Project Assignment in 42114 Integer Programming

Jesper Larsen
jesla@dtu.dk

Technical University of Denmark – Fall 2023
October 2, 2023– version 1.0

Introduction

The project period formally starts on **Monday 2 October** and the report must be handed in at the latest **Monday 6 November at 12:00**. Questions to be answered are clearly marked in "question boxes". Please read the text and the questions carefully before answering them.

Info: Your final report must be uploaded to the appropriate group assignment on DTU Learn of the course. The report is expected to be not more than five pages long **excluding** tables, figures and appendices and must be written in Danish or in English. The project may be solved in groups and permissible group sizes are up to four persons. We strongly encourage you to work in groups of three or four.

1 Crew dispatching

Your Operations Research background has landed you the job as senior dispatch coordinator in the planning department. You are in charge of dispatching engineering teams to different field service assignments. Dispatching the engineers has to be done as cost efficiently as possible (that is as cheap as possible), yet it cannot take too long time for the team to arrive at the destination.

The problem can be defined using a *directed* graph $G = (V, A)$ where V is the set of vertices and A the set of edges. Each edge (i, j) represents a transport opportunity and has an associated cost c_{ij} and a travel time t_{ij}. We assume $c_{ij} \geq 0$ and $t_{ij} \geq 0$. In addition, a source node s and a destination node d is given. The aim is to determine the cheapest path (itinerary) from s to d such that the accumulated travel time does not exceed T hours. As an example your new manager shows you the following small example (see Figure 1).

In the example you need to find the cheapest itinerary for an engineering team that needs to go from the HQ in node 1 to the destination at node 6 with a travel time of at most 10 hours. For each edge (i, j) two values are given; the first one is the cost of travelling along this edge and the second is the time in hours it takes to traverse this edge.

A feasible solution to the problem in this case is a path from node 1 to node 6. We will use the terms path and itinerary interchangeably. A path is a sequence of edges where one edge starts in the node of its predecessor, the first edge starts in the predefined source and the last edge finishes in the predefined destination. An example of a path from 1 to 6 could be to use edges $(1, 3), (3, 5), (5, 6)$ or $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$. This path has a total cost of 24 and takes 8 hours to complete.

Your manager has produced a binary programming model of the problem. This model has only one set of variables, namely a binary variable x_{ij} for each edge (i, j). This variable is 1 if

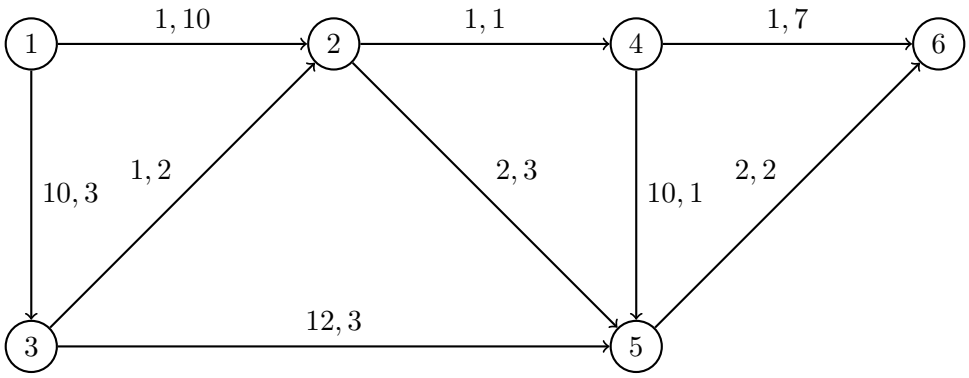

Figure 1: Example problem. Values on the edges are c_{ij}, t_{ij}

edge (i, j) is used in the solution and 0 otherwise. The manager's attempt looks like this:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
& \sum_{(i,j)\in A} t_{ij}x_{ij} \leq T \\
& \sum_{(s,j)\in A} x_{sj} = 1 \\
& \sum_{(i,d)\in A} x_{id} = 1 \\
& x_{ij} \in \{0, 1\} \quad \text{for } (i, j) \in A
\end{aligned}$$

Question 1

In the manager's model a constraint is missing. Whenever a node is used by the path there need to be exactly one entering edge and one leaving edge. Adding such a constraint/set of constraints does not require any new variables. Determine these constraints.

