DF Coverage Formulation after Jobs Solved

We've just solved, using the PT NyVA >= and potentially stem-and-blender, the vehicle routing problem with time windows (VRPTW). Now it's time to integrate jobs.

I have a few mechanisms for doing so. One way is to take a formulation (direct) and use column generation directly on the whole thing, both sets x (coverage arc binary decision variables) and z (same, but for job vehicles). Another is this method: Take a solution we already have from the (quite fast) PT NyVA >=, and build a coverage solution 'around it'. It should be pretty good, though I don't know if we can in theory prove that this will yield a strict optimum.

So let's look at the latter method. Suppose we know what routes will be selected for job vehicles. What parameters can we gather?

Times of Work, W_{it} - it's either 0 or 1, and it tells us if work is being done at node i at time t (since we're solving coverage, we don't care which route is the one containing the work. All that matters is a job is being solved there).

That's it! That's all we need from our routes. If we now think about this problem from the perspective of coverage, all you need to know is that you have to be close enough if $W_{it} = 1$.

From coverage, there are a few more parameters involved.

You can only cover a certain job at location i if you are sufficiently close. So we will define a parameter L_{ji} that is 1 if "spot" (our equivalent of location for coverage, I don't want to write coverage location) j (now our default index letter for coverage) is sufficiently close to node i.

We will also need to know the cost of each arc, which is c_a. This is distance.

Now, onto the formulation itself. Decision variables-wise, we will use the direct formulation first. It will be very hard to build a route system for coverage vehicles, because there are literally no time window constraints, allowing far more routes than for job vehicles. However, colgen can help us circumvent this problem—that's for later.

Coverage Arcs
Among all the parameters, this is the hardest to conceptualize.

Suppose we have V vehicles for coverage. We also have coverage arcs (specifically, m times m+1 times O(T) of them, where m is the number of spots: m+1 choices of first spot, including depot, and m choices for next spot). Our decision variable is x_{va} = 1 if vehicle v uses arc a, and 0 otherwise. Regarding coverage, spot coverage is represented by γ_{jt}: if, at time t, coverage spot j is covered.

Coverage Arcs: Three types. From the origin-depot to some place, from some place to the end-depot, from a non-depot to a non-depot.

Origin-depot to some spot: Start at [0, t]. Go to [j, t+d(0, j)]. t is allowed to start at between 0 and T - 2 * d(0, j) (because you have to make sure you can get back) inclusive.

Some place to end-depot: Start at [j, t]. Go to [m+1, t+d(0, j)]. t is allowed to start at between 1 and T-d(0, j) inclusive (so you can get back in time).

Between non-depots: Start at [j, t]. Go to [j', t+d(j, j')]. t is allowed to start at 1 and end at T-1. Yes, j is allowed to equal j', in which case we add 1 to the time.

Trivial: [[0, 0], [m+1, 0], 0].

Subtle Points
I'd like to cover two subtle points now.

First.

It is critical to note that our arc system for coverage, which incorporates (j, t) going to (j+?, t+?), does not allow for time wasting. Why not? At least in the direct formulation, if you need to stall for time, you will get there and stall for time at the coverage location itself—it can do no harm, and might even be good. Therefore, by not including arcs that waste time or go slowly along the way, we are not unintentionally removing valid solutions, and we are saving ourselves a massive amount of space and time complexity.

Subtly, this is different than in the colgen for job vehicles, in which we can waste a lot of time in between routes if we need to wait for a job window to open. It is beautifully true that in the colgen for coverage routes as well, we will be allowed to waste time because the subproblem LSA will look very similar to that for the job vehicles, and in coverage, if we don't need to arrive too early, and we have some leeway (e.g. if we take 2 distance to get from a place where the last coverage need at the old node is at time 17 and the first coverage need at the new node is at time 23), we'll do the same thing as we did for job vehicles: arrive at 23. The route will "look like" it took 6 units of time to get from the old node to the new node. In truth, we will never know what the route *really* intended to do between then, but it doesn't matter at all, because we've minimized our objective, which is cost, no matter how we spend our time. This is the same tactic that allowed me to delete the y-variables from the jobs problem: the cost is still minimized as long as our route and feasible times don't get compromised.

Second.

Another even more subtle, and pathological, point is that although individually, i.e. considered on their own (job vehicles only, or coverage vehicles only assuming static aka solved job vehicles), it doesn't matter when in a leeway-existing situation the vehicle, job or coverage, actually gets there, and whether it wastes time on the route, when we do consider them together, we might be forced to reconsider our assumption that if

there's leeway, any time you choose, as long as it doesn't change the cost, is fine. Specifically, since my algorithm for the job vehicle routes chooses the earliest possible start times at a new node and wastes as little time as possible, we might accidentally cause coverage to require more vehicles than if we had wasted a little time and arrived later (with no increase to job vehicle route cost), causing total cost to increase. This is possible if the coverage vehicle is forced to travel from one spot to another to ensure coverage, and would've not reached the new coverage spot had the new job started at a sufficiently early time.

For example: Suppose I have a job that takes 4 hours with window [7, 11], and my next job takes 4 hours at [19, 25], and the former is covered by spot A but not spot B while the latter is covered by spot B but not spot A. Furthermore, suppose the time to travel between the jobs is 7 hours, but the time to travel between coverage spots is 10 hours.

My SP algorithm would create a route that, for jobs, went: 7-11 - stay at first job; 11-19 - travel (time is wasted; in this specific scenario it's irrelevant); 19-23 - do second job. Notice that the coverage vehicle that stayed at spot A could not sufficiently cover this job route, because it leaves spot A at 11 at the earliest, and gets to spot B at 11 + 10 = 21. But this is too late, as the second job has already started! As a result, I will need another coverage vehicle, wasting time.

