University of Toronto at Scarborough
CSCC73 - Algorithm Design and Analysis

Assignment #8 Approximation Algorithms

Due: April 11, 2023 at 11.59pm This exercise is worth 5% of your final grade.

Warning: Your electronic submission on Gradescope affirms that this exercise is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on Gradescope. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

1. A trucking company needs to load trucks with n boxes to send to a distribution center. We'll say that the size of a truck is 1 and that the i^{th} box has size $0 \leq s_i \leq 1$. It doesn't matter which boxes go into which trucks. Each truck can hold a subset of the boxes that does not exceed 1 unit. Consider the heuristic that takes each box in turn and places it into the first truck that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

   (a) Determine the minimum number of trucks required by the optimal solution. Justify your answer.

   (b) Prove an upper bound on the number of trucks that this heuristic leaves less than half full.

   (c) Prove an upper bound on the number of trucks used by the heuristic.

   (d) Determine α and prove an approximation ratio of α for this heuristic.

   (e) Give an efficient implementation for this algorithm and analyze it's running time.

2. Consider the load balancing problem with just two machines M_1 and M_2. The goal is to keep the loads balanced. In particular, there are n jobs, and each job j has a required processing time t_j. The jobs need to be partitioned into two groups A and B, where set A is assigned to M_1 and B is assigned to M_2. The time needed to process all of the jobs on the two machines is $T_1 = \sum_{j \in A} t_j$ and $T_2 = \sum_{j \in B} t_j$. The problem is to have the two machines work roughly for the same amount of time–that is, to minimize $|T_1 - T_2|$. This problem is known to be $NP - hard$ which is why a good local search algorithm may be a good option.

   Consider the following proposal. Start by assigning jobs to the two machines arbitrarily (say jobs $1, \ldots, n/2$ to M_1, the rest to M_2). The local moves are to move a single job from one machine to the other, and we only move jobs if the move decreases the absolute difference in the processing times. We want to answer some basic questions about the performance of this algorithm.

   (a) The first question is: *How good is the solution obtained?* Assume that there is no single job that dominates all the processing time–that is, $t_j \leq \frac{1}{2} \sum_{i=1}^{n} t_i$ for all jobs j. Prove that for every locally optimal solution, the times the two machines operate are roughly balanced: $\frac{1}{2} T_1 \leq T_2 \leq 2T_1$. A *locally optimal solution* is one that is not an optimal solution but one in which the algorithm terminates as moving a single job cannot improve the solution.

(b) Next we consider the running time of the algorithm: *How often will jobs be moved back and forth between the two machines?* Consider the following modification to the algorithm. If, in a local move, many different jobs can move from one machine to the other, then the algorithm should always move the job j with maximum t_j. Prove that, under this variant each job will move at most once. Hence the local search terminates in at most n moves.

(c) The algorithm given above is a *local search algorithm* as it starts with a solution and searches "neighbouring" solutions for an improved solution until it can do no better. In this algorithm with the variation in Part 2b), prove that the search will not always lead to an optimal solution by giving an example.

(d) Express this problem as a linear programming problem. In other words, define an objective function which needs to be minimized/maximized and constraints on the variables.