CS170 - Fall 2016 - Final Project Writeup

Name: Brandon Teo
SID: 26494656
Login: cs170-akr
Email: brandon.teo@berkeley.edu

In this algorithm, we model the relationship between the horses as a directed graph with the horses as vertices and the “liking” relationship between horses as edges. In the big picture, for each iteration, we randomly pick the first vertex in the graph and then we try two approaches. We first check if the chosen vertex leads to other friend horses. If no, then we declare the chosen horse as the last horse in the team it belongs to and pick another horse to start another team. If yes, then we have two approaches: (1) We randomly pick a horse among its friend horses. (2) We greedily pick the horse among its friend horses that has the highest performance rating. By keeping track of which horses have already been assigned to a team, we repeat this process until all the horses have been assigned to a team and end the iteration. We then run numerous iterations for the two approaches and keep track of all the returned results. We then pick the assignment that gives the best result as the output of the algorithm.

To improve the runtime of this algorithm, we make the following optimizations. By analyzing the problem, we can see that each instance of the problem must have a maximum value when all the horses can be assigned into one team. This means that during each iteration of assigning horses to teams, if we are able to assign all of them into one team in the midst of running through the iterations, we can end the algorithm early since we have already maximized the output. Also, since the algorithm is randomized, we can first run a lesser amount of iterations and check if we can get to assign all the horses into one team. If after running through this first pass we cannot get to assign the horses into one team, we can perform a second pass with more iterations to search for the maximal assignment. By performing this optimization, we will be able to solve “easier” instances more efficiently and spend more time on “harder” instances by running more iterations on it and search for the best assignment.