

A Multiplier Adjustment Method  
for Dynamic Resource Allocation Problems

Tassio A. Carvalho  
Warren B. Powell  
Department of Civil Engineering  
and Operations Research  
Princeton University  
Princeton, NJ 08544

Statistics and Operations Research  
Technical Report SOR-96-03

December 23, 1998

## **Abstract**

Dynamic fleet management problems (with a homogeneous fleet) are classically formulated as dynamic networks, or linear programs with side constraints. Recently, a new dynamic control approach was introduced called a logistics queueing network. Instead of a large linear program, the problem is decomposed into small subproblems, that are guided by two control variables that push these local problems to produce a solution that is close to a global optimum. In prior work, these control variables were updated using a subgradient approximation. In this paper, we propose a multiplier adjustment method for solving the same problem. Numerical experiments show that this method produces better solutions with greater stability. The new method is somewhat slower, and is more difficult to implement. We believe that both methods will represent reasonable choices for solving the problem.

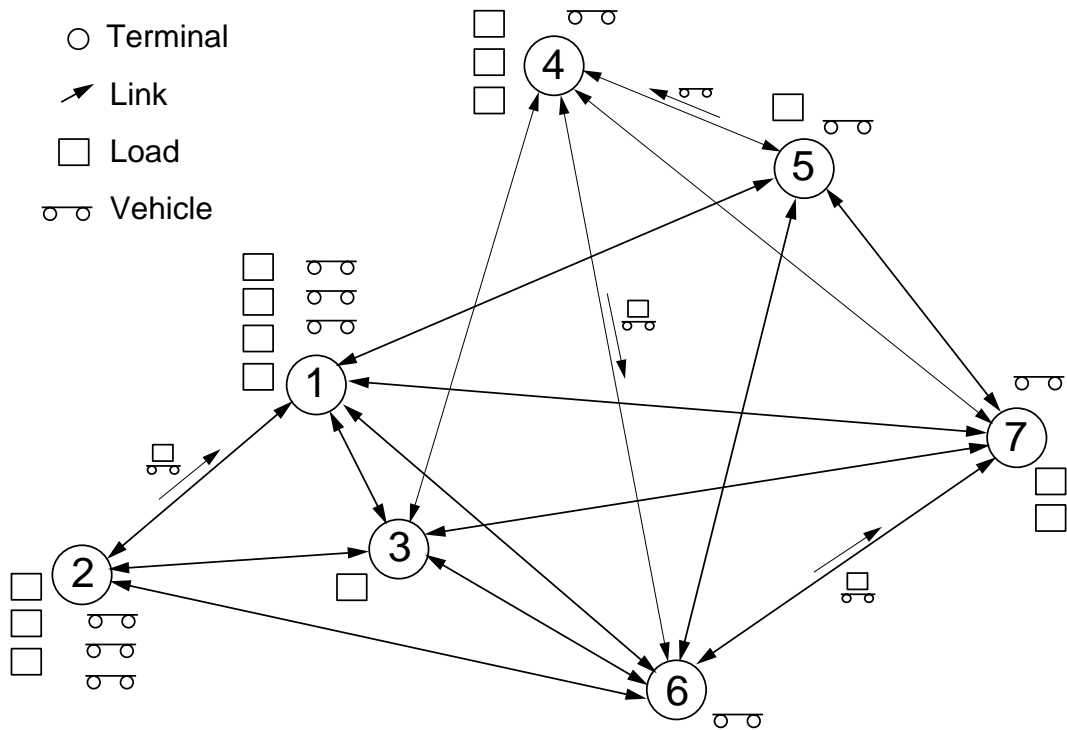


Figure 1: Network of transportation services.

Resource allocation problems consist of assigning resources to tasks. These problems are *dynamic* when tasks appear in a non-predictable way. The problem is to assign resources to tasks over time to maximize total profits over a given horizon. Applications of dynamic resource allocation problems arise in many settings, ranging from managing emergency vehicles, medical testing vehicles, sales people, and military force deployment. However, our motivating application is fleet management, which arises in truckload trucking, rail and shipping. In this context, a fleet of vehicles (resources) must be assigned to loads (tasks) that have the effect of moving the vehicle from one location to the next. Of course, vehicles are *reusable* and therefore must be managed after they have finished a task.

A fleet management problem is illustrated by figure 1. This figure shows a snapshot of a freight transportation system. Nodes represent terminals and links represent the possibility of offering transportation service between a pair of terminals. Throughout this work, we assume that the fleet of vehicles is homogenous. Any vehicle can be assigned to satisfy a load, provided that the vehicle is at the terminal where the load originates. Once the load is delivered, the vehicle that had been

assigned to it becomes available at the destination of the load at the time it arrives there.

Loads are characterized by a fairly simple set of data. Each load has a *departure time window* which gives the earliest and latest departure times for a load. Time windows may be one sided (as often happens in freight transportation). A load can be assigned a vehicle at any time within the time window. If a load is not assigned a vehicle by the latest departure time, it is considered lost. A contribution, or profit, is collected when a load is moved. Loads are distributed unevenly over space and time. Some terminals may have many more inbound than outbound loads over a period of time. In order to satisfy more loads and increase the total profit, vehicles can be repositioned empty at a cost.

Models of dynamic fleet management problems can be categorized by heterogeneity of the fleet, types of demand forecasts, whether demand backlogging or time windows are considered, and their consideration of other operating issues (primarily for rail). The simplest model assumes a homogeneous fleet, no demand backlogging (departure times for each load are fixed) and deterministic demand forecasts (White & Bomberault (1969), White (1972), Herren (1973), Herren (1977), Turnquist (1986) and Joborn (1995)). Stochastic demands without backlogging have been considered by Powell (1996), Frantzeskakis & Powell (1990), and Cheung & Powell (1996). Jordan & Turnquist (1983) considered both stochastic demands and demand backlogging, but assumed that once a car was sent empty, it would never be moved again. More recently, Crainic, Gendreau & Dejax (1993) suggest a stochastic, dynamic model for empty container distribution, but do not report any computational experiments.

Interestingly, the issue that appears to have received the least attention is the presence of time windows or demand backlogging – the ability to serve a demand at different points in time. Some models formulate the problem effectively as a distribution problem (once a vehicle is assigned to a demand, it leaves the system) where time windows are easily handled ( Turnquist (1986), Joborn (1995)). The problem is much harder when we want to model our ability to use a vehicle over again. Airline applications typically assume departure times are fixed, which greatly simplifies the modeling. In freight applications, on the other hand, these time windows can be quite wide.

Efforts to explicitly model time windows typically result in a linear programming with a GUB (generalized upper bounding) constraint (see Magnanti & Simpson (1978) or Powell, Jaillet & Odoni (1995)). Non-integer solutions can be obtained by solving the linear relaxation of the problem. This approach results in using rounding heuristics to obtain integer solutions, as branching is impractical

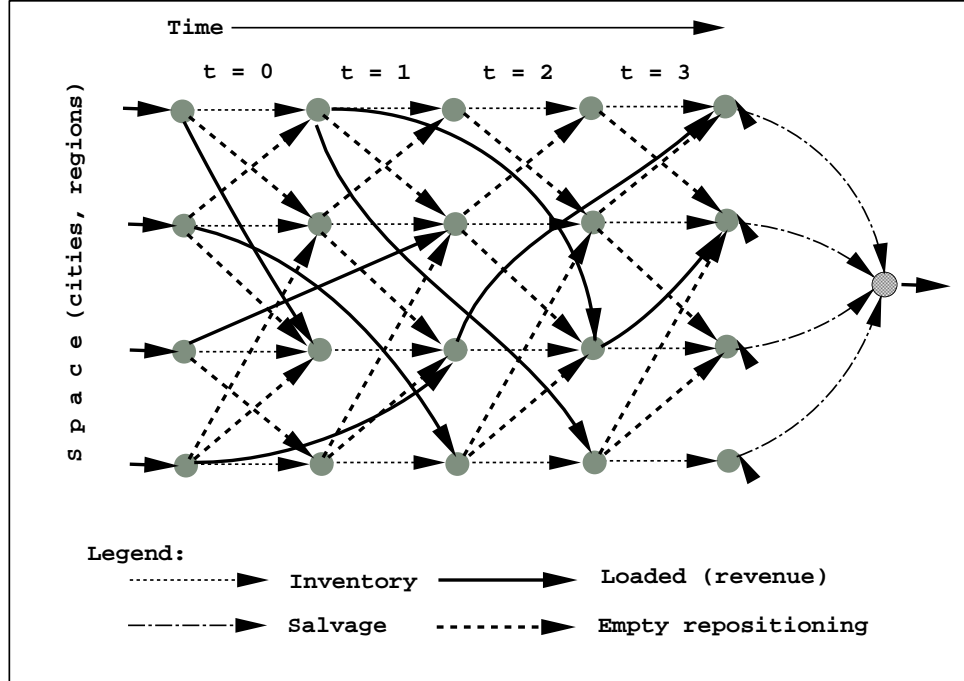


Figure 2: Time-space network for fleet management

for the big problems found in the real world. In case the departure time windows are wide, the resulting linear program is very degenerate and solution times suffer. Furthermore, problems have to be highly simplified to fit within this framework.

Powell & Carvalho (1998a) introduces an approach that addresses some of the shortcomings of the more traditional methods to solve dynamic resource allocation problems. Instead of the more classical formulation as a time-space network (as illustrated in figure 2), the problem can be viewed as controlling a network of queues, a formulation which is termed a *logistics queueing network*. The LQN approach uses a linear approximation of the future, which results in the problem decomposing into *local problems* for each terminal at each time period. In the single commodity case (homogeneous resources), each local problem reduces to a sorting of the different movement (loaded or empty) options out of a terminal at a given time. The global effect of the decisions at each local problem is captured by the presence of *control variables* in each local problem. The local problems are solved iteratively, and the control variables are reevaluated after each solution.

The LQN approach has several advantages over conventional methods. It provides integer solu-

tions. It allows for considering a wide array of real world constraints that cannot be easily modeled as linear constraints, such as crew labor regulations, load priorities, maintenance requirements and exceptions. Using the gradient approximation method developed in Powell & Carvalho (1998*a*), this approach returned solutions that are within 2–3 percent of the optimal value for the linear relaxation of the problem.

