Algorithm Design Assignment 6

Jianan Lin, 662024667, linj21@rpi.edu

Problem 7.16. There are n users and m advertisers. There are k groups called G and each user belongs to one or several groups, and each advertiser belongs to one or several groups. Each advertiser i has a requirement of at least r_i ads per minute and its ads can only be sent to users with same group(s). A user can receive at most 1 ad per minute.

Write an algorithm to judge whether we can achieve this requirement and if we can, then output the specific allocation between ads and users.

Solution. Our algorithm has three parts.

The first part is to construct a directed graph. We have a super source node s and super sink / destination node t. Besides, we have two sets of nodes called A and B which are also in the graph, and A includes all the users, B includes all the advertisers. For all $a_j \in A$, there is a directed edge (s, a_j) with capacity 1, and for all $b_i \in B$, there is a directed edge (b_i, t) with capacity r_i. If there is some group G_x that both user j and advertiser i belong to, then there is a directed edge (a_j, b_i) with capacity 1.

To judge whether we should connect a_j and b_i, we should consider the method. There are at most k groups and we regard the list of groups "that a user / advertiser belongs to" as a hash map. We know to judge whether an element belongs to a hash map costs $O(1)$ time. So it takes $O(k)$ to judge whether the two hash maps (one is the user and the other is the advertiser) have intersections, because we can select each element from hash map 1 and judge whether it belongs to hash map 2. Therefore we know the time complextiy of this part is $O(m + n + mnk) = O(mnk)$. Notice there are at most $m + n + mn$ edges and $m + n + 2$ nodes.

Why should we construct such a graph? Since a user can receive at most 1 advertisement per minute, the capacity of the edge connected a user should be 1. Since the goal of advertisement i is to have at least r_i ads per minute, the capacity of the edge connected an advertiser and t should be r_i. Therefore, it is easy to find that, we can achieve the requirement if and only if there is a feasible circulation with demands in this graph. Here, the demand of sink t is $R = \sum_i r_i$ and the demand of source s is $-R$. The demand of any other node is 0.

The second part is to find the maximum $s-t$ flow in this graph. There are many different algorithms and we select the Ford-Fulkerson algorithm. The time complexity of this algorithm is $O(EF)$ where E is the number of edges and F is the max $s - t$ flow. Since we know $E \leq mn + m + n$ and $F \leq \sum_i r_i$ (because the in-degree of t is at most $\sum_i r_i$, called $R = \sum_i r_i$), we know the time complexity of this part is $O(mnR)$.

The third part is to compare the max $s - t$ flow (called F) and the sum of demand $\sum_i r_i$ (called R). If $F = R$, then we can achieve this according to the theorem 7.50 in the textbook, i.e., there is a feasible circulation with demands $d_t = R, d_s = -R$ in the graph, if and only if the maximum $s - t$ flow is R. And we print each user-advertiser pair if there is a flow between the user and advertiser. If $F \neq R$, then we cannot achieve it. The time complexity of this part is $O(mn)$.

According to the complexity of each part, the time complexity of this algorithm is $O(mn(k + R))$.

Problem 7.28. There are n TAs and k time slots. Each TA is available in some time slots. Each time slot can have at most 1 TA. Each TA should work at least a time slots per week and at most b time slots. We should have c time slots totally exactly.

(1). Write a polynomial algorithm to judge whether we can achieve this assignment. If we can, then output the time schedule.

(2). We add an extra restriction. In each day, d_i denotes the least time slots needed in day i. Also write a polynomial algorithm to achieve the same goal.

Solution.

(1).

Our algorithm has three parts. Obviously if $c < an$ or $c > bn$ or $k < c$, then we can never achieve this. Therefore we assume $an \leq c \leq b$ and $c \leq k$, otherwise this algorithm outputs "False" directly.

The first part is to construct two graphs G and G' with the same nodes and edges (but different capacity, demands and lower bounds). In fact, the algorithm only needs G' but G is helpful to our analysis. There is a source node s and super sink node t in G and G' respectively. We use two sets of nodes A and B which are also in G and G'. A includes all the TAs and B includes all the time slots.

For any $a_i \in A$, we have a directed edge (s, a_i) with capacity b in G and capacity $b - a$ in G'. Besides, there is a lower bound a for each (s, a_i) in G. For any $b_j \in B$, we have a directed edge (b_j, t) with capacity 1 both in G and G'. For any $a_i \in A, b_j \in B$, there is a directed edge (a_i, b_j) with capacity 1 both in G and G' if and only if TA i is available in time slot j. The time complexity in this part is $O(nk)$ because there are at most $n + k + 2$ nodes and $nk + n + k$ edges, and it only takes $O(1)$ time to judge whether TA i is available in time slot j.

