Column Generation for Coverage

Direct formulation for coverage is a total nightmare. Increase the number of potential coverage locations beyond 20, and you can expect the model to take far too long. Even with only column generation on the line, because the number of nodes is simply too big.

Set partitioning is hopelessly infeasible. Because there are virtually no time constraints, the number of potential routes will be far larger than for job vehicles.

Instead, our only hope is column generation. Our feasible start columns will be represented by: For each valid coverage spot j in 1:m_cov, put a vehicle there until time T - distance, then scurry back.

The formulation proceeds as follows.

We have lots of routes $p \in \mathcal{P}$. Decision variable x^p if route p is used. Binary.

If a vehicle is at a spot j, then j is covered.

Parameters: W_{it}, 1 if job i is done at time t.
L_{ji}, if spot j can cover job i.
y_{jt}^p, if spot j is covered by route p at time t.
C^p, cost of the route.

Minimize $\sum_p C^p x^p$, such that:

$\sum_p x^p \leq V$ there are no more routes than there are vehicles

$\forall i,t,\ \sum_p \sum_j L_{ji} y_{jt}^p x^p \geq W_{it}$: Whenever work is being done, we must have coverage.

min $\sum_p C^p x^p$

st $\sum_p x^p \leq V$ (dual: β) - beta

st $\forall i,t,\ \sum_p \sum_j L_{ji} y_{jt}^p x^p \geq W_{it}$ (dual: ξ_{it}) - xi

Finding the Dual
Rewrite constraints as

$\sum_p x^p - V \leq 0 \rightarrow -\sum_p x^p + V \geq 0$ (dual: β)
$\forall i,t,\ \sum_p \sum_j L_{ji} y_{jt}^p x^p \geq W_{it} \rightarrow \sum_p \sum_j L_{ji} y_{jt}^p x^p - W_{it} \geq 0\ \forall i,t$ (dual: ξ_{it})

as

$\ldots - x^p - \ldots \geq -V$ (dual: β)
$\ldots + L_{ji} y_{jt}^p x^p + \ldots \geq 0$ (dual: ξ_{it})

so that our constraint in the dual will look like
$-\beta + \sum_i \sum_t \xi_{it} L_{ji} y_{jt}^p \leq C^p$ for each p.

Thus, our goal is to find new routes p' such that

$$C^p + \beta - \sum_i \sum_t \xi_{it} L_{ji} y_{jt}^p \leq 0.$$

We can try to reroute this so that we consider j and t instead of i and t, otherwise we cannot solve the shortest problem. We make the following observations:

If L_{ji} is zero (j out of range of i), we don't bother. So we can rewrite the double sum as $\sum_{L(j)} \sum_t \xi_{it} y_{jt}^p$, where $L(j)$ denotes all the nodes which are close enough to j to be covered. This is great! Now we have forcefully reframed the problem in terms of j, so that we can run shortest paths on spots j rather than nodes i. Indeed, $L(j)$ can easily be found through e.g. a dictionary.

It does not matter whether the xi dual variable is calculated before or after we reach the node, because we just need the correct reduced cost to be calculated. However, if we are to use pruning algorithms, maybe it will?

Here's the full algorithm for column generation.

Run a shortest paths algorithm, where each path start collects β. As we move to a coverage spot j, add the cost between the two spots as a penalty. Simultaneously, obtain a list $L(j)$ of all nodes i which are close enough to j. For each of those i, subtract ξ_{it} as a reward. And just like μ_{it} from last time, we will need to subtract every time t we are at that spot j.

As a good heuristic (this will definitely be updated; I just want an answer to be produced to get a foothold), we stay until the W_{it} of all nodes in $L(j)$ are exhausted, and then leave.

Let's see how this does.