Colgen on Jobs and Coverage Simultaneously

We apply column generation on both jobs and coverage together. I want to see how well this method works. For the time being, I will prioritize getting a result/benchmark over theoretical security. For instance, there are bound to be time issues and intricacies that I won't address until I get a solution.

From last time, we created a colgen for job vehicles that looked like the following:

Decision Variable: z^q - if route q is used

Constraint: $\sum_q u_i^q z^q = 1 \ \forall i$ - each job is visited only once

Constraint: $\sum_q z^q \leq B$ - there at least as many job vehicles as there are routes

Objective: minimize $\sum_q C^q z^q$ - total travel distance of all routes

Parameter: C^q - cost of route q
Parameter: B - number of job vehicles
Parameter: u_i^q - if route q passes through job i

I also created a colgen for only coverage vehicles, assuming job routes were finished:

Decision Variable: x^p - if route p is used

Constraint: $\sum_p x^p \leq V$ - there are no more routes than there are coverage vehicles

Constraint: $\sum_p \sum_j L_{ji} y_{jt}^p x^p \geq W_{it} \ \forall i, t$ - when work is being done, there must be coverage

Objective: minimize $\sum_p C^p x^p$ - total travel distance of all routes

Parameter: C^p - cost of route p
Parameter: V - number of coverage vehicles
Parameter: y_{jt}^p - if route p is at spot j at time t
Parameter: L_{ji} - if spot j can cover node i
Parameter: W_{it} - if work is done on node i at time t

The last parameter W_{it}, highlighted in red, is the only thing preventing us from immediately turning this into a double column generation (dcg). W_{it} can only be calculated after we solve job vehicles, but we want to solve job and coverage together. There is a trick that can alleviate this. Notice that in my formulation, if you are at a job location, then you are doing work there, because otherwise you could waste time on the road, and I don't care if you waste time because it won't change the objective. In other words, using the assumption "at a node i → doing work at node i" does not delete any possibilities that would've remained had I not made this assumption.

Therefore, we can encapsulate work by a parameter including time as well, δ_{it}^q, telling us if we are at job i at time t in route q. Then $W_{it} = \sum_q z^q \delta_{it}^q$, so we rewrite the last constraint of the coverage as $\sum_p \sum_j L_{ji} y_{jt}^p x^p \geq \sum_q z^q \delta_{it}^q \ \forall i, t$.

Anything else we need to include? Let's think about it. Routes themselves will already incorporate time information, which we will guarantee beforehand is feasible. The real challenge is guaranteeing feasibility for a new route configuration of both coverage and job in the subproblem.

The full formulation is produced below.

Decision variable: z^q - if job route q is used
Decision variable: x^p - if coverage route p is used

Constraint: $\sum_q u_i^q z^q = 1 \ \forall i$ - each job is visited only once

Constraint: $\sum_q z^q \leq B$ - there at least as many job vehicles as there are routes

Constraint: $\sum_p x^p \leq V$ - there are no more routes than there are coverage vehicles

Constraint: $\sum_p \sum_j L_{ji} y_{jt}^p x^p \geq \sum_q z^q \delta_{it}^q \ \forall i, t$ - when work is being done, there must be coverage

Parameter: C^q - cost of route q
Parameter: B - number of job vehicles
Parameter: u_i^q - if route q passes through job i
Parameter: δ_{it}^q - if route q passes through job i at time t
Parameter: C^p - cost of route p
Parameter: V - number of coverage vehicles
Parameter: y_{jt}^p - if route p is at spot j at time t
Parameter: L_{ji} - if spot j can cover node i

Objective: Minimize $\sum_q C^q z^q + \sum_p C^p x^p$

Completing the Dual
The associated dual variables are produced below.

$\sum_q u_i^q z^q = 1 \ \forall i$ - dual: π_i [I constraints]

$\sum_q z^q \leq B \rightarrow -\sum_q z^q \geq -B$ - dual: ρ [1 constraint]

$\sum_p x^p \leq V \rightarrow -\sum_p x^p \geq -V$ - dual: β [1 constraint]

$\sum_p \sum_j L_{ji} y_{jt}^p x^p$ - $\sum_q z^q \delta_{it}^q \geq 0 \ \forall i, t$ - dual: ξ_{it} [IT constraints]

Primal Objective: Min $\sum_q C^q z^q$ + $\sum_p C^p x^p$ [$Q + P$ variables]

Our primal looks like this:

Objective has variables $[z^q \ \ x^p]$ of length $Q + P$, with coefficients $[C^q \ \ C^p]$ of length $Q + P$.

