Adv Alg Lab 0 Report

Cassandra Overney, Nick Steelman

February 2020

Part 1: Setting up the problem

1. From the current standings, Vicky can't win because she only has 77 wins and 3 games left. Even if she wins all 3 of those games, she wouldn't get enough wins to surpass the number of Emily's wins ($77 + 3 \leq 83$). Prava will also be eliminated because even if she wins all 3 of her remaining games and gets 83 wins, either Emily or Shashank will get more wins. Emily and Shashank have to play 6 more games against each other. If Emily wins any of them, then she would get more than 83 wins and surpass Prava. If we want to ensure that Emily won't win anymore games, then Shashank would have to win all 6 of the games that he plays with Emily, which puts his number of wins to 84. Either way, Prava will not have the highest number of wins. Using the same reasoning, Shashank and Emily are not eliminated yet. If Shashank wins all his games with Emily, and Emily loses all her games, then he will win. If Emily wins any game, then she will at least tie or beat Shashank.

2. For each player, we would build a network to determine whether he/she would be eliminated. Since the procedure would be the same for each player, we will explain it using Emily as an example, but the same strategy applies to anyone.

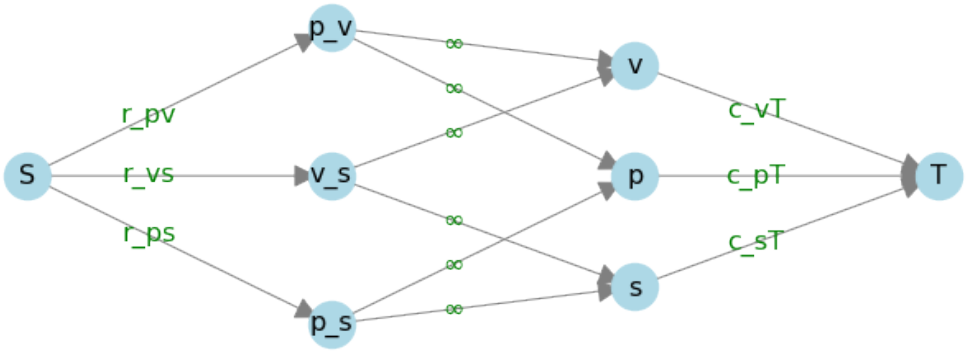

Figure 1: Emily's network flow in variable form

According to Figure 1, r_pv, r_vs, and r_ps refer to the number of remaining games that will be played between two players. For example, r_pv is the number of games left between Prava and Vicky. c_vT, c_pT, and c_sT refer to the difference in wins between Emily and the other players assuming that Emily wins all of her remaining games. The differences can be calculated as follows:

- $c\_\text{vT} = w_{\text{Emily}} + r_{\text{Emily}} - w_{\text{Vicky}} = 83 + 8 - 77 = 14$
- $c\_\text{pT} = w_{\text{Emily}} + r_{\text{Emily}} - w_{\text{Prava}} = 83 + 8 - 80 = 11$
- $c\_\text{sT} = w_{\text{Emily}} + r_{\text{Emily}} - w_{\text{Shashank}} = 83 + 8 - 78 = 13$

In essence this graph represents an ideal scenario of games played, where the flow coming from the source represents games played and the flow going into the sink represents the wins from a team. To

maintain logical consistency for the result to be a valid scenario where our team wins, all capacities from the source should be filled to show that all games have been played. Additionally, since the capacity of edges going into the sink represent how many games a team can win before they beat our team, all flows going into the sink should not overflow their capacity as this would represent that a team has beaten us and we have no chance of winning the tournament.

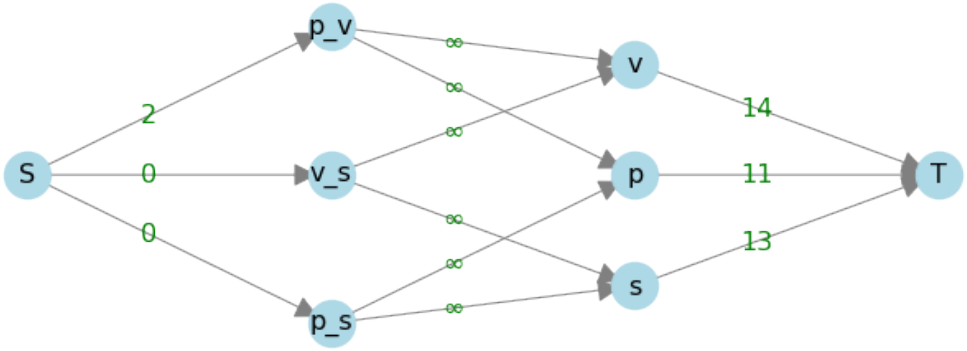

Figure 2: Emily's network flow with numerical values

Figure 2 displays the values of the variables based on the current standings. We can determine whether Emily will be eliminated by using Ford-Fulkerson or Edmonds-Karp to find the maximum flow and resulting network, essentially "playing out" the games while maintaining that this is a valid scenario. Once we have the maximum flow we can look at the network and determine if there is still capacity in the edges leaving the source. If there is, that means that not all games have been played and another game being played would break a capacity, causing a team to have more wins than us, which would result in our team being eliminated. We could also look at the minimum cut of the residual network. If the minimum cut is not 0, then we know that the limiting factor when finding the max flow is at least one of the edges entering the sink, which means that either Vicky, Prava, or Shashank would be able to win more games than is necessary to tie with Emily, thus eliminating her. If the minimum cut is 0 then no one would be able to win enough games to beat Emily, so she wouldn't be eliminated. After applying Edmonds-Karp to Figure 2, we get a maximum flow of 2 (as shown in Figure 3) and a minimum cut of 0, so Emily won't be eliminated. Basically, assuming that Emily wins all of her remaining games, there is no way that Vicky, Prava, or Shashank would be able to win enough games played amongst themselves for any of them to surpass 91 wins.