One difficulty faced in the gradient approximation method is that the gradients tend to change quite abruptly from iteration to iteration. The strategy used in Powell & Carvalho (1998*a*) to lessen the effect of these changes over the control variables is to smooth the gradients of each solution with gradients obtained in previous iterations.

The contributions of this paper are two-fold. 1) This is the first paper to formally present the dynamic fleet management problem in a control-theoretic setting. While the methodology is an extension of the approach suggested in Powell & Carvalho (1998*a*), no prior work has formally proposed solving the problem as a dynamic control problem. 2) Using this new formulation, we introduce a multiplier adjustment algorithm that improves on the subgradient approximation used in Powell & Carvalho (1998*a*). We give the steps of the method, and show that it produces higher quality solutions with greater stability (less jumping from one iteration to another). The new algorithm, called LAMA (linear approximation, multiplier adjustment), is shown to produce objective function values (on deterministic datasets) that are 0.7 percent better, on average, than the gradient approximation method. Viewed differently, the new method produces solutions that are 97.7 percent of the optimal linear relaxation, as opposed to the 97.0 percent produced using the gradient approximation algorithm. These results were produced with run times that were three to four times slower than the gradient approximation method, but still 10 times faster than the linear programming code used to develop the optimal noninteger solution.

In Section 1 we present a review of the main analytical results from Powell & Carvalho (1998*a*), including the integer program for the problem and the equations to compute the gradient approximations. In section 2 we introduce the basic idea for the LAMA method and derive equations to update multipliers. The algorithm is presented in section 3. In section 4 we present numerical results and contrast them with those obtained in Powell & Carvalho (1998*a*). The conclusion and directions for future research are presented in Section 5.

# 1 Review of the LQN Approach

This section is divided in three parts. In the first part we present the integer program for the problem. The integer program is necessary to evaluate the quality of the solutions obtained using the LAMA method. In the second part we briefly review the subgradient algorithm developed in Powell & Carvalho (1998a). The third part is composed of the equations to compute the gradients, which are also used in the LAMA method to update the multipliers and the upper bounds.

## 1.1 Integer Program

In order to formulate this problem as an integer program, we use the following notation.  $t$  always refers to a discrete point in time, and  $T$  is the (integer) length of the model horizon. The indices  $i$  and  $j$  always refer to points in space, and the pair  $(i, t)$  refers to a point in space time. If  $\tau_{i,j}$  is the travel time from city  $i$  to city  $j$ , then we use the triplet  $(i, j, t)$  to denote a link from space-time node  $(i, t)$  to node  $(j, t + \tau_{i,j})$ .

Network variables:

- $\mathcal{C}$  is the set of terminals  $i$  in the network.
- $\tau_{i,j}$  is the travel time between terminal  $i \in \mathcal{C}$  and terminal  $j \in \mathcal{C}$ . We let  $\tau^{max}$  be the longest possible travel time.
- $\tau_l$  The travel time for load  $l$ .
- $\mathcal{N}$  is the set of nodes  $(i, t), i \in \mathcal{C}, t \leq T$ , in the dynamic network.

Activity variables:

- $\mathcal{L}$  is the set of loads  $l$  available within the planning horizon,  $T$ .
- $\mathcal{T}_l$  is the set of feasible departure times for satisfying load  $l \in \mathcal{L}$ , otherwise known as the *departure time window*.
- $\mathcal{L}_{i,j,t}^s$  is the set of loads  $l \in \mathcal{L}$  with origin  $i$  and destination  $j$  having  $t$  as a feasible departure time. This is the static version of this set (hence the superscript  $s$ ). Later we define a dynamic version of this set, denoted simply  $\mathcal{L}_{i,j,t}$  which includes the loads that

are available to be moved at time  $t$ . For the purposes of formulating the initial integer program, we need to define the set of loads that might be moved at time  $t$  (but which may have been moved prior to time  $t$ ).

- $R_{i,t}$  is the net inflow ( $R_{i,t} > 0$ ) or outflow ( $R_{i,t} < 0$ ) of vehicles at terminal  $i$  at time  $t$ .  $R_{i,0}$  is the set of vehicles available at time 0. In our numerical work, we assume that  $R_{i,t} = 0, t > 0$ , but in practice this will rarely be true.
- $r_{l,t}$  is the revenue generated by choosing time  $t$  to satisfy load  $l$ .
- $c_{i,j}$  is the cost of repositioning one vehicle over link  $(i, j, t)$ . This may depend on time, but in our work, we have assumed that it is stationary.

Decision variables:

- $x_{l,t} = 1$  if load  $l$  is served at time  $t$ .
- $z_l = 1$  if load  $l$  is never served (within the time window).
- $y_{i,j,t}$  is the number of vehicles being repositioned empty along link  $(i, j, t)$ . If  $i = j$ ,  $y_{i,i,t}$  represents the number of vehicles in inventory at terminal  $i$  from time  $t$  to time  $t + 1$ . In general,  $y_{i,j,t}$  is the number of vehicles moving empty from  $(i, t)$  to  $(j, t + \tau_{i,j})$ .
- $w_{i,j,t}$  is the total flow of vehicles on the dynamic link  $(i, j, t)$ .

The variables  $z$  and  $w$  are included primarily for notational convenience. The primary decisions variables are the loaded and empty movement variables  $x$  and  $y$ .

The objective function we are maximizing is stated as:

$$F(x, y) = \sum_{t=0}^T \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \left( \sum_{l \in \mathcal{L}_{i,j,t}^s} r_{l,t} x_{l,t} - c_{i,j} y_{i,j,t} \right) \quad (1)$$

This problem can be formulated as:

$$\max_{x,y} F(x, y) \quad (2)$$

subject to

$$\sum_{t \in \mathcal{T}_l} x_{l,t} + z_l = 1 \quad \forall l \in \mathcal{L} \quad (3)$$

$$\sum_{l \in \mathcal{L}_{i,j,t}^s} x_{l,t} + y_{i,j,t} - w_{i,j,t} = 0 \quad \forall i, j \in \mathcal{C}, \forall t \leq T \quad (4)$$

$$\sum_{j \in \mathcal{C}} w_{i,j,t} - \sum_{j \in \mathcal{C}} w_{j,i,t-\tau_{j,i}} = R_{i,t} \quad \forall (i,t) \in \mathcal{N} \quad (5)$$

$$y_{i,j,t}, w_{i,j,t} \geq 0 \quad (6)$$

$$x_{l,t} = (0, 1) \quad (7)$$

where constraints (3) restrict the maximum number of vehicles to be assigned to each load to one and constraints (4) and (5) enforce flow conservation. In the experimental section, the linear relaxation of this integer program is solved in order to provide a bound on the objective function. We compare this bound to values of the objective function using the multiplier adjustment method developed in this paper.

## 1.2 The Subgradient Algorithm for LQN

In order to state the basic equations for the subgradient algorithm, we need the additional activity variables:

- $\bar{\mathcal{L}}_{i,t}$  is the set of loads  $l$  with origin  $i$  having  $t$  as a feasible departure time.
- $\mathcal{L}_{i,j,t}$  is the set of loads  $l$  with origin  $i$  and destination  $j$ , which are available to move at time  $t$  and have not been moved at a time prior to time  $t$  at a given solution. This is the dynamic version of the set  $\mathcal{L}_{i,j,t}^s$ .
- $\mathcal{L}_{i,t}$  is made of the union of all sets  $\mathcal{L}_{i,j,t}$  for all the destinations  $j \in \mathcal{C}$ .
- $\mathcal{L}_t = \bigcup_{i \in \mathcal{C}, t' \geq t} \mathcal{L}_{i,t'}$
- $\mathcal{L}_{i,t}^0$  is the set of loads  $l$  with origin  $i$ , where  $t$  is the beginning of the time window  $\mathcal{T}_l$ .
- $\mathcal{L}_{i,t}^f$  is the set of loads  $l$  with origin  $i$ , where  $t$  is the end of the time window  $\mathcal{T}_l$ .

Let  $V_{i,t}$  be the number of vehicles available at node  $(i, t)$ , i.e., the total flow entering node  $(i, t)$ . Following Powell et al. (1995), the objective function can be expressed in the recursive form by:

$$g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) = \sum_{l \in \mathcal{L}_{i,t}} r_{l,t} x_{l,t} - \sum_{j \in \mathcal{C}} c_{i,j} y_{i,j,t} \quad (8)$$

$$G_t(V_t, \mathcal{L}_t) = \max_{x_t, y_t} \left\{ \sum_{i \in \mathcal{C}} g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) + G_{t+1}(V_{t+1}, \mathcal{L}_{t+1}) \right\} \quad (9)$$

subject to constraints governing flow conservation and system dynamics (which we present later in the context of the recursive form).  $g_{i,t}$  is the contribution to the objective function of the decisions taken at time  $t$  at terminal  $i$  and  $G_t$  is the contribution to the objective function of the decisions taken from time  $t$  to the end of the planning horizon.

The LQN approach consists of choosing an approximation for the value function, represented by  $G_{t+1}(V_{t+1}, \mathcal{L}_{t+1})$  in equation (9) and constraining the variables that represent empty moves  $y$  with upper bounds. These bounds are necessary because the approximation for the value function uncouples the decisions for different terminals. We define  $u_{i,j,t}$  as the upper bound on  $y_{i,j,t}$ .

