COMP 3010 Assignment 4

Ameer Karas

44948956

Problem A

One can argue that the nodes wi1 and wi2 representing the pet types are not required in above flow network because we can directly link the pets to the human. Do you think these nodes are necessary? Explain your answer and then give an example to support your answer.

a) These nodes are necessary to ensure that a person receives the correct amount of pets that they are willing to take; the ‘W’ nodes regulate how many pets a person can take.
   For example, in figure 1 we can observe R2. R2 wants only one pet and it must be a cat. If W21 is removed and there is a path from P1 and P2 to R2, each with a weight of 1, then it would be valid for R2 to take both P1 and P2 as the capacity from R2 to the sink is 3.

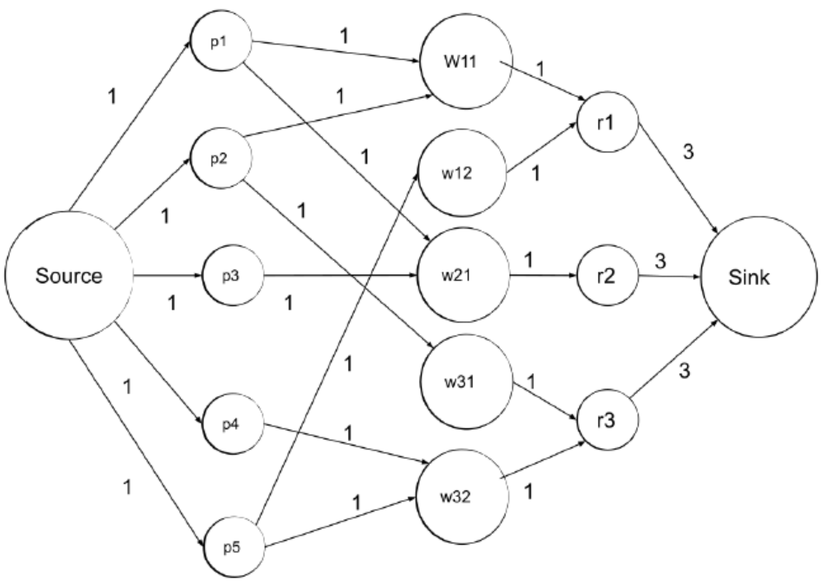


Figure 1: The flow network for Problem A

If the total capacity of the edges coming out of the source node is equal to the total capacity of the edges coming into the sink node, does this mean that all the pets can be adopted? Explain your answer and then give an example to support your answer.

b) If the total capacity coming from the source node equals the total capacity going to the sink then theoretically, every pet can be adopted. However, whether or not every pet can be adopted will depend on the pets’ preferences, the peoples’ preferences; it will depend on whether there is a flow that is able to traverse the graph from the source to the sink such that the flow is equal to the capacity leaving the source and entering the sink.

For example, if we changed figure 1 in the following ways:

- The capacity from r1 to the sink set to 2
- The capacity from r2 to the sink set to 1
- The capacity from r3 to the sink set to 2

Then we can have:

P1 → w11 → r1; flow of 1.

P2 → w31 → r3; flow of 1.

P3 → w21 → r2; flow of 1.

P4 → w32 → r3; flow of 1.

P5 → w12 → r1; flow of 1.

Here, we have every pet being adopted when the capacity leaving the source is equal to the capacity entering the sink (5).

Problem B

Is it possible to use the greedy approach to solve the job assignment problem? Explain your answer.

The job-compatibility problem uses a set J of jobs and a set W of workers. Workers complete jobs and in our example we have:

