EECS 336 Homework 5

Tom Large

1. This problem has a feasible schedule when the $n \cdot c \geq m$, or in other words, there are enough shuttles with enough capacity to fit all employees. If $n \cdot c < m$, then there is no way to create a schedule where every employee can get on a shuttle.

2. **Algorithm**

We first sort our list of employees is sorted in increasing order by their release times, r_j. Then, we construct the following subproblem, $C(i, j, c')$, which is the minimum cost of placing passenger j on shuttle i which currently has c' employees scheduled to leave with this shuttle.

$$C(i,j,c') = \begin{cases} \infty & c' > c \text{ or } i > n \\ 0 & j = m \text{ and } c' < c \\ \min\left((c'+1)\cdot(r_{j+1} - r_j) + C(i, j+1, c'+1), C(i+1, j+1, 0)\right) & \text{else} \end{cases}$$

Essentially, our algorithm steps through each passenger in order of release time, and places this passenger on the shuttle. It then makes a decision, whether to leave at that moment, thus leaving the next passenger $j+1$ to get on shuttle $i+1$ with no additional cost for waiting, or the shuttle waits for passenger $i+1$, and incurs a cost of $(c'+1)\cdot(r_{j+1} - r_j)$, since the shuttle will wait for at least $(r_{j+1} - r_j)$ more time with $c'+1$ passengers aboard.

If the shuttle has allowed on too many passengers or we have scheduled too many shuttles, we return ∞ to indicate an unfeasible solution. If we have just let on the last passenger, and the rest of the constraints are still satisfied, then the shuttle can leave, and we have found a feasible solution, so we end the recursion and return 0.

3. We prove this part with 2 Lemmas.

Lemma 1: We claim that an optimal solution has for each shuttle i, $d_i = r_{j'}$, where j' corresponds to the employee on the shuttle with the latest release time, $r_{j'}$.

The intuition is that it is wasteful for a shuttle to wait for any amount of unnecessary time. Thus, suppose there is an optimal solution such that there exists shuttle i such that $d_i > r_{j'}$, where j' is defined as in the claim. Since this shuttle allowed no one else on after employee j', then leaving at $r_{j'}$ would reduce the cost of the trip by at least $d_j - r_{j'}$ since employee j' would wait no time for a shuttle. Thus, since this change would still yield a feasible solution, then this could not have been an optimal solution to begin with. □

Lemma 2: We claim that an optimal solution must have that each employee j get on the shuttle that leaves the soonest after their release time r_j.

Again, the intuition here is that this allows passengers to wait for the least amount of time when waiting for their shuttle. So suppose there were an optimal solution such that there exists shuttles i and i' where employees $j_{i_1}, \ldots, j_{i_k}$ ride on shuttle i and employees $j_{i'_1}, \ldots, j_{i'_{k'}}$ ride on shuttle i'.

Assume $d_{i'} > d_i$ and that there exists employee $j_{i'_s}$ with $r_{i'_s} < d_i$, so that $j_{i'_s}$ was ready by the time shuttle i left, but rode on shuttle i'. By the previous lemma, since this is an optimal solution, shuttle i must carry a passenger j_{i_l} such that $r_{i_l} = d_i$, and thus $r_{i'_s} \leq r_{i_l}$. Suppose we swapped these two passengers, so $j_{i'_s}$ rode on shuttle i and j_{i_l} rode on shuttle i'. If $r_{i'_s} = r_{i_l}$, then there is no change in cost. Moreover, shuttle i can leave earlier, in particular we can say $\hat{d}_i = r_{i_{l-1}}$, which is the second to last passenger to be ready on shuttle i, which could be $r_{i'_s}$. In either case, employee $j_{i'_s}$ waits for $0 \leq \hat{d}_i - r_{i'_s} < d_{i'} - r_{i'_s}$ and employee $j_{i'_s}$ waits for $d_{i'} - r_{i'_s}$, and thus total waiting time for these two employees has decreased since $\hat{d}_i - r_{i'_s} + d_{i'} - r_{i_l} < d_i - r_{i_l} + d_{i'} - r_{i'_s}$. Thus, the cost will be minimized by making this switch, and we preserve the feasibility of the solution since no shuttle went over capacity, thus proving the Lemma. □

4. Feasibility: Our algorithm satisfies the four constraints in the following way. We assume that there is at least one feasible solution to begin with. Thus, our algorithm returns a finite value only when we have reached the last passenger to account for. The recursive formula always proceeds to the next passenger, either placing this passenger on the current shuttle i or leaving it for shuttle $i + 1$ to decide. Moreover, if we ever reach a point where the capacity for a shuttle c' exceeds the max capacity c, or if we have scheduled too many shuttles ($i > m$), we return an infinite value. Lastly, since we have sorted our passenger release times in increasing order, then we know that a shuttle leaves with only passengers that are ready by the depart time, and thus this constraint is also satisfied.

Optimality: In our base case, if we have a shuttle that just let on the last passenger, then there is no additional cost for putting this passenger on the shuttle since the shuttle will leave at r_m, so 0 is the proper value for this case.

In our recursion, use Lemma 1 and Lemma 2 to simplify the decision-making. Given that we are not letting on the last passenger, we see that we have two decisions. Since we just let on passenger j, shuttle i can leave immediately, and let passenger $j + 1$ get on shuttle $i + 1$, which will incur no waiting costs for passengers $1, \ldots, j$ (by Lemma 2, there should be no passengers leaving on shuttle $i + 1$ before employee $j + 1$). Or, shuttle i can wait to let on employee $j + 1$, incurring a waiting cost of it's capacity after letting on employee j, which is $c' + 1$, times the amount of time waiting for passenger $j + 1$, which is $r_{j+1} - r_j$. Moreover, we only ever leave immediately after a passenger has boarded, which we showed to be optimal in Lemma 1. Thus, our algorithm considers every possible decision from these assumptions, and returns the one with the minimum cost, which must be optimal.

5. We see that for each subproblem, excluding recursive calls, we either return ∞, 0, or the minimum of two numbers. These are all $O(1)$ operations, so each subproblem runs in constant time. We consider at most n shuttles and m passengers, however, since we take at least one passenger at every iteration, we will not consider more than m shuttles. Moreover, we can bound the number of considered values for c' by m, since a shuttle taking m passengers will terminate the algorithm before reaching the capacity if $c > m$. Thus, there are maximum $O(m^3)$ subproblems, so our total run time accounting for sorting m employees is $O(m \log m) + O(m^3) \cdot O(1) = O(m^3)$.