Our solution approach begins by replacing  $G_{t+1}$  in equation (9) with a linear approximation. If all movements required one time period, we would write:

$$\tilde{G}_t(x_t, y_t, V_t, \mathcal{L}_t, \xi_{t+1}) = \sum_{i \in \mathcal{C}} g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) + \xi_{t+1} V_{t+1} \quad (10)$$

(Here and elsewhere, the multiplication of two vectors, as in  $\xi_{t+1} V_{t+1}$ , is assumed to be a scalar product.) We can modify (10) to handle multiperiod travel times if we define:

$$\begin{aligned} V_{i,t,t'} &= \text{The number of vehicles inbound to location } i \text{ at time } t' \text{ that} \\ &\quad \text{were dispatched at time } t. \\ &= \sum_{j \in \mathcal{C} | \tau_{j,i} = t' - t} \left\{ \sum_{l \in \mathcal{L}_{j,i,t}} x_{l,t} + y_{i,j,t} \right\} \end{aligned}$$

Equation (10) can now be stated for multiperiod travel times as:

$$\tilde{G}_t(x_t, y_t, V_t, \mathcal{L}_t, \xi_{t+1}) = \sum_{i \in \mathcal{C}} g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) + \sum_{t'=t+1}^{\tau^{max}} \xi_{t'} V_{t,t'} \quad (11)$$

We now let (using the simpler single-period travel time form):

$$\hat{G}_t(V_t, \mathcal{L}_t) = \max_{x_t, y_t} \tilde{G}_t(x_t, y_t, V_t, \mathcal{L}_t, \xi_{t+1}) \quad (12)$$

$$= \max_{x_t, y_t} \{g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) + \xi_{t+1} V_{t+1}\} \quad (13)$$

where the slope  $\xi_{t+1}$  is termed the *spatial potential function* for vehicles at terminal  $i$  at time  $t+1$  because  $\xi_{i,t}$  is a measure of how useful a vehicle at node  $(i,t)$  can be. We find it useful to also define the optimal solution (which we use shortly):

$$\begin{aligned} x_{l,t}(\xi, u) &= x_{l,t}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{i,t}) \\ &= \text{The optimal value of } x_t \text{ in equation (13)} \quad . \\ y_{i,j,t}(\xi, u) &= y_{i,j,t}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{i,t}) \\ &= \text{The optimal value of } y_t \text{ in equation (13)} \quad . \end{aligned}$$

After evaluating the scalar product  $\xi_{t+1} V_{t+1}$ , we arrive at:

$$\tilde{G}_t(x_t, y_t, V_t, \mathcal{L}_t, \xi_{t+1}) = \sum_{i \in \mathcal{C}} \left( \sum_{j \in \mathcal{C}} \sum_{l \in \mathcal{L}_{i,j,t}} (r_{l,t} + \xi_{j,t+1}) x_{l,t} + \sum_{j \in \mathcal{C}} (-c_{i,j} + \xi_{j,t+1}) y_{i,j,t} \right) \quad (14)$$

or, for multiperiod travel times:

$$\tilde{G}_t(x_t, y_t, V_t, \mathcal{L}_t, \xi_{t+1}) = \sum_{i \in \mathcal{C}} \left( \sum_{j \in \mathcal{C}} \sum_{l \in \mathcal{L}_{i,j,t}} (r_{l,t} + \xi_{j,t+\tau_l}) x_{l,t} + \sum_{j \in \mathcal{C}} (-c_{i,j} + \xi_{j,t+\tau_{i,j}}) y_{i,j,t} \right) \quad (15)$$

Henceforth, we will only refer to the multiperiod travel time form of the problem.

Let us define  $\nu_{i,t}$  as a subgradient of  $\hat{G}_t$  with respect to  $V_{i,t}$ :

$$\nu_{i,t} = \frac{\partial \hat{G}_t}{\partial V_{i,t}} \quad (16)$$

In the subgradient algorithm, the spatial potential function at node  $(i,t)$  is defined as an averaged right gradient of  $\hat{G}_t$  with respect to  $V_{i,t}$  over previous iterations (call it  $\bar{\nu}_{i,t}^{+n}$  for iteration  $n$ ). The approximation uncouples the problems at different terminals, so that we arrive at the *local problem* for each terminal  $i$  at each time period  $t$ :

$$\max_{x_t, y_t} \sum_{j \in \mathcal{C}} \left( \sum_{l \in \mathcal{L}_{i,j,t}} (r_{l,t} + \bar{\nu}_{j,t+\tau_l}^{+n}) x_{l,t} + (-c_{i,j} + \bar{\nu}_{j,t+\tau_{i,j}}^{+n}) y_{i,j,t} \right) \quad (17)$$

subject to:

$$x_{l,t} \leq 1 \quad \forall l \in \mathcal{L}_{i,t} \quad (18)$$

$$y_{i,j,t} \leq u_{i,j,t} \quad \forall j \in \mathcal{C} \quad (19)$$

$$\sum_{l \in \bar{\mathcal{L}}_{i,t}} x_{l,t} + \sum_{j \in \mathcal{C}} y_{i,j,t} \leq V_{i,t} \quad (20)$$

$$x_{l,t}, y_{i,j,t} \geq 0 \quad (21)$$

where the set of loads,  $\mathcal{L}_{i,t+1}$ , and the number of vehicles,  $V_{i,t+1}$ , available at the next time period are computed by:

$$\mathcal{L}_{k,t+1} = \{\mathcal{L}_{kt} \setminus \{\mathcal{L}_{kt}^s \cup \mathcal{L}_{kt}^f\}\} \cup \mathcal{L}_{k,t+1}^0 \quad \forall k \in \mathcal{C} \quad (22)$$

$$V_{k,t+1} = V_{k,t} - \sum_j w_{k,j,t} + \sum_i w_{i,k,t+1-\tau_{i,j}} + R_{k,t+1} \quad \forall k \in \mathcal{C} \quad (23)$$

where the set  $\mathcal{L}_{k,t}^s$  is the set of loads with origin  $k$  that were assigned a vehicle at time  $t$ .

In passing, we note that problem (17) is solved using a simple sort. We create a list of *options*, where an option is to take a load  $l$  or move empty to some location  $j$ . A loaded option has a value  $(r_{l,t} + \bar{v}_{j,t+\tau}^+)$  while an empty option has a value  $(-c_{i,j} + \bar{v}_{j,t+\tau_{i,j}}^+)$ . Options are ranked from highest to lowest value, and flow is assigned to an option up to the bound for that option. Loaded options all have a bound of 1, while empty options are limited by the upper bound  $u_{i,j,t}$ . We generally assume that the option of holding a vehicle (represented by the empty flow  $y_{i,i,t}$ , is unbounded to preserve feasibility).

We now turn our attention to the *functions*  $x_{l,t}(\xi, u)$  and  $y_{i,j,t}(\xi, u)$ . In our original formulation,  $x_t$  and  $y_t$  were decision variables. Now we find that we can view  $\xi$  and  $u$  as the decision vectors which, along with  $V_{i,t}$  and  $\mathcal{L}_{i,t}$ , determine  $x_t$  and  $y_t$ . This view allows us to reformulate the problem in an elegant way. Let:

$$f_{i,t}(\xi, u) = \sum_{l \in \mathcal{L}_{i,t}} r_{l,t} x_{l,t}(\xi, u) - \sum_{j \in \mathcal{C}} c_{i,j} y_{i,j,t}(\xi, u)$$

Thus our master problem consists of finding the control variables that maximize the overall profits:

$$F = \max_{\xi, u} \sum_{t=0}^T \sum_{i \in \mathcal{C}} f_{i,t}(\xi, u) \quad (24)$$

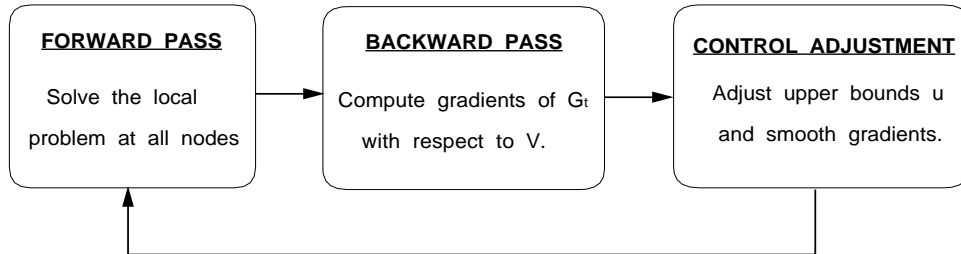


Figure 3: Subgradient Algorithm for the LQN approach.

Equation (24) now represents the dynamic fleet management problem as a dynamic control problem, which involves finding the values for  $\xi$  and  $u$  to maximize total profits.

The LQN approach consists of three main steps (figure 3). The forward pass consists of solving the local problem at each node. The backward pass consists of finding values for the gradients  $\nu_{i,t}$ . There are two types of control that must be adjusted. One is the upper bound on empty moves. The gradients computed in the backward pass are used to estimate the impact in the objective function of increasing or decreasing an upper bound. The other control is the spatial potential function. In the case of the subgradient algorithm, they consist simply of weighting the gradients at the present iteration with the gradients from the previous iteration. For iteration  $n + 1$ :

$$\bar{\nu}_{i,t}^{n+1} = \gamma \nu_{i,t}^n + (1 - \gamma) \bar{\nu}_{i,t}^n \quad (25)$$

where  $\gamma$  is the smoothing factor such that  $0 \leq \gamma \leq 1$ . The subgradient algorithm returned solutions that are within 2 – 3 percent of optimality for most cases.

### 1.3 Gradients of the Objective Function

In this section we summarize the gradient derivations from Powell & Carvalho (1998a). In the LAMA method, we use the supply gradients  $\nu$  to update the multipliers  $\xi$  and compute the upper bound gradients. We use the upper bound gradients to update the upper bounds using the same strategies presented in Powell & Carvalho (1998a).

## Supply Gradients :

In order to compute the gradient  $\nu_{i,t}^+$ , we first compute the following finite differences:

$$\frac{\partial x_{l,t'}}{\partial V_{i,t'}^+} = X_{l,t'}^+ = x_{l,t'}(V_{i,t'}+1, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) - x_{l,t'}(V_{i,t'}, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) \quad (26)$$

$$\frac{\partial y_{i,j,t'}}{\partial V_{i,t'}^+} = Y_{i,j,t'}^+ = y_{i,j,t'}(V_{i,t'}+1, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) - y_{i,j,t'}(V_{i,t'}, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) \quad (27)$$

Then the right gradient at each node is approximated by

$$\nu_{i,t}^+ \simeq \begin{cases} -c_{i,j'} + \nu_{j',t'+\tau_{i,j'}}^+ & \text{if } \exists j' \text{ such that } Y_{i,j',t'}^+ = 1 \\ r_{l,t'} + \nu_{j',t'+\tau_l}^+ & \text{if } \exists l \text{ such that } X_{l,t'}^+ = 1 \text{ and } x_{l,t} = 0 \forall t > t' \\ & \text{where } j' \text{ is the destination for load } l. \\ r_{l,t'} - r_{l,t''} + \nu_{j',t'+\tau_l}^+ + \nu_{i,t''}^+ - \nu_{j',t''+\tau_l}^- & \text{if } \exists l \text{ and } t'' > t' \text{ such that } X_{l,t'}^+ = 1 \text{ and} \\ & x_{l,t''} = 1 \text{ where } j' \text{ is the destination for load } l. \\ \nu_{i,t'+1}^+ & \text{otherwise} \end{cases} \quad (28)$$

Since the derivation of this expression is given more carefully in Powell & Carvalho (1998a), we provide here instead a brief intuitive justification. If there is one more vehicle at node  $(i, t)$ , then the vehicle will either move loaded or empty, or do nothing. If it moves empty to some location  $j'$ , then we incur a negative reward  $-c_{i,j'}$ , and we add another vehicle to node  $(j, t + \tau_{i,j'})$ . If we move loaded, we have to consider two cases. The first case is where we move a loaded that otherwise was not covered. In this case, we capture the additional revenue, and we add another vehicle at the destination of the load. The situation is somewhat more complicated if we use the additional vehicle at node  $(i, t)$  to cover a load that would otherwise have been covered at a later time  $t' > t$ . In this case, we have start with the same impact of covering a load at time  $t$ , and then we have to consider the impact of not covering the same load at  $t'$ . This effect is the same as losing the future revenue (which might be different if revenues are time dependent), which then adds another vehicle at location  $(i, t')$  (because it is no longer covering the load), but takes a vehicle away from the destination of the load at node  $(j', t + \tau_l)$ .