$w_1 \rightarrow j_1, j_3$

$w_2 \rightarrow j_1, j_2, j_3, j_4$

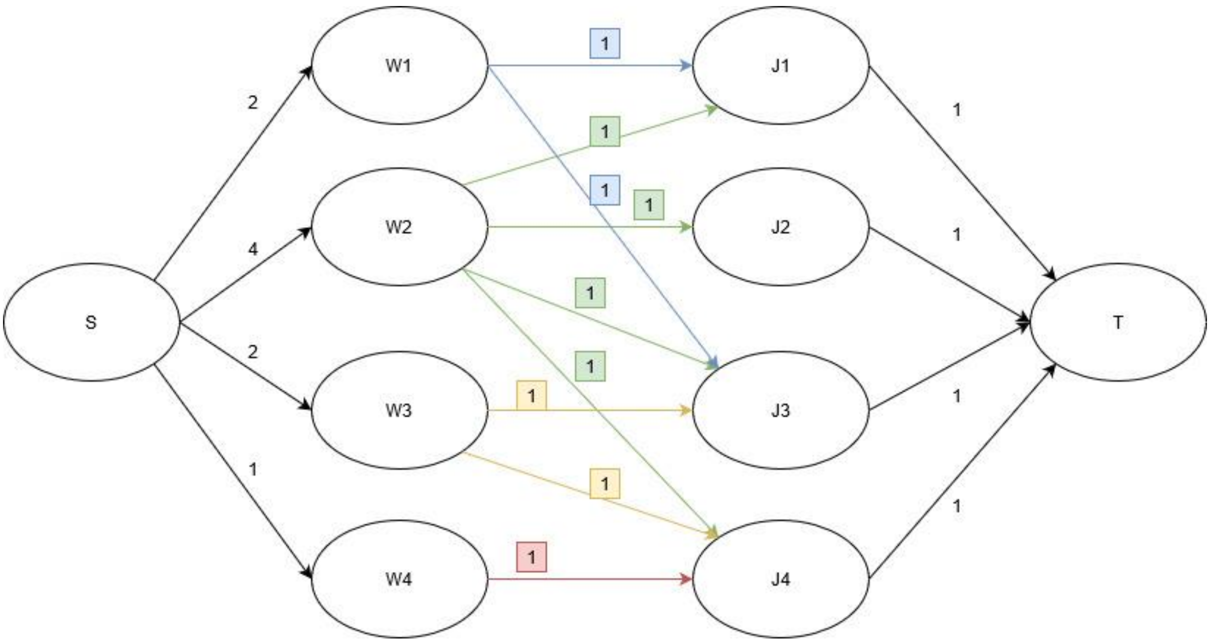
$w_3 \rightarrow j_3, j_4$

$w_4 \rightarrow j_4$

Where a worker's qualification denotes the jobs they are able to complete.

A greedy approach can be used to solve this problem. Let there exist a set J containing all of the jobs and a set W containing all of the workers. We can construct a flow network such that:

- The capacity from the source to the worker nodes signify how many jobs they can compete.
- The worker nodes point to the nodes of the corresponding jobs they are able to perform.
- The capacity from the job nodes to the sink is 1 as (in this example) each job can only be completed by 1 worker.



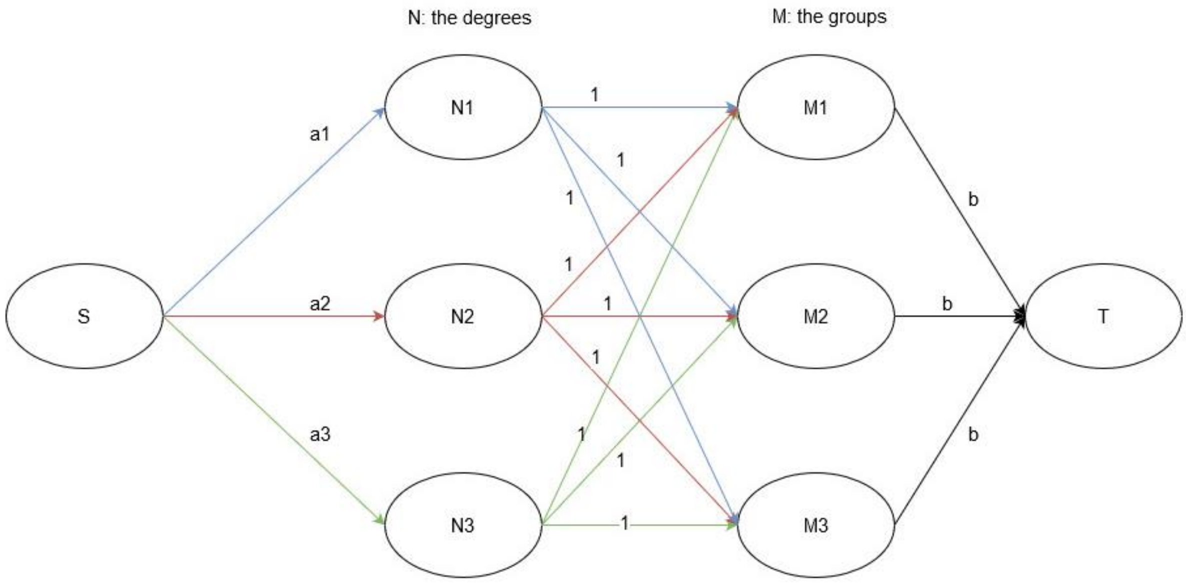
A greedy approach would be to select the flow paths in order of their lowest capacity from the source, i.e. make sure the least qualified worker (in this case W4) has the first pick. This is because a more qualified worker (e.g. W2) would be able to complete more jobs than a worker with less qualifications, so it is allowable for a more qualified worker to select a job after a less qualified worker. For the above example this would be:

- Path 1: S → W4 → J4 → T
- Path 2: S → W3 → J3 → T
- Path 3: S → W1 → J1 → T
- Path 4: S → W2 → J2 → T

With this method, we are saying that the locally optimal solution is the path from the source node to the worker node with the smallest capacity.

As the greedy method we have applied here returns a correct result, it can be said that it is possible to use a greedy approach to solve this problem.

Problem C



In the solution for your flow network, do you need all the edges coming into the sink node to be saturated, that is, filled to their capacity? Explain your answer.

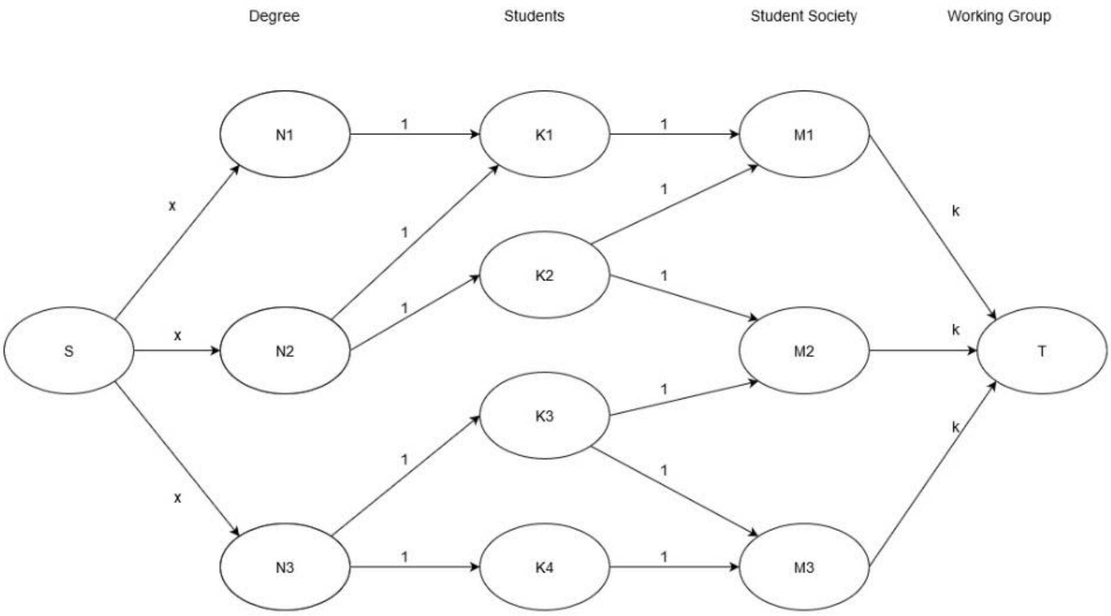
a) Each group can have at most *b* members, so we can say that each group must have a number of members that is less than or equal to *b*. This means that *b* is the maximum number of students that a group can have. However, this doesn't mean that every group is full. Because of this, you do not need to have every edge going into the sink equal to *b*, as you may have some groups whose amount of members is less than *b*, meaning that the edges going to the sink won't be saturated.

Obviously the computing students greatly outnumber the students from the other degrees, so you decided to allow at most 2 students from computing in each group. How would you modify your flow network in this case?

b) Let's assume that N_1 is a computer science degree. If we add an extra potential student from COMP to every group, then we just need to add one extra capacity to every edge heading to each group from COMP, as well as an additional one extra capacity to the sink (i.e. *b* + 1 is the new capacity from the M vertices to the sink) from the groups.