But imagine I instead wasted even more time and started the second job only at 21, ending at 25. Then my route is: 7-11 - first job; 11-21 - travel (waste a lot of time); 21-25 - second job. Now, one single coverage vehicle is enough, because it stays at spot A until 11, at which point it takes 10 hours to get to spot B, perfectly covering the second job.

For now, I won't worry about the second point. It will be addressed down the line, but it's not our priority to resolve right now. My goal is to build a functioning algorithm.

The Formulation

Our objective is to minimize the total cost traveled by coverage vehicles. $\sum_{v} \sum_{a} c_a x_{va}$.

Our constraints are as follows.

Flow Constraints

For all vehicles v, we exit the origin-depot exactly once. (This allows us to use the trivial route that stays at the depot, because the origin-depot and the end-depot are locationally numerically distinct.)

The constraint is: $\sum_{(0,t)^+} x_{va} = 1 \ \forall v$, where (spot)$^+$ refers to the outgoing arcs from that spot; for instance, if spot 3 has an arrow towards spot 4, then 4 is in the set 3^+.

Similarly, we go into the end-depot exactly once, $\sum_{(m+1,t)^-} x_{va} = 1 \ \forall v$, where (spot)$^-$ refers to all arcs that go into that spot.

And for all in-between spots, flow is preserved. For all vehicles v and all spots $m = (j,t),\ i \neq 0,\ m+1$, $\sum_{a \in m^-} x_{ka} = \sum_{a \in m^+} x_{ka}$.

Coverage Constraints

Remember our variables: W_{it} is if there is work done at node i at time t.
L_{ji} is if spot j can cover node i. And γ_{jt}, which is binary, is if spot j is covered at time t.
Gamma functions similarly to y_{it} for the jobs.

First, we can only cover a spot j if there is a vehicle there. How do we count how many vehicles are at a place? We discretize by all j and t so we only have to deal with whether that node is still there.

Let's make sure we understand the process. We have a spot arc system which includes: arcs that indicate travel from a spot to a spot; arcs that indicate stationary placement. For example, if I intend to stay at spot 5 from times 17 to 21, then I will have arcs indicating [[5, 17], [5, 18], 0], ..., [[5, 20], [5, 21], 0].

Wouldn't it be nice if we had a function which could tell us if each arc actually is AT the specified j and t? Then we could filter among those arcs which ones are occupied.

Among all valid arcs, sum up over x_{va} for all vehicles. If this is at least 1, it means gamma is possibly 1. If 0, then gamma must be 0.

Isn't the loosening of gamma (it could be 0 when the constraint allows for 1) a problem? No, because in optimization, if gamma = 1 is better, it will be done.

The mathematics is: $\forall j,t,\ \sum_{v} \sum_{\text{valid } a} x_{va} \geq \gamma_{jt}$. "valid a" refers to arcs that incorporate j,t provided.

Next, for all $W_{it} = 1$, there must be a coverage at that time and that i. We can set this up as a 1-0 constraint: W_{it} is less than or equal to the (coverage function).

What's the coverage function? Whether coverage vehicles are 'there'. What's 'there'? Only the locations that are close enough to that i. Specifically, this can be determined by $L_{ji} \times \gamma_{jt}$. If spot j is not close enough to node i, it doesn't matter whether there's a vehicle covering spot j. If spot j is close enough to node i, then gamma functions as intended.

So sum up over all spots j the quantity $L_{ji}\gamma_{jt}$, and this has to be greater than or equal to W_{it}. In mathematics, $\forall i,t,\ \sum_{j} L_{ji}\gamma_{jt} \geq W_{it}$.

Formal Formulation

Decision Variables:
γ_{jt} Binary, whether spot j at time t is covered (1 if so, 0 if not)
x_{va} Binary, whether vehicle v uses arc a

Parameters and Calculations:
L_{ji} binary, whether spot j is close enough to cover node i
W_{it} binary, whether work is done at time t on node i
c_a, the cost of arc a as a distance
valid(j,t), a function returning all arcs that visit spot j at time t

Objective:
MINIMIZE $\sum_v \sum_a c_a x_{va}$, the cost of all travels = distance

Constraints:
Flow Constraints:
$\sum_{(0,t)^+} x_{va} = 1 \ \forall v$, a vehicle leaves the origin-depot exactly once

$\sum_{(m+1,t)^-} x_{va} = 1 \ \forall v$, a vehicle goes into the end-depot exactly once

$\sum_{a \in m^-} x_{ka} = \sum_{a \in m^+} x_{ka} \ \forall v, \ \forall m = (j,t), \ j \neq 0, \ m+1$, flow balance at non-depot spots

Coverage Constraints:
$\forall j,t, \ \sum_v \sum_{\text{valid } a} x_{va} \geq \gamma_{jt}$, there can only be coverage if a vehicle is there

$\forall i,t, \ \sum_j L_{ji}\gamma_{jt} \geq W_{it}$, if you do work, you have to be covered

Alternate Point
The time complexity of valid(j,t) is going to be bad. Maybe we can instead declare a calculated parameter P_{ajt} telling us if arc a passes by/contains as 1st or 2nd element time t at spot j. Then our summation for the first coverage constraint is:

$\forall j,t, \ \sum_v \sum_a P_{ajt} x_{va} \geq \gamma_{jt}$.

We'll see how fast this is, but I think it will do well because this is an array of size |A|, but we'll have to see.