduce max flow by deleting edges. Therefore, we consider which kind of edges are inevitable to get the max flow. By strong duality, $\underset{flow f}{max}\, v(f) = \underset{st-cut(S,T)}{min} c(S,T)$; if we reduce the min-cut, the max flow would be reduced as well. By definition of min-cut,$c(S,T) \doteq \sum_{e \in S \times T} c(e)$. Thus, if we delete edge going $T \to S$, it would not affect the capacity of the cut. If we delete the edge going $S \to T$, the capacity of the st-cut would be reduced by 1 when delete one edge for each edge's capacity is 1. Therefore, we want to delete the edge going $S \to T$ as much as possible to reduce max flow.
Symbol Description: N=(V,E,c) is the given network.N_f is the residual network. N' is the graph with the k edges deleted. f is the max-flow of N, v(f) is the value of max-flow;c(S,T) is the sum of capacity on edges from S to T. k is the number of edges deleted in N;k' is the number of edges going S→T on the min-cut(S,T), which is Number($\forall e \in E \cap S \times T$). n is the total number of vertices in N,m is the total number of edges in N.

Algorithm 1 DeleteEdges

Input: a flow network N=(V,E,c) in which every edge has capacity 1; a given integer k.
Output: k edges deleted in N, making the maximum value of a flow in the remaining network is as small as possible.

```
procedure DeleteEdges(N = (V, E, c), k)
Find max-flow f of N                                  {▷ Find min-cut (S,T) in N}
Build the residual network N_f
Based on the N_f, do BFS starting from starting s, all reachable vertices will be included in S,
  and T= V\S. (S,T) is the the minimum s-t cut of G
k'← number of edges going S→T on the min-cut(S,T)
if k' < k then
    Choose all the e ∈ E ∩ S × T
    Choose k - k' edges from left edges on N randomly
else
    Choose k edges which e ∈ E ∩ S × T
end if
return  k selected edges for deleting
```

Proof of Correctness:
Given a black-box algorithm MaxFlow(runtime O(nm)), we can get the correct max flow of N.

Given the Theorem 1 on page 10 in the lecture notes, we know statement (1) is equivalent to statement (3).
(1)$\Longleftrightarrow$ (3)
So cut(S,T), where S is the set of vertices reachable from s in N_f searched by BFS, is an s-t cut. cut(S,T) furthermore satisfies the property that, for every edge e in E(S,T), f(e) = c(e), and for every dege e in E(T,S), f(e) = 0.
This implies $\underset{flow f}{max}\, v(f) = c(S,T)$.
By **Strong Duality**, $\underset{flow f}{max}\, v(f) = \underset{st-cut(S,T)}{min} c(S,T)$
$\Rightarrow c(S,T) = \underset{st-cut(S,T)}{min} c(S,T)$
So cut(S,T) is a min-cut of N.
Then, we want to prove that from line 4-10, under 2 situations, the algorithm outputs k edges will make the maximum value of a flow in the remaining network is as small as possible.

Case1: Line 5-7, when the number of edges going S$\rightarrow$T is less then k,we delete all of them, then delete the left k-k' edges from the left graph(Because they are not edges S$\rightarrow$T, they do not affect c(S,T)). Therefore, no S$\rightarrow$T edges in N', c(S,T) = 0.By weak duality $\underset{flow f}{max}\, v(f) \leq \underset{st-cut(S,T)}{min} c(S,T)$, therefore v(f)$\leq$c(S,T)=0,and v(f) could not be negative by definition. Therefore, v(f) of max-flow of N is 0 which is the smallest.

Case2: Line 8-9, when the number of edges going S$\rightarrow$T is greater or equal to k, we delete k edges from these k' edges (Because all the edge has capacity = 1, they have the same affect to c(S,T). Therefore, no matter which edges of the k' edges). The min-cut value would become c(S,T) = k'-k. Then, we want to show this the smallest st-cut capacity we can get.
For N, its min-cut is (S,T) with k' edges going S$\rightarrow$T in the min-cut, c(S,T) = k' which is the minimum capacity of all st-cut in N by definition.
If the k deleted edges are not all S to T edges on min-cut, and some other st-cut with capacity k" are also reduced, the capacity of that st-cut would be reduced by at most $k * 1$=k (Because the deleted edges may not be all S to T edges of this cut). The new capacity of this s-t cut is at least k"-k.
For such a deletion strategy, if the min-cut of the original network remains to be min-cut, the reduction of min-cut capacity is less than the reduction in our algorithm (k) because not all edges deleted are on the min-cut
Or in the other situation, some other s-t cut with original capacity k" becomes min-cut of the reduced network, the new min-cut capacity will be at least k"-k. In other words, the min-cut capacity of the new network is larger than k"-k, which implies that it is also larger than k'-k because k" must be larger than k'.
Therefore, under such deletion strategy (the deleted edges are not all S to T edges on min-cut), the new min-cut capacity is larger than k'-k.
the correctness of our algorithm is proved.