Problem D

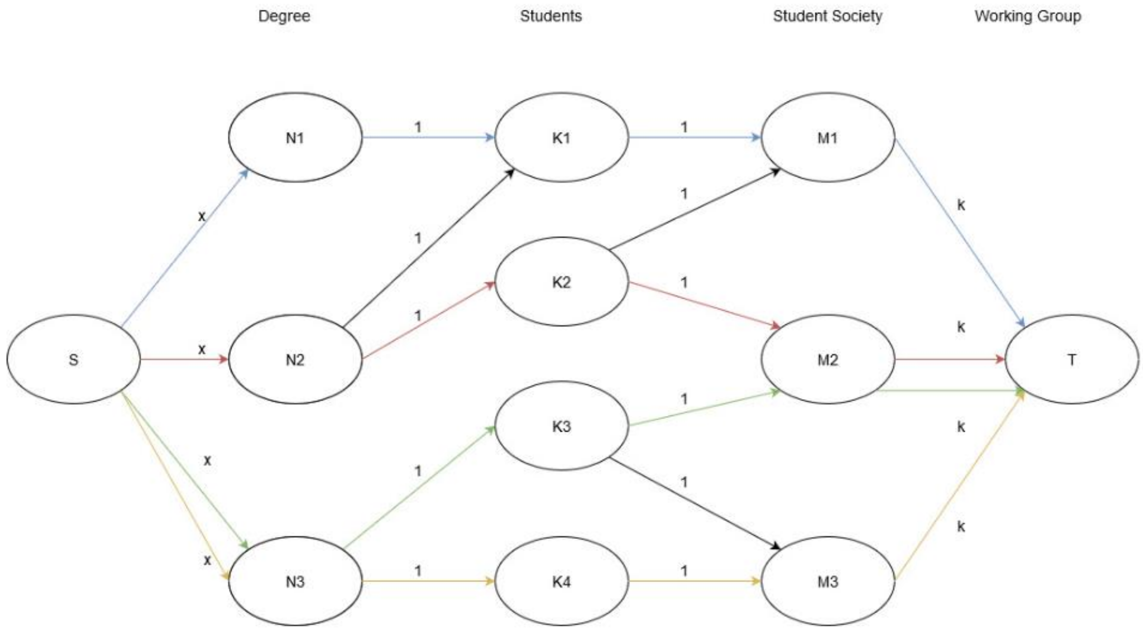
Questions: none, 100% of the marks comes from the constructing and explaining the flow network.



Explanation:

- A degree has multiple students.
- A student may be a part of multiple student societies.
- Multiple students may be part of the same student society.
- The source sends *x* students to each degree. This is because we want at most *x* students from each degree to be a part of the working group.
- Each student can do at most one degree, so the capacity from the degree nodes to the student nodes must be 1.
- Students can be members of multiple student societies, and each student society can only be represented by one student. Therefore, the capacity from the student nodes to the student society nodes is 1.
- There are *k* students, so the capacity from the student society nodes to the working group is *k*.

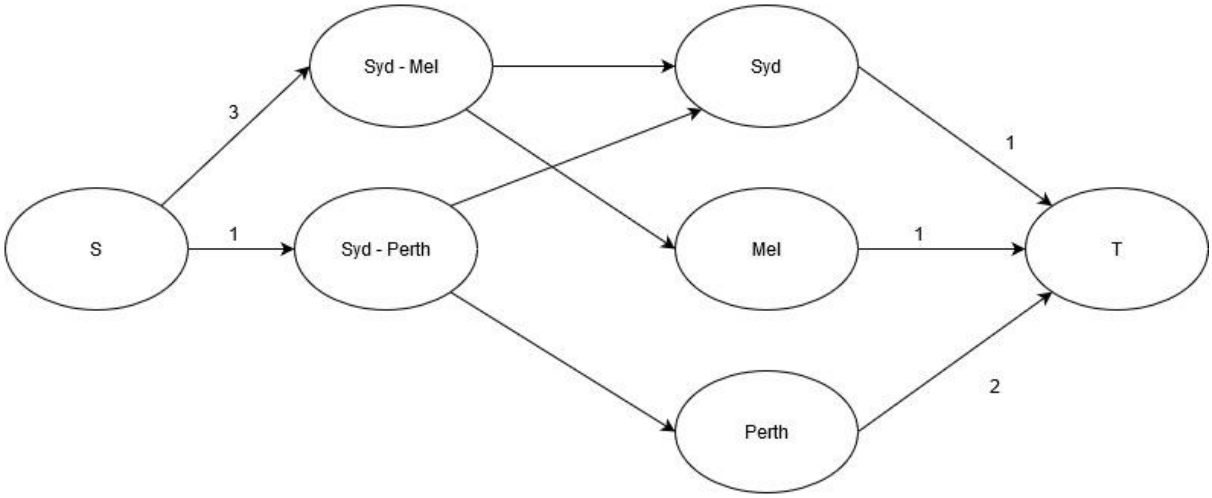
Maximum-flow:



