TDA251
Solutions for Assignment 2

Wang QuFei
900212-6952
qufei@student.chalmers.se

December 30, 2019

Excrcise 3

Let c denote the minimum total weight of set $F \subset E$ such that the removal of F disconnects s and t, let c' denote the capacity of a minimum s,t cut (A^*, B^*) in G'. We prove that $c = c'$ and the edges connect A^* and B^* constitute exactly the solution F of G.

- Fist, for the flow f that gives the minimum cut (A^*, B^*), we claim that for all opposite directed edges $e = (u, v)$ and $e' = (v, u)$, f can be modified such that either $f(e) = 0$ or $f(e') = 0$. For if $f(e) \neq 0$, $f(e') \neq 0$, let δ be the smaller of the two values, and modify f by decreasing the flow value on both e and e' by δ. The resulting flow f' is feasible, has the same value as f, and the capacity condition and conservation condition on vertex u and v still hold. We thus assume that for all opposite directed edges, the flow value of one of them must be zero under f.
- Second, for all undirected edges that connect A^* and B^* in G with a weight $w > 0$, we claim that flow f in G' gives directed edge $e = (u, v)$ flow value $f(e) = w$, gives directed edge $e' = (v, u)$ flow value $f(e') = 0$, where $u \in A^*$, $v \in B^*$. For otherwise there would be a s, v path in the residual graph of G' which contradicts our assumption that $v \in B^*$.
- We say $c \leq c'$. For from

$$c' = \sum_{\substack{e \, out \, of \, A^* \\ e \in G'}} f(e) = \sum_{\substack{e \, connects \, A^*, B^* \\ e \in G}} w(e)$$

we know that c' is the total weight of a set of edges in G, with the removal of which we can disconnect s and t in G, whereas c is by definition the total weight of edges in G with the same property with minimum value.

- We say $c' \leq c$. Assume $F \subset E$ is the set of edges that yields minimum total weight c. Denote A' as the set of nodes reachable from s in G after the removal of F, $B' = V \setminus A'$. Clearly, (A', B') forms a cut in G', and for each of these edges $e = (u, v) \in F, u \in A', v \in B'$, we assign both directed edges $e_1 = (u, v), e_2 = (v, u)$ in G' with flow value $w(e)$, assign other edges in G' with flow value 0. This satisfies capacity condition and conservation condition for every node in G', and the capacity of cut (A', B')

$$c(A', B') = \sum_{\substack{e \, out \, of \, A' \\ e \in G'}} f(e) = \sum_{\substack{e \, connects \, A', B' \\ e \in G}} w(e) = c$$

Because c' is by definition the value of minimum cut of G', so we have $c' \leq c(A', B') = c$.

□

Excrcise 4

4.1 Denote the size of F as $|F| = n$. Considering only the free squares in F, we now reduce this problem to Maximum Flow problem as follows.

- Each square represents a node.
- Add two nodes s and t as the source and the sink.
- Add directed edges from s to each white square with capacity 1.
- Add directed edges from each white square to all of its neighboured(N-S or W-E direction) black squares with capacity 1.
- Add directed edges from each black square to node t with capacity 1.
- Denote $G = (V, E)$ as the directed flow network with the nodes and edges stated above.

We claim that a value k of maximum flow in G exists if and only if an optimal solution of k boxes exists in the storage hall problem.

- Let k and k' respectively be an optimal solution to the maximum flow problem and the storage hall problem.

- Each edge from graph G has capacity 1, a flow of value k means k pairwise disjoint paths from node s to node t. Each path covers a pair of nodes, which can be seen as a box placed on this pair of nodes. Thus we have a solution of k boxes for the storage hall problem. Since k' is the optimal value, we have $k \leq k'$.
- For the k' boxes that placed on the set F of squares, each box can be viewd as an edge connecting a white node and a black node with flow of value 1. Since two boxes can not overlap, we have k' pairwise disjoint paths from node s to node t with a flow of value k'. Since k is the maximum value of flow, we have $k' \leq k$.
- We proved that $k = k'$

Running *Ford-Fulkerson Algorithm* in G costs $O(mC)$ time, where m represents the number of edges and C represents sum of the capacity of the edges leaving s. For the set F with n free squares, m has size $O(n)$, C has size $O(n)$, thus time bound of this problem would be $O(n^2)$ □

4.2 Suppose f is a max-flow we get from 4.1 by running the *Ford-Fulkerson Algorithm*. Denote G_f as the residual graph with respect to f, we know that there's no s-t path in G_f.
Now, one square is added to F. Add node e_n in G_f as the representation of this newly freed square. By the way of reduction in 4.1, if this new square is white, we need to add one directed edge from s to e_n, and at most 4 directed edges from e_n to the nodes which represent the neighboured black squares. Otherwise, we need to add one directed edge from e_n to t, and at most 4 directed edges from the nodes which represent the neighboured white squares to e_n. The capacity of each of these newly added edges is 1, and the flow value on these edges is 0. Denote the graph with this new node and edges as G'. A more efficient method can be given as follows, it searches all s-t paths in the G', the number of these possible s-t pahts in G' is the extra flow of value that can be added, which is also the number of extra boxes that be added for the original problem.

Initialization:

- A boolean value *complete* signifies whether a s-t path $p_{s,t}$ has been found, initialize it with 0.
- Initialize stack S with e_n as its top element.
- A boolean value *direction*, when set 1, means we are constructing $p_{s,t}$ forwardly from s to t, otherwise, we are constructing $p_{s,t}$ backwardly from t to s.
- If e_n is white, set *direction* to 1, since we already have a partial s-t path $s \to e_n \to \ldots$, what we need next is to complete the construction of this path forwardly. Mark s and e_n as *explored*.

- If e_n is black, set *direction* to 0, since we already have a partial s-t path $\cdots \to e_n \to t$, what we need next is to complete the construction of this path backwardly. Mark t and e_n as *explored*.

while S not empty **and** *complete* == 0 **do**
  $e = pop(S)$;
  if *direction* == 1 **then**
    if $e == t$ **then**
      assign *complete* = 1;
    else
      for all unexplored nodes n in G' for which (e, n) is a directed edge **do**
        mark n as *explored*;
        push n to S;
      end for
    end if
  else
    if $e == s$ **then**
      assign *complete* = 1;
    else
      for all unexplored nodes n in G' for which (n, e) is a directed edge **do**
        mark n as *explored*;
        push n to S;
      end for
    end if
  end if
end while
return *complete*

If this method returns 1, then we conclude that a better solution exists and one more box can be placed. Otherwise there isn't a better solution. Since each node is examined only once, with some constant time searching for its neighbours when using adjacency list as the representation of G', the time bound of this method is $O(n)$.

Next we prove that at most one more box can be inserted. The procedure described above ends when it finds one s-t path, which can be denoted as δ. By setting the flow value on all the edges in δ to 1, we get a flow f' which yields one more value than f, which means we can at least add one more box for the original problem.

Suppose there are situations where more than one boxes can be inserted. Because the capacity of all the edges in G' is 1, this means that there must be another s-t path δ'. We claim that δ' can not pass e_n, otherwise if e_n represents a white square, there's only one edge from s to e_n; If e_n represents a black square, there's only one edge from e_n to t. On both cases it would mean δ and δ' share an edge from s or into t, which is impossible due to the fact that the

capacity of each edge is 1.This means δ' is indenpent of e_n, which means δ' exists in G_f. This contradicts to the fact that there's no s-t path in G_f. Thus we have proved that at most one box can be added if a better solution exists.

□