$C^1 z^1 + \ldots + C^Q z^Q + C^1 x^1 + \ldots + C^P x^P$ (length $Q + P$) (yes I am aware of the C repeats, but conceptually I think you know what I mean)

Two constraints that look like this:
$-z^1 - \ldots - z^Q \geq -B$ (length: Q) (dual: ρ)
$-x^1 - \ldots - x^P \geq -V$ (length: P) (dual: β)
Taken together, the coefficient for each z^q is -1. The coefficient for each x^p is -1.

I constraints that look like this:
$u_1^1 z^1 + u_1^2 z^2 + \ldots + u_1^Q z^Q = 1$ (length: Q) (dual: π_1)
$u_2^1 z^1 + u_2^2 z^2 + \ldots + u_2^Q z^Q = 1$ (length: Q) (dual: π_2)
$\ldots$
$u_I^1 z^1 + u_I^2 z^2 + \ldots + u_I^Q z^Q = 1$ (length: Q) (dual: π_I)
Consider one equation for a specific value i.
The coefficient for each z^q is a single u_i^q.

IT constraints that look like this:
$\sum_p \sum_j L_{ji} y_{jt}^p x^p$ - $\sum_q z^q \delta_{it}^q \geq 0 \ \forall i, t$ (length: $Q + P$) (dual: ξ_{it})

Consider one equation for a specific value i and t.
The coefficient for each z^q is a single $-\delta_{it}^q$. The coefficient for each x^p is $\sum_j L_{ji} y_{jt}^p$.

Now we are going to distill our variables for the dual.

Gathering the Dual
Let's look at a generic formulation for the primal/dual.

PRIMAL:
MIN $b_1 w_1 + \ldots + b_m w_m$ (length: m)
st $a_{11} w_1 + \ldots + a_{m1} w_m \geq c_1$ (dual: x_1)
st ...
st $a_{1n} w_1 + \ldots + a_{mn} w_m \geq c_n$ (dual: x_n) (length: m, height: n)

DUAL:
MAX $c_1 x_1 + \ldots + c_n x_n$ (length: n)
st $a_{11} x_1 + \ldots + a_{1n} x_n \leq b_1$
st ...
st $a_{m1} x_1 + \ldots + a_{mn} x_n \leq b_m$ (length: n, height: m)

Our current formulation looks like:
PRIMAL:
MIN $C^1 z^1 + \ldots + C^Q z^Q + C^1 x^1 + \ldots + C^P x^P$ (length: $Q+P$)
st $-1z^1 - \ldots - 1z^Q + 0x^1 + \ldots + 0x^P \geq -B$ (dual: ρ)
st $0z^1 + \ldots + 0z^Q + -1x^1 - \ldots - 1x^P \geq -V$ (dual: β)
st $u_i^1 z^1 + \ldots + u_i^Q z^Q + 0x^1 + \ldots + 0x^P = 1$ (dual: π_i) - there are I of these constraints!
st $-\delta_{it}^1 z^1 - \ldots - \delta_{it}^Q z^Q + \left(\sum_j L_{ji} y_{jt}^1 \right) x^1 + \ldots + \left(\sum_j L_{ji} y_{jt}^P \right) x^P \geq 0$ (dual: ξ_{it}) - there's IT of these!
Length: $Q+P$, height: $2+I+IT$.

So the DUAL (unkempt, put here as an intermediate step):

MAX $-B\rho - V\beta + \sum_{i=1}^{I} 1\pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} 0\xi_{it}$ (length: $2+I+IT$)

st $-1\rho + 0\beta + \sum_{i=1}^{I} u_i^1 \pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} -\delta_{it}^1 \xi_{it} \leq C^1$ (the first of Q constraints)
st ...
st $-1\rho + 0\beta + \sum_{i=1}^{I} u_i^Q \pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} -\delta_{it}^Q \xi_{it} \leq C^Q$ (the last of Q constraints)

st $0\rho - 1\beta + \sum_{i=1}^{I} 0\pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} \left(\sum_j L_{ji} y_{jt}^1 \right) \xi_{it} \leq C^1$ (the first of P constraints)
st ...
st $0\rho - 1\beta + \sum_{i=1}^{I} 0\pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} \left(\sum_j L_{ji} y_{jt}^P \right) \xi_{it} \leq C^p$ (the last of P constraints)

Now we simplify our dual and throw out zeroes:

MAX $-B\rho - V\beta + \sum_{i=1}^{I} \pi_i$

st $\forall q = 1, \ldots, Q, \ -\rho + \sum_{i=1}^{I} u_i^q \pi_i - \sum_{i=1}^{I} \sum_{t=1}^{T} \delta_{it}^q \xi_{it} \leq C^q$ (this is our first Q constraints)