Question 2

To get an idea of the optimal solution for the example in Figure 1 solve the LP relaxation. State the optimal solution of the LP relaxation and the value of the optimal solution.

Your next step would be to use LP-based branch and bound, but you realise that if it was not for the annoying time constraint the problem would be really easy to solve. There are special algorithms for the problem without the time constraint and just solving the LP relaxation of that problem would give you integer solutions.

In order to get rid of the time constraint you decide to use Lagrangian relaxation, where "difficult constraints" are removed and instead punished in the objective function. Denote the Lagrangian multiplier u (we only need one as we only relax a single constraint) and the value of the optimal solution of the Lagrangian relaxation $z(u)$.

Question 3

Write up the Lagrangian relaxation of your binary programming model from Question 1 relaxing the time constraint. Remember the relaxed term is transferred to the objective function as RHS minus LHS. The term should be included in the objective function so that $u \geq 0$.

If $u = 0$ you find the cheapest path from source to destination without any consideration to the travel time.

Question 4

What is the value of the cheapest path in the example when $u = 0$? In general (not only the example): Looking only at the real cost of the itinerary (disregarding the term introduced by the Lagrangian relaxation), what happens as u is increased? And what about travel times when increasing u?

Clearly as u is gradually increased from 0 different itineraries will become the optimal solution to the Lagrangian relaxation. In order to get a better understanding of the problem you decide to figure out what is the value of the cheapest itinerary for a given value of u. This can be done by enumerating the potential itineraries in the network and determine the objection function value.

Question 5

For the example complete Table 1 below. What is the best dual bound that can be achieved?

Range of u	Path cost $\sum c_{ij}$	Travel time $\sum t_{ij}$	$z^L(u)$
$0 \leq u \leq$?			
.	.	.	.
.	.	.	.
.	.	.	.

Table 1: Fill out the table. Column 1 is the range in which the given u values leads to the best Lagrangian relaxation solution. Column two is the accumulated cost and column three is the travel time cost. With the values of column two and three you also have the Lagrangian objective function $z^L(u)$ which goes into column four.

Question 6

Looking at your table from question 5 what is the Lagrangian Dual of the example problem. Is the value of the Lagrangian dual a surprise? Why/why not?

2 Crew Work Assignment

As part of your job in the planning department you must also oversee the distribution of tasks between teams when several teams dispatched for large safety inspections of warehouses, shopping malls and other large infrastructures.

The problem of assigning tasks to crew can be proposed in the following way: We have a set S of n tasks and we have k teams that need to take care of the n tasks. In addition, for safety reasons a team can only handle consecutive tasks in the set of tasks. That means that we cannot assign tasks $1, 3$ and 4 to a team as that leaves a gap consisting of task 2. Therefore, the assignment of a team can be characterised by the first and the last task to be handled.

For task i s_i denotes the time in minutes it takes to carry out the task. The aim is now to divide the set S of tasks into k assignments, one for each team, such that the maximum accumulated time of the tasks over all the assignments is minimised.

An example could be:

$$S = \{100, 200, 400, 500, 900, 700\} \text{ and } k = 3$$

Here we have six tasks and three teams. One way to make the assignments could be to let one team handle task 1, one team tasks $2, 3, 4, 5$ and 6, and finally give the last task to the third team. The total time allocated to team one is 100 minutes, 2000 minutes for team two and 700 minutes for team three. The objective function would then be 2000 as this is the largest value, and this we seek to minimise.

Dynamic Programming

This problem can be solved by dynamic programming. First let us consider the optimal substructure of the problem. Here we notice that the kth assignment starts after we place the $(k-1)$st "divider". If we were to construct an optimal solution then the question could be to look at where we can place the last divider. It can be placed between some tasks, suppose between ith and $(i+1)$st task. The cost can now be split into two parts:

- cost of the optimal way to partition the tasks from 1 to i into assignment, so the optimal solution to an identical (but smaller) problem.
- cost of the last assignment.

