DM872 assignment 2

sagra16
(Valdemar Grange - 081097-2033)

June 2020

It should preemptively be noted that the `ipynb` notebook supplied along side with the code shows results and the way the code was developed.

Task 1

The (capacitated vehicle routing problem) CVRP can be formulated as a set-partitioning problem as follows.

$$\begin{aligned} \text{minimize} \quad & \sum_{r \in \Omega} c_r \theta_r \\ \text{subject to} \quad & \sum_{r \in \Omega} a_{i,r} \theta_r = 1, \quad \forall i \in N \\ & \sum_{r \in \Omega} \theta_r \leq m \\ & \theta_r \in \{0, 1\}, \qquad \forall r \in \Omega \end{aligned}$$

Compared to the tasked problem we need a couple of things to complete the model. We implicitly only allow routes that satisfy the capacity limit by only allowing such sets in the problem.

The "additional" aspect is the time windows, which we can model quite easily by the following constraint.

$$\theta_r \, a_{i,r} \, q_{i,r} \leq l_i, \qquad \forall i \in N, \forall r \in \Omega$$

We let $q_{i,r}$ denote the time it takes to reach customer i in route r, where we include accounting for waiting for each customer in the route to be ready (eg the time window beginning). This can simply be implemented by observing `makeQ` from the source code. Observe that we can realize that this constraint can be applied before solving the model, meaning that effectively we can filter out solutions that do not satisfy this constraint. The optimal solution yields an objective value of 41.

Task 2

By introducing linear relaxation we can find a good, but not necessarily optimal solution. We perform linear relaxation by relaxing the binary constraint, such that our model becomes.

$$\begin{aligned} \text{minimize} \quad & \sum_{r\in\Omega} c_r\theta_r \\ \text{subject to} \quad & \sum_{r\in\Omega} a_{i,r}\theta_r = 1, \quad \forall i \in N \\ & \sum_{r\in\Omega} \theta_r \leq m \\ & \theta_r \geq 0, \qquad\qquad \forall r \in \Omega \end{aligned}$$

The result from the above model yields $m = 3$ which is not a violation of the original problem, furthermore an objective value of 41, which is in fact the same as the IP model. Luckily we did reach an integer solution which must mean that our polytope's best value is also the integer solution! I tried to not preprocess the time-window constraint, which did not yield an IP solution, such that it needed rounding but luckily the LP solution was integer (because of unimodularity?). If we inspect the produced routes:

$$\begin{bmatrix} [0 & 1 & 0 & 1 & 0 & 0 & 0 & 0] \\ [1 & 0 & 1 & 0 & 0 & 0 & 0 & 1] \\ [0 & 0 & 0 & 0 & 1 & 1 & 1 & 0] \end{bmatrix}$$

Which in fact is exactly the same as the IP solution:

$$\begin{bmatrix} [0 & 1 & 0 & 1 & 0 & 0 & 0 & 0] \\ [1 & 0 & 1 & 0 & 0 & 0 & 0 & 1] \\ [0 & 0 & 0 & 0 & 1 & 1 & 1 & 0] \end{bmatrix}$$

A Lagrangian relaxation might fit better in some more complicated case where LP relaxation would not yield a good enough solution, since lagrangian is both tighter and does not pose so many questions about solution correctness. I will relax the amount of present vehicles.

$$\begin{aligned} & f(\lambda) = \\ & \text{minimize} \quad \sum_{r\in\Omega} c_r\theta_r - \lambda\left(\sum_{r\in\Omega}\theta_r - m\right) \\ & \text{subject to} \quad \sum_{r\in\Omega} a_{i,r}\theta_r = 1, \qquad\qquad \forall i \in N \\ & \qquad\qquad \theta_r \in \{0,1\}, \qquad\qquad \forall r \in \Omega \end{aligned}$$

To solve the problem finding the best λ for the lagrangian relaxation, I will use the Held and Karp method.

$$\theta = \mu\frac{z_{LR}(\lambda^k) - z}{\sum_i \lambda_i^2}$$

The solution found from the gradient decent with 10 iterations and $\mu = 2$ was $\lambda = 0$, the code can be viewed for my solution. The actual solution reached an objective of 41 with the same three rows as with the IP and LP relaxed models. Finding the specific sequence of customers to visit is simply a shortest path problem on each vehicle route. Of course the shortest-path algorithm should include the time-window constraint as the cost might not be the same as the travel-time.