We write the expression for  $\nu_{i,t}^+$  as an approximation since, as argued in Powell & Carvalho (1998a), it is not even a valid subgradient for the problem, but appears to work very well in practice. The expression would be a valid subgradient if there were no time windows (that is, if loads had to be served at a fixed point in time).

For the computation of  $\nu_{i,t}^-$ , we first find the following finite differences:

$$\frac{\partial x_{l,t'}}{\partial V_{i,t'}^-} = X_{l,t'}^- = x_{l,t'}(V_{i,t'}, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) - x_{l,t'}(V_{i,t'} - 1, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) \quad (29)$$

$$\frac{\partial y_{i,j,t'}}{\partial V_{i,t'}^-} = Y_{i,j,t'}^- = y_{i,j,t'}(V_{i,t'}, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) - y_{i,j,t'}(V_{i,t'} - 1, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{i,t'}) \quad (30)$$

Then the left gradients are approximated by:

$$\nu_{i,t'}^- \simeq \begin{cases} -c_{i,j'} + \nu_{j',t'+\tau_{i,j'}}^- & \text{if } \exists j' \in \mathcal{C} \text{ such that } Y_{i,j',t'}^- = -1 \\ r_{l,t'} + \nu_{j',t'+\tau_{i,j'}}^- & \text{if } \exists l \in \mathcal{L}_{i,t'} \text{ such that } X_{l,t'}^- = -1 \text{ and } \frac{\partial x_{l,t'}}{\partial V_{i,t'}^-} = 0 \\ r_{l,t'} - r_{l,t''} + \nu_{j',t'+\tau_{i,j'}}^- + \nu_{i,t''}^- - \nu_{j',t''+\tau_{i,j'}}^+ & \forall t > t' \text{ where } j' \text{ is the destination for load } l. \\ & \text{if } \exists l \in \mathcal{L}_{i,t'} \text{ and } t'' > t' \text{ such that } X_{l,t''}^- = -1 \text{ and } \frac{\partial x_{l,t''}}{\partial V_{i,t''}^-} = 1 \text{ where } j' \text{ is the destination for load } l. \\ \nu_{i,t'+1}^- & \text{otherwise} \end{cases} \quad (31)$$

### Upper Bound Gradients :

The gradients of  $G_t$  with respect to the control variable  $u$  are computed by:

$$\eta_{i,j,t'}^+ = \frac{\partial G_t}{\partial u_{i,j,t'}^+} \begin{cases} \simeq -c_{i,j} + \nu_{j,t'+1}^+ - \nu_{i,t'}^- & \text{if } -c_{i,j} + \nu_{j,t'+1}^+ - \nu_{i,t'}^- > 0 \\ = 0 & \text{otherwise} \end{cases} \quad (32)$$

$$\eta_{i,j,t'}^- = \frac{\partial G_t}{\partial u_{i,j,t'}^-} \begin{cases} \simeq c_{i,j} - \nu_{j,t'+1}^- + \nu_{i,t'}^+ & \text{if } c_{i,j} - \nu_{j,t'+1}^- + \nu_{i,t'}^+ > 0 \\ = 0 & \text{otherwise} \end{cases} \quad (33)$$

## 2 The LAMA method

As stated in section 1, the LQN approach consists of choosing an approximation to the value function in (9) so that:

$$\hat{G}_t(V_t, \mathcal{L}_t) = \max_{x_t, y_t} \sum_{i \in \mathcal{C}} g_{i,t}(x_t, y_t, V_{i,t}, \mathcal{L}_{i,t}) + \xi_{t+1} V_{t+1}$$

This approximation results in decoupling the local problems for different terminals. Therefore, in order to prevent the flooding of some terminals with vehicles, we introduce upper bounds  $u_{i,j,t}$  on the empty moves.

After replacing the approximation for the recourse function in (9), we arrive at the local problem, which is the problem to be solved at every node  $(i, t)$ . The objective function for the local problem at node  $(i, t)$  is represented in a general form by:

$$\max_{x_t, y_t} \sum_{j \in \mathcal{C}} \left( \sum_{l \in \mathcal{L}_{i,j,t}} (r_{l,t} + \xi_{j,t+\tau_{i,j}}) x_{l,t} + (-c_{i,j} + \xi_{j,t+\tau_{i,j}}) y_{i,j,t} \right) \quad (34)$$

subject to constraints (18) to (21).

In the subgradient algorithm, the spatial potential function for iteration  $n + 1$  is defined as a weighted average of the gradient  $\nu$  and the spatial potential function from iteration  $n$  as in the updating equation (25). The gradient approximations may change abruptly from one iteration to the next. As the spatial potential function appears in the objective function of the local problem (17)–(21), a change in  $\bar{\nu}$  may alter the optimal ordering of tasks. This results in unexpected changes in the solution and an objective function that bounces up and down with the iteration number.

In this section we propose a multiplier adjustment method to adjust  $\xi$  as an improving step. The control of the ordering of tasks at each node is done directly through the spatial potential function. We then proceed to the derivation of the equations to update the multipliers  $\xi$ , first to increase a multiplier, then to decrease it. Finally, we describe the algorithm and discuss the issues that come along with it.

## 2.1 Basic Idea

Each local problem consists of sorting activities (loaded moves or empty moves allowed by upper bounds) that can originate at that node. The spatial potential function  $\xi$  at node  $(i, t)$  shows in the objective function of the local problem at node  $(j, t - \tau_{j,i})$  whenever it is possible to move a vehicle from  $(j, t - \tau_{j,i})$  to  $(i, t)$ .

We need to find values for  $\xi_{i,t}$  so that the set of decision variables obtained by solving the local problems for all nodes  $(i, t)$  returns a solution that is very close to the optimal solution of the original linear program stated in section 1. However, altering the values of  $\xi$  for several nodes at the same time can have effects difficult to account for.

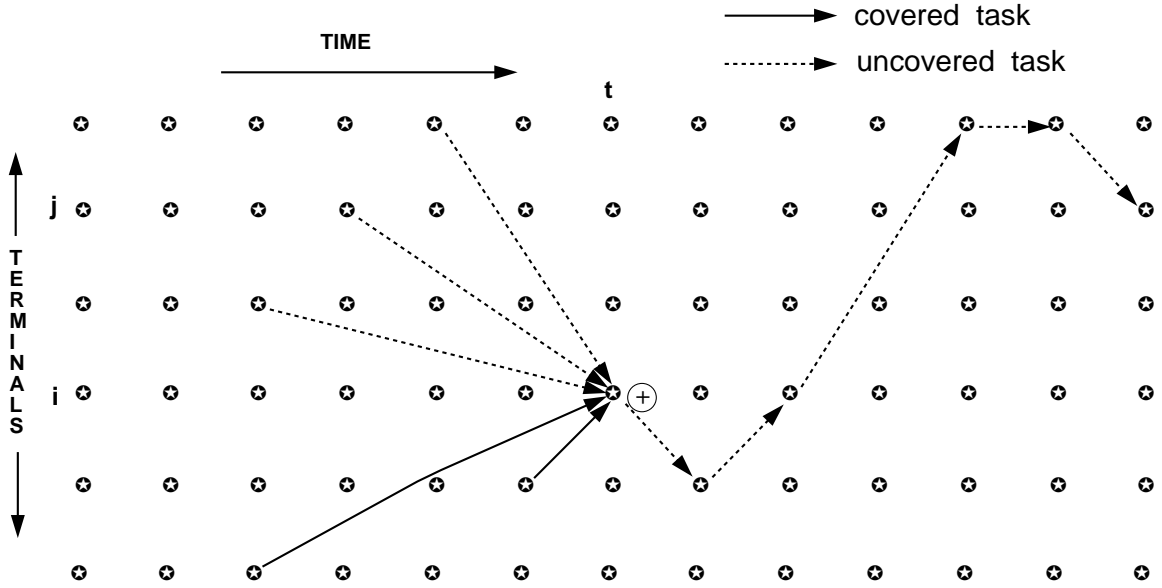


Figure 4: Tasks having node  $(i, t)$  as a destination and the flow augmenting path out of node  $(i, t)$ .

The idea behind the LAMA method is to carefully control the spatial potential function. By increasing or decreasing  $\xi$  at one node per iteration we make one predictable change in the overall solution. To illustrate the effect of increasing  $\xi_{i,t}$ , we have figure 4. This figure shows all the tasks that could possibly have node  $(i, t)$  as a destination. In the current solution some of these tasks are covered, others are not. This figure also shows the flow augmenting path out of node  $(i, t)$ , to indicate which tasks an additional vehicle at node  $(i, t)$  would satisfy.