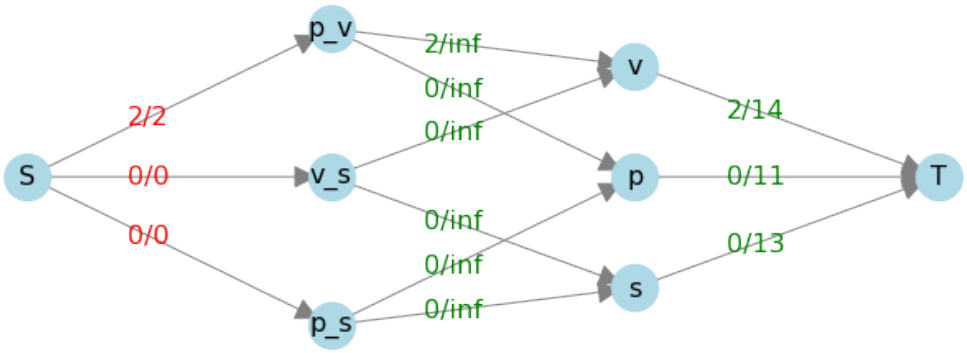

Figure 3: One possible maximum flow of Emily's network

3. In order to represent this network as a linear programming problem, we first need to represent all the rules of the network by linear equations as follows:

Source Edge Constraints:
First we can define the flows and capacities of the edges coming out of the source., If we define the series for arbitrary player a, b, c and we define variables with x to mean flow of an edge while variables with c signify capacity of an edge, we can set up the edges from the source as follows.
It's worth noting that all capacities and flows mentioned should be greater than or equal to 0, if the capacities are not, then it means victory is impossible and Prava should go home and sulk.

$$x_{s-ab} \leq c_{s-ab} \qquad x_{s-ac} \leq c_{s-ac} \qquad x_{s-bc} \leq c_{s-bc}$$

Where for example x_{s-ab} is the flow from the source to the node representing games played between player a and b.

Sink Edge Constraints:
We can also define variables to signify the capacity and flow of edges leading to the sink.

$$x_{a-t} \leq c_{a-t} \qquad x_{b-t} \leq c_{b-t} \qquad x_{c-t} \leq c_{c-t}$$

Where for example x_{a-t} represents the flow from node a to the sink which represents the number of games won by team a.

Node flow definitions:
We then also define variables for the edges representing who wins the games between each team. For the flow for nodes representing games played, we can define them as follows:

$$x_{ab-a} + x_{ab-b} = x_{s-ab} \qquad x_{ac-a} + x_{ac-c} = x_{s-ac} \qquad x_{bc-b} + x_{bc-c} = x_{s-bc}$$

Where for example x_{ab-a} is the number of games that team a wins when played against b. (Basically saying number of games from ab is the wins from a plus the wins from b)
Additionally for flows around nodes associated with how much each team wins, we can set up some equations as follows:

$$x_{ab-a} + x_{ac-a} = x_{a-t} \qquad x_{bc-b} + x_{ab-b} = x_{b-t} \qquad x_{ac-c} + x_{bc-c} = x_{c-t}$$

(Basically saying the amount a wins is its wins vs b plus its wins vs c)

Filling in constraints:
Finally we can define the capacities in terms of the given information. First the capacities of edges from the source are the games remaining between teams, represented by symbol r

$$c_{s-ab} = r_{ab} \qquad c_{s-ac} = r_{ac} \qquad c_{s-bc} = r_{bc}$$

Where r_{s-ab} is the games remaining between team a and team b.
Then, the capacities of the edges to the sink are the amount of points each team would need to beat the team in question (if our team won all their remaining games). So if our wins are represented by w_{us}, our remaining games by r_{us}, and the wins of other teams by w_a, w_b, w_c:

$$c_{a-t} = w_{us} + r_{us} - w_a \qquad c_{b-t} = w_{us} + r_{us} - w_b \qquad c_{c-t} = w_{us} + r_{us} - w_c$$

If any of these are below 0, then we are eliminated.
Finally we can maximize any cut for this network, but let's use the easy one, the edges coming out of the source:

$$max(x_{s-ab} + x_{s-ac} + x_{s-bc})$$

Once the maximization is done, we'll know if we have a shot at winning if the flow of these variables is equal to their constraints.

This formulation makes sense since, all the constraints of the network (e.g. nodes having equal flow) are represented inside the equations and our maximization goal is equivalent to the max flow of the network. So, reworking this series of linear equation to find values that maximize our goal will be equivalent to finding the max flow of the network.

Part 2: Implementation

We implemented the network flows and the linear programming approach to this problem. Our code can be found in *baseball_elimination.py*. We decided not to add any extra tests because the current tests are sufficient to determine whether we can accurately determine if a player will be eliminated or not. There is a nice variety of tests ranging from 4 teams to 24 teams. Looking at the smallest test, it covers a wide range of different cases, including (1) a player should be eliminated if their current and possible wins would not exceed the current wins of another player (e.g. Vicky), (2) a player should be eliminated if for sure another player would get at least one more win after playing the remaining games (e.g. Prava). We could add a few more edge test cases, such as when we only have one team (they shouldn't be eliminated) or two teams (including a case when they could tie).