Dynamic Control of Logistics Queueing Networks for Large-Scale Fleet Management

Warren B. Powell
Tassio A. Carvalho

Department of Civil Engineering
and Operations Research
Princeton University
Princeton, NJ 08544

Statistics and Operations Research
Technical Report SOR-96-01

February 6, 1998

Abstract

Dynamic fleet management problems are normally formulated as networks over dynamic networks. Additional realism usually implies the inclusion of complicating constraints, typically producing exceptionally large integer programs. In this paper, we present for the first time the formulation of dynamic fleet management problems in an optimal control setting, using a novel formulation called a Logistics Queueing Network (LQN). This formulation replaces a single, large optimization problem with a series of very small problems that involve little more than solving a single sort at each point in space and time. We show that this approach can produce solutions that are within a few percent of a global optimum, but providing for considerably more flexibility than standard linear programs.

We consider the problem of managing a homogeneous fleet of vehicles over time to serve a set of loads, each with a known origin and destination, and a specified time window in which they must be served. The problem has been widely studied. The most common formulation is a dynamic network with three types of arcs: revenue generating arcs representing the market demand between two cities at a point in time, empty repositioning arcs, which represent the movement of capacity from one location to the next at a point in time (at a positive cost), and inventory arcs, capturing the cost of holding capacity at the same location over time.

The first published formulation of this problem as a linear network appears to be by White & Bomberault (1969) and White (1972), although the basic formulation had been well known prior to this time (see, for example, Dantzig & Fulkerson (1954)). The work of White and Bomberault focused on presenting specialized algorithms for this model, and said little about the model itself. Magnanti & Simpson (1978), in an unpublished technical report, give a series of linear programming models, extending the basic dynamic network formulation to handle multiple fleet types and time windows for task arcs. An extensive review of these models can be found in Powell, Jaillet & Odoni (1995*a*).

The formulation and solution of dynamic networks for fleet assignment models has received attention in an airline context, but has not proved effective in large scale fleet management problems arising in rail, containers and trucking. These problems involve the routing of tens of thousands of vehicles, serving thousands of tasks per day. The difficulty is that tasks (representing the movement of freight over space and time) can be moved within time windows that can range from very narrow to exceptionally wide. We are not aware of any research demonstrating the feasibility of linear programming-based models for these large fleet assignment problems with time windows.

Jordan & Turnquist (1983) approach the fleet management problem for rail as a nonlinear inventory distribution problem for empty freight cars, and allow for backlogging of unsatisfied demands. Powell (1988), Frantzeskakis & Powell (1990), and Powell & Cheung (1994), show how the dynamic fleet management problem can be modified to handle uncertainty in demand forecasts, but these models require demands to be served in a specific time period. Powell (1996) shows how fleet management problems with time windows can be solved on a rolling-horizon basis, but does not explicitly solve the fleet assignment problem with time windows.

Recently, Powell, Carvalho, Godfrey & Simao (1995*b*) presented a new formulation of the fleet management problem, called a *logistics queueing network*. The solution approach starts with the

classical linear programming formulation, and is then reformulated as a recursive dynamic program. Initial computational results suggested that the approach showed promise, producing solutions that were within four percent of a (continuous) optimal solution produced by a commercial linear programming solver.

The goal of this paper is to develop more fully the logistics queueing network formulation introduced in Powell et al. (1995*b*). Although we build on this prior work, the paper is self contained.

The contributions of this paper are as follows: a) we introduce for the first time a motivation of the LQN algorithm based on a linear approximation of the dynamic programming formulation; b) we provide a new derivation of the gradients used in the optimization; c) we present new algorithms for updating the control variables; d) we conduct a much more thorough set of computational experiments, which yield insights into the effect of data characteristics on solution quality; and finally e) we show that the version of the LQN algorithm in this paper achieves solutions that are generally within three percent of the optimal solution of the linear programming relaxation. This performance improves with solution size, and degrades when it becomes more tightly constrained. It is also substantially faster than a linear programming solver, and easily handles problems that are substantially larger than those tested here. However, the most important benefit of the approach is the ease with which complex operational details are handled.

This paper is organized as follows. Section 1 formulates the problem as an integer program. The linear relaxation is used later to produce a bound on the quality of our solution. Then, section 2 offers a control theoretic solution approach based on an approximation of a dynamic programming formulation. This approach depends on the use of an estimate of the gradient of the value function in the dynamic program. Section 3 presents an intuitive development of these equations, while a more detailed derivation is given in appendix A. Section 5 provides a complete description of the algorithm, and introduces key variations. Experimental results are described in Section 6 and a summary of the paper is given in Section 7.

1 Linear Programming Model

This section formulates the fleet management problem as an integer program. Later, we use the linear relaxation as a bound on the algorithm that we propose.

We assume that time is divided into a set of discrete instants $\mathcal{T} = (0, 1, \ldots, T)$ where T is the length of the planning horizon.

We present the complete notation for the problem here. Some of this notation is not used until later.

Network variables:

- $\mathcal{C}$ is the set of terminals i in the network.
- τ_{ij} is the travel time between terminal $i \in \mathcal{C}$ and terminal $j \in \mathcal{C}$.
- $\mathcal{N}$ is the set of nodes $(i, t), i \in \mathcal{C}$, $t \leq T$, in the dynamic network.

We simplify many of our derivations in this paper by assuming that $\tau_{ij} = 1$, but all of our results are easily generalized to positive, integer (and finite) travel times.
Activity variables:

- $\mathcal{L}$ is the set of loads l available within the planning horizon T.
- $\mathcal{L}_{ijt}$ is the set of loads $l \in \mathcal{L}$ with origin i and destination j having t as a feasible departure time.
- $\mathcal{L}_{it}$ is made of the union of all sets $\mathcal{L}_{ijt}$ such that $j \in \mathcal{C}$.
- $\bar{\mathcal{L}}_{ijt}$ 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.
- $\bar{\mathcal{L}}_{it}$ is made of the union of all sets $\bar{\mathcal{L}}_{ijt}$ for all the destinations $j \in \mathcal{C}$.
- $\mathcal{L}_t$ is made of the union of all sets $\mathcal{L}_{it'}$ such that $t' \geq t$.
- $\mathcal{L}^0_{it}$ is the set of loads l with origin i, where t is the beginning of the time window $\mathcal{T}_l$.
- $\mathcal{L}^f_{it}$ is the set of loads l with origin i, where t is the end of the time window $\mathcal{T}_l$.
- R_{it} is the net inflow ($R_{it} > 0$) or outflow ($R_{it} < 0$) of vehicles at city i at time t. Normally, we will assume that $R_{it} = 0$ for $t \geq 1$.
- V_{it} is the total number of vehicles at node (i, t) waiting to be assigned to a load, or moved empty.

Parameters

- $\mathcal{T}_l$ is the set of feasible departure times for satisfying load $l \in \mathcal{L}$, otherwise known as the *departure time window*.
- r_{lt} is the profit generated by choosing time t to satisfy load l.
- c_{ij} is the cost of repositioning one vehicle over link (i, j, t).

Decision variables:

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

The objective function is given by:

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

The complete integer program can now be expressed as follows:

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

subject to:

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

$$\sum_{l \in \mathcal{L}_{ijt}} x_{lt} + y_{ijt} - w_{ijt} = 0 \quad \forall\, i,\ j \in \mathcal{C},\ \forall\ t \leq T \tag{4}$$

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

$$y_{ijt}\,,\ w_{ijt} \geq 0 \tag{6}$$

$$x_{lt} = (0, 1) \tag{7}$$

Our problem consists of maximizing profits by assigning up to one vehicle to each load – constraints (3) – and enforcing conservation on the flow of vehicles at each node – constraints (4) and (5). This formulation can be regarded as a network problem with GUB side constraints, which are represented by (3).