The fact that some of the tasks bound to node  $(i, t)$  were not assigned a vehicle may stem from two reasons: either there were no vehicles available at the origin node ( $V_{j,t-\tau_{j,i}} = 0$ ) or, if there were, those particular tasks were not priced high enough to get a vehicle. If we increase  $\xi_{i,t}$  by a very small amount, it is likely the current solution will not change. But if we sharply increase  $\xi_{i,t}$ , node  $(i, t)$  could be flooded with vehicles. By increasing  $\xi_{i,t}$  by the right amount, call it  $\Delta\xi_{i,t}^+$ , we are able to cover *one* more task bound to node  $(i, t)$ , and thus increase  $V_{i,t}$  by one unit. Let  $j$  be the origin of that task. The effect of the correct increase in  $\xi_{i,t}$  can be seen in figure 5. By satisfying the task out of  $(j, t - \tau_{j,i})$  bound to  $(i, t)$ , one vehicle out of  $(j, t - \tau_{j,i})$  is diverted from its original path into  $(i, t)$ .

By applying the same strategy, we can look at decreasing  $\xi_{i,t}$  and estimate the change in the

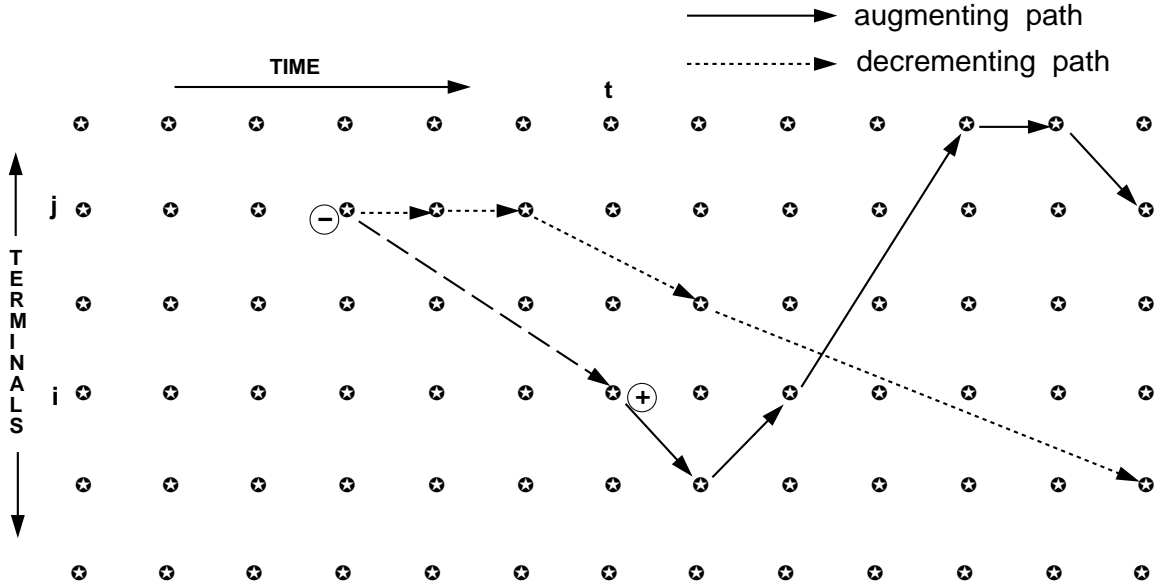


Figure 5: Flow augmenting path and flow decrementing path resulting of increasing  $\xi_{i,t}$ .

objective function resulting from this decrease. One must expect that decreasing  $\xi_{i,t}$  decreases the supply of vehicles at node  $(i, t)$ , but may result in an increase in the supply at some other node, which might have a very desirable global effect.

In the coming section we derive the equations to compute  $\Delta\xi_{i,t}^+$  and  $\Delta\xi_{i,t}^-$  and the corresponding changes in the objective function:  $\Delta G(\Delta\xi_{i,t}^+)$  and  $\Delta G(\Delta\xi_{i,t}^-)$ .

## 2.2 Increasing the Spatial Potential Function

Let us look at increasing the multiplier  $\xi_{i,t}$ . Whenever  $(i, t)$  is the destination node for a task in a local problem,  $\xi_{i,t}$  must appear in its objective function. The multiplier  $\xi_{i,t}$  must then appear in the local problems at nodes  $(k, t - \tau_{k,i})$ , for all  $k \in \mathcal{C}$ . Consider the local problem at node  $(k, t - \tau_{k,i})$  and assume that  $V_{k,t-\tau_{k,i}} > 0$ . If we assume otherwise ( $V_{k,t-\tau_{k,i}} = 0$ ), regardless of the increase in  $\xi_{i,t}$ , there will be no vehicle directed from  $(k, t - \tau_{k,i})$  to  $(i, t)$ .

As the local problem reduces to sorting the variables in (34) by their coefficients, let the decreasing sequence  $\mathcal{K} = \{\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n\}$ , where  $\bar{c}_1 \geq \bar{c}_2 \geq \dots \geq \bar{c}_n$ , represent the ordered coefficients for the  $n$  tasks at  $(k, t - \tau_{k,i})$ , as illustrated by figure 6. These tasks can be either loads or empty moves allowed by positive upper bounds out of node  $(k, t - \tau_{k,i})$ . If task  $m$  is a load, then  $\bar{c}_m = r_{l,t-\tau_{k,i}} + \xi_{i,t}$ .

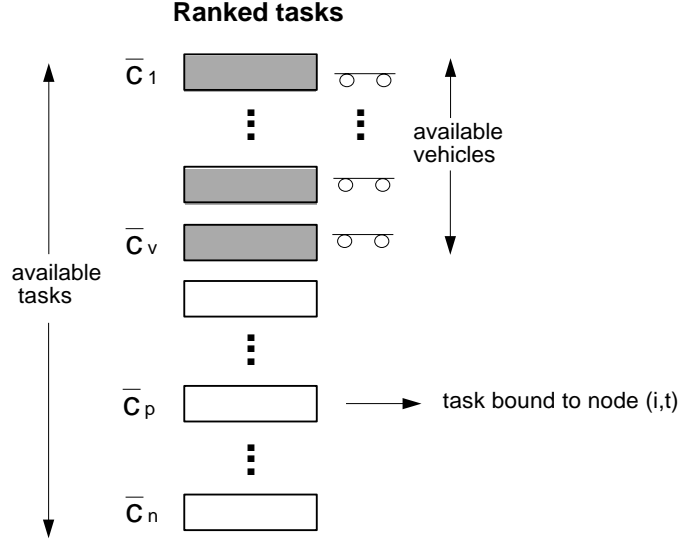


Figure 6: Local problem at node  $(k, t - \tau_{k,i})$  with a task bound to node  $(i, t)$ .

If task  $m$  is an empty move then  $\bar{c}_m = -c_{k,i} + \xi_{i,t}$ . Let  $\mathcal{P}_{k,t-\tau_{k,i}}(i)$  be the set of indices for tasks in  $(k, t - \tau_{k,i})$  having  $i$  as the destination terminal. When solving the local problem at node  $(k, t - \tau_{k,i})$ , the first  $V_{k,t-\tau_{k,i}}$  tasks in the sequence  $\mathcal{K}$  would be assigned a vehicle. Let  $\bar{c}_v$  be the lowest coefficient among all tasks being assigned a vehicle where the task has a destination node other than  $(i, t)$  (the number  $\bar{c}_v$  should be indexed by the origin node  $(k, t - \tau_{k,i})$  and the destination node  $(i, t)$ ; we suppress this indexing to reduce notational clutter). Suppose there exists a task in node  $(k, t - \tau_{k,i})$  having node  $(i, t)$  as its destination where this task is not among those that are assigned a vehicle in the current solution. Let  $p$  be its index in the sequence  $\mathcal{K}$ . Then by increasing  $\xi_{i,t}$ , we can alter the ordering of the tasks at node  $(k, t - \tau_{k,i})$  so that this task will be assigned a vehicle, increasing the number of vehicles flowing towards  $(i, t)$ . We are looking for the smallest increase in  $\xi_{i,t}$  that results in one more unit of flow being directed from  $(k, t - \tau_{k,i})$  to  $(i, t)$ . As there might be more than one task at node  $(k, t - \tau_{k,i})$  bound to terminal  $i$ , we compute the increase in  $\xi_{i,t}$  by

$$\delta_{k,t-\tau_{k,i}}^+(i) = \min_{p \in \mathcal{P}_{k,t-\tau_{k,i}}(i)} \{\bar{c}_v - \bar{c}_p \mid \bar{c}_v - \bar{c}_p > 0\} \quad (35)$$

The value  $\delta_{k,t-\tau_{k,i}}^+(i)$  is the smallest increase in  $\xi_{i,t}$  that would result in increasing the flow of vehicles between node  $(k, t - \tau_{k,i})$  and node  $(i, t)$  by (at least) one. (Of course, equation (35) creates a tie which we can break by incrementing  $\delta^+$  by a small amount.) We need the smallest increase in  $\xi_{i,t}$  that will divert one vehicle towards  $(i, t)$ , regardless of where it comes from, increasing  $V_{i,t}$  by one

unit:

$$\Delta\xi_{i,t}^+ = \min_{k \in \mathcal{C}} \{\delta_{k,t-\tau_{k,i}}^+(i)\} \quad (36)$$

The value  $\Delta\xi_{i,t}^+$  is the smallest increase in  $\xi_{i,t}$  that would result in one more vehicle being directed to node  $(i,t)$ . Let  $j$  be the origin of such a task. We must then compute the impact on the objective function  $G$  of increasing  $\xi_{i,t}$  by  $\Delta\xi_{i,t}^+$ . Let this impact be  $\Delta G(\Delta\xi_{i,t}^+)$ . Let  $\hat{x}$  and  $\hat{y}$  be the current solution of the local problem at node  $(j,t-\tau_{j,i})$  and  $\tilde{x}$  and  $\tilde{y}$  represent the solution of the local problem at the same node with  $\xi_{i,t}$  increased by  $\Delta\xi_{i,t}^+$ . We compute the differences:

$$\bar{X}_{l,t-\tau_{j,i}}^+ = \tilde{x}_{l,t-\tau_{k,i}} - \hat{x}_{l,t-\tau_{j,i}} \quad \forall l \in \mathcal{L}_{j,i,t-\tau_{j,i}} \quad (37)$$

$$\bar{Y}_{j,i,t-\tau_{j,i}}^+ = \tilde{y}_{j,i,t-\tau_{j,i}} - \hat{y}_{j,i,t-\tau_{j,i}} \quad (38)$$

Depending on the value of these indicator variables, three cases to compute the increase in the objective function arise. For each case  $m$  we present a computation for the difference  $\Delta G^m(\Delta\xi_{i,t}^+)$  and the condition for that case to apply.