Proof of Time Complexity :
On pseudo-code line 1, by the lecture, block-box algorithm to find the max-flow takes O(nm) time. For line 2, by construction of the residual network N_f taught on lecture, we need to copy all the n nodes to N_f, and go over all m edges in N to compute forwards and backwards capacity in N_f, total time is O(n+m). For line 3, BFS search also takes O(n+m) time to find the min st-cut. For line 4, we need to count number of edges from S to T. Consider the use array data structure to store vertices, we need to loop over m edges (u,v) and check if $u \in S$ and $v \in T$ every time, and count the number of them. It takes totally O(nm) time.For line 5-6, we need to choose all edges going S$\rightarrow$T in the min-cut (S,T). Then choose k - k' edges from left edges on N randomly.Find edge operations are similar with line 4, whose runtime is O(nm). For line 7-8, it choose k edges going S$\rightarrow$T in the

min-cut (S,T) and does similar operations with line 6, therefore, runtime is O(nm) too. Add these run time together, the total run time is O(nm), which is polynomial in n and m.

2 Question 3

2.1 a:

Input: A given network G number with integer capacities
Output: A densest minimum s-t cut of G
Symbol Description: G is the given network with integer capacities. G' is the modified G.$G'_f$is the residual network of G'. f is the max-flow of G'. (S',T') IS the minimum s-t cut of G'. n is the total number of vertices in G,m is the total number of edges in G. c(S,T) is the sum of original capacity on edges from S to T, c'(S,T) is the sum of modified capacity on edges from S to T.
Algorithm:

1. Modify G to a new network G': decrease the capacity of each edge on G by $\frac{1}{m}$.
2. Call the MaxFlow algorithm on G' and get the max-flow f.
3. Build the residual network G'_f of G'
4. Based on the G'_f got from step 3, do BFS starting from s, all reachable vertices will be included in S', and T' = V\S'. (S',T') is the the densest minimum s-t cut of G

Proof of Correctness:
Given a black-box algorithm MaxFlow, we can get the correct max flow of the modified network G'. Given the Theorem 1 on page 10 in the lecture notes, we know statement (1) is equivalent to statement (3).
(1)$\Longleftrightarrow$ (3)
So cut(S',T'), where S' is the set of vertices reachable from s in G'_f searched by BFS in step 4, is an s-t cut. cut(S',T') furthermore satisfies the property that, for every edge e in E(S',T'), f(e) = c'(e), and for every dege e in E(T',S'), f(e) = 0.
This implies $\underset{flowf}{max}\, v(f) = c(S', T')$.
By **Strong Duality**, $\underset{flowf}{max}\, v(f) = \underset{st-cut(S,T)}{min} c(S,T)$
$\Rightarrow c(S', T') = \underset{st-cut(S,T)}{min} c(S,T)$
So cut(S',T') is a min-cut of G'.
We want to show (S',T') is a densest minimum s-t cut of G by proving the correctness of 2 statements:
1.Let $cut(S_{min}, T_{min})$ be a min-cut in the original network G. For an arbitrary cut $cut(S_{non-min}, T_{non-min})$ which is not min-cut in the original network G, $c'(S_{non-min}, T_{non-min}) > c'(S_{min}, T_{min})$

2. The min-cut in G with more S to T edges will have smaller capacity in G'

For 1: Suppose ($S_{non-min}$,$T_{non-min}$) is a s-t cut of G but not the min-cut with number of S to T edges($density_{non-min}$). The min-cut (S_{min}, T_{min}) of G with number of S to T edges($density_{min}$). Because $cut(S_{non-min}, Tnon-min)$ is not the min-cut of G, so the $c(S_{non-min}, T_{non-Min})$ is strictly larger than $c(S_{min}, T_{Min})$
Also, all the capacities are integers, which implying that

$$c(S_{non-min}, T_{non-min}) \geq c(S_{min}, T_{min}) + 1 \quad (1)$$

. By step 1, decrease the capacity of each edge on G by $\frac{1}{m}$. Every edge's capacity - $\frac{1}{m}$.