Why should we construct these two graphs? First we see graph G. Since each TA has a lower bound a and an upper bound b of number of time slots per week, each edge (s, a_i) should have a lower bound a and capacity b. Since each time slot can contain at most 1 TA, the capacity of edges connected a time slot should be 1 (and no lower bound). So we can achieve our goal to have an assignment if and only if, there is a feasible circulation with demands in this graph. Here, the demand of sink t is c and the demand of source s is $-c$. The demand of any other node is 0.

Also notice G'. This time the demand changes. The demand of s is $an - c \leq 0$, the demand of $a_i \in A$ is $-a$, the demand of t is c and the demand of others is 0. According to theorem 7.52 in textbook, there is a feasible circulation in G if and only if there is a feasible circulation in G'. In this way, we should also add a super source node s^* for G' and for any node $v \in A$, there is a directed edge (s^*, a_i) with capacity a and there is an extra directed edge (s^*, s) with capacity $c - an$. So we only need to find the maximum $s^* - t$ flow in G'.

The second part is to find the max $s^* - t$ flow in G'. We still use Ford-Fulkerson algorithm, which costs $O(EF)$ time where E is the number of edges and F is the max flow. We know $E \leq nk + n + k$ and $F \leq c$ (because the out-degree of s^* is at most c). Therefore the time complexity for this part is $O(nkc)$.

The third part is to compare the max $s^* - t$ flow F and c. If $F = c$, then we can achieve this goal according to theorem 7.50 in the textbook, since there is a feasible circulation if and only if the max $s^* - t$ flow is c. Then we output each TA and time slot pair if there is a flow between them. If $F \neq c$, then obviously we output "False", which means we cannot achieve the goal. The time complexity of this part is $O(nk)$.

Therefore the total time complexity of our algorithm is $O(nkc)$.

(2).

Suppose there are m work days per week and in day i, there are x_i time slots and at least d_i needed. In fact we can regard m as a constant, but here we firstly use it as a variable (especially during the time complexity discussion) and then point out that it can be omitted even if it is not a constant. Also our algorithm has three parts. Of course besides $an \leq c \leq bn$ and $k \geq c$, we have to add two new

restrictions $\forall i, d_i \le x_i$ and $c \ge \sum_i d_i$. Otherwise we output "False" directly.

The first step. We still construct two graphs G and G'. They are similar to (1) but there are some differences. Since the time complexity analysis and circulation analysis are the same, we will use short language this time. Still, G and G' have the same nodes and edges. There is a source s, a super sink t. We have three sets of nodes A, B, C. A includes all the TAs, B includes all the time slots, and C includes all the days.

For all $a_i \in A$, we have directed edge (s, a_i) with capacity b and lower bound a in G, and capacity $b - a$ in G'. For all $a_i \in A, b_j \in B$, we have directed edge (a_i, b_j) with capacity 1 both in G and G' if and only if TA i is available in time slot j. For all $b_j \in B, c_i \in C$, we have directed edge (b_j, c_i) with capacity 1 both in G and G' if and only if time slot j belongs to day i. For all $c_i \in C$, we have directed edge (c_i, t) with capacity x_i and lower bound d_i in G, and capacity $x_i - d_i$ in G'.

