1. The nodes which are reached by fanning out are called labelled nodes. The others are unlabelled nodes. We use labelling algorithm in this case as, We know a non-integer maximum flow in a directed network with integer arc capacities. We need an algorithm for converting this flow into an integer maximum flow.

Labelling algorithm labels a node by going through the adjacency list of a particular chosen node. Then the algorithm sends the maximum possible flow on the path from node s to t. The nodes are discarded and the process is repeated.

Algorithm ⇒

Labeling Algorithm:

begin:
  label node t:
  While (node t is labeled):
  do:
    unlabel all nodes
    set prev[j] = 0 for each $j \in N$.
    Label node s and set adjacency list = (s).
    While (List is not empty) or (t is unlabeled):
    do:
      remove a node from list.
      for each arc (i,j) in residual network from node i :

do:

if $r(i,j) > 0$ and node j is unlabeled then

set prev[j] = i

Label node j

Add j to List.

if t is labeled:

then augment

Augment:

begin:

use the predecessor labels to trace back from the sink to the source.

Obtain augment path P from $s \to t$.

$\delta = \min(r(i,j), EP)$

augment δ units of flow along P and update the residual capacities.

end.

→ In a capacitated network, the maximum value of flow from s (source node) to t (sink node) is equal to the minimum residual capacity among all s-t cuts. (Max Flow Min Cut)

→ If the residual network does not contain augmenting path, ~~only~~ then the flow 'f' is maximum. (Augmenting Path Theorem).

→ The maximum flow problem has an integer result (max flow) if all arc capacities are integer. (Integrality Theorem).

The above labeling algorithm has a running time complexity of $O(nmU)$.

2. Let us illustrate this problem as a maximum flow problem with the help of an example where $M=3$.
We rank all the release and due dates (r_j, d_j for all jobs) in ascending order.
We calculate $P \leq 2|J|-1$; mutually disjoint intervals of intervals of dates between consecutive milestones.
Let,
$T_{k,l}$ ← interval that starts at beginning date k and ends at a date $l+1$.
For the example under consideration let us consider order of release and due dates as 1, 3, 4, 5, 7, 9. We have 5 intervals ($T_{1,2}$, $T_{3,3}$, $T_{4,4}$, $T_{5,6}$, $T_{7,8}$).

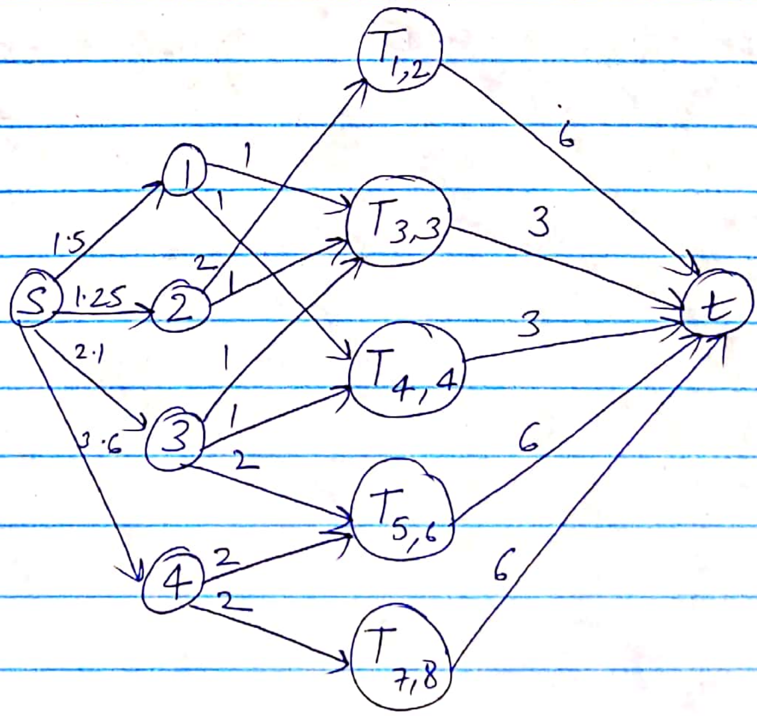
We can schedule the given problem as a maximum flow problem on a bipartite network as follows ⇒
s ← source node ; t ← sink node
The source node is connected to every job j with an arc of capacity p_j.
p_j ← we need to assign p_j days to j.
The sink node is connected to each interval node with an arc of capacity $(l-k+1)M$. This means ~~that~~ the total no. of days available on days from k to l.
We connect a job node j to every interval node $T_{k,l}$ if $r_j \leq k$ and $l+1 \leq d_j$ with an arc of

capacity $(l-k+1)$. This indicates the maximum number of days we can allot to job j on days from k to l.

The scheduling problem has a feasible schedule iff maximum flow value $= \sum_{j \in J} p_j$.

We can validate this by representing a one-to-one correspondence between feasible schedules and flows of value $\sum_{j \in J} p_j$ from the source node s to the sink node t.

j	1	2	3	4
p_j	1.5	1.25	2.1	3.6
r_j	3	1	3	5
d_j	5	4	7	9



3. Let $i \leftarrow$ each leg flight
Start $i \leftarrow$ starting for i
end $i \leftarrow$ ending for i

We use an arc with lower bound 1 to connect start i and end i. This indicates that atleast 1 plane is required to complete this flight leg.
If the same plane has to be used for both ~~legs~~ flights we connect two legs.
if $b_i + r_{ij} \leq a_j$ then End_i —— $Start_j$ (connect).