Paths for maximum-flow:

- Path 1: S → N1 → K1 → M1 → T
- Path 2: S → N2 → K2 → M2 → T
- Path 3: S → N3 → K3 → M2 → T
- Path 4: S → N3 → K4 → M3 → T

Problem E



Based on the table above, use your flow network to show that Brisbane has no chance of coming first in the rankings.

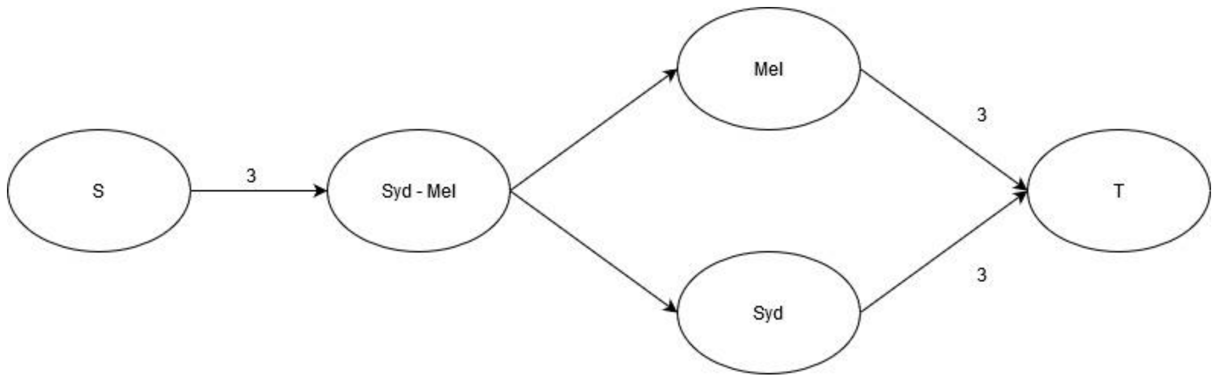
a) We want to see if Brisbane can win the tournament if it wins all of its remaining games. For this reason, we omit it from the graph and focus on if there is a path that would prevent Brisbane from coming first, even if it wins all of its remaining matches.
We graphically construct the flow network by setting the capacity between S and $city_i$ – $city_j$ as the number of games left between each city.
The capacity from the $city_i$ – $city_j$ nodes to the city nodes is set as infinity.
We calculate the capacity from $city_i$ to the sink as the sum of the potential wins Brisbane has (by adding the wins with the remaining games, we can find the potential wins of Brisbane) and subtracting this value by the number of games that $city_i$ has won.
We can tell that a city has no chance of coming first if max-flow does not saturate all the edges leaving the source; if the max-flow entering the sink is less than the sum of the capacities leaving the source, Brisbane has no chance of winning.

Sum of capacities leaving the source = 4.
Max-flow entering sink = 3.

Thus, Brisbane has no chance of coming first.

Similarly, use your flow network to show that Perth still has a chance of coming first in the rankings.

b) Using a similar method for constructing the graph in part (a), we have:



Recall that a team has no chance to win if the max-flow entering the sink is less than the sum of the capacities leaving the source. Here we have:
Sum of capacities leaving the source = 3.
Max-flow entering sink = 6.

So Perth still has a chance to come first in the rankings.

The flow network corresponding to this problem in the Wikipedia page does not restrict the flow on the edges between the game nodes and the team nodes. If you followed the same approach, explain why this is the case. If you did not, then explain if you implemented something similar in your flow diagram.

c) The capacities of the edges from the game nodes to the team nodes are set to infinity because the number of additional games won can only be restricted by the number of games that a team has left.