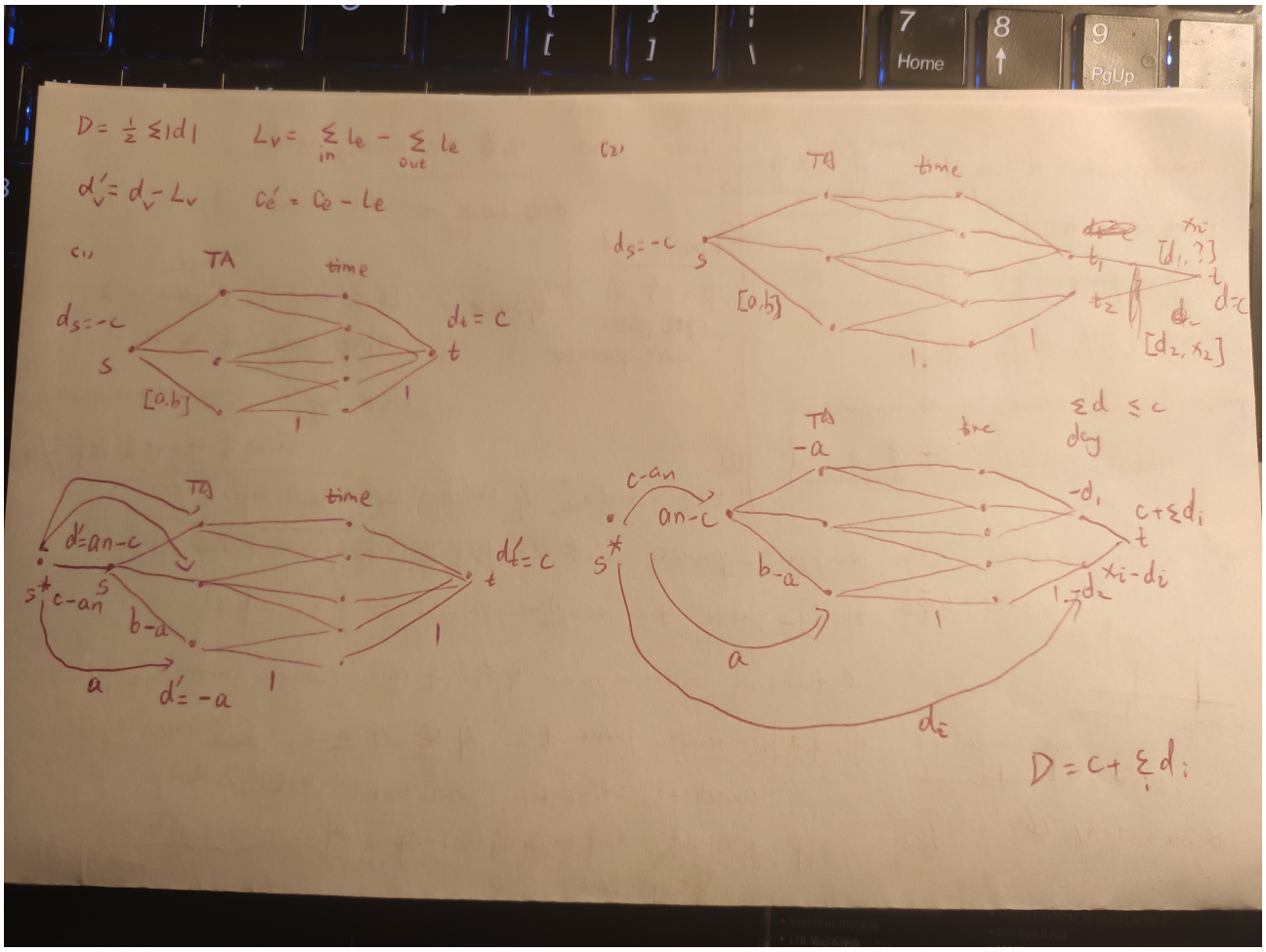
Why should we construct these two graphs? First we see graph G. We can achieve our goal if and only if there is a feasible circulation in G with demand of s equal to $-c$ and demand of t equal to c. Then we see G'. Let demand of s be $an - c \le 0$, demand of $a_i \in A$ be $-a$, demand of $c_i \in C$ be $-d_i$ and demand of t be $c + \sum_i d_i$.

Therefore we should add another super source node s^* to G'. We let edge (s^*, s) have capacity $c - an$, let any edge (s^*, a_i) have capacity a (here $a_i \in A$), and let any edge (s^*, c_i) have capacity d_i (here $c_i \in C$). So according to the theorem and previous results, there is a feasible circulation if and only if the max $s^* - t$ flow is $c + \sum_i d_i$. The time complexity of this part is $O(kn + m) = O(nk)$. Here we should notice that, a time slot can only belong to one day, so the number of edges between B and C is always k. Also notice that $m \le k$, so $O(kn + m) \le O(kn + k) = O(kn)$.

The second step. We use Ford-Fulkerson algorithm to find the max $s^* - t$ flow in G'. This part takes $O(EF)$. Notice $E \le O(nk)$ and $F \le c + \sum_i d_i$ because the out-degree of s^* is at most $F \le c + \sum_i d_i$. Therefore we know the time complexity is $O(kn(c + \sum_i d_i))$.

The third step. If max flow $F = c + \sum_i d_i$, then we can achieve our goal. We only need to output each pair of TA and time slot if there is a flow between them. This costs $O(nk)$. So the total time complexity of this algorithm is $O(kn(c + \sum_i d_i))$.

Last we provide a picture about the graph in this question in order to better understand. The left two are for question (1) and the right are question (2). The up is G and the bottom is G'.


D = ½ Σ|d|
L_v = Σ_in l_e − Σ_out l_e
d'_v = d_v − L_v
c'_e = c_e − l_e
(1)
TA
time
d_s = −c
s
[a,b]
d_t = c
t
d'=an−c
s*
c−an
b−a
a
d'=−a
d'_t = c
(2)
d_s = −c
[a,b]
x_i
[d_1, ?]
[d_2, x_2]
d=c
-a
c−an
an−c
-d_1
c+Σd_i
b−a
x_i − d_i
-d_2
d_i
Σd ≤ c
D = c + Σd_i