Now, we create a supernode (super-source) and connect it to the start of each leg and also create a super-sink node and connect it to the end of each leg. This way we can find out about putting a plane out of service or into service.
We can do the following steps now, to achieve our goal.

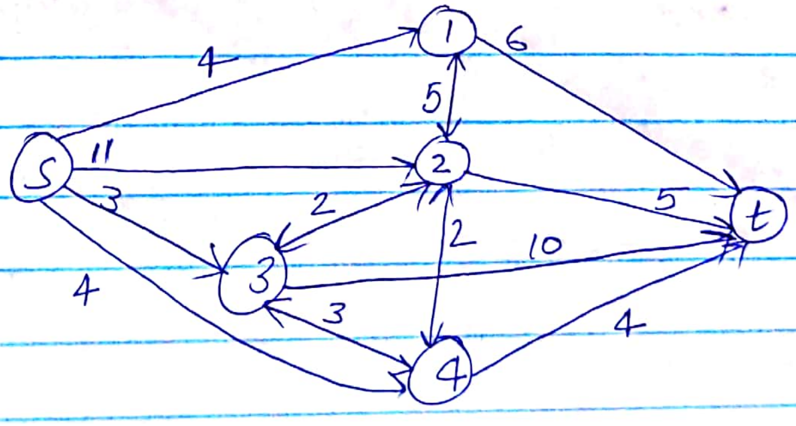
a) Find a feasible flow
b) Push as much flow back from the super-sink to the super-source. Find a flow that uses minimum number of planes.

Hence, the problem has been formulated as minimum flow problem.

4. The given problem is a minimum-cut problem
Let
$s \leftarrow$ source node (processor 1)
$t \leftarrow$ sink node (procesor 2)
node $i = 1, \ldots n$; nodes for the modules of the prog.
We include an arc (s,i) with capacity $u_{si} = \beta_i$ and $u_{it} = \alpha_i$ for every node $i \neq s$ and $i \neq t$.
During the execution, if module i interacts with j we include arc (i,j) and (j,i) with capacities $u_{ij} = u_{ji} = c_{ij}$

Let us consider an example with 4 modules, interprocessor costs $\cancel{c_{12} = 8}$, $C_{12} = 5$, $C_{23} = C_{24} = 2$, $C_{34} = 3$, $c_{ij} = 0$ for all others.
Processing costs $\Rightarrow$

i	1	2	3	4
α_i	6	5	10	4
β_i	4	11	3	4

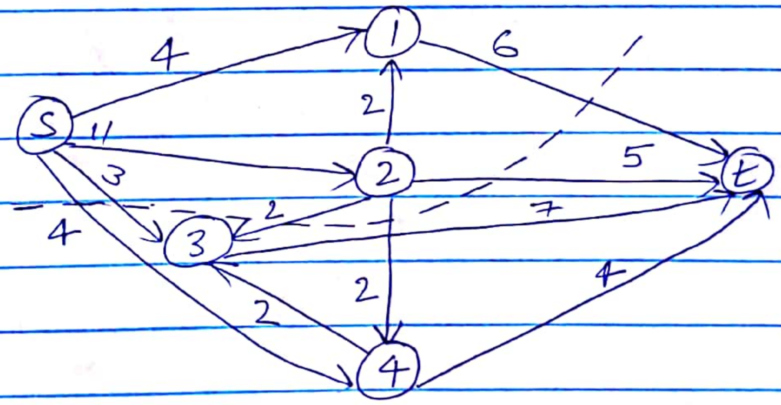


This fig represents the arc capacities

Let A_1 and A_2 be assignments of modules to processors 1 and 2.
Then, $[\{S\} \cup A_1, \{t\} \cup A_2] \leftarrow$ s-t cut and this expression has a value equal to assigment costs which are calculated using the formula

$$\sum_{i \in A_1} \alpha_i + \sum_{i \in A_2} \beta_i + \sum_{(i,j) \in A_1 \times A_2} c_{ij}$$

The maximum flow and minimum cut are represented as follows $\Rightarrow$ $4 + 11 + 3 + 4 = 22$ — ①



The value ① is equal to the capacity $4+3+2+2+5+6 = 22$. of the s-t cut. $[\{S, 1, 2\}, \{t, 3, 4\}]$.
This cut defines an optimal solution to the problem. Therefore, the modules of the program are allocated to processor 1 and 2 such that the total cost of processing and interprocessing communication is minimized.

5. We need to block all communications between the commander and his subordinates with minimal efforts.
In order to solve this problem in polynomial time we use Bellman Ford Algorithm. This algorithm will give us the shortest path from a single source p and all other vertices in the set S. We can then calculate the effort of each path this way and choose the minimum.

2nd Approach.

5. One possible way to solve this problem would be to add one super-source and a super-sink to the graph. The super-source is connected to the commander's node p. The arc connecting the supernode and the commanders node is given a capacity of infinity. Similarly, we connect the super-sink node to the subordinates node with an arc which has a capacity of infinity.
The minimum cut between the super-source and the super-sink will give us the minimum effort edges. The infinity capacity arcs we have added to the problem would never cross the cut which results in the commander and subordinate on the different sides of the cut. This way we determine the minimal effort required to block all communications in polynomial time.

Scanned with CamScanner