The purpose of this linear program is to evaluate the quality of the solutions obtained by the LQN approach. To obtain the optimal solution, one needs to solve the linear relaxation and then use branching to find the optimal integer solution. We limit ourselves to finding the optimal solution of the linear relaxation of this problem. Using branching on a problem with tens of thousands of variables is not practical using current technology. Hane, Barnhart, Johnson, Marsten, Nemhauser & Sigismondi (1994) investigated problems with a similar structure to the ones in this paper but with half as many constraints and variables as our smallest data set. In some cases more than $6,000$ seconds were spent in the branching procedure with customized branching strategies. We formulate large scale, dynamic fleet management problems as a Logistics Queueing Network (LQN). This formulation easily accomodates arbitrary time windows on loads to be served, and can be modified to handle a wide variety of operational issues that may arise in specific applications. In this paper, we propose a solution approach that decomposes fleet management decisions by time and space, using gradient information and control variables to guide the solution toward a global optimum. Each iteration of the algorithm involves a simulation of the dispatching process, after which gradients and control variables are updated to improve the solution. Even using deterministic data, we find that dynamic problems with long planning horizons exhibit a chaotic property, motivating the use of techniques found in stochastic optimization. Comparisons to solutions produced by a linear programming package suggest that our method will produce solutions within three percent of the global (but noninteger) optimum, with run times that may be 10 to 100 times faster.

This formulation ignores other constraints that arise in real-world applications like terminal capacities, load prioritization and labor regulations of the crews assigned to vehicles. It could indeed be extended to allow for several equipment types or loads that use a sequence of links through intermediate terminals from origin to destination. However, we have chosen the simplest setting, as it already leads to somewhat large linear programs.

2 A control theoretic formulation

What makes the formulation in section 1 so complex is not the presence of the integer variables, but rather the fact that we are trying to solve the problem over an extended planning horizon. In this section, we use a dynamic programming formulation to set up the problem in a control theoretic setting. In this formulation, the pair (x_t, y_t) represents the set of decisions at time t, which is normally referred to as the *control vector*. Shortly, we will show that this is not the correct perspective. In addition, the state of our system is given by $\mathcal{S}_t = (V_t, \mathcal{L}_t)$, which gives the vector of inventories of vehicles at time t, and the set of uncovered tasks at time t.

The problem can be formulated as a dynamic program using the optimality recursion:

$$G_t(V_t\,, \mathcal{L}_t\,) \;\; = \;\; \max_{x_t, y_t} \sum_{i \in \mathcal{C}} g_{it}(x_t, y_t, V_{it}\,, \mathcal{L}_{it}\,) + G_{t+1}(V_{t+1}\,, \mathcal{L}_{t+1}\,) \tag{8}$$

where:

$$g_{it}(x\,, y, V_{it}\,, \mathcal{L}_{it}\,) \;\; = \;\; \sum_{l \in \mathcal{L}_{it}} r_{lt} x_{lt} - \sum_{j \in \mathcal{C}} c_{ij} y_{ijt}$$

g_{it} 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. It follows trivially that

$$F(x\,, y) = G_0(V_0\,, \mathcal{L}_0\,)$$

The equations for the LQN method can be derived by mixing two ingredients together. The first one is to choose an approximation for the value function, represented by $G_{t+1}(V_{t+1}, \mathcal{L}_{t+1}\,)$ in equation (8). The second is to constrain the unbounded decision variables by control variables, in our case by setting upper bounds for the y variables that represent empty moves. The bounds on these variables are necessary because the approximation for the value function uncouples the decisions for different terminals. Thus we define u_{ijt} as the upper bound on y_{ijt} for every link (i, j, t).

The value function $G_t(V_t, \mathcal{L}_t)$ is, of course, intractably complex. Instead, we replace it in

equation (8) with a linear approximation. Let:

$$L_{ijt} = |\mathcal{L}_{ijt}|$$

be the number of tasks in the set $\mathcal{L}_{ijt}$, and let:

$$L_t = (\ldots, L_{ijt}, \ldots)$$

be the vector giving the number of tasks between every pair of cities at time t. We can now adopt the linear approximation:

$$\tilde{G}_t = \xi_t V_t + \mu_t L_t \tag{9}$$

where ξ_t is an estimate of the slope of G_t with respect to the supply of vehicles V_t, and μ_t is an estimate of the slope with respect to the number of tasks waiting to be served, given by L_t. We refer to ξ_t as the *supply gradient* and μ_t as the *task gradient*. In this paper, we further simplify our approach by ignoring the task gradient in $\tilde{G}_t$. This creates a new approximation:

$$\hat{G}_t = \xi_t V_t \tag{10}$$

Substituting $\hat{G}_t$ from equation (10) into (8) then gives:

$$\hat{G}_t(V_t, \mathcal{L}_t) = \max g_{it}(x, y, V_{it}, \mathcal{L}_{it}) + \xi_{t+1} V_{t+1} \tag{11}$$

The gradient ξ_t can be thought of as giving an estimate of the value of an incremental unit of capacity at each point in space. For this reason, we refer to ξ_t as the *spatial potential function*.

The scalar product $\xi_{t+1} V_{t+1}$ can be written as:

$$\sum_{j \in \mathcal{C}} \xi_{j,t+1} V_{j,t+1} = \sum_{j \in \mathcal{C}} \xi_{j,t+1} \sum_{k \in \mathcal{C}} w_{kjt} \tag{12}$$

$$= \sum_{j \in \mathcal{C}} \xi_{j,t+1} \sum_{k \in \mathcal{C}} \Big(\sum_{l \in \mathcal{L}_{kjt}} x_{lt} + y_{kjt} \Big) \tag{13}$$

Replacing (13) and the definition of g_{it} in (11),

$$\hat{G}_t(x, y, V_t, \mathcal{L}_t, \xi_{t+1}) = \sum_{i \in \mathcal{C}} \left(\sum_{j \in \mathcal{C}} \sum_{l \in \mathcal{L}_{ijt}} (r_{lt} + \xi_{j,t+1}) x_{lt} + \sum_{j \in \mathcal{C}} (-c_{ij} + \xi_{j,t+1}) y_{ijt} \right) \tag{14}$$

We have adopted a linear approximation of G_t since it is the simplest to estimate and use. An important contribution of this paper is demonstrating that this approach can provide excellent results for certain problem classes. The biggest limitation of linear approximations is that they can be unstable. For this reason, we introduce an additional decision variable:

$$u_{ijt} = \text{an upper bound on the flow of empties from } i \text{ to } j \text{ at time } t$$

We choose in our approximation to only limit the flow of empty vehicles. Loaded vehicles are already limited by market demands. This does not mean that our solution would not benefit from a separate bound on loaded movements, or redefining u_t to apply to the total flow (loaded and empty).

One difficulty with the use of the upper bound vector u_t is that the problem can be potentially infeasible. Normally, these problems are feasible, since we always have the option of doing nothing. For this reason, we introduce an additional variable $\tilde{y}_{iit}$ which represents flow that is held in a region from node (i,t) to node $(i,t+1)$ at zero cost. The total flow held in inventory, then, is given by:

$$\bar{y}_{iit} = y_{iit} + \tilde{y}_{iit}$$

We include, then, an upper bound on the variable y_{iit}, but not on $\tilde{y}_{iit}$.