Task 3

Like in task 2, the master problem is actually just the linear relaxation of the original problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{r\in\Omega} c_r\theta_r \\ \text{subject to} \quad & \sum_{r\in\Omega} a_{i,r}\theta_r = 1, \quad \forall i \in N \\ & \sum \theta_r \leq m \\ & \theta_r \geq 0, \qquad\qquad \forall r \in \Omega \end{aligned}$$

The restricted master problem is a subset of the master problem where we do not yet consider a majority of the available problem. We denote this arbitrary subset by $\hat{\Omega} \subset \Omega$.

$$\begin{aligned} \text{minimize} \quad & \sum_{r\in\hat{\Omega}} c_r\theta_r \\ \text{subject to} \quad & \sum_{r\in\hat{\Omega}} a_{i,r}\theta_r = 1, \quad \forall i \in N \\ & \sum \theta_r \leq m \\ & \theta_r \geq 0, \qquad\qquad \forall r \in \hat{\Omega} \end{aligned}$$

We also need to develop a pricing model for evaluating the sub-problem. The sub-problem should minimize the reduced cost, where the negative reduction means that the column we are inspecting will enter the basis. In fact if a non-negative reduced cost is found, the solution is optimal. We must construct the dual to our problem, to determine the reduced cost. The dual variables can be observed by inspecting the primal tableau. In the problem we have two constraints, such that we have two sets of dual variables. One of the constraints is a singleton (the $\leq m$ constraint), which I will denote λ_0. The second set of constraints is over $i \in N$, so I will denote them $\lambda_{1,r}$. Our reduced cost of a variable θ_r can be realized by $\hat{c}_r = c_r - \lambda_T A_r$. In fact, constraints will be a valid pathing and with the reduction in cost it becomes a shortest path problem. Every vertex must exactly consume one edge and produce one, or rather every vertex must be the sink of exactly one edge and the source of one. Additionally we use big-M to model capacity constraints, and include the time window constraint. Let A be the set of edges, V be the set of vertices, Q the the max capacity and $x_{i,j}$ be the activation variable of path i, j. Finally let $y_i \in \mathbb{R}$ be the load on the vehicle when leaving vertex $i \in V$.

To express the "path taken" we must accumulate and check all time window constraints for violation.

$$\sum_{0,j\in A} x_{0,j}t_{0,j} + \sum_{j,i\in A} x_{j,i}t_{j,i} \cdots \sum_{s,z\in A} x_{s,z}t_{s,z} \leq l_z \sum_{s,z\in A} x_{s,z} \quad \forall z \in V$$

As one might notice, this is not easily expressible, as the amount of sum's should be equal to the length of the path from 0 to z.

1. One solution is to make the shortest-path algorithm aware of the domain of the routes, in a sense pre-generating all possible permutations of path's and excluding bad ones.

2. Another could be to check the constraint after the problem has been solved, if the constraint is violated then solve it again with the previously chosen path excluded.

3. A solution could also be to introduce an additional variable like y, but for time window's as this can also be viewed as a resource.

4. There also exists a polynomial time algorithm to solve this based on labelling in chapter two of [1], but seems way out of scope of this assignment.

I will use item 3, as it seems the most appropriate (but not simple, that would certainly be item 2). To do this I will introduce a new variable t_i, which will be the time spent travelling to position t_i. t_i should be bounded by $0 \leq t_i \leq l_i$, where l_i is the latest possible time to arrive. This method uses big-M, denoted by M, and is formulated such that the solver must assign the time-spent correctly to the variable. Additionally we still need to account for waiting before the customer "opens" in case that the vehicle is too early, which is $max(e_i, t_i)$. We can model this by.

$$max(e_j, time_{i,j} + t_i) + Mx_{i,j} \leq t_j + M \quad \forall i, j \in A : j \neq 0$$

Once again there is a problem, max is not linear thus we must find a way around this. Luckily we are minimizing and want to find the maximum, so we can simply introduce a variable to the model which specifies the time of delivery j given we are coming from i.

$$\alpha_{i,j} \geq e_j \wedge \alpha_{i,j} \geq time_{i,j} + t_i \quad \forall i, j \in A : j \neq 0$$

The solver will pick the smallest value to assign to α that is, the maximum of the two, without compromising the solution quality. Keep in mind that "serving time" can also easily be modelled here if wished by simply adding it to α. The time constraint can now be expressed more formally as.

