1 Book 7.28: TA Scheduling

1.1 Mathematical Formulation

(a) Given an input of T TA's, S sessions, and parameters a, b, and c where the number of sessions each TA can hold a week is t_i, $a \leq t_i \leq b, \forall i \in 1..T$ and the total number of seesions that can be held a week must be $\leq c$. Determine which TAs cover which sessions if all requirements are met.

(b) Given the above information, add in a density d_i representing the minimum number of sessions that must be held that day of the week. Again determine which TAs cover which sessions if all requirements are met.

1.2 Solution

In both cases we will begin by determining if the number of TA's T can acceed the minimum capacity c by checking $c \geq T \cdot a$. We then construct a circulation digraph max flow representation with $2 + T + S + 2$ nodes: 1 source, 1 virtual source, T TA, S session, 1 virtual sink, and 1 sink node. To ensure that the minimum capacity for each TA is met, we connect the source to each TA node with capacity a, then, to account for this we will connect the source to the virtual source with with a capacity $c - T \cdot a$ and the virtual source will connect to each of the TA nodes with a capacity of $b - a$ so that the total sum of in-capacities for each TA node is $a + b - a = b$. Therefore our total out-degree from the source node will be $T \cdot a + (c - T \cdot a) = c$. Now, according to availablility of the TA's, we connect them to the corresponding session nodes with a capacity of 1. This is the same build for both parts (a) and (b).

For part **(a)**, we will assign edges from each of the sesssion nodes to the virtual sink with capacity 1 to represent that each session can only have 1 TA. Then to ensure that the cycle is complete and that the maximum number of sessions is $\leq c$, we connect the virtual sink with a capacity of c to the sink.

For part **(b)**, we will add in 7 extra nodes, each representing a day with a correspoding value of $d_i, i \in 1..7$. We continue our connections by connecting each seesion, to their correspoding day with a capacity of 1 to represent that each session can only have 1 TA. The for each day, i, we connect it directly to the sink with a capacity of d_i to represent the minimum density that must be met. Next, we connect it to the virtual sink with a capacity of ∞ since they have met the minimum requitrement of d_i. Now we just connect the virtual sink to the sink with a capacity of $c - \sum_{i=1}^{7} d_i$ such that the total in-capacity of the sink is $\sum_{i=1}^{7} d_i + c - \sum_{i=1}^{7} d_i = c$.

Now that we are fully connected we run Ford Fulkerson and if the max flow is $< T \cdot a$ then we have not fufilled the required minimum flow. Otherwise we go through

each TA node and check its edges, if the edge has a flow value, then print the TA and the Session information.

For Purposes of simplicity we will show the algorithm for **(b)** as it is part (a) with an additional constaint.

Algorithm 1 Build Network (part b)

```
procedure CONNECT(out, in, capacity)
procedure MAIN(T, S, a, b, c)
    if c ≥ T · a then Not Possible To Compute
    G ← initialize nodes
    for each TA node t ∈ 1..T do
        G.CONNECT(source, t, a)
        G.CONNECT(cSource, t, (b-a))
        for each session, s ∈ 1..S, that t can lead do
            G.CONNECT(t, s, 1)
    CONNECT(source, vSource, c − T · a)
    for each session, s ∈ 1..S do
        d ← day that s occurs on
        G.CONNECT(s, d, 1)
    for each day, i do
        d_i ← minimum sessions day i must have
        G.CONNECT(i, sink, d_i)
        G.CONNECT(i, vSink, ∞)
    G.CONNECT(vSink, sink, c − Σ_{i=1}^{7} d_i)
    G.MAXFLOW( )
    for edge, e, from source to not vSource do
        if e.flowValue() != a then Not Possible To Compute
    for each day, i do
        d_i ← minimum sessions day i must have
        edge, e, from i to sink
        if e.flowValue() != d_i then Not Possible To Compute
    for each TA node t ∈ 1..T do
        for edge, e, from t do
            if e.flowValue() == 1 then
                session s ← e.to()
                Print t and s
```

1.3 Correctness

Proposition 1.
This algorithm will determine (if one exists) a valid schedule for office hours, specifying which TA will cover which time slots. If one does not exist, then it will report so.

Proof.
Above we have described how to construct the flow network in detail. As so we show now that this supports the constraints set forth by the question. The first constraint is that each TA must have a minimum of a office hours and a maximum of b each week. It is simple to upper bound each TA by b by ensuring that the sum of the maximum capacity of all edges coming into each TA is equal to b. Therefore the only thing we need to constrain is that we have at least a for each TA. We acheive this by making an edge of maximum capacity a between the source and each TA, which will be filled out first. Similarly we will have a "whatever we have left" node which we call the virtual source. The connections from the virtual source, we connect to each TA node with edges with max capacity of $b - a$ to ensure that each node has and input of b.

Now to account for the second constraint that at most c office hours may occur each week. We do this by making the maximum capacity of all edges leaving the source and entering the sink to be c. Therefore (since $T \cdot a$ has already left the source going to each TA), we connect the source to the virtual source with a value of $c - T \cdot a$.

The last constraint of the problem is that each day i have a specific density d_i that acts as a minimum value. To account for this, we have each Session connect to its respective day with a max capacity of 1 so that only one TA can be present at each session. Then from each day, we will connect directly to the sink with maximum capacity d_i therefore we need to adjust our edge capacity from the virtual sink to the sink to be $c - \sum d_i$ and connect each day to the virtual sink with max capacity ∞.

Now we have shown that each constraint has been accounted for, but to ensure that each one has held up, we simply will check the flow values of each of the edges from the source to the TA nodes are full and those from each day to the sink are full. □

1.4 Analysis

For this section we will use the following variables: T for number of TAs, S for number of sessions, C for number of connections between TAs and Sessions, $V = T + S + 7 + 4$ for total number of nodes and $E = 2 \cdot T + C + 2 \cdot S$ for total number of edges. We say then that we can make our calculations based off V + E.

Proposition 2. *The space complexity of this algorithm is* ***O(V+E)***

Proof.

This is due to the fact that we only store the nodes and the connections between them in a datastructure and the rest is stored in constant variables. □

Proposition 3. *The time complexity of this algorithm is* ***O(T·S + MaxFlowValue·E + C)***

Proof.

We can break this down into the loops which we have described in our algorithm:

1. TA Node Connection: we go through each TA node (T) and connect it to the source and virtual source (2) and then go through every session and see if it is a valid time for each TA to work (S) $\implies T \cdot (2 + S) \implies O(T \cdot S)$

2. Session Node: we have already connected each session to its corresponding TAs, now we just connect each one (S) to the specific day that it belongs to (1) $\implies O(S)$

3. Days: when we have to connect each day to the sink and virtual sink $(7 \cdot 2)$ which is considered arbitrary in the large cases $\implies O(1)$

4. MAX FLOW: Known to be $O(MaxFlowValue \cdot E)$

5. Check TAs and days: we check that each TA and day have met the minimum requirements $\implies O(T)$

6. Print out: Go through every edge going from TA to Session nodes $\implies O(C)$

Thus summing to be $T \cdot S + S + 1 + MaxFlowValue \cdot E + T + C$ which in Big-O becomes **O(T·S + MaxFlowValue·E + C)** □