The resulting linear program for stage t can now be expressed as:

$$\max_{x,y} \hat{G}_t(x, y, V_t, \mathcal{L}_t, \xi_{t+1}) \tag{15}$$

subject to:

$$\sum_{t \in \mathcal{T}_l} x_{lt} \leq 1 \qquad \forall l \in \mathcal{L}_{it}, \ \forall i \in \mathcal{C} \tag{16}$$

$$y_{ijt} \leq u_{ijt} \quad \forall i, j \in \mathcal{C} \tag{17}$$

$$\sum_{l \in \mathcal{L}_{it}} x_{lt} + \sum_{j \in \mathcal{C}} y_{ijt} \leq V_{it} \quad \forall i \in \mathcal{C} \tag{18}$$

The attraction of this subproblem is that it decomposes by node (i,t). For the problems we are considering in this paper (homogeneous resources and tasks) this subproblem is a simple sort.

We rank all the options of moving loaded or empty, where the value of a loaded move is given by $r_{lt} + \xi_{j,t+1}$ (where j is the destination of load l), while the value of an empty move (to destination j) is given by $-c_{ij} + \xi_{j,t+1}$. We now rank all the options, and put as much flow on the options with the highest value. The total flow on a loaded option is bounded by L_{ijt} while empty moves are bounded by u_{ijt}.

Given the simplicity of the subproblem, we can think of the optimal solution (x_t, y_t) as being functions of the *control variables* ξ_t and u_t. That is, given (ξ_t, u_t), we can find (x_t, y_t). This functional dependence can be expressed as:

$$x_{lt} = x_{lt}(V_{it}, \xi_{t+1}, u_{it}, \mathcal{L}_{it}) \tag{19}$$

$$y_{ijt} = y_{ijt}(V_{it}, \xi_{t+1}, u_{it}, \mathcal{L}_{it}) \tag{20}$$

After finding x_{lt} and y_{ijt} at a given time t, we must advance the state of our system. This is done using:

$$\bar{\mathcal{L}}_{k,t+1} = \{\bar{\mathcal{L}_{kt}} \setminus \{\mathcal{L}^s_{kt} \cup \mathcal{L}^f_{kt}\}\} \cup \mathcal{L}^0_{k,t+1} \quad \forall k \in \mathcal{C} \tag{21}$$

$$V_{k,t+1} = V_{k,t} - \sum_j w_{k,j,t} + \sum_i w_{i,k,t+1-\tau_{i,k}} + R_{k,t+1} \ \ \forall k \in \mathcal{C} \tag{22}$$

We can now state our problem as one of optimizing the control variables ξ_t and u_t for all points in time t, as follows:

$$\max_{\xi,u} \sum_{t=0}^{T} \sum_{l \in \mathcal{L}_t} r_{lt} x_{lt}(V_{it}, \xi_{t+1}, u_{it}, \mathcal{L}_{it}) - \sum_{j \in \mathcal{C}} c_{ij} y_{ijt}(V_{it}, \xi_{t+1}, u_{it}, \mathcal{L}_{it})$$

subject to (16) - (18) and (21) - (22).

The challenge at this point is to devise a strategy to update ξ and u so that the solution of the control problem closely matches the global optimization formulation. Since ξ_t is an estimate of the slope of $\hat{G}_t$, the dual associated with the constraint (18) provides us with an estimate of this slope. Define:

$$\nu_{it} = \frac{\partial G_t}{\partial V_{it}}$$

In addition, let ν^+_{it} and ν^-_{it} represent, respectively, the right and left derivatives of $\hat{G}_t$ with respect to V_{it}. In section 5, we show how the vector ν_t is used to update both ξ_t and u_t. Thus, the

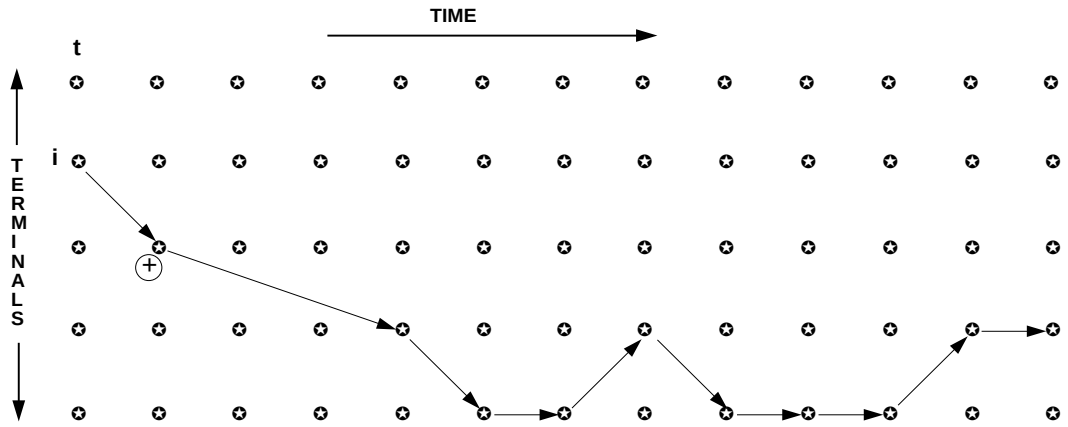

Figure 1: Example of a flow augmenting path.

calculation of ν_t is central to the algorithm. We begin this development in section 3 by developing an appreciation of the dynamics of the problem. Then, section 4 provides an intuitive derivation an estimate of the gradient ν_t (a more detailed derivation is given the appendix). After this, section 5 gives a detailed summary of the algorithm.

3 Understanding the dynamics

Central to the LQN algorithm is understanding the effect of a perturbation of the vehicle supply vector V_t. In the case of fleet management problems where the tasks move at predetermined times, the effect of increasing the supply of vehicles by one unit at a node in the network is well known to be described by a flow augmenting path into a supersink (see, for example, Powell (1989)). In the presence of time windows, however, the problem is much more complex. This section uses a series of illustrations to communicate the dynamics of fleet management problems in the presence of time windows. In doing so, we show that even deterministic problems can exhibit a near chaotic property (when time horizons are sufficiently long). Later, we use this property to justify the adoption of certain smoothing procedures from stochastic optimization.

Assume that we have a problem defined by the external supply of vehicles R_t and the set of tasks to be covered $\mathcal{L}_t$, $t = 0, 1, 2, \ldots, T$. Let the optimal set of flows be denoted (x^0, y^0). Now assume that we solve the problem again, but increment V_{it} by one unit for some node (i, t), and let the new set of flows be denoted (x^1, y^1). The change in flows $(x^1 - x^0, y^1 - y^0)$ in the simplest case will be as represented in figure 1. This figure depicts a simple flow augmenting path, giving

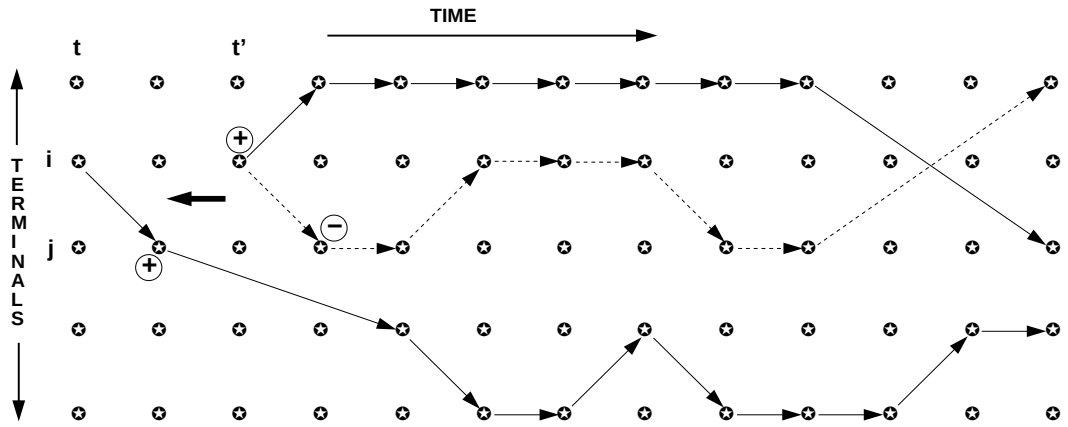

Figure 2: Example of flow augmenting paths with no interactions

the effect of adding one vehicle at terminal i starting at time t, through the rest of the planning horizon. Let $p^+(i,t)$ be the total contribution along this path. This situation will arise only if the incremental unit of flow is used to cover a task ℓ that was not covered in the original solution (this will always occur, for example, in problems with no time windows). Let ∇G_t be the actual change in G_t, while ν_t is the gradient of G_t with respect to V_t. We let ν_t^+ and ν_t^- represent the right and left derivatives. In this simple case,

$$\begin{aligned}
\nabla G_{it} &= \nu_{it}^+ \\
&= p^+(i,t) \\
&= r_{\ell,t} + p^+(j,t+1) \\
&= r_{\ell,t} + \nu_{i,t+1}^+
\end{aligned}$$

where we assume that the task terminates at j one time period later.