Question 7

What is the cost of the last assignment given that we place the divider between i and $i+1$?

Now let $C(i, j)$ be the minimum cost over all assignments of the first i tasks into j assignments for $1 \leq i \leq n$ and $1 \leq j \leq k$. Given the definition of $C(i, j)$ we can build our dynamic programming method. Base cases are $C(1, j)$ for $1 \leq j \leq k$ and $C(i, 1)$ $1 \leq i \leq n$. The recurrence will be:

$$C(i, j) = \min_{1 \leq i' < i} \max\{C(i', j-1), \text{"cost of the last assignment"}\}$$

With the recurrence we can put the different parts together to one dynamic programming method.

Question 8

- How would you compute the values of the base cases?
- To compute the $C(i, j)$ you will fill in a table, one value for each $C(i, j)$. How many entries will there be in the table?
- Where do I find the value of the optimal solution in the table?
- Worst case, hoe many other entries does an entry depend upon?

Question 9

Compute the value of the optimal solution for the example on page 4. Show the entire table.

Often in dynamic programming the first thing we find is the value of the optimal solution, and then afterwards a solution with this value has to be constructed.

Question 10

How would you find the optimal solution given the dynamic programming given? You do not have to do it on the example, but give a general description for the problem on how to construct the solution.

Integer Programming

An alternative to a dynamic programming method would be to make an integer programming model. We will start working on the integer programming solution by removing the requirement that tasks in an assignment have to be consecutive. The integer programming model will lead to an LP-based branch-and-bound, and in order to give it a good start it sometimes makes sense to make a greedy heuristic to generate a good bound.

Question 11

Is the problem, defined by removing the requirements that tasks in an assignment have to be consecutive, a relaxation of the originally defined problem? Why/why not?

Question 12

We still look at the problem defined by removing the requirements that task have to be consecutive. Present a greedy heuristic and use it on the example from Question 9. Is the bound generated by the greedy heuristic a primal or a dual bound? And is it an upper or lower bound?

Now on to the integer programming model. For the model you have a binary variable x_{ia} which is 1 if task i is in assignment a and 0 otherwise. You are not allowed to have any other binary variables in your model. With this set of variables we need to model that each task has to be assigned to exactly one assignment, which can be stated ad:

$$\sum_{p=1}^{k} x_{ip} = 1$$

And then we also need to ensure that each assignment contains at least one task:

$$\sum_{i=1}^{n} x_{ip} \geq 1$$

The rest of the constraints need to model the objective function.

Question 13

Formulate a mixed integer programming model for the problem where the tasks does not have to be consecutive. What is the optimal solution to the problem based on the data from Question 9.

Now we shift back to the problem where assignments can only consist of consecutive tasks. The "consecutiveness" is not easy to model in an integer programming model. One approach is to generate all feasible consecutive sequences. We can make a matrix D where each column is a feasible assignment, and each row represents a task. If $D_{ij} = 1$ then task j is part of assignment i and 0 otherwise. Let W be the number of feasible assignments, that is, the number of columns in the matrix. As soon as I have defined the content of an assignment I can also easily compute the cost. We will denote the cost of assignment i c_i.

Given the definition of W and D we can define the constraints making sure that all tasks are part of a chosen assignment by defining a binary variable x_i for each feasible assignment, that is, $i = 1, \ldots, W$. We have that $x_i = 1$ if assignment i is part of the solution and 0 otherwise. Modelling that each task need to be part of a solution can then be modelled as:

$$\sum_{i=1}^{W} D_{ij} x_i = 1$$

for each task j.

Question 14

Formulate an integer programming model for the problem. Implement the model for the example problem. Attached to the project assignment is also a preliminary Julia file that can help you. Your code should be attached as an appendix to the report (also if you choose to solve the problem in anther modelling language than Julia). Find the optimal solution of the example problem. Do you, in a general setting, see any disadvantages of this approach (enumerating all assignments with consecutive tasks)?