**Case 1:** There exists a terminal  $j$  such that  $\bar{Y}_{j,i,t-\tau_{j,i}}^+ = 1$ .

The increase in  $\xi_{i,t}$  resulted in  $y_{j,i,t-\tau_{j,i}}$  increasing by one unit.

$$\Delta G^1(\Delta\xi_{i,t}^+) = \frac{\partial G}{\partial y_{j,i,t-\tau_{j,i}}^+} \quad (39)$$

$$= \frac{\partial G_{t-\tau_{j,i}}}{\partial y_{j,i,t-\tau_{j,i}}^+} \quad (40)$$

As in Powell & Carvalho (1998a), we use an approximation to evaluate this derivative:

$$\Delta G^1(\Delta\xi_{i,t}^+) \simeq -c_{j,i} + \nu_{i,t}^+ - \nu_{j,t-\tau_{j,i}}^- \quad (41)$$

**Case 2:** There exists a load  $l \in \mathcal{L}_{j,i,t-\tau_{j,i}}$  such that

$$\bar{X}_{l,t-\tau_{j,i}}^+ = 1$$

and

$$\hat{x}_{l,t} = 0 \quad \forall t > t'$$

Thus the increase in  $\xi_{i,t}$  results in  $x_{l,t-\tau_{j,i}}$  increasing from zero to one.

$$\Delta G^2(\Delta \xi_{i,t}^+) = \frac{\partial G}{\partial x_{l,t-\tau_{j,i}}^+} \tag{42}$$

$$= \frac{\partial G_{t-\tau_{j,i}}}{\partial x_{l,t-\tau_{j,i}}^+} \tag{43}$$

Again, in accordance with the results discussed in Powell & Carvalho (1998a),

$$\frac{\partial G_{t-\tau_{j,i}}}{\partial x_{l,t-\tau_{j,i}}^+} \simeq r_{l,t-\tau_{j,i}} + \nu_{i,t}^+ - \nu_{j,t-\tau_{j,i}}^- \tag{44}$$

We arrive at

$$\Delta G^2(\Delta \xi_{i,t}^+) \simeq r_{l,t-\tau_{j,i}} + \nu_{i,t}^+ - \nu_{j,t-\tau_{j,i}}^- \tag{45}$$

**Case 3:** There exists a load  $l \in \mathcal{L}_{j,i,t-\tau_{j,i}}$  such that

$$X_{l,t-\tau_{k,i}}^+ = 1$$

and suppose that there exists  $t' > (t - \tau_{j,i})$  such that:

$$\hat{x}_{l,t'} = 1$$

Thus the increase in  $\xi_{i,t}$  results in  $x_{l,t-\tau_{k,i}}$  increasing from zero to one. As there can only be one time to serve a load, it also implies that  $x_{l,t'}$  decreases, i.e.,  $\tilde{x}_{l,t'} = 0$ . We do not present the proof of the result for this case. However, it is similar to the derivation of  $\nu_{i,t}^+$  for case 3 in Powell & Carvalho (1998a).

$$\Delta G^3(\Delta \xi_{i,t}^+) \simeq r_{l,t-\tau_{k,i}} - r_{l,t'} + \nu_{k,t'}^+ + \nu_{i,t}^+ - \nu_{i,t'+\tau_{k,i}}^- - \nu_{k,t-\tau_{k,i}}^- \tag{46}$$

Finally,

$$\Delta G(\Delta \xi_{i,t}^+) \simeq \begin{cases} \Delta G^1(\Delta \xi_{i,t}^+) & \text{if } \bar{Y}_{j,i,t-\tau_{j,i}}^+ = 1 \\ \Delta G^2(\Delta \xi_{i,t}^+) & \text{if } \exists l \text{ such that } \bar{X}_{l,-\tau_{j,i}}^+ = 1 \text{ and } \hat{x}_{l,t'} = 0 \forall t' > t - \tau_{j,i} \\ \Delta G^3(\Delta \xi_{i,t}^+) & \text{if } \exists l \text{ and } t' > (t - \tau_{j,i}) \text{ such that } \bar{X}_{l,t-\tau_{j,i}}^+ = 1 \text{ and } \hat{x}_{l,t'} = 1 \end{cases} \quad (47)$$

### 2.3 Decreasing the Spatial Potential Function

Using the same approach, we can find  $\Delta \xi_{i,t}^-$ , the amount by which we must decrease  $\xi_{i,t}$  in order to reduce  $V_{i,t}$  by one unit, and derive  $\Delta G(\Delta \xi_{i,t}^-)$ , the resulting impact in the objective function.

We again resort to the decreasing sequence of coefficients  $\mathcal{K} = \{\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n\}$  in the objective function for the  $n$  tasks at node  $(k, t - \tau_{k,i})$ . We define the set  $\mathcal{P}_{k,t-\tau_{k,i}}(i)$  as before. The first  $V_{k,t-\tau_{k,i}}$  tasks would be assigned a vehicle. If  $\bar{c}_v$  is the coefficient of the lowest valued task that is assigned a vehicle that is not headed to node  $(i, t)$ , then  $\bar{c}_{v+1}$  is the coefficient of the highest valued task among those rejected (not headed for  $(i, t)$ ). Let us assume that there exists a task in  $(k, t - \tau_{k,i})$  having node  $(i, t)$  as its destination and that this task is assigned a vehicle in the current solution. If there is no task satisfying these conditions, the flow of vehicles between nodes  $(k, t - \tau_{k,i})$  and  $(i, t)$  is already at its lowest level, i.e., no flow. Let  $\bar{c}_p$  be the coefficient of a task satisfying these conditions in our ordered sequence. As there might be more than one task in node  $(k, t - \tau_{k,i})$  bound to terminal  $i$ , we then compute the decrease in  $\xi_{i,t}$  by

$$\delta_{k,t-\tau_{k,i}}^-(i) = \min_{p \in \mathcal{P}_{k,t-\tau_{k,i}}(i)} \{\bar{c}_p - \bar{c}_v \mid \bar{c}_p - \bar{c}_v > 0\} \quad (48)$$

The value  $\delta_{k,t-\tau_{k,i}}^-(i)$  is the smallest decrease in  $\xi_{i,t}$  that would result in decreasing the flow of vehicles between node  $(k, t - \tau_{k,i})$  and node  $(i, t)$  by one (again, we may need to add an incremental amount for tie-breaking purposes). We need the smallest decrease in  $\xi_{i,t}$  that will divert one vehicle away from  $(i, t)$ , decreasing  $V_{i,t}$  by one unit:

$$\Delta \xi_{i,t}^- = \min_{k \in \mathcal{C}} \{\delta_{k,t-\tau_{k,i}}^-(i)\} \quad (49)$$

Let  $j$  be the origin that solves (49). We then compute the increase in the objective function  $G$  resulting from decreasing  $\xi_{i,t}$  by  $\Delta \xi_{i,t}^-$ . Let this impact be  $\Delta G(\Delta \xi_{i,t}^-)$ . Let  $\hat{x}$  and  $\hat{y}$  be the current

solution of the local problem at node  $(j, t - \tau_{j,i})$  and  $\tilde{x}$  and  $\tilde{y}$  represent the solution of the local problem at the same node with  $\xi_{i,t}$  decreased by  $\Delta\xi_{i,t}^-$ . We compute the differences:

$$\bar{X}_{l,t-\tau_{j,i}}^- = \tilde{x}_{l,t-\tau_{j,i}} - \hat{x}_{l,t-\tau_{j,i}} \quad \forall l \in \mathcal{L}_{j,i,t-\tau_{j,i}} \quad (50)$$

$$\bar{Y}_{j,i,t-\tau_{j,i}}^- = \tilde{y}_{j,i,t-\tau_{j,i}} - \hat{y}_{j,i,t-\tau_{j,i}} \quad (51)$$

Again, we arrive at three cases:

**Case 1:** There exists  $j \in \mathcal{C}$  such that

$$\bar{Y}_{j,i,t-\tau_{j,i}}^- = -1$$

$$\Delta G^1(\Delta\xi_{i,t}^-) \simeq c_{j,i} - \nu_{i,t}^- + \nu_{j,t-\tau_{j,i}}^+ \quad (52)$$

**Case 2:** There exists a load  $l \in \mathcal{L}_{j,i,t-\tau_{j,i}}$  such that

$$\bar{X}_{l,t-\tau_{j,i}}^- = -1$$

and

$$\hat{x}_{l,t} = 0 \quad \forall t > t'$$

Then:

$$\Delta G^2(\Delta\xi_{i,t}^+) \simeq -r_{l,t-\tau_{j,i}} - \nu_{i,t}^- + \nu_{j,t-\tau_{j,i}}^+ \quad (53)$$

**Case 3:** There exists a load  $l \in \mathcal{L}_{j,i,t-\tau_{j,i}}$  such that

$$\bar{X}_{l,t-\tau_{j,i}}^- = -1$$

and suppose that there exists  $t' > (t - \tau_{j,i})$  such that:

$$\tilde{x}_{l,t'} = 1$$

Then:

$$\Delta G^3(\Delta \xi_{i,t}^+) \simeq -r_{l,t-\tau_{j,i}} + r_{l,t'} - \nu_{j,t'}^- - \nu_{i,t}^- + \nu_{i,t'+\tau_{j,i}}^+ + \nu_{j,t-\tau_{j,i}}^+ \quad (54)$$

Finally,

$$\Delta G(\Delta \xi_{i,t}^-) \simeq \begin{cases} \Delta G^1(\Delta \xi_{i,t}^-) & \text{if } \bar{Y}_{j,i,t-\tau_{j,i}}^- = -1 \\ \Delta G^2(\Delta \xi_{i,t}^-) & \text{if } \exists l \text{ such that } \bar{X}_{l,t-\tau_{j,i}}^- = 1 \text{ and } \tilde{x}_{l,t} = 0 \forall t > t' \\ \Delta G^3(\Delta \xi_{i,t}^-) & \text{if } \exists l \text{ such that } \bar{X}_{l,t-\tau_{j,i}}^- = 1 \text{ and } \exists t' > (t - \tau_{j,i}) \text{ such that } \tilde{x}_{l,t'} = 1 \end{cases} \quad (55)$$

### 3 The LAMA algorithm

By solving the local problems at each terminal for all times from time  $t = 0$  to the end of the planning horizon (*Forward Pass*), we get a feasible solution to the problem originally stated as an integer program. Then, by using the equations from section 1.3 we can approximate the gradients of the objective function  $G$  with respect to to the number of vehicles at each node (*Backward Pass*). These gradients can be used to adjust the upper bound controls and/or adjust the multipliers  $\xi$  (*Control Adjustment*).

