Network Flows Lab

Adam Selker, Nick Sherman

Due February 25, 2020

1 Part 1: The Algorithm

1.1 Question 1: Who's eliminated?

The first question we need to answer is who is eliminated in the sample problem in Lab 0.

Emily has not been eliminated. If she wins all eight games, she'll have 91 wins, which is more than anyone else can have.

Prava is eliminated. If she wins all three games, she'll have 83 wins. If Emily beats Shashank at least once, Emily will have at least 84 wins; if Shashank beats Emily all six times, then Shashank will also have 84 wins. This causes Prava to lose as her maximum number of wins is fewer than the max wins of Shashank and Emily.

Shashank is not eliminated. If Shashank wins all of his games, he will be left with 84 wins, which is more than Prava or Vicky could achieve. If Emily loses one or both of her games against Prava and Vicky, then she will have either 83 or 84 wins total, proving that it is possible for Shashank to win.

Vicky is eliminated. If she wins all three games, she'll have 80 wins, which is already less than Emily. rip.

1.2 Question 2: Network Flow

Throughout this problem, we only consider games that have yet to be played. So, for instance, l_p is the number of *remaining* wins that Prava needs to beat Emily; it excludes the 80 games she's already won. Also, we assume that Emily wins all 8 games she has left, giving her a total of 91 wins.

These are the variables we use in this problem. Those with fixed values also have their values listed.

- $r_{pv} = 2$: The number of remaining matches between Prava and Vicky
- $r_{vs} = 0$: The number of remaining matches between Vicky and Shashank
- $r_{sp} = 0$: The number of remaining matches between Shashank and Prava
- $l_p = 11$: The number of wins Prava needs to be guaranteed to tie Emily

- $l_v = 14$: The number of wins Vicky needs to be guaranteed to tie Emily
- $l_s = 13$: The number of wins Shashank needs to be guaranteed to tie Emily
- f_p : Prava's wins in our "capped" scenario
- f_v : Vicky's wins in our "capped" scenario
- f_s : Shashank's wins in our "capped" scenario
- f_{vp} : The number of wins that Vicky had over Prava
- f_{pv} : The number of wins that Prava had over Vicky
- f_{vs} : The number of wins that Vicky had over Shashank
- f_{sv} : The number of wins that Shashank had over Vicky
- f_{ps} : The number of wins that Prava had over Shashank
- f_{sp} : The number of wins that Shashank had over Prava
- g_{pv} : The number of games played between Prava and Vicky
- g_{vs} : The number of games played between Vicky and Shashank
- g_{sp} : The number of games played between Shashank and Prava

In order to solve this problem, the first thing we do is construct the graph. The source S and sink T don't represent much; they're for mathematical convenience. The left-hand nodes (p_v, v_s, s_p) represent games played between people other than Emily. Naming convention is by the first letter of the two people playing; e.g. p_v is all games played between Prava and Vicky. The right-hand nodes (V, P, S) represent each person's wins; e.g. V represents the games that Vicky wins.

The connections between S and the left-hand nodes represent the number of remaining games. For example, p_v is supplied with up to $r_{pv} = 2$ games because Prava and Vicky have two more games to play. The connections between the left-hand and right-hand nodes are unlimited, and their flows represent who wins what games; for instance, f_{vp} is the number of (upcoming) games in which Vicky will beat Prava. The connections from the right-hand nodes to the sink T represent the fact that we're looking for any solution where Emily *does not* lose, so they are limited to however many wins would make that person tie with Emily. For Prava, $l_p = 12$ because if Prava wins 12 games, then she'll match Emily's score of 91 wins.

By limiting the capacities of the connections to the sink node, we are forcing all players' win counts to be less than or equal to Emily's. If we find that we exhaust the supply from the source before the far right flows are maximized, then we know that it is possible for Emily to still win, since we have found an arrangement of wins and losses where Emily would have the most wins, or be tied for it. If, however, there is still flow left to be added and the right sides are maximized, then we know that there is no configuration that allows for Emily to still win. This will happen if there is some group of people where no arrangement of wins and losses will keep them all to 91 or fewer wins. Therefore, this strategy will allow us to see if any combination exists that allows for Emily to still win, and prove whether she is eliminated or not.

Putting all of this together, we ended with the graph that can be seen in 1.