$$c'(S_{min}, T_{min}) = c(S_{min}, T_{min}) - \frac{density_{min}}{m} \tag{2}$$

$$c'(S_{non-min}, T_{non-min}) = c(S_{non-min}, T_{non-min}) - \frac{density_{non-min}}{m} \tag{3}$$

$$\begin{aligned} & c'(S_{non-min}, T_{non-min}) - c'(S_{min}, T_{min}) \\ = & (c(S_{non-min}, T_{non-min}) - c(S_{min}, T_{min})) - (\frac{density_{non-min}}{m} - \frac{density_{min}}{m}) \\ = & c(S_{non-min}, T_{non-min}) - c(S_{min}, T_{min}) - \frac{density_{non-min} - density_{min}}{m} \\ \geq & 1 - \frac{density_{non-min} - density_{min}}{m} > 0 \end{aligned}$$

Explanation for $1 - \frac{density_{non-min} - density_{min}}{m} > 0$:
if $density_{non-min} - density_{min} \leq 0$, $-\frac{density_{non-min} - density_{min}}{m} \geq 0$, therefore $1 - \frac{density_{non-min} - density_{min}}{m} \geq 1 > 0$
if $density_{non-min} - density_{min} > 0$, because m is the total number of edges of G and G', $m \geq density_{non-min} > density_{min} \geq 0$, therefore $0 \geq density_{non-min} - density_{min} < m$ ($density_{non-min} - density_{min} = m$ only when $density_{non-min} = m, density_{min} = 0$,but by the theory that when all capacities are integral, there must be max-flow as well as the min-cut. $density_{min}$ =!0), $0 \geq \frac{density_{non-min} - density_{min}}{m} < 1$. So $1 - \frac{density_{non-min} - density_{min}}{m} > 0$
Combine the above situations, $1 - \frac{density_{non-min} - density_{min}}{m} > 0$

For 2: By step 1, decrease the capacity of each edge on G by $\frac{1}{m}$. Every edge's capacity - $\frac{1}{m}$. Suppose $cut(S_1, T_1)$ is a min-cut of G with number of S1 to T1 edges $density_1$; $cut(S_2, T_2)$ is another min-cut of G with number of S2 to T2 edges $density_2$. Let $density_1 > density_2$

$$c'(S_1, T_1) = c(S_1, T_1) - \frac{density_1}{m} \tag{4}$$

$$c'(S_2, T_2) = c(S_2, T_2) - \frac{density_2}{m} \tag{5}$$

Because $cut(S_1, T_1)$ and $cut(S_2, T_2)$ are both min-cut of G, so $c(S_1, T_1) = c(S_2, T_2)$.

$$\Rightarrow c'(S_1, T_1) - c'(S_2, T_2) = \frac{density_2}{m} - \frac{density_1}{m} = \frac{density_2 - density_1}{m} < 0 \tag{6}$$

$$\Rightarrow c'(S_1, T_1) < c'(S_2, T_2) \tag{7}$$

Therefore, the densest min-cut will have the smallest capacity in G' among all min-cut.

Combine the 1 and 2 proof, statement 1 and statement2 are both correct, which implies that the densest min-cut will have smallest capacity in G' among all cuts. So The densest min-cut of G will be a min-cut of G'. We have already proved that $cut(S', T')$ is a min-cut of G', which means $cut(S', T')$ is the densest min-cut in G.

Proof of Time Complexity :
The algorithm is made up of 5 steps.For step 1, decreasing the capacity of each edge on G by $\frac{1}{E}$.It needs to go over the edge and doing standard arithmetic operations(constant time) on each edges,

therefore, runtime of step 1 is O(m) which is linear time. For step 2, it call the MaxFlow algorithm on G'. For step 3, by construction of the residual network taught on lecture, we need to copy all the n nodes to $G^{'}_{f}$, and go over all m edges in G' to compute forwards and backwards capacity in $G^{'}_{f}$, total time is O(n+m) which is linear time. For step 4, BFS search takes O(n+m) time to find the min st-cut of $G^{'}_{f}$ and bulid T' need to check each nodes with O(n). Step 4's also runs in linear time.Our algorithm runs in linear time outside of MaxFlow.

