Writing and Implementing the Direct Formulation (DF) for Job Vehicles

After completing the subproblem LSA and the colgen loop, it is time to evaluate what the true, exact optimal solution is. Let's recall the formulation given for the set partitioning with column generation method:

Set Partitioning Formulation (over routes)

There are three sets of decision variables.
z^q telling us if we use each route $q \in \mathcal{Q}$, and
time variables y_{it}^S and y_{it}^E telling us if we already started/finished the job *i* by time *t*.
In colgen they are NOT binary, only 0 <= var <= 1. In DF they are binary.

Meanwhile, we use a few parameters.
C^q denotes the cost of route *q*.
u_i^q tells us if route *q* intersects node *i* at some point.
K is the number of vehicles we have.
δ_{it}^q references whether route q is at node *i* at time *t*.
T_i^S is the start of time window for job *i*.
T_i^E is the end of time window for job *i*.
τ_i^W is the duration of the time window at job *i*.

MINIMIZE $\sum_{q \in \mathcal{Q}} C^q z^q$ - the sum of route distances traversed by all vehicles.

SUCH THAT

- JOB CONSTRAINT -
$\sum_{q \in \mathcal{Q}} u_i^q z^q = 1 \quad \forall i \in \mathcal{I}$ - exactly one vehicle visits each job. [DUAL: π_i]

- VEHICLE CONSTRAINT -
$\sum_{q \in \mathcal{Q}} z^q \leq K$ - there are at most num_of_vehicles routes. [DUAL: ρ]

- TIME CONSTRAINTS -
$y_{it}^S \leq y_{i,t+1}^S \quad \forall i,t$ - if a job has started it will have started at a later time
$y_{it}^E \leq y_{i,t+1}^E \quad \forall i,t$ - if a job has ended it will have ended at a later time
$y_{it}^S = 0 \quad \forall t \leq T_i^S \; \forall i$ - a job must have not started before the time window
$y_{it}^E = 1 \quad \forall t \geq T_i^E \; \forall i$ - a job must have ended after the time window

$\sum_t (y_{it}^S - y_{it}^E) \geq \tau_i^W \; \forall i$ - for all jobs, the completion time is at least the job load

$y_{it}^S - y_{it}^E \leq \sum_q z_q \delta_{it}^q \quad \forall i,t$ - job done timespan cannot exceed available timespan

Direct Formulation (over arcs)

Like before, there are 3 sets of variables. Now the identity of the vehicle matters, because we cannot have a route [1, 4, 5] meant for one vehicle having arc [1, 4] traversed by vehicle 1 and [4, 5] traversed by vehicle 2. All arcs are denoted a and go from one $(i_1, t_1) \rightarrow (i_2, t_2)$.

There are three sets of decision variables, and they read similarly to before.
x_{ka} telling us if vehicle number k goes to arc $a \in \mathscr{A}$.
Time variables y_{it}^S and y_{it}^E defined as before!

For convenience, define $\mathscr{A}^+(n)$ as the list of nodes that go out of n (outgoing). Also define $\mathscr{A}^-(n)$ as the list of nodes that goes into n (incoming).

There is one parameter: c_a, denoting the cost of the arc (distance).

Objective: MINIMIZE $\sum_k \sum_a c_a z_{ka}$. Sum over all vehicles and over all arcs: if we visit that arc, then add its cost.

Constraints:

- VEHICLES -
$\sum_{(0,t)} \sum_{a \in \mathscr{A}^+((0,t))} z_{ka} = 1 \quad \forall k$ - Every vehicle only leaves the depot once. Exactly once, in fact.
Good to test if you say <= if there's any difference.

$\sum_{(n+1,t)} \sum_{a \in \mathscr{A}^-((n+1,t))} z_{ka} = 1 \quad \forall k$ - Every vehicle only goes back into the depot once. *Same cautionary note as the earlier constraint.*

$\sum_{a \in \mathscr{A}^-(n)} z_{ka} = \sum_{a \in \mathscr{A}^+(n)} z_{ka} \quad \forall k, \forall (i,t) \text{ st } 1 \leq i \leq n$ - For each vehicle, flow is consistent: if you enter a node, you must leave that node.

$\sum_k \sum_{a \in \mathscr{A}^-(n)} z_{ka} = 1 \quad \forall (i,t) \text{ st } 1 \leq i \leq n$ - For each job node, exactly one vehicle enters it.

- TIME WINDOWS -
These read very similar to before.
$y_{it}^S \leq y_{i,t+1}^S \quad \forall i, t$ - if a job has started it will have started at a later time
$y_{it}^E \leq y_{i,t+1}^E \quad \forall i, t$ - if a job has ended it will have ended at a later time
$y_{it}^S = 0 \quad \forall t \leq T_i^S \ \forall i$ - a job must have not started before the time window
$y_{it}^E = 1 \quad \forall t \geq T_i^E \ \forall i$ - a job must have ended after the time window

$\sum_t (y_{it}^S - y_{it}^E) \geq \tau_i^W \ \forall i$ - for all jobs, the completion time is at least the job load

The last constraint will be modified a bit.

$y_{it}^S - y_{it}^E \leq \sum_k z_{k,(i,t)\to(i,t+1)} \; \forall i,t$ - job done timespan cannot exceed available timespan - here, $(i,t) \to (i,t+1)$ is an arc which goes from the first point to the next.

Further notes on each constraint are provided in section *[3.3] Constraints*.

Implementing DF

The direct formulation takes the same setup as the set partitioning one. We take the old Jupyter notebook, clone it, and use the first section to do direct formulation.

One of the main steps is to build our entire arc system that tells us all of the valid arcs. Note that the DF is entirely helpless if we actually have an invalid arc (if it takes 5 hours to get from node 6 to node 8, and we have an arc that says (6, 8h) → (8, 12h), then that should've been removed earlier), so we can't just go arc-hopping from everywhere to everywhere.

In theory, since our time goes from 1 to 50, and there are 27 nodes for 25 jobs, we could have up to on the order of $27^2 50^2 = 729 \cdot 2500 = 1'822'500$ nodes. Obviously, given time window restrictions and travel distance limits (mostly the former), the number will be much smaller.

In section *[1] Setup* we build all the necessary variables for the process. Parameters are established in *[1.1] Initialize variables*; locations, time windows, job loads, and distances are created in the namesake *[1.2]*; the node and arc networks which we saw in formulation are made in *[1.3] Create node and arc networks*; and two helper functions (actually only the second is necessary) are made in *[2] Helper Functions*.

Obtaining Results

We will output an array z_ka_v which represents value.(z). This will give us all the 0's and 1's of the most interpretable decision variable.

The array is 25 times 11387, because 11387 is A (the number of arcs).

Step 1 Set up an array that nabs arcs from each vehicle. Each element of the array is a list of arcs that belongs to that element's corresponding vehicle. For each list within the array, traverse the entire z_k and tell me which arcs are active.

Step 2 The resulting list for each vehicle is almost guaranteed not to be in order. Sort these arcs by the first time (second time is fine, same thing).

Step 3 Extract the first location from each arc. This will form a list. Finally, append (n_jobs + 1) to it for the depot.

Results

The result is an absolute answer, 3'023.663.

The routes used are:

0 16 17 18 19 13 14 26
0 1 2 3 26
0 24 25 26
0 20 21 22 23 26
0 4 5 6 7 26
0 8 9 10 26
0 11 12 15 26