Robot Movement

Paul A. Rubin

September 8, 2024

Problem

This document describes an integer programming model for a bicriterion routing problem posted on Operations Research Stack Exchange. The problem involves a robot moving through a rectangular grid, with each move to an adjacent cell (up, down, left or right but not on a diagonal). The robot must visit each cell at least once and return to the cell from which it started. In addition, each cell has a nonnegative priority value (weight). There are two competing criteria to be minimized. One is to minimize the total time (or equivalently the number of movements) required for the robot to make its rounds and return to base. The other is to minimize weighted service delay, which is the sum over all cells of the time at which the cell is first visited multiplied by its priority.

Parameters

C is the set of cells in the grid

$c_0 \in C$ is the origin (and eventual destination) of the robot

$N(c)$, $c \in C$ is the set of cells adjacent to cell c (i.e., the set of cells to which the robot could move when leaving cell c)

$w_c \geq 0$, $c \in C$ is the weight (priority) assigned to cell c

$H \gg |C|$ is the planning horizon (the maximum number of movements allowed in the model; see notes below)

Variables

$x_{c,t} \in \{0,1\}$, $c \in C, t \in \{1,\ldots,H\}$ is 1 if and only if the robot is in cell c at time t

$y_{c,t} \in \{0,1\}$, $c \in C, t \in \{1,\ldots,H\}$ is 1 if and only if the first visit to cell c occurs at time t

z_t, $t \in \{1,\ldots,H\}$ is 1 if and only if the robot returns to c_0 for the last time (with all cells visited at least once) at time t

Objectives

$\ell = \sum_{t=1}^{H} t \cdot z_t$ is the length of the route (the number of movements required for the robot to complete its rounds) (technically, one more than the route length if you are being picky)

$d = \sum_{c \in C} w_c \left(\sum_{t=1}^{H} y_{c,t} \right)$ is the weighted delay in servicing all cells

Constraints

- With apologies to Erwin Schrödinger, the robot can only be one place at each time.

$$\sum_{c \in C} x_{c,t} = 1 \quad \forall t \in \{1,\ldots,H\}$$

- The robot starts in cell c_0.

$$x_{c_0,1} = 1$$

- The robot ends where it began.

$$x_{c_0,H} = 1$$

- Every node must be visited at least once.

$$\sum_{t=1}^{H} x_{c,t} \geq 1 \quad \forall c \in C$$

- To reside in any node other than the origin, the robot must have come from a neighboring node.

$$x_{c,t} \leq \sum_{n \in N(c)} x_{n,t-1} \quad \forall c \in C \backslash \{c_0\}, \forall t \in \{2, H\}$$

- To reside at the origin, the robot must have come from a neighboring node or already been in the origin.

$$x_{c_0,t} \leq x_{c_0,t-1} + \sum_{n \in N(c_0)} x_{n,t-1} \quad \forall t \in \{2, H\}$$

- Every node has a unique first visit.

$$\sum_{t=1}^{H} y_{c,t} = 1 \quad \forall c \in C$$

- To get credit for a first visit, the robot must actually be at the node.

$$y_{c,t} \leq x_{c,t} \quad \forall c \in C, \forall t \in \{1, \ldots, H\}$$

- The robot must eventually be officially done.

$$\sum_{t=1}^{H} z_t = 1$$

- To be done, the robot must be at the origin.

$$z_t \leq x_{c_0,t} \quad \forall t \in \{1, \ldots, H\}$$

- To be done, there can be no further visits outside the origin.

$$z_t + x_{\tau,c} \leq 1 \quad \forall t \in \{1, \ldots, H-1\}, \forall \tau \in \{t+1, \ldots, H\}, \forall c \in C \backslash \{c_0\}$$

Notes

There are some redundancies or trivial simplifications available. For instance, $z_1 = 0$ since at time 1 the robot has not yet visited any node other than the origin. The presolve operation of any decent solver will remove these.

The final constraint (no further visits after the robot is done) could be rewritten as

$$z_t + \sum_{\tau=t+1}^{H} \sum_{c \in C \backslash \{c_0\}} x_{\tau,c} \quad \forall t \in \{1, \ldots, H-1\}.$$

It would reduce the number of constraints but make the constraint matrix denser, which is generally not desirable when using contemporary solvers.

Combining the two objectives (both to be minimized) is left to the reader as an exercise. Possibilities include minimizing one with an upper bound on the other (in which case selecting the bound is tricky), minimizing a weighted combination of the two (in which case selecting weights is tricky), or using lexicographic minimizing (minimizing the higher priority objective and then minimizing the lower priority objective subject to the higher priority objective being bounded by its optimal value) (in which case the issue is choosing the higher priority objective).

All parameters are given *except* the planning horizon H. Choosing it is a compromise between model size and confidence in the optimal solution. Clearly some finite limit is required. Choosing too large a horizon increases solution time (and at some point would cause the solver to run out of memory). Choosing too small a horizon might result in an "optimal" solution that is actually suboptimal (because the truly optimal solution needs more than H moves). A similar issue arises in other types of models, and it is often the case that the presence of slack in the final solution (in this context, the robot returning to the origin before time H and then staying there until time H) ensures that the true optimum has been found. In this case, that might not hold: a suboptimal solution might contain slack because any improvement in the route would lengthen it by more than the amount of slack. It is not clear here how much slack would be enough to ensure optimality, but it is reasonable to say that the more slack is found in the solution, the more confidence one can have in it.