$\forall p = 1, \ldots, P, \ -\beta + \sum_{i=1}^{I} \sum_{t=1}^{T} \left(\sum_{j} L_{ji} y_{jt}^P \right) \xi_{it} \leq C^p$ (this is our last P constraints)

So we are looking for a new route q^n that violates this criterion; namely, that

$$C^q + \rho - \sum_{i=1}^{I} u_i^q \pi_i + \sum_{i=1}^{I} \sum_{t=1}^{T} \delta_{it}^q \xi_{it} \leq 0.$$

Similarly, we are looking for a new route p^n that has reduced cost

$$C^p + \beta - \sum_{i=1}^{I} \sum_{t=1}^{T} \left(\sum_{j} L_{ji} y_{jt}^P \right) \xi_{it} \leq 0.$$

Subproblem Considerations
This is where it gets tricky. There are a few questions to consider.

Any fixes we need before we can run a subproblem?
For q: no. Everything is presented as a shortest paths, and the variables are straightforward. We borrow the very successful PT NyVA >= algorithm we used before, except that now we have to add ξ_{it} variables. See the question below too.

Thus, for each route: Start with ρ. Then, every time you go to a new node (i, t), add the cost of that distance, subtract π_i ($u_i^q = 1$), and add, for all times you plan to stay there (let's forecast into the future), ξ_{it}.

For p: yes. Currently the formulation is written in terms of i instead of j; bad. So instead we write the last term as
$$\sum_{i \in L(j)} \sum_{t=1}^{T} \left(\sum_{j} L_{ji} y_{jt}^P \right) \xi_{it}$$
where $L(j)$ is all nodes $i*$ that are close enough to j to be covered.

Thus, for each route: Start with β. Then, every time you go to a new spot (j, t), add the cost of that distance. Then, for all times you plan to stay there (forecast into the future) and all nodes $i*$ close enough to j, subtract ξ_{i*t}. (The sum over j isn't intimidating: it's either 1 or 0 depending on whether you're visiting there and you can cover the node $i*$.)

If we use pruning algorithms, should we calculate ξ_{it} before or after we forecast its times?

This is an important question. If we only had π_i, we would calculate it before we decide whether to prune. Why shouldn't it hold for ξ_{it} as well? So my answer is: "Before, meaning forecast into the future when you decide to stay until time t, then calculate the sum of all ξ_{it}." Afterwards, prune.

Is the process of adding q^n and p^n independent?

I think so, there's no such concept as a 'route pair' q^n p^n, especially since there is no fundamental linkage between q^n and p^n. For instance, it may end up that the z- and x -variables don't have overlap at which route index was 1.0, because it's all arbitrary.

Does the order of how I add q^n and p^n affect correctness?

It shouldn't, right? Whether I structure my algorithm to add p^n after q^n after p^n, as is most proper, or I add 10 q^n's and then 10 p^n's, shouldn't matter: because we are dealing with a linear problem, we don't need to worry about concavity or accidentally stumbling on a local minimum unable to hit the global minimum.

If the correct answer is "no", that means I can first generate a set of routes such that feasibility is guaranteed among them (so that eventually I can just generate whatever I want without fear of losing feasibility, because in the worst case we just adopt this old set). In other words, if I generate a list of dummy job routes and dummy feasible coverage routes (in which a vehicle literally 'sits' at that coverage location), then I do column generation repeatedly on the q^n to get something great, then I do p^n, it's guaranteed to converge to the optimum (barring route-splitting). This would be awesome because it's basically the same as my first approach "solve jobs entirely, then solve coverage entirely given those jobs", except with dummy coverage routes.

If you generate q^n or p^n independently, will that affect when we end the algorithm?

My current belief is this: We keep generating, and at any point, if either q^n or p^n report that the least reduced cost >= 0 (aka -1e-8), then generate the other. Until both hit least reduced cost >= 0, we are not done, because maybe if we had a better e.g. coverage route, we can find even better job routes.

(Previous question, before I understood the first point:

Do we add a route set q^n, p^n if both reduced costs are less than 0, or is one of their reduced costs' being less than 0 enough?

In favor of "both": Our decision variables can be thought of as one vector set, which we happened to partition into z^q and x^p for ease. But in reality, they are chained together, and both constraints have to be violated.

In favor of "one": If we violate any constraint at all, it should be added.

We can test this empirically to see what happens, and then determine which course of action was right. This is what I did for μ_{it} when I wasn't sure if we should add it to a vehicle every time t it stayed at node i, or only when it first arrives.)