$$\begin{array}{ll} \alpha_{i,j} \geq e_j \wedge \alpha_{i,j} \geq time_{i,j} + t_i & \forall i, j \in A : j \neq 0 \\ \alpha_{i,j} + Mx_{i,j} \leq t_j + M & \forall i, j \in A : j \neq 0 \\ 0 \leq t_i \leq l_i & \forall i \in V \end{array}$$

$$\alpha \in \mathbb{R}$$

The problem is as follows.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in A} (c_{i,j} - \lambda_{1,i}) x_{i,j} - \lambda_0 & \\
\text{subject to} \quad & \sum_{i,k \in A} x_{i,k} = \sum_{k,j \in A} x_{k,j} & \forall k \in N \\
& \sum_{0,j \in A} x_{0,j} = 1 & \\
& \sum_{i,0 \in A} x_{i,0} = 1 & \\
& & \\
& \alpha_{i,j} \geq e_j \wedge \alpha_{i,j} \geq time_{i,j} + t_i & \forall i,j \in A : j \neq 0 \\
& \alpha_{i,j} + M x_{i,j} \leq t_j + M & \forall i,j \in A : j \neq 0 \\
& 0 \leq t_i \leq l_i & \forall i \in V \backslash \{0\} \\
& & \\
& y_i + q_j + M x_{i,j} \leq y_j + M & \forall i,j \in A : j \neq 0 \\
& 0 \leq y_i \leq Q & \forall i \in N \\
& & \\
& x_{i,j} \in \{0,1\} & \forall i,j \in A \\
& \alpha_{i,j} \in \mathbb{R} & \forall i,j \in A
\end{aligned}
$$

Notice that we cannot preemptively remove path's to be checked based on time windows, like before. This is because the shortest path algorithm works different in that the permutation of visiting order dictates when we arrive, thus we must include the time-window constraint.

Task 5

Initially we have the subset $\hat{\Omega} \subset \Omega$.

r_r	$\hat{\Omega}_r$	c_r
r_0	[0 0 0 1 0 1 1 0]	15
r_1	[0 0 0 0 1 1 0 1]	13
r_2	[1 0 0 0 1 0 0 0]	22
r_3	[0 1 0 1 1 0 0 0]	17
r_4	[1 0 1 0 0 0 0 0]	15
r_5	[1 1 0 0 0 0 1 0]	22
r_6	[0 1 1 0 0 0 0 1]	18
r_7	[0 1 0 1 0 0 0 0]	10
r_8	[0 0 1 0 1 0 1 0]	15
r_9	[0 0 0 0 1 0 1 1]	13
r_{10}	[0 1 1 1 0 0 0 0]	17
r_{11}	[0 0 0 0 0 1 1 1]	13

And the solution values after the first iteration, the first r rows are λ_1's and the final ≤ 3 row is λ_0.

λ \ c_r	15	13	22	17	15	22	18	10	15	13	17	13	
13.33	0	0	1	0	1	1	0	0	0	0	0	0	$= 1$
7.66	0	0	0	1	0	1	1	1	0	0	1	0	$= 1$
7.0	0	0	0	0	1	0	1	0	1	0	1	0	$= 1$
7.66	1	0	0	1	0	0	0	1	0	0	1	0	$= 1$
6.99	0	1	1	1	0	0	0	0	1	1	0	0	$= 1$
6.33	1	1	0	0	0	0	0	0	0	0	0	1	$= 1$
6.33	1	0	0	0	0	1	0	0	1	1	0	1	$= 1$
5.0	0	1	0	0	0	0	1	0	0	1	0	1	$= 1$
-5.33	1	1	1	1	1	1	1	1	1	1	1	1	≤ 3
θ	0.33	0.66	0.0	0.0	0.66	0.33	0.0	0.33	0.0	0.33	0.33	0.0	

First I introduce the route $[0, 3, 4, 6, 0]$, or rather $[0, 0, 1, 1, 0, 1, 0, 0]$, which yields an objective value of -4.33. Moreover the algorithm will keep solving the shortest path sub-problem and pick the best reduced cost to approach the point of primal and dual feasibility. The final objective is 37 with a fractional solution of θ_r.

θ_{11}	0.5
θ_{12}	1.0
θ_{19}	0.5
θ_{20}	0.5
θ_{30}	0.5

Which is quite odd, when the linear relaxation and lagrangian gave optimal integer solutions.

References

[1] B. Petersen and D. Pisinger, *Shortest paths and vehicle routing*. PhD thesis, PhD thesis, 2011.