CS170 Final Project Report

Group member: Yicheng Chen, Shuyu Tang
Language: Python
Outside package: networkx for Python

Strategies:

Greedy 1—largest node first: we find the sources of the graph first and then start with the source node with largest score (if no source, start with the largest node in the graph). And each time we include and go to the child node with the largest score until no child available. These nodes form a path in the graph and then we remove this path from the graph and repeat the process on the residual graph until all nodes are selected. A variant of the basic greedy algorithm is to start from the sink and go to the parent node, this also guarantees a valid partition of the graph.

Greedy 1 with random selection/weighted random selection: Similar to the basic greedy scheme, but each time we select the nodes randomly. In the first strategy, we select the child nodes with equal probability, while in the second (weighted random selection) we select the child among all the children with the likelihood equal to their scores. These random strategies were also be adapted for the greedy scheme starting from sinks.

Greedy/DP --Longest_weighed_path_forward/backward: In this strategy, we try to find the path with highest performance rating in the graph, choose this path as a relay team, remove the path and recursively do it in the remaining graph. We guess this strategy is not too far from the optimal solution. First, graph is converted to a DAG. Cycles are removed by using a DFS search and if a node has been visited twice and if it has a path to its parent, remove the edge from parent to that node. In topological order, “cumulative value” and “cumulative number” of each node are calculated. “cumulative number” of a node = “cumulative number” of the parent with highest “cumulative value” + 1. “cumulative value” of a node = highest “cumulative value” of its parents / “cumulative number” of that parent * “cumulative number” of the node itself. Also, previous nodes that contribute to “cumulative value” of a node are recorded. Then we could found the node with highest “cumulative value” and put its previous nodes and itself in a relay team, and remove this path. “backward” in the name indicate the method is implemented in the reverse graph.

Try all possible combinations: As the name indicates, this algorithm tries all possible combinations of relay teams and return the best one. Although this is an exponential algorithm, it runs in a reasonable time when the number of nodes or edges is smaller. It’s useful to check if the solution is optimal.