By changing several upper bounds and/or changing several multipliers in the same iteration we may affect the objective function in an unexpected manner, as the impact of every change is estimated using marginal values. As an example, we have found that very seldom is  $2\nu_{i,t}^+$  a good predictor of the downstream value of adding two vehicles to node  $(i, t)$ .

In order to obtain initial estimates for the upper bound values running a limited number of iterations we chose to follow the strategy outlined in Powell & Carvalho (1998a). Initially, the upper bounds are set to *zero*. By using the gradient estimates  $\eta$ , a gradient step is performed on  $u$  using the steepest ascent direction. The spatial potential function is computed as in the subgradient algorithm (equation 25) with the value of the smoothing factor set to  $\gamma = 0.15$ . This procedure is used in the first 50 iterations and is regarded as the initialization step. Let  $\omega_k$  represent the gradient vector in iteration  $k$ . The control vector  $u$  for iteration  $k + 1$  is updated by

$$u^{k+1} = u^k + s_k \omega_k \quad (56)$$

where the step size  $s_k$  is computed by

$$s_k = \frac{\lambda_k(G_k^u - G_k^l)}{\|\omega_k\|^2} \quad (57)$$

where  $\lambda_k$  is a coefficient for which we have chosen the initial value  $\lambda_1 = 0.5$  and is halved whenever five iterations are performed without improvement in the objective function.  $G_k^u$  and  $G_k^l$  are upper and lower bounds on the objective function. For the lower bound, we use the best value of  $G$  found up to iteration  $k$  for  $G_k^l$ . For the upper bound, we have two choices: one quick and approximate, and the second slower but more strict. The first would involve optimally solving a static approximation of the problem. This is not strictly an upper bound, because it implicitly enforces flow conservation constraints at the beginning and ending of the planning interval. However, because it relaxes all timing restrictions, this will generally give an upper bound for all but the shortest planning horizons. A strict bound can be obtained by solving the LP relaxation. This is much more expensive, but it can be done and does provide a true bound. We chose the latter approach (primarily because we already had to calculate it) but it is generally well known that subgradient stepsize formulas are not especially sensitive to the choice of upper and lower bound. This procedure leads to non-integer values for  $u$ . Therefore, whenever using the values obtained by using (56) they are rounded to the closest integer.

The idea behind the LAMA method is to carefully control  $u$  and  $\xi$  so that between any two iterations, the actual change in the objective function is always very close to the expected change. In order to accomplish this, after the initialization step of 50 iterations, we only change one upper bound or one multiplier per iteration. So, we alternate between iterations where one upper bound is adjusted or the spatial potential function at one node is updated.

**Updating an upper bound:** The procedure to update an upper bound consists of employing a coordinate search. First, the gradients  $\eta^+$  and  $\eta^-$  are computed as in equations (32) and (33). Let  $\mathcal{W} = \{w_1, w_2, \dots, w_N\}$  be the set of gradient values, where an element  $w_n$  represents a value  $\eta_{i,j,t}^+$  or  $\eta_{i,j,t}^-$ . Let  $\ell_n$  be the corresponding element, which might be  $(i, j, t)^+$  or  $(i, j, t)^-$ . So, if  $\ell_n = (i, j, t)^+$ , then  $w_n = \eta_{i,j,t}^+$ . We increase or decrease the upper bound on the link that corresponds to the highest valued  $w_n$  by one unit depending on whether  $\ell_n = (i, j, t)^+$  or  $\ell_n = (i, j, t)^-$ .

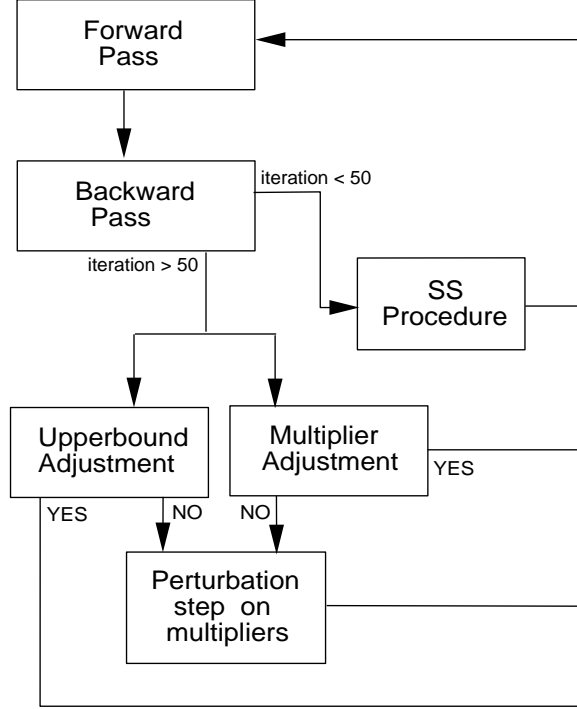


Figure 7: Iterative procedure for the LAMA method.

**Updating the spatial potential function at a node:** In iterations where  $\xi$  is adjusted we also employ a coordinate search. The values of  $\Delta G(\Delta \xi_{i,t}^+)$  and  $\Delta G(\Delta \xi_{i,t}^-)$  are used to update one component of the vector  $\xi$ . Let  $\mathcal{W} = \{w_1, w_2, \dots, w_N\}$  be the set of variations in the objective function, where an element  $w_n$  represents a value  $\Delta G(\Delta \xi_{i,t}^+)$  or  $\Delta G(\Delta \xi_{i,t}^-)$ . Let  $\ell_n$  be the corresponding element, which might be  $(i, t)^+$  or  $(i, t)^-$ . So, if  $\ell_n = (i, t)^+$ , then  $w_n = \Delta G(\Delta \xi_{i,t}^+)$ . Now, let  $\ell_{max}$  be the highest-valued element. We increase or decrease  $\xi_{i,t}$  by the corresponding  $\Delta \xi_{i,t}$ , depending on whether  $\ell_{max} = (i, t)^+$  or  $\ell_{max} = (i, t)^-$ .

However, our preliminary runs have shown that after running many iterations it is very likely that no upper bound and no multiplier can be adjusted yielding an improvement in the objective function. This is due to our restrictive policy of small changes from iteration to iteration. We then added a perturbation step on the multipliers, which we describe below, arriving at the iterative procedure shown in figure 7.

**Perturbation step on the spatial potential function:** The perturbation step on the multipliers consists of slightly altering their values so that they get a little closer to the values of the gradients  $\nu$ . Whenever we get to iteration  $n$  and no upper bounds or multipliers can be adjusted

improving the overall objective function, we update the multipliers for iteration  $n + 1$  using:

$$\xi_{i,t}^{n+1} = (1 - \alpha)\xi_{i,t}^n + \alpha\nu_{i,t}^n \tag{58}$$

where  $\alpha = 0.01$ .

We chose to run a total of 500 iterations, because very little, if any additional improvement at all could be obtained by running more iterations.

## 4 Numerical Results

We use two measures of performance. The first one is the CPU ratio, which is the ratio between the CPU time that our linear solver, CPLEX, took to find the optimal solution of the linear program and the CPU time to run the 500 iterations of the LAMA method. The second measure of performance is the OPT ratio, which is the ratio between the value of the objective function obtained by the LAMA method and the optimal value obtained by CPLEX.

Table 1 shows the data sets used to evaluate the performance of the LAMA method. Figure 8 shows the evolution of the objective function through 500 iterations for data set 1 using the LAMA and the subgradient algorithm. The first 50 iterations are performed using the gradient step procedure to adjust the upper bounds. On subsequent iterations, the small drops in the CPU ratio for LAMA are the result of resorting to the overall perturbation of the multipliers by a small percentage of the value of the gradients  $\nu$ . This figure shows that the LAMA method returns better solutions and also has a much more stable objective function. The stability arises because the more controlled changes in  $\nu$  reduces the shuffling of which loads are served and when. Large changes to the vector  $\nu$  can shift the prioritization of tasks at each point in time, which introduces tremendous downstream instabilities in problems with long planning horizons.

The data sets in table 1 are identical to those used in Powell & Carvalho (1998a). The results obtained in that work using the subgradient algorithm are reproduced in table 2 together with the results obtained by using the LAMA method. Clearly, the LAMA method returns results that are closer to the optimal solution at the expense of more computation time. (Note: the CPU times for the LAMA algorithm exclude the time required to obtain the linear relaxation used

