Delivery of expiring products CO 2021/2022

January 3, 2022

This document describes the problem for the case of Combinatorial Optimization 2021-2022. In addition, the formats of the instance and solution data are presented.

Problem description

This case is based on a problem faced by a company that delivers groceries. The planning horizon consists of a period of consecutive days, numbered 1, 2, and so on.

Customers place an order (request) for several products on a specific day in the time horizon. So a request consist of a day, delivery location, and the ordered amount of each product measured in kilograms, see the following example:

request	**day**	**location**	**strawberries**	**potatoes**	**milk**
1	1	5	0	0	8
2	1	6	0	2	0
3	2	4	2	4	3
4	2	5	1	3	0
5	3	6	1	1	0

In this example, there is a customer in location 5 that has placed two requests with different delivery dates: on day 1 they need milk, and on day 2 they need potatoes and strawberries. A request must be fulfilled in one stop, that is: everything that was ordered is delivered, and it is not allowed to split the products from one request over multiple delivery moments.

All products begin their journey at a central *depot*. Products are first transported by truck to a number of *hubs*. This truck transport happens overnight, between 0:00 and 08:00. Afterwards, from 08:00 till 17:00, the products are

(a) Truck

(b) Van

transported from hubs to customers, by van. All products must go via a hub; it is not possible to go from the central depot to a customer directly.

Your task is to make routes such that all customer requests are fulfilled. A route is a sequence of stops within one day, that starts and ends at the same location. For a truck route, you need to determine this sequence of stops as well as how much of each product the driver should load; for a van route it is assumed that the driver always loads exactly the required amount. You need to ensure there is enough of each product at the hub at the right day. The product does not necessarily have to be delivered to the hub the night before: each product stay fresh for a given number of days. For example, if a product stays fresh for three days it can be put in the van 0, 1 or 2 days after it comes out of the truck. You may assume the van drivers use a first-in-first-out policy. At the depot, unlimited product is available of each type. The number of trucks and vans is also unlimited; however, using them does incur a cost. The ultimate goal is to find a feasible solution with minimal cost (further explained below).

In order to determine the traveled distances, integer coordinates are provided for the depot, the potential hubs, and the customer locations. The distance between coordinates (x_1, y_1) and (x_2, y_2) is defined as $\left\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \right\rceil$, i.e., the ceiling of the Euclidean distance.

Feasibility

A solution is feasible if all requests are delivered. Moreover, each truck route begins and ends at the depot, and a van route begins and ends at the *same* hub. It is allowed to use the same van at another hub on the next day, and you can ignore those relocation costs. It is *not* allowed for a vehicle to visit its starting point several times in a day to reload. A vehicle never has more product on board than their capacity allows. Vehicles travel at most a predefined maximum distance within a day.

Costs

The objective is to minimize the total cost. There are costs per unit of distance traveled by truck, for using a truck for a day, and for using a truck at all during

the planning horizon. Additionally, there are costs per unit of distance traveled by a van, for using a van for a day, and for using a van at all during the planning horizon.

Part I

Every customer order must be delivered on the desired day.

Clearly describe your algorithm and analyze its run time in Big-O notation. In your report, add a table with the costs of your best found solutions to Instance 1-10. Also include the .txt files of the solutions as separate documents.

Part II

Consider the problem of Part 1, but now with the possibility of delivering a customer order on an earlier day than desired. If the delivery is n days early, this incurs a penalty that is exponential in the number of days: p^n where the coefficient p is specified in each problem instance. It is not possible to delay a customer's order.

Clearly describe the changes made to the algorithm from Part 1. Explain whether the run time of your algorithm is polynomial or exponential in the length of the planning horizon? In your report, add a table with the costs of your best found solutions to Instance 11-20. Also include the .txt files of the solutions as separate documents.

Part III

Consider the problem of part II, but now each customer can only be serviced from a predefined subset of all hubs. Moreover, you can reduce costs by closing one or several hubs. Each hub that you use incurs a specified cost which is added to the objective value.

