CS 170 Project Deliverable 1

Tina Zhao, Lily Bhattacharjee, Thien Phan

November 11, 2018

1 Introduction

We reduced the overall problem into two subproblems we had to solve: maximize the number of friends in the same bus and minimize the number of rowdy groups. For our explanations, we will assume we are given a graph with each node representing a student and k buses with capacity c. We also assume that the size of each rowdy group cannot be greater than c.

2 Kernighan-Lin Algorithm Approach

For this approach, we rely on partitioning because we interpreted the problem as finding k partitions that fit the conditions. After some research, we came across the Kernighan-Lin algorithm. The algorithm takes in an undirected graph and partitions the vertices into two disjoint sets of (nearly) equal size while minimizing the sum of the weights of the edges crossing the two subsets. Some sources state that we start off with two subsets and swap the elements until we get a minimum sum of the edges crossing the two subsets, but for now, we will follow the algorithm from the source given. It also has a runtime of $O(n^2 \log n)$. We decided to use this algorithm to handle our second problem of minimizing the number of rowdy groups.

We start off by addressing our first subproblem by partitioning the original graph of students into $k/2$ sets of size $2c$ (this is so we can run Kernighan-Lin on each set and end up with k sets of size c). We will create these $k/2$ partitions by dividing along friend groups as optimally as possible to minimize cuts across partitions. This will ensure that friends have the highest chance of ending up on the same bus.

After splitting the graphs into $k/2$ subsets, for each subset, we will now add "rowdy" edges and assign edge weights to all of the edges. Since each rowdy group is constrained to be at most size c, we will assign the edge weights of the edges belonging to the original graph to be some integer d greater than c. For "rowdy edges", we will add edges between nodes in the same rowdy group. The edge weight will depend on how many rowdy groups a particular node is in and how many of those rowdy group members are present in the subset. If all the members of a rowdy group are present, the corresponding edge weights of the graph will be 1. If only some members are present, the edge weight will correspond to the number of total rowdy group members - the number of members present in the subset + 1. If a node is in multiple rowdy groups, the corresponding "rowdy" edges of the node will be weighted lower. For now, we decided on an arbitrary heuristic in which the weights would be the absolute value of the difference between the two original "rowdy" edge weights.

Now, we just have to run the Kernighan-Lin algorithm on each of those $k/2$ sets and end up with k groups.