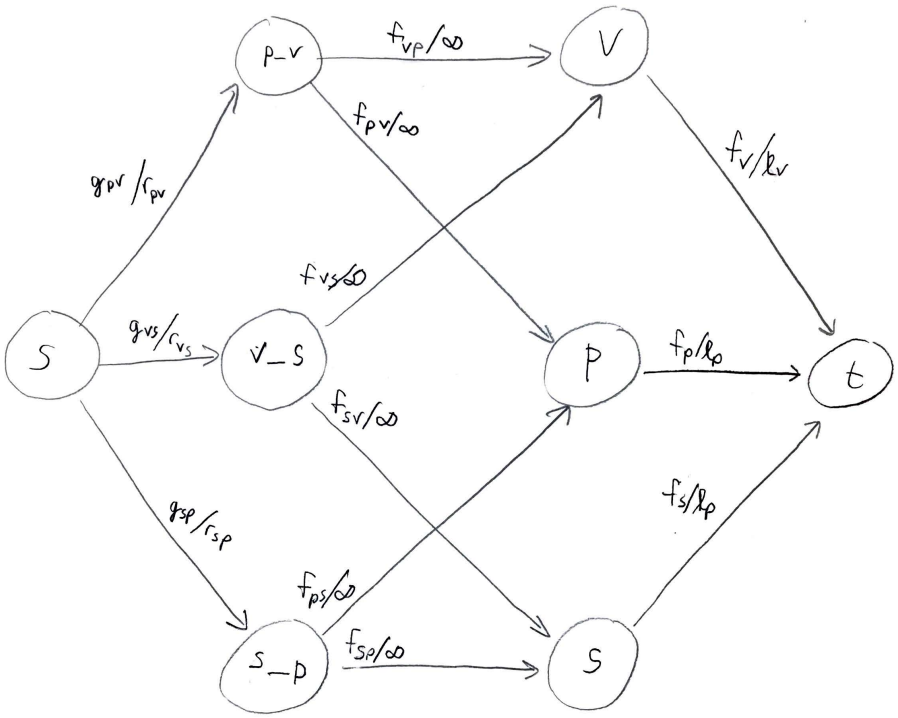

Figure 1: Our labelled graph of the example problem which is checking to see if Emily is eliminated.

Finally, in order to use this method to test whether Emily is eliminated, we first find the maximal flow of the network. We can do this with the Ford-Fulkerson algorithm. This algorithm is preferable over the Edmonds-Karp algorithm because all paths from sink to source are the same length, which means a depth-first search of the path space will be faster than a breadth-first search. Once we have the maximal flow, we check to see whether a min cut consists of only the source or not. If there is no excess flow that can come from the source, then we know that there is a combination of game wins/losses that results in all other players having at most the same number of wins as a best-case scenario for the current player. This means that the current player, Emily, can still win. If, however, the min cut occurs not next to the source node and there is flow that could flow from the source node into the system, then the current player is eliminated because this means there is no scenario where they could be part of the team with the most wins.

1.3 Question 3: Linear Program Conversion

Based on the network flow diagram that we finished above, we obtained the following linear program:

We optimize for a profit variable p, which is defined as:

$$p = g_{pv} + g_{vs} + g_{sp}$$

This corresponds to the number of games played, before it's impossible to play another game without someone overtaking Emily. If the profit p hits its maximum of $r_{pv} + r_{vs} + r_{sp}$, then all games have been played, and Emily is not eliminated. If, after optimizing this linear equation, p is less than this maximum, then Emily has been eliminated.

$$g_{pv} \leq r_{pv}$$
$$g_{vs} \leq r_{vs}$$
$$g_{sp} \leq r_{sp}$$

The first set of equations dictates the maximal flow that can go through each of the edges attached to the source. These limits are derived from the total number of each type of game left to be played, as the model must be limited by the number of games left to play to be solvable.

$$g_{pv} - f_{vp} - f_{pv} = 0$$
$$g_{vs} - f_{vs} - f_{sv} = 0$$
$$g_{sp} - f_{sp} - f_{ps} = 0$$

The second set of equations connect the source to the first layer of nodes, which represent the total number of games of each type left to play. This equation is balanced due to flow conservation, as the r component of each equation is the flow from the source, which is then split up into two separate flows out from the node, indicating the wins to each of the two possible people for a game. This means that every game has a winner, with a net gain/loss of 0 at a given node.

$$f_{ps} + f_{pv} - f_p = 0$$
$$f_{vp} + f_{vs} - f_v = 0$$
$$f_{sv} + f_{sp} - f_s = 0$$

The third set of equations connect the sink to the second layer of nodes, which represent the games that each person wins. These equations are balanced as the number of wins each person gets is equal to the summed number of wins they have over each other person (e.g. Vicky's total wins is equal to her wins over Pravallika plus her wins over Shashank).

$$f_v \leq l_p$$
$$f_v \leq l_v$$
$$f_v \leq l_s$$

The fourth set of equations govern the limits of the flow to the sink. For each of the edges, the maximum amount of flow that can pass through must be limited by the total number of wins that a given person can have before beating Emily's maximal score. (e.g. The maximum number of victories Shashank could win assuming Emily won all of her remaining games would be $w_e + r_e - w_s$, or $83 + 8 - 78$ or 13.) This makes sense as we are solving the equations such that either all games are played and the max wins are less than Emily's total number of wins or to prove that it is impossible for Emily to win enough and still be a contender for first place.

$$f_p, f_v, f_s, f_{pv}, f_{vs}, f_{sp}, g_{pv}, g_{vs}, g_{sp} \geq 0$$

The sixth and last set of equations limits all of the flows to be non-negative since, for instance, it is not possible to play or win a negative number of games.

2 Part 2: Implementation

You can find our code in our Github repository at the following link: https://github.com/NickShermeister/Lab0.

2.1 Test Case Analysis

We spent a little bit of time trying to think of situations that would make our solution break when we were trying to figure out what the answer to Team 13 should be. However, as we were working through the other solutions, we found that most of the things we would have come up with (basic example, a Prava situation in the example given to us, and larger, more complicated examples) seemed to go smoothly. The only other test that may have been useful is a very large-scale test (>100 teams), mostly as an efficiency test (as we think that our linear programming solution could be more efficient due to its long runtime, and having a large benchmark would help prove this).