2.2 b:

Input: A given network G number with integer capacities
Output: Whether G contains a unique densest minimum s-t cut.
Symbol Description: Algorithm:

1. Modify G to a new network G': decrease the capacity of each edge on G by $\frac{1}{m}$.
2. Call the MaxFlow on G' to get the max-flow f
3. Build the residual network $G^{'}_{f}$ of G'
4. Based on the $G^{'}_{f}$ got from step 3, do BFS starting from s, all reachable vertices will be included in S', and T' = V\S'. (S',T') is one densest minimum s-t cut of G.
5. Reverse each edge's direction on $G^{'}_{f}$ to get the new graph G_{new}
6. Based on the G_{new} got from step 5, do BFS starting from t. Include all reachable vertices from t into a new set T_{new}
7. If $|T_{new}| = |T'|$, then the densest min-cut is unique, otherwise, there must be some other densest min-cut

Proof of Correctness:
3b algorithm is looking for all vertices that can be reached from t in G_{new}. So for every vertex v_t in T_{new}, there is at least one path from t to v_t in G_{new}. G_{new} is just reversing every edges in G'_f, so it implies that there is at least one path from v_t to t in G'_f.

Firstly, we will show that T_{new} must be a subset of T'.
This prove will be done by contradiction.
Assume T_{new} is not a subset of T', which means

$$\exists v \in T_{new}, v \notin T' \tag{8}$$

Because $T' = V \setminus S'$

$$v \notin T' \Rightarrow v \in S' \tag{9}$$

This implies there is at least one path from s to v in G'_f. Also $v \in T_{new}$, which means there is at least one path form v to t in G'_f.
So there will be an augmenting path in G'_f, which is contradict to f is a max flow of G'. (By Theorem 1 on page 10 in lecture notes). The paradox shown above prove that T_{new} must be a sutset of T'.
If $|T_{new}| < |T'|$, which means $T_{remain} = T' \setminus T_{new}$ is not empty, for each vertex v_{remain} in T_{remain}, there is no path from v_{remain} to t in G'_f
Let $S_{new} = S' \cup T_{remain}$.
What we want to prove is that $cut(S_{new}, T_{new})$ is also a min-cut of G', in other words, it's another densest min-cut of G.

This is equivalent to:

$$f(e) = c(e), e \in E(S_{new}, T_{new}) \tag{10}$$

$$f(e) = 0, e \in E(T_{new}, S_{new}) \tag{11}$$

Let r(e) be the capacity of edges e in the residual network G'_f, equation (8) and (9) are equivalent to:

$$r(e) = 0, e \in E(S_{new}, T_{new}) \tag{12}$$

$$r(e) = c(e), e \in E(T_{new}, S_{new}) \tag{13}$$

We have already known that in G'_f, v_{remain} doesn't have any path to t, which means v_{remain} doesn't have any path to any vertices in T_{new}. Because if there is a path from v_{remain} to a vertex in T_{new}, v_t, v_{remain} will also be able to reach t through v_t. Also, we know $cut(S', T')$ is a min-cut of G', so there is no path from vertices in S' to T' in G'_f neither. Because T_new is a subset of T', there is no path from S' to T_{new}. In summary, there is no path from S_{new} to T_{new} in G'_f, which implies equation (8) and (9). So $cut(S_{new}, T_{new})$ is another min-cut of G'.

If $|T_{new}| = |T'|$, and we know T_{new} is a subset of T', which means $T_{new} = T'$. So that $cut(S_{new}, T_{new})$ is exactly the same cut with $cut(S', T')$.
For any other cut $cut(S_a, T_a)$
There are 2 situations:
(1). $\exists v \in S', v \in T_a$:
Because v is reachable from s in G'_f, in G'_f,$\exists e = (u, v), r(e) > 0$, where $u \in S_a, v \in T_a$, which means e = (u,v) is an edge from S_a to T_a which is not full in G'. That means such cut is not a min-cut of G'. (2) $\exists v \in T', v \in S_a$:
Because v is reachable from t in G_{new}, in G_{new}, $\exists e = (u, v), u \in T_a$, the capacity of the edge is larger than 0. It is equivalent to: in G'_f, $\exists e = (v, u), r(e) > 0$, which means e = (v, u) is an edge from S_a to T_a which is not full in G'. That means such cut is not a min-cut in G'.
For a cut different from $cut(S', T')$, at least one situation above will exist. So the other cuts can not be min-cut.

So the min-cut of G' is unique. Referring to the proof in 3a, the densest min-cut of G must be the min-cut of G'. So the densest min-cut of G is also unique.

Proof of Time Complexity :
For step 1-4, it is the same as 3(a). They runs in linear time outside of MaxFlow. For step 5, reverse each edges' direction and copy all the nodes to the new graph takes O(m+n) time.For step 6, use BFS to search takes O(n+m) time. For step 7, we need to calculate the number of vertices on T_{new} and T',which takes O(n) time and then comparison takes constant time.Add all the step time together, the total run time is linear time outside of MaxFlow.