Now assume that the additional vehicle at node (i,t) allows us to cover a task ℓ that was otherwise covered at a later time t'. This situation is shown in figure 2. Here, we again have the flow augmenting path starting at node (i,t), but we also have a different set of perturbations created by the removal of task ℓ from the set of tasks to be moved at time t'. This change has the effect of freeing one vehicle at node (i,t'), and removing a vehicle from node $(j,t'+1)$. These two changes create flow augmenting and flow decrementing paths out of node (i,t). This situation nicely depicts how a change in supply of vehicles at time t can change the tasks to be covered at a future time t', which has further downstream impacts on the supply of vehicles at other terminals.

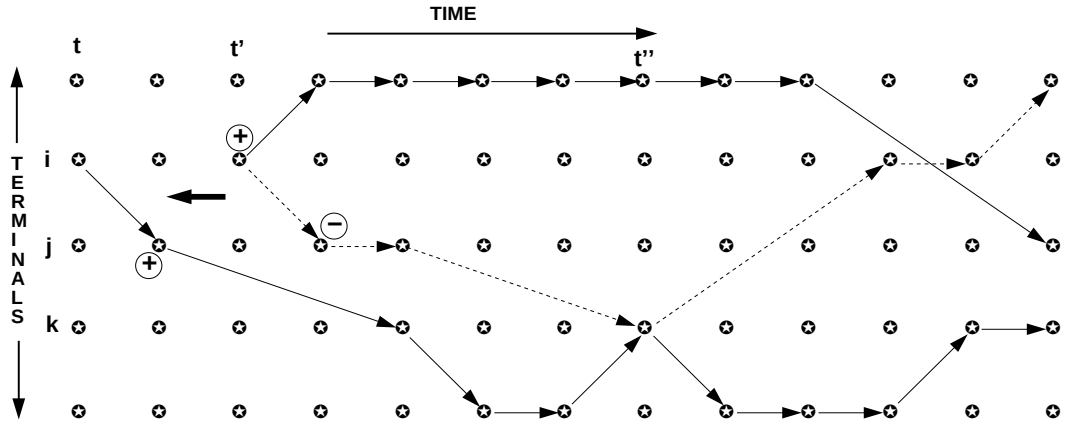

Figure 3: Interaction between one flow augmenting path and one flow decreasing path.

The cost of this change is seen to be given by:

$$\begin{aligned} \nabla G_t &= \nu_{it}^{+} \\ &= p^{+}(i,t) + [-r_{\ell,t'} + p^{+}(it') - p^{-}(j,t'+1)] \\ &= p^{+}(i,t) + [-r_{\ell,t'} + \nu_{it'}^{+} - \nu_{j,t'+1}^{-}] \end{aligned} \tag{23}$$

where $p^{-}(j,t'+1)$ is the cost of the flow decrementing path from $(j,t'+1)$. Note that $p^{+}(i,t)$ will include $r_{\ell,t}$, giving the contribution of covering task ℓ at time t. The term in brackets captures the downstream impact from dropping a task from the set of available tasks at time t'.

So far, we have been able to estimate global changes exactly. In both these instances we were able to provide an exact estimate of the effect of a change in the supply of vehicles using the left and right derivatives ν_{it}^{+} and ν_{it}^{-}. Now consider the situation depicted in figure 3. Here we have an intersection of two paths at terminal k at time t''. Let $p^{+}(i,t;k,t'')$ be the cost of a flow augmenting path from (i,t) to (k,t''), and let $p^{-}(i,t;k,t'')$ be the corresponding flow decrementing path. If the flow augmenting path out of (i,t) passes through (k,t''), then clearly:

$$p^{+}(i,t) = p^{+}(i,t;k,t'') + p^{+}(k,t'') \tag{24}$$

Correspondingly, if the flow decrementing path from $(j,t'+1)$ also passes through node (k,t'') then

$$p^{-}(j,t'+1) = p^{-}(j,t'+1;k,t'') + p^{-}(k,t'') \tag{25}$$

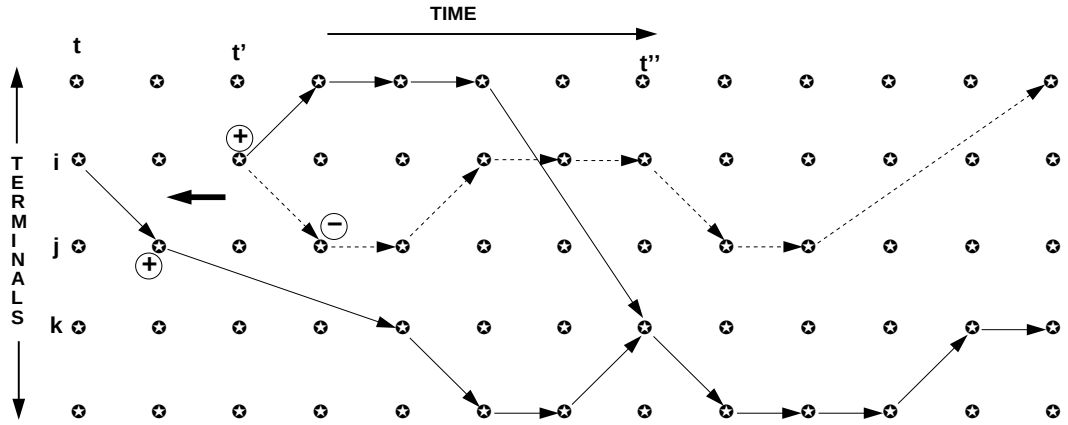

Figure 4: Interaction between two flow augmenting paths.

Because the flow changes cancel at (k, t''), the actual change in costs is given by:

$$\nabla G_{it} = p^+(i,t;k,t'') - p^-(j,t';k,t'') + [-r_{\ell,t'} + p^+(it')] \tag{26}$$

In appendix A.2, we show that (23) gives an exact expression for the gradient ν_{it}^+ if there are no path interactions. Consider now what happens when we use this same expression in the presence of path interactions. Substituting (24) and (25) into equation (23) gives:

$$\begin{aligned} \nu_{it}^+ &= p^+(i,t;k,t'') + p^+(k,t'') + [-r_{\ell,t'} + p^+(it') - p^-(j,t';k,t'') - p^-(k,t'')] \\ &= [p^+(i,t;k,t'') - p^-(j,t';k,t'')] + [p^+(k,t'') - p^-(k,t'')] + \\ &\quad [-r_{\ell,t'} + p^+(it')] \end{aligned} \tag{27}$$

From concavity we have:

$$p^+(k,t'') - p^-(k,t'') \leq 0$$

Comparing the actual change in (26) to the gradient estimate given in (27) we see that:

$$\nu_{it}^+ \leq \nabla G_{it}$$

Now turn to the situation depicted in figure 4. Here we have two flow augmenting paths combining at node (k, t''). If $p^+(k,t'')$ is the contribution of the first increment of flow at (k, t''), let

$p^{2+}(k,t'')$ be the contribution of the second unit of flow. Now the actual change in the objective function is given by:

$$\begin{aligned}\nabla G_t &= p^+(i,t;k,t'') + p^+(k,t'') + \left[-r_{\ell,t'} + p^+(i,t';k,t'') + \right. \\ &\quad \left. p^{2+}(k,t'') - p^-(j,t'+1)\right]\end{aligned}$$

Our estimate of the gradient, by contrast, uses:

$$\begin{aligned}\nu_{it} &= p^+(i,t;k,t'') + p^+(k,t'') + [-r_{\ell,t'} + p^+(i,t';k,t'') + \\ &\quad p^+(k,t'') - p^-(j,t'+1)]\end{aligned} \tag{28}$$

Since $p^{2+}(k,t'') < p^+(k,t'')$:

$$\nu_{it}^+ \geq \nabla G_{it}$$

Thus, our estimate of the gradient is neither an upper or lower bound on the actual change in the objective function. But empirically it does seem to be a good approximation.