Clearly describe the changes made to the algorithm from Part 1.

In your report, add a table with the costs of your best found solutions to Instance 21-30. Also include the .txt files of the solutions as separate documents.

Format of the instance data

Within each section, the entries are sorted by their (consecutive) IDs. All IDs in the input are positive integers, and the depot has location ID 1 in every instance.

```
DATASET = CO2022_11

DAYS = 3
TRUCK_CAPACITY =  20
TRUCK_MAX_DISTANCE = 550
VAN_CAPACITY = 10
VAN_MAX_DISTANCE = 400

TRUCK_DISTANCE_COST = 10
TRUCK_DAY_COST = 250
TRUCK_COST = 100000
VAN_DISTANCE_COST = 5
VAN_DAY_COST = 400
VAN_COST = 20000

DELIVER_EARLY_PENALTY = 5
```

This is the penalty for delivering a request a day early. In this example, delivering two days early gives a penalty of 25. If P = 0 this should be interpreted as: delivering early is not allowed.

```
PRODUCTS = 3
1 2
2 14
3 4
```

The product IDs are followed by the number of days it stays fresh. In this example, it is reflected that strawberries expire quicker than potatoes.

```
HUBS = 2
1 0 3,5,6
2 100 1,2,4,5,6
```

The hub IDs are followed by the cost of having that hub open (in this example, hub 1 is free and hub 2 comes at a cost). The remaining numbers are the requests that are allowed to be served from this hub.

```
LOCATIONS = 6
1 143 190
2 50 121
3 128 196
4 163 246
5 130 134
6 120 149
```

These location entries are an ID followed by x- and y-coordinates. The first location is always the depot. The following locations listed are the hubs, in the same order as the hubs are specified above (in this example location 2 and 3 are hubs). The remainder are customer locations.

```
REQUESTS = 5
1 1 5 0 0 8
2 1 6 0 2 0
3 2 4 2 4 3
4 2 5 1 3 0
5 3 3 1 1 0
```

These request entries represent ID, day, location, amount of product 1, ..., amount of product n. All entries are nonnegative integers.

Format of the solution

Specify your solution in a .txt file with a name that reflects the instance number that it belongs to. There needs to be an entry for each day of the planning horizon, followed by the activities of the trucks and the vans on that day.

```
DATASET = CO2022_11

DAY = 1
NUMBER_OF_TRUCKS = 1
1 H2 0,2,8 H1 2,4,3
```

The first number is the truck ID, followed by the hubs that it visits. Hub IDs are preceded with a capital H to make the lines easier to read for humans (not necessarily computers). The numbers separated by commas represent the amount of each product that the truck brings to that hub.

```
NUMBER_OF_VANS = 1
1 H2 1 2
```

The first number is the van ID, followed by which hub it belongs to that day. In this example, the van belongs to hub 2 (H2). Then follows a sequence of requests that it serves, in this case it first delivers request 1 and then request 2. The load of the vans is not specified; it is assumed to be exactly the amount corresponding to those requests.

```
DAY = 2
NUMBER_OF_TRUCKS = 1
1 H2 2,4,0
NUMBER_OF_VANS = 2
1 H1 3
2 H2 4

DAY = 3
NUMBER_OF_TRUCKS = 0
NUMBER_OF_VANS = 1
1 H2 5
```

In another example with six hubs and three truck routes on the same day, the syntax would be:

```
NUMBER_OF_TRUCKS = 3
1 H2 4,10,1 H6 0,9,9 H4 8,1,0
2 H1 3,3,0
3 H5 10,9,9 H3 12,8,7
```

Validator

A so-called validator is provided: a python program that runs with an instance file and a solution file. It computes whether the solution is feasible, and calculates the total cost of that solution. You can use this to check that your own computations are correct. Moreover, it is very important that all solutions that you submit are accepted by the validator!