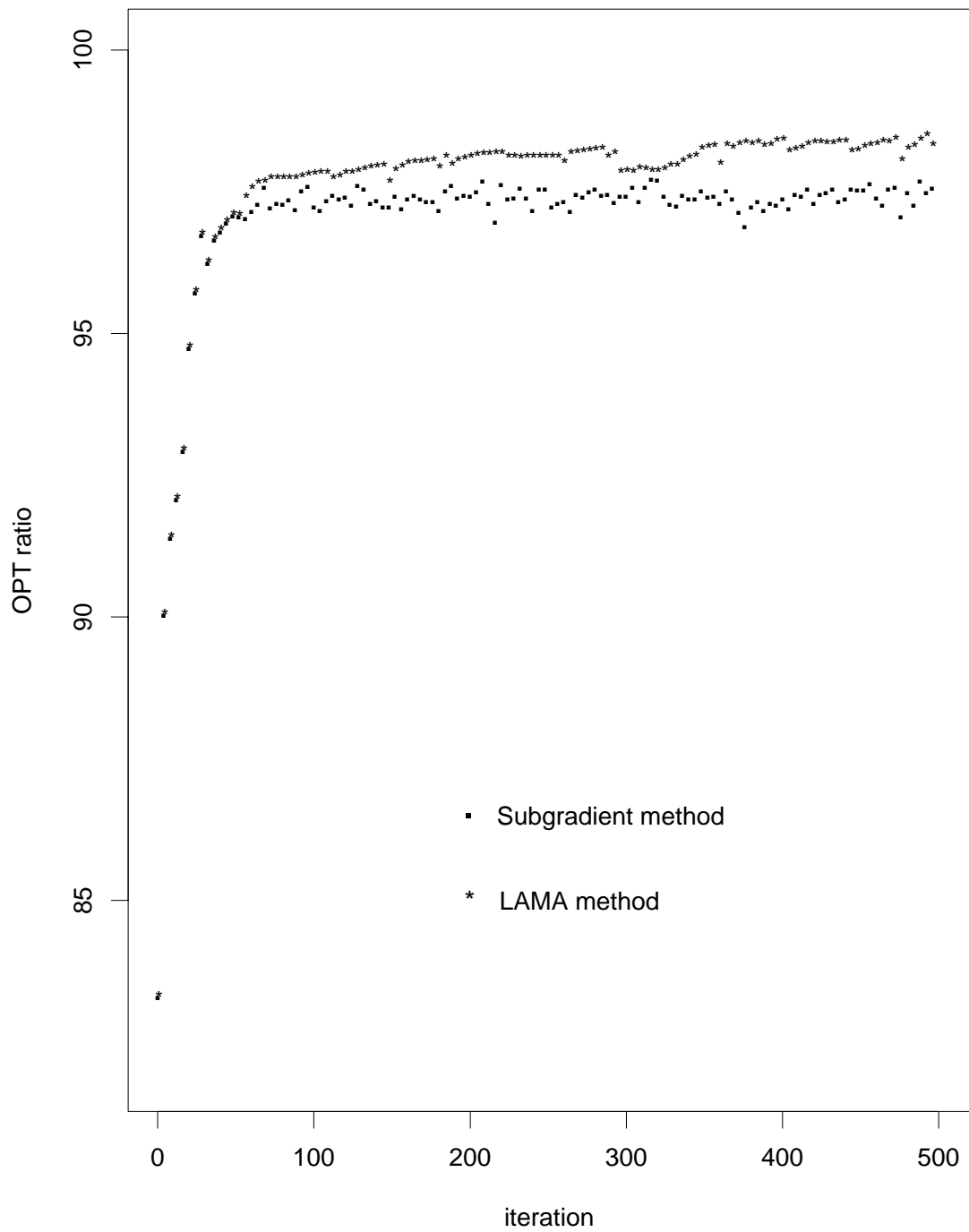


Figure 8: Percentage of optimal solution as a function of the iteration number for data set 1 using the LAMA and the subgradient methods. Notice that only one in every four iterations is shown.

data set	loads	vehicles	horizon (hours)	time period (hours)	dist. time window (hours)
1	2000	400	120	4	U[0,40]
2	3000	600	120	4	U[0,40]
3	4000	800	120	4	U[0,40]
4	5000	1000	120	4	U[0,40]
5	6000	1200	120	4	U[0,40]
6	8000	1600	120	4	U[0,40]
7	10000	2000	120	4	U[0,40]
8	2000	200	120	4	U[0,40]
9	2000	300	120	4	U[0,40]
10	2000	500	120	4	U[0,40]
11	2000	600	120	4	U[0,40]
12	2000	400	120	2	U[0,40]
13	2000	400	120	6	U[0,40]
14	2000	400	120	8	U[0,40]
15	1000	400	60	4	U[0,40]
16	4000	400	240	4	U[0,40]
17	6000	400	360	4	U[0,40]
18	2000	400	120	4	0
19	2000	400	120	4	20
20	2000	400	120	4	40
21	2000	400	120	4	60

Table 1: Summary of problem sets used for testing.

for the upper bound; these were excluded because we can obtain bounds very quickly by solving a static approximation.) This table indicates that the largest gain with LAMA as opposed to the subgradient algorithm is on problems with long horizons and loads with narrow departure time windows. The average increase in the OPT ratio over all data sets is 0.7 percent. Viewed differently, this represents a 25 percent reduction in the optimality gap. From a practical perspective, the most significant advantage of the LAMA algorithm is its stability. We are comparing results to the best solution that the subgradient algorithm ever produced; in some applications (in particular, real-time applications) it is necessary to use the most current solution, which may be far from the best available.

The longer CPU time for the LAMA method results from our choosing to run a larger number of iterations when compared to the 150 iterations the subgradient algorithm ran. It is indeed expected to take more iterations to reach a “good” solution with the LAMA method, as the solution obtained

data set	Subgradient		LAMA	
	CPU ratio	OPT ratio	CPU ratio	OPT ratio
1	17.5	97.6	4.8	98.4
2	24.4	97.7	6.6	98.3
3	43.6	97.8	11.4	98.3
4	38.4	97.8	11.1	98.4
5	67.4	98.1	18.5	98.6
6	83.8	98.0	25.6	98.3
7	88.1	98.0	25.1	98.6
8	41.0	92.5	10.3	92.7
9	35.8	95.7	10.2	96.3
10	8.6	98.1	2.1	98.6
11	12.1	98.0	3.3	98.5
12	85.5	95.8	23.4	96.1
13	6.3	97.5	1.5	98.2
14	4.0	97.9	1.1	98.5
15	2.0	97.5	0.5	98.0
16	98.4	95.5	34.6	96.6
17	231.0	93.4	66.6	96.0
18	1.2	96.5	0.3	98.1
19	17.1	97.6	3.9	98.2
20	25.3	97.5	7.4	97.7
21	22.5	98.1	6.1	98.6
Average	45.4	97.0	13.1	97.7

Table 2: Comparison between the subgradient algorithm and LAMA. LAMA CPU times exclude the time required to obtain the relaxed upper bound.

in the Forward Pass changes very little from iteration to iteration. Also, LAMA takes a slightly longer time per iteration, due to the additional computation required to estimate the variations in the spatial potential function and in the objective function. For data set 1, the subgradient algorithm took an average of 0.687 seconds per iteration, while the LAMA method took 0.754 seconds. The increase in computation time per iteration is roughly 10 percent.

## 5 Conclusion

This paper shows that the LAMA algorithm produces better objective functions than the subgradient algorithm originally presented in Powell & Carvalho (1998a). This result, however, is reached at the expense of longer CPU times. Out of the 21 data sets, 16 resulted in solutions that were within 2.0 percent of the optimal value of the linear relaxation. Some of this gap is due to the integrality constraints. Hane, Barnhart, Johnson, Marsten, Nemhauser & Sigismondi (1995) reports integrality gaps of up to 0.2 percent for problems with similar structure but which are much smaller. Thus, we expect that only a fraction of our optimality gap is due to integrality. Some of the gap is due to the use of simple adjustment strategies for the control vector  $u$ . This paper shows that some of the optimality gap reported in Powell & Carvalho (1998a) is due to the instabilities in the gradient adjustment strategy originally proposed in that paper.

This paper also demonstrates the potential of a control theoretic formulation. This approach is inherently hierarchical, which offers advantages for problems in areas such as rail and container shipping where many decisions are made locally. The experimental work in this paper suggests that it is possible for these problems to be solved locally, given the right network information (contained in the control vector  $(\xi, u)$ ). Of particular interest is the high quality obtained using this approach.

The ultimate goal of this research is not to produce an algorithm that provides optimal solutions for simple, deterministic fleet management problems, but rather to provide a tool that has the flexibility to solve much more complex problems, as well as to handle the uncertainties that always accompany these problems. For example, Powell & Carvalho (1998b) uses the method to optimize the flows of flatcars for a railroad. The “loads” to be moved are trailers and containers that are placed on top of the flatcars in various configurations. The LQN formulation allowed us to relatively easily handle the complex rules governing the configurations of trailers and containers that would fit on a particular type of flatcar.

## Acknowledgement

This research was supported in part by grant AFOSR-F49620-93-1-0098 from the Air Force Office of Scientific Research. We would also like to gratefully acknowledge the careful attention given by one referee whose many comments improved the overall presentation.

## References

- Cheung, R. and W.B. Powell, “An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management”, *Operations Research* **44**, 951–963 (1996).
- Crainic, T., M. Gendreau and P. Dejax, “Dynamic stochastic models for the allocation of empty containers”, *Operations Research* **41**, 102–126 (1993).
- Frantzeskakis, L. and W.B. Powell, “A successive linear approximation procedure for stochastic dynamic vehicle allocation problems”, *Transportation Science* **24**(1), 40–57 (1990).
- Hane, C., C. Barnhart, E. Johnson, R. Marsten, G. Nemhauser and G. Sigismondi, “The fleet assignment problem: Solving a large-scale integer program”, *Math. Prog.* **70**, 211–232 (1995).
- Herren, H. , “The distribution of empty wagons by means of computer: An analytical model for the Swiss Federal Railways (SSB)”, *Rail International* **4**(1), 1005–1010 (1973).
- Herren, H. , “Computer controlled empty wagon distribution on the SSB”, *Rail International* **8**, 25–32 (1977).
- Joborn, M., Empty freight car distribution at Swedish Railways - analysis and optimization modeling, Ph.D. thesis, Department of Mathematics, Linköping University, Sweden, (1995).
- Jordan, W. and M. Turnquist, “A stochastic dynamic network model for railroad car distribution”, *Transportation Science* **17**, 123–145 (1983).
- Magnanti, T. and R. Simpson, Transportation network analysis and decomposition methods, Report no. dot-tsc-rspd-78-6, U.S. Department of Transportation, 1978.
- Powell, W. , “A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers”, *Transportation Science* **30**, 195–219 (1996).
- Powell, W. and T. A. Carvalho, T. A., “Dynamic control of logistics queueing network for large-scale fleet management”, *Transportation Science* **32**, 90–109 (1998a).
- Powell, W. and T. A. Carvalho, T. A., “Real-time optimization of containers and flatcars for intermodal operations”, *Trans. Sci.* **32**, 110–126 (1998b).
- Powell, W. B., P. Jaillet and A. Odoni, Stochastic and dynamic networks and routing, in C. Monma, T. Magnanti & M. Ball, eds, ‘*Handbook in Operations Research and Management Science*, Volume on *Networks*’, North Holland, pp. 141–295 (1995).
- Turnquist, M., Mov-em: A network optimization model for empty freight car distribution, School of Civil and Environmental Engineering, Cornell University (1986).
- White, W., “Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers”, *Networks* **2**, 211–236 (1972).
- White, W. and A. Bomberault, “A network algorithm for empty freight car allocation”, *IBM Systems Journal* **8**, 147–171 (1969).