The examples just presented offer fairly simple interactions of flows. We see that the presence of time windows on tasks can create complex downstream effects when the supply of vehicles is changed. An increase in supply of vehicles at any node can result in a task sliding to a later departure time, while a reduction in supply can push tasks to an earlier departure time. Any sliding of a task in turn can create a cascading of downstream changes in flows. Figure 5 attempts to illustrate a more complex set of interactions. In a large network, the effect of a single perturbation in supply early in the simulation can produce a nearly chaotic impact on downstream activities, complicating the problem of accurately estimating the effect of a change in flows. We believe this behavior is characteristic of this class of dynamic systems, and we have used it to simplify our algorithm in the belief that it is virtually impossible, in any realistic setting, to accurately estimate these downstream impacts.

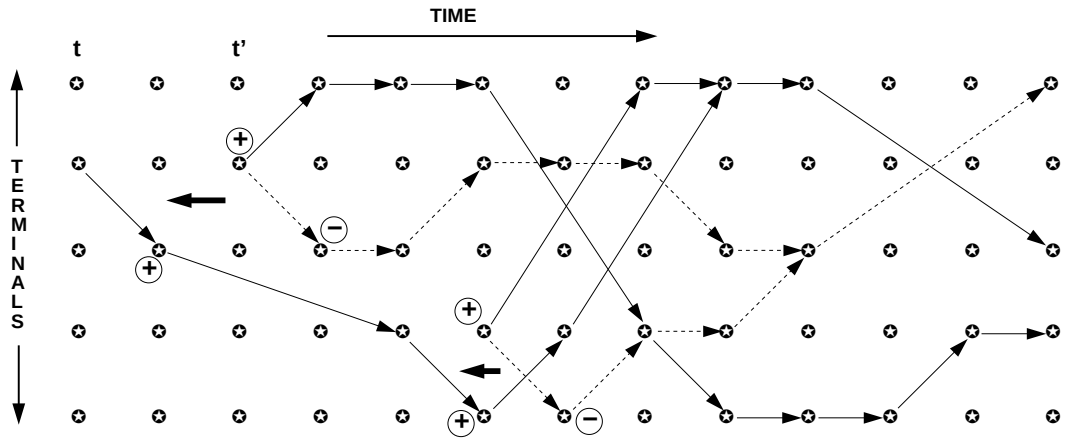

Figure 5: Several downstream interactions between different paths.

4 The gradients

In this section, we draw on the intuition provided in section 3 to develop expressions for the right derivative ν_{it}^+, defined as:

$$\nu_{it}^+ = \frac{\partial G_t(V_t, \mathcal{L}_t)}{\partial V_{it}^+}$$

Our goal is to use ν_t^+ to build an estimate of ξ_t in equation (11). As in the previous section, our derivation is somewhat intuitive. Appendix A gives a more precise development of the expressions given here.

4.1 The basic calculation

Assume that instead of differentiating $G_t(V_t)$, we differentiate $\hat{G}_t(V_t)$ in equation (14). In this case, we obtain an approximate gradient:

$$\begin{aligned} \hat{\nu}_t^+ &= \frac{\partial \hat{G}_t(V_t, \mathcal{L}_t)}{\partial V_{it}^+} \\ &= \frac{\partial g_{it}}{\partial V_{it}^+} + \sum_{j \in \mathcal{C}} \xi_{j,t+1} \frac{\partial V_{j,t+1}}{\partial V_{jt}^+} \end{aligned} \tag{29}$$

To obtain the derivative of g_{it} with respect to V_{it}, we need to find the change in the flows x_t and y_t with respect to V_{it}. Assuming $l \in \mathcal{L}_{it}$, define:

$$\frac{\partial x_{lt}}{\partial V_{it}^+} = X_{lt}^+ = x_{lt}(V_{it}+1, \nu_{t+1}, u_{it}, \mathcal{L}_{it}) - x_{lt}(V_{it}, \nu_{t+1}, u_{it}, \mathcal{L}_{it}) \tag{30}$$

$$\frac{\partial y_{ijt}}{\partial V_{it}^+} = Y_{ijt}^+ = y_{ijt}(V_{it}+1, \nu_{t+1}, u_{it}, \mathcal{L}_{it}) - y_{ijt}(V_{it}, \nu_{t+1}, u_{it}, \mathcal{L}_{it}) \tag{31}$$

Note that we index X_{lt} by only l and t; implicit is that load l originates at location i. We also note that since:

$$V_{i,t+1} = \sum_{l \in \mathcal{L}_{i,t-1}} x_{l,t-1} + \sum_{k \in \mathcal{C}} y_{k,i,t-1}$$

we can write:

$$\frac{\partial V_{i,t+1}}{\partial V_{it}^+} = \sum_{l \in \mathcal{L}_{i,t-1}} X_{l,t-1}^+ + \sum_{k \in \mathcal{C}} Y_{k,i,t-1}^+ \tag{32}$$

Substituting (30), (31) and (32) into (29) gives:

$$\hat{\nu}_{it}^+ = \sum_{j \in \mathcal{C}} \sum_{l \in \bar{\mathcal{L}}_{ijt}} \hat{X}_{lt}^+ (r_{lt} + \xi_{j,t+1}) + \sum_{j \in \mathcal{C}} \hat{Y}_{ijt}^+ (-c_{ij} + \xi_{j,t+1}) \tag{33}$$

Equation (33) gives a reasonable estimate for ν. Initial experiments, however, showed that a better approximation was obtained by adding a term that provides an estimate of the task gradient μ (see equation (9)). Rather than using the aggregated set L_{ijt} defined in equation (9), we use the entire set $\mathcal{L}$, which implies that there is an element μ_{lt} for each task l, for each time period t. In this case, we replace equation (9) with:

$$\tilde{G}_t = \xi_t V_t + \mu_t \tilde{x}_t \tag{34}$$

where:

$$\tilde{x}_{lt} = \begin{cases} 1 & \text{if task } l \text{ has not been served by time } t \\ 0 & \text{otherwise} \end{cases}$$
$$= 1 - \sum_{t' < t} x_{lt'}$$

We estimate μ_{lt} as follows. A given task l might be served before time t, at time t, after time t, or not at all. The indicator that the task was served after time t is given by

$$1 - \tilde{x}_{lt} - x_{lt} = \sum_{t' \geq t+1} x_{lt} \tag{35}$$

If this expression is 1, and if $X^+_{lt} = 1$ (which means that $x_{lt} = 0$), then the perturbation had the effect of moving task l from some time $t' > t$ to time t. An estimate of the impact of this change on the system is written:

$$\tilde{\mu}_{l,t+1} = \sum_{t' \geq t+1} (r_{lt'} - \nu^+_{it'} + \nu^-_{j,t'+\tau_{ij}}) x_{lt'} \tag{36}$$

The intuition behind this derivation is presented in section 3. If $x_{lt'}$ is increased from 0 to 1, then we pick up the revenue $r_{lt'}$, we subtract a unit of flow from node (i, t') and we add a unit of flow to node $(j, t' + \tau_{ij})$. Equation (36) gives this calculation assuming that $x_{lt'} = 1$. In our work, $\nu^+_{it'}$ and $\nu^-_{j,t'+1}$ are replaced by $\xi_{it'}$ and $\xi_{j,t'+1}$. Finally, we observe that the derivative of $\tilde{x}_{t+1}$ with respect to V_{it} is given by $-X^+_{lt}$. This provides a new approximation for ν^+_{it} which we denote $\tilde{\nu}^+$ (consistent with the fact that we are now approximating $\tilde{G}_t$), given by:

$$\tilde{\nu}^+_{it} = \hat{\nu}^+_{it} - \sum_{l \in \mathcal{L}_{it}} X^+_{lt} \tilde{\mu}_{l,t+1} \tag{37}$$

The numerical experiments in this paper use equation (37), since it produced the best results. However, the original approximation for ν given in equation (36) is much simpler, and may prove adequate for many applications.

4.2 Smoothing

At each iteration n, we obtain a new set of flows (x^n_t, y^n_t), new vehicle inventories V^n_t, and a new set of gradients $\tilde{\nu}^n_t$. As pointed out in section (3), even a small change in an early time period can be magnified into larger changes in later time periods. As a result, we expect the estimates $\tilde{\nu}^n_t$ to bounce around considerably. For this reason, we smooth the estimates using:

$$\bar{\nu}^{n+1}_{it} = \gamma^n \tilde{\nu}^{n+1}_{it} + (1 - \gamma^n) \bar{\nu}^n_{it}$$

where γ^n, $0 < \gamma^n \leq 1$, is a smoothing factor that must be chosen experimentally. We now propose to use, at iteration n in the algorithm:

$$\xi_t^n = \bar{\nu}_t^n$$

for the linear approximation in (9). We also use $\bar{\nu}_t^n$ in equation (36) to compute $\tilde{\mu}_t^n$.

Our solution algorithm closely mimics that used in stochastic optimization (see, for example, Ermoliev (1988), Gupal & Bazhenov (1972)). While our problem is not stochastic, the dynamics of the problem impart a stochastic flavor, since the downstream effects of changes can be highly complex and difficult to predict. Of course, real problems *are* stochastic, and it is useful that our algorithm can be used for stochastic problems with almost no changes.

4.3 Variations

Our estimate of the gradient is based on an expression for the derivative of $\hat{G}_t(V_t)$ as a function of V_t. However, there is more changing from one iteration to the next than simply V_t. Our approximation $\hat{G}_t$ depends on ξ_t and μ_t, both of which now depend on $\bar{\nu}^n$, which also changes from one iteration to the next. As a result, while we are trying to estimate the effect of changes in V_t, we are also changing ξ (and μ) which impacts the change in the objective function from one iteration to the next. We do not make an explicit attempt at estimating the change in $\hat{G}_t(V_t)$ due to changes in ξ_t, but strategies can be developed which incorporate these effects.

Changes in ξ_{t+1} can be captured through their impact on the flows x_t and y_t, which in turn is measured through the difference variables X_t and Y_t (we drop the superscript designations X^+ and Y^+ for simplicity here). Originally, (X_t, Y_t) were calculated using the difference between a solution where $V_{it}^{n+1} = V_{it}^n + 1$ and $\xi_t^{n+1} = \xi_t^n = \bar{\nu}_t^n$. This suggests two variations: the first variation uses $\xi_t^{n+1} = \bar{\nu}_t^{n+1}$ and $\xi_t^n = \bar{\nu}_t^n$. The second variation uses $\xi_t^{n+1} = \bar{\nu}_t^{n+1}$ and $\xi_t^n = \bar{\nu}_t^{n+1}$.

All three methods are summarized as follows:

- Compute the marginal value of a vehicle at node (i,t) according the gradient computation derived in the previous section, i.e., look at the marginal value of a vehicle in the current solution at node (i,t). In order to compute ν_{it}^+ we reproduce equations (30) and (31) here with the superscript n on the gradients added to indicate the iteration

number:

$$\begin{aligned} X^+_{lt'}(n+1) &= x_{lt'}(V_{it'}+1, \nu^n_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - x_{lt'}(V_{it'}, \nu^n_{t+1}, u_{it}, \mathcal{L}_{it'}) \\ Y^+_{i,j',t'}(n+1) &= y_{i,j,t'}(V_{it'}+1, \nu^n_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - y_{i,j,t'}(V_{it'}, \nu^n_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \end{aligned}$$

- Take advantage of the new gradients computed in iteration $n+1$ to solve again the local problem at node (i,t) with an additional vehicle and compare it to the current solution at node (i,t). In order to compute the indicator variables for ν^+_{it}, one would then have to use:

$$\begin{aligned} \hat{X}^+_{lt'}(n+1) &= x_{lt'}(V_{it'}+1, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - x_{lt'}(V_{it'}, \nu^n_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \\ \hat{Y}^+_{i,j',t'}(n+1) &= y_{i,j,t'}(V_{it'}+1, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - y_{i,j,t'}(V_{it'}, \nu^n_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \end{aligned}$$

- Take advantage of the new gradients computed in iteration $n+1$ to solve again the local problem at node (i,t) with an additional vehicle and compare it to the resolved local problem *without* the additional vehicle:

$$\begin{aligned} \tilde{X}^+_{lt'}(n+1) &= x_{lt'}(V_{it'}+1, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - x_{lt'}(V_{it'}, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \\ \tilde{Y}^+_{i,j',t'}(n+1) &= y_{i,j,t'}(V_{it'}+1, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - y_{i,j,t'}(V_{it'}, \nu^{n+1}_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \end{aligned}$$

We call these procedures *SUBG1*, *SUBG2* and *SUBG3* respectively. The indicator variables X^- and Y^- to compute ν^-_{it} are adapted in the same fashion.

It is important to observe that methods *SUBG2* and *SUBG3* both require knowledge of $\bar{\nu}^{n+1}_{t+1}$ in order to calculate $\bar{\nu}^{n+1}_t$. These calculations can only be done in a backward pass that starts at time $t = T$ and progresses back to $t = 0$. Backward propagation of derivative information is common in dynamic systems and is expected to provide better answers. However, the backward pass is more complex to implement.

5 The LQN Algorithm

The LQN algorithm is summarized in figure 6.

The forward pass

STEP 1 Initialization:

- Set $\bar{\nu}^0 = 0, u^0 = 0$.
- Set $n = 1$.

STEP 2 Forward pass:

- For $t = 0, 1, 2, \ldots, T$:
  - Find $x_t^{n+1}(V_t^n, \bar{\nu}_{t+1}^n, u_t^n, \mathcal{L}_t^n)$ and $y_t^{n+1}(V_t^n, \bar{\nu}_{t+1}^n, u_t^n, \mathcal{L}_t^n)$.
  - Find V_{t+1}^{n+1} and $\mathcal{L}_{t+1}^{n+1}$ given x_t^{n+1} and y_t^{n+1}.

STEP 3 Backward pass:

- For $t = T, T-1, T-2, \ldots, 0$:
  - Find $X^{(n+1)+}$ and $Y^{(n+1)+}$ using one of the methods *SUBG1*, *SUBG2* or *SUBG3*.
  - Calculate $\tilde{\nu}_t^{n+1}$ and $\tilde{\mu}_t^{n+1}$.
  - Calculate the smoothed gradients $\bar{\nu}_t^{n+1}$.

STEP 4 Global update:

- Upate the control variables u^{n+1}.
- Check termination criteria; if not satisfied, return to Step 2.

Figure 6: The LQN Algorithm

The forward pass takes as input the gradients $\bar{\nu}_t^n$ and bounding variables u_t^n and simply calculates $x_t^{n+1}(V_t^n, \bar{\nu}_{t+1}^n, u_t^n, \mathcal{L}_t^n)$ and $y_t^{n+1}(V_t^n, \bar{\nu}_{t+1}^n, u_t^n, \mathcal{L}_t^n)$ by solving $\hat{G}_t^n$, which involves little more than solving a series of simple sorts. The strength of this procedure is that its sheer simplicity makes it relatively easy to add a number of operational issues. For example, if the real problem is stochastic, it is relatively simple to sample any random variables, such as random demands or travel times. If the application is simulating the movements of containers that need to move by train or ship on a fixed schedule, it is easy to reflect the fact that certain containers can only depart at certain points in time.

The backward pass

The backward pass executes one of the procedures *SUBG1*, *SUBG2* or *SUBG3*, starting at the end of the horizon and working backward. We first calculate the vector of changes in flows, represented by $X^{(n+1)+}$ and $Y^{(n+1)+}$, and then the gradients $\tilde{\nu}^{n+1}$ and $\tilde{\mu}^{n+1}$.

Upper bound adjustment

Increasing an upper bound u_{ijt} can have the effect of increasing the flow of empties from i starting at time t and arriving to j at time $t+1$. Let η_{ijt}^+ and η_{ijt}^- be, respectively, the right or

left derivatives of the total costs as a function of u_{ijt}. In appendix A.3, we develop the following approximations:

$$\eta^{+}_{ijt'} \begin{cases} \simeq -c_{ij} + \tilde{\nu}^{+}_{j,t'+1} - \tilde{\nu}^{-}_{it'} & \text{if } -c_{ij} + \tilde{\nu}^{+}_{j,t'+1} - \tilde{\nu}^{-}_{it'} > 0 \\ = 0 & \text{otherwise} \end{cases} \tag{38}$$

The derivation for $\eta^{-}_{ijt'}$ is similar and leads to

$$\eta^{-}_{ijt'} \begin{cases} \simeq -c_{ij} + \tilde{\nu}^{-}_{j,t'+1} - \tilde{\nu}^{+}_{it'} & \text{if } -c_{ij} + \tilde{\nu}^{-}_{j,t'+1} - \tilde{\nu}^{+}_{it'} < 0 \\ = 0 & \text{otherwise} \end{cases} \tag{39}$$

We have looked at two strategies to adjust the upper bounds. The first one is to employ a coordinate search (CS). We use η^{+}_{ijt} and η^{-}_{ijt} to determine whether we should increase or decrease any component of the control vector. Let $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$ be the set of gradient values, where an element w_n represents a value η^{+}_{ijt} or η^{-}_{ijt}. Let ℓ_n be the corresponding element, which might be $(i,j,t)^{+}$ or $(i,j,t)^{-}$. So, if $\ell_n = (i,j,t)^{+}$, then $w_n = \eta^{+}_{ijt}$. Now, let $\mathcal{M} = \{\ell_n, n = 1,2,\ldots,M; w_n > 0\}$ be the highest-valued M elements. We increase or decrease $u_{ijt}, (i,j,t) \in \mathcal{M}$ by one, depending on whether $\ell_n = (i,j,t)^{+}$ or $\ell_n = (i,j,t)^{-}$.

The second strategy is to take a gradient step (SS) in the steepest ascent direction. Let ω_n represent the gradient vector in iteration n. The control vector for iteration $n+1$ is updated by

$$u^{n+1} = u^{n} + s_n \omega_n \tag{40}$$

where the step size s_n is computed by

$$s_n = \frac{\lambda_n (F^{u}_{n} - F^{l}_{n})}{\|\omega_n\|^2}$$

where λ_n is a coefficient for which we have chosen the initial value $\lambda_1 = 0.5$ and is halved whenever five iterations are done without improvement in the objective function. F^{u}_{n} and F^{l}_{n} are upper and lower bounds on the objective function. We use the optimal value of the linear relaxation of the problem as F^{u}_{n}. As we have solved the relaxation to optimality, that number is readily available. In real world applications, we recommend two procedures to come up with an upper bound on the objective function. One can add up the profit of all loads in the problem. This should be a fairly tight bound in problems found in practice. It is possible to obtain a tighter bound resorting to a network problem where nodes are terminals, links represent loaded and empty moves and the total

supply of vehicles at each terminal is the inflow at each terminal. Whatever the choice, the value of the parameter λ_1 must be properly adjusted. We use the best value of F found up to iteration n for F_n^l.

This procedure leads to non-integer values for u. Therefore, the values obtained by using (40) are rounded to the closest integer to be used in the Forward Pass.

Another experimental question is whether we should use $\tilde{\nu}^n$ in equations (38) and (39), or $\bar{\nu}^n$. We call the method that uses $\tilde{\nu}^n$ "Smoothing after upper bound adjustment" (SAU), and the method that uses $\bar{\nu}^n$ "Smoothing before upper bound adjustment" (SBU).

6 Numerical experiments

In this section, we try to evaluate the quality of the LQN algorithm in terms of traditional measures such as objective function and execution time. It is important to realize, however, that an important strength of the algorithm is its tremendous flexibility. Just the same, it is important to quantify the quality of the solution, and provide a measure of its ability to solve large, realistic problems.

Section 6.1 describes a test bank of problems that arise in fleet management. These problems are all large enough to be interesting, and small enough to allow us to solve them to optimality using a commercial linear programming solver. Section 6.2 then uses a single base problem to tune the algorithm, determine the best solution strategies and smoothing parameters. Finally, section 6.3 tests the algorithm on the full bank of problems to provide a measure of the performance of the algorithm and its sensitivity to certain characteristics of the datasets.

6.1 Experimental design

The data sets we selected for testing are described in table 1. Data set 1 is the standard data set we used to evaluate the algorithmic issues. The loads and the initial distribution of vehicles were randomly generated over a real set of 40 terminals spread across the Eastern United States. We chose 50 cents per mile as the cost of moving empty and 20 cents of profit per loaded mile. The data generation accounted for the demand imbalances so common in real-world problems by using different probability weights of a terminal being an origin or a destination of a load.

The linear relaxation of each dataset was solved to optimality by solving the linear program

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.

described in section 1. Throughout our experiments, the objective function is expressed as a percentage of this optimal linear relaxation. We do not have any indication of the size of the gap between the linear relaxation and the optimal integer solution, but we suspect it to be quite small.

6.2 Algorithm Selection and Calibration

The algorithmic research questions we posed were:

1. What is the comparative performance of the algorithm when procedures SUBG1, SUBG2 and SUBG3 are employed in the Backward Pass?

2. What is the comparative performance of the coordinate search (CS) and the gradient step (SS) when one of these procedures is used to adjust the control vector? Does a combination of both procedures offer an even better performance?

3. Should the control adjustment be performed with the smoothed gradients (SBU) or the raw ones (SAU)?

4. What is the best value for the smoothing factor γ^n?

The vectors u, ν and μ had their components initially set to *zero*. Gradients with a time coordinate beyond the planning horizon, i.e., ν_{it} for $t > T$ are always *zero*. Our preliminary experiments have shown that smoothing factors γ^n in the lower range do yield better results than high ones. Thus we chose to show the result of the combination of the procedures described in the previous section for $\gamma^n = 0.2$, $\gamma^n = 0.4$ and $\gamma^n = 0.6$. Whatever the combination of procedures, the objective function has shown to reach somewhat stable values after at most a few hundred iterations. Table 2 shows the best OPT ratio for these values of γ^n obtained when the procedure is run for 600 iterations.

Using the smoothed gradients to perform the upper bound adjustment (SBU) has systematically yielded better results than using the raw gradients. Meanwhile, when comparing the variations of the Backward Pass, none of the procedures had a clear lead, as it can be seen in table 2. Our conclusion was also based on further investigation that we do not report. We chose to adopt SUBG1 as the standard Backward Pass procedure for its consistent performance and because the gradient computations used in it are thoroughly derived in Appendix A.

Also from the same table it can be seen that the coordinate search (CS) provides marginally

Back Pass procedure	Control procedure	γ^n	OPT ratio SAU	OPT ratio SBU
SUBG1	CS	0.2	97.1	97.7
		0.4	96.8	97.3
		0.6	96.1	96.5
	SS	0.2	96.9	97.5
		0.4	96.6	96.8
		0.6	95.3	95.8
SUBG2	CS	0.2	97.3	96.8
		0.4	96.9	97.1
		0.6	96.3	96.6
	SS	0.2	96.8	97.4
		0.4	96.3	97.0
		0.6	96.0	96.5
SUBG3	CS	0.2	97.2	97.4
		0.4	96.7	97.3
		0.6	96.0	96.8
	SS	0.2	97.0	97.4
		0.4	96.5	96.8
		0.6	95.4	95.9

Table 2: OPT ratio for different procedures.

better results than the gradient step (SS) procedure to adjust the control vector. Figure 7 shows the evolution of the objective function using both control adjustment strategies for $\gamma^n = 0.2$ and $\gamma^n = 0.6$. The gradient step procedure has much better convergence properties. Rather than choosing one procedure over the other, these plots suggest that a better strategy is to start using procedure SS and then switch to CS after several iterations.

The next issue is the choice of smoothing factor. Figure 8 presents the best OPT ratio obtained in 600 iterations by using either the coordinate search or the gradient step procedures to adjust the control vector for values of γ^n such that $0.05 \leq \gamma^n \leq 0.6$. For this plot we have used the procedure we selected for the Backward Pass, SUBG1, and the control adjustment was performed with the smoothed gradients.

Considering figures 7 and 8 we decided on the following strategy to constrain the number of iterations the algorithm runs. Choose a value for γ^n such that $0.1 \leq \gamma^n \leq 0.3$. Run the algorithm for a fixed number of iterations using the gradient step (SS) procedure for control adjustment. Switch to the coordinate search procedure (CS) for a fixed number of iterations and then exit. We call this the hybrid *SS+CS* procedure and the iterations using CS are performed with $M = 1$, i.e., only one upper bound is adjusted per iteration. For our standard data, we chose $\gamma^n = 0.15$ and ran SS for 50 iterations and CS for 100. We obtained 97.6% of optimality with a CPU time of 103 seconds.

Our second measure of performance is the *CPU ratio*, which is the ratio between the CPU time for our linear program solver, CPLEX, to reach optimality and the CPU time spent by our hybrid algorithm to run 150 iterations. Notice, however, that the linear program solver was set to find the solution of the linear relaxation of the problem. Further computation would be necessary in order to find integer solutions. The linear program solver took 1798 CPU seconds to reach optimality for the standard data set. Thus, the CPU ratio was 17.5.

In order to validate our restriction on the number of iterations, we have generated five other data sets shown in table 3. They were generated using the same distributions and parameters used to generate the standard data set. The table shows that the results are similar to those obtained using the standard data set. This table also shows the OPT ratio obtained when using either the CS or the SS procedures alone for 600 iterations. Even though CS returned results that are sometimes slightly better than the hybrid procedure, they were obtained at the expense of much longer computation times.

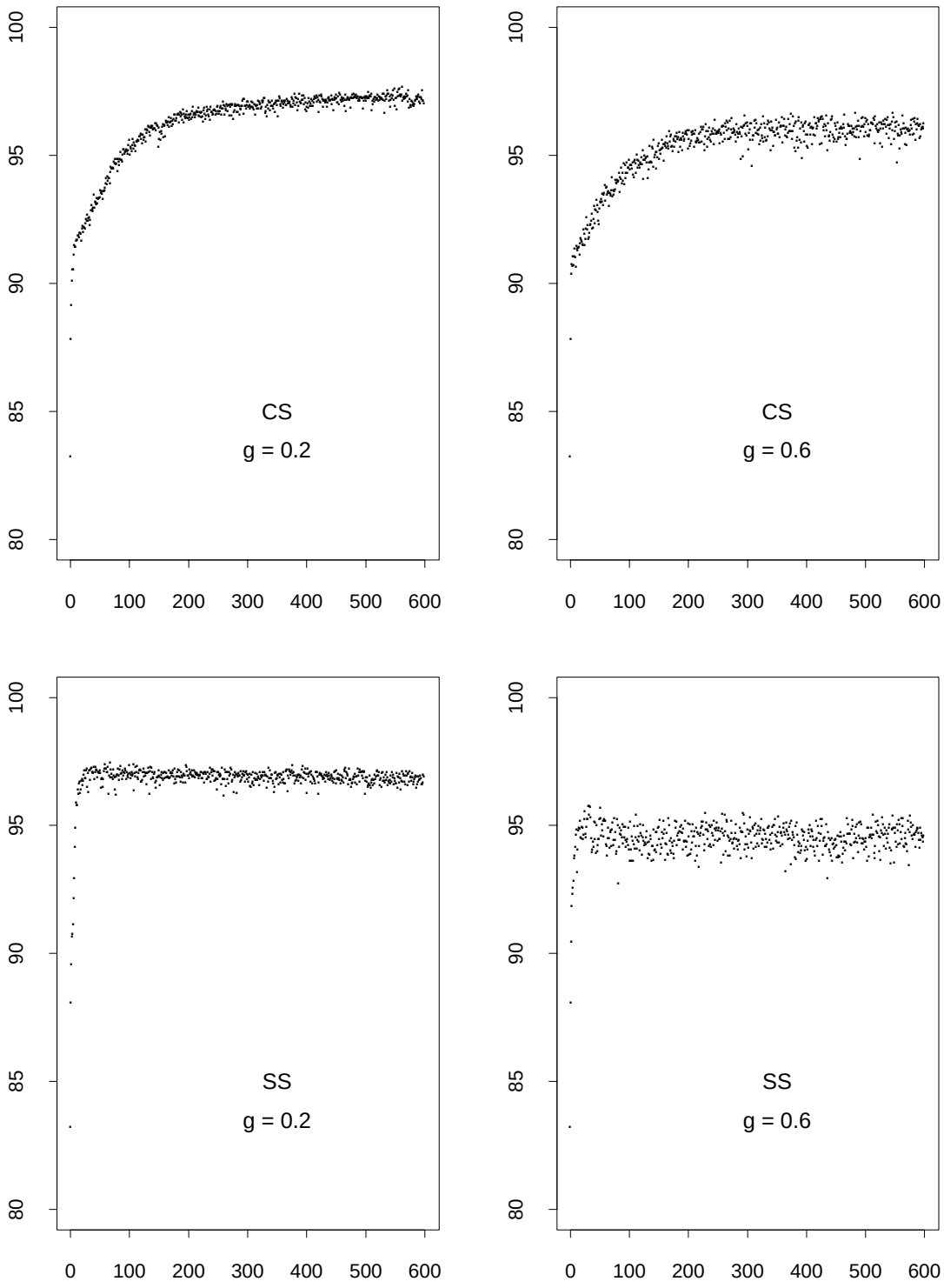

Figure 7: Percentage of optimal solution as a function of the iteration number for selected values of γ^n for control adjustment through coordinate search (CS) and gradient step (SS).

