Assignment 1: Crew Scheduling
Transportation and Scheduling (FEM21043)

Wessel Boosman (589273)

January 12, 2022

Question 1

(a) *First, discard the requirement for duties to contain a break. Consider the LP-relaxation of the CSP and assume it is solved by column generation. Describe the pricing problem as a shortest path problem with additional constraints. Develop an algorithm to solve this pricing problem. What is the running time of your algorithm? (Is it polynomial?) Also discuss how many columns (or: duties) are found when the pricing problem is solved by your algorithm, and how many you would add to the master problem.*

The pricing problem a shortest path problem with the constrain that the length of the duty should not be larger than the maximum allowed length, which is nine and a half hour. To solve this problem I implement the following algorithm. I create a list of tasks which could be the start of the duty, this are tasks which start from a depot. From this starting task, I make a list of all the tasks which are within the allowed time period of 9.5 hours of this task. Arcs are drawn from the starting task to tasks in this neighborhood if they start from the same platform as for which the starting task ends. Then for each of those newly reached tasks, I draw new arcs towards tasks which are in the neighborhood of the first task, begin after the ending time of that task and, start from the platform at which that tasks ends. Also for each reached task, I check I they end at the same platform at which the first task starts (so the depot). If this is the case, I also draw an arc from this task to the sink, because then a duty can be ended. In total, I construct a new directed and acyclic graph for each of the tasks at which a duty can start.

From there I use the dual variables from the RMP to update the weights on each of the arcs. Then for each directed acyclic graph I can calculate the shortest path (corresponding to a duty) to the sink and store the reduced costs. The shortest path corresponding to the lowest reduced cost (and the lowest reduced cost must of course be negative) is added to the RMP. This process is repeated until no paths can be found anymore with a negative reduced cost.

This time complexity of finding the shortes path in a directed acyclic graph is polynomial, in fact it is of $\mathcal{O}(V{+}E)$. The LP relaxation of the RMP (a set covering problem) can also be solved in polynomial time. Therefore, the pricing problem as well as the RMP can be solved in polynomial time, and the complete algorithm should be running in polynomial time. I created an algorithm that only adds one duty each time the pricing problem is solved, namely the duty with the lowest reduced cost. An alternative approach could be to add multiple duties which have the most negative reduced costs.

Like I mentioned, I update the weights on the arcs using the dual variables of the RMP. I initialize my algorithm with a big M. So each of the tasks gets its own duty, even if this corresponds to a infeasible duty. I give each of this duties the price it should have if the duty is feasible and a price of 2090 if it is not feasible. This big M is based on the start of the first task and the ending time of the last task. It is not possible to have a duty with a higher price than this big M, even if the 9.5 hours constraint is relaxed.

(b) *See Table 1*

(c) *Suggest a heuristic to find an integer solution for the CSP. Describe and implement this heuristic and report the solution value and computation time for the small and large instance. For both instances, provide a text file that describes the solution of the CSP.*

The LP relaxation returns a whole bag of duties and a value between zero and one, corresponding to how much this duty is in the optima solution. You can't perform duties partially, so unless the LP relaxation return binary variables it is infeasible. To go from infeasible to feasible I construct the following heuristic:

1. Take the LP solution
2. Fix the largest value of the LP solution to 1
3. Solve LP relaxation of RMP
4. Fix the largest value of the LP solution to 1 which is not fixed yet
5. Repeat until each variable is binary

(d) *Now, do consider the requirement for duties to contain a break. Answers questions a., b., and c. again, now incorporating the break requirement in the pricing problem of the column generation algorithm.*

The heuristic stays the same. The results are given in Table 1. To allow for the break I make some changes to the algorithm explained in question A. Again I make a list of all possible starting tasks, these are all the tasks starting from a depot. For each of this tasks I construct a graph as in part A, but now with a time window of 5.5 hours instead of 9.5 hours. Using this graph I construct a list of tasks which can be performed before the break, this are all the tasks in the graph. So for each starting task I have a list of tasks which can be performed before the break. For each of the tasks in the list you can decide to take a break afterwards, resulting in a different graph. So instead of a graph for each of the starting points as in the

Table 1: Table summarizing all instances

	Small no Break	**Large no Break**	**Small with Break**	**Large with Break**
LP objective	108,861.2	259,234.1	113,092.0	-
Heuristic objective	115,674	283,287	119,710	-
Time spent in PP[s]	45	1767	242	-
Time spent RMP [s]	152	2584	140	-
Total running time LP[s]	212	4361	399	-
Total running time heuristic[s]	12	185	11	-

algorithm without a break, we now have a list of graphs for each starting point. Namely, for each starting task there is a list of tasks after which you can decide to have a break, and for each of these tasks I construct a new graph. I construct the graph in the following way:

1. Create graph with tasks between starting task and task after which is decided to take a break
2. Consider all tasks after the break (of 30 minutes) which satisfy the constraint of working no longer as 5.5 hours without break or not longer than 9.5 hours in total.
3. Connect graph from before the break with the graph after the break and finish the graph

The pricing problem remains the same, I still look for the shortest path among all graphs, only the number of graphs is now considerably higher. This resulted in the problem that for the large instance I ran into memory errors.

(e) *We now aim to improve the time required to solve the LP-relaxation of the CSP. Explain some possibilities to solve the LP-relaxation faster, imple- ment them, and report whether they indeed improve the overall computa- tion time.*
To improve the running time of the LP-relaxation I would first look at the amount of time the problem is spending on the pricing problem and the amount of time the problems spends on the restricted master problem. If most time is spent in the restricted master problem, then this could suggest that the solver finds it hard to find an optimal solution. We could look at the duties which are not in the optimal solution at least for a certain number of iterations and delete them from the bag of duties. In this way the problem becomes smaller and should become easier to solve. A disadvantage is that you maybe throw away duties which could be beneficiary later on, or which could be beneficiary in the heuristic. If most of the time is spent on the pricing problem, we should reduce the number that this problem is solved. We could do this by adding multiple duties at once. A disadvantage of this method is that we are adding a lot of duties to the RMP, which on its turn could be harder to solve. None of the proposed methods in implemented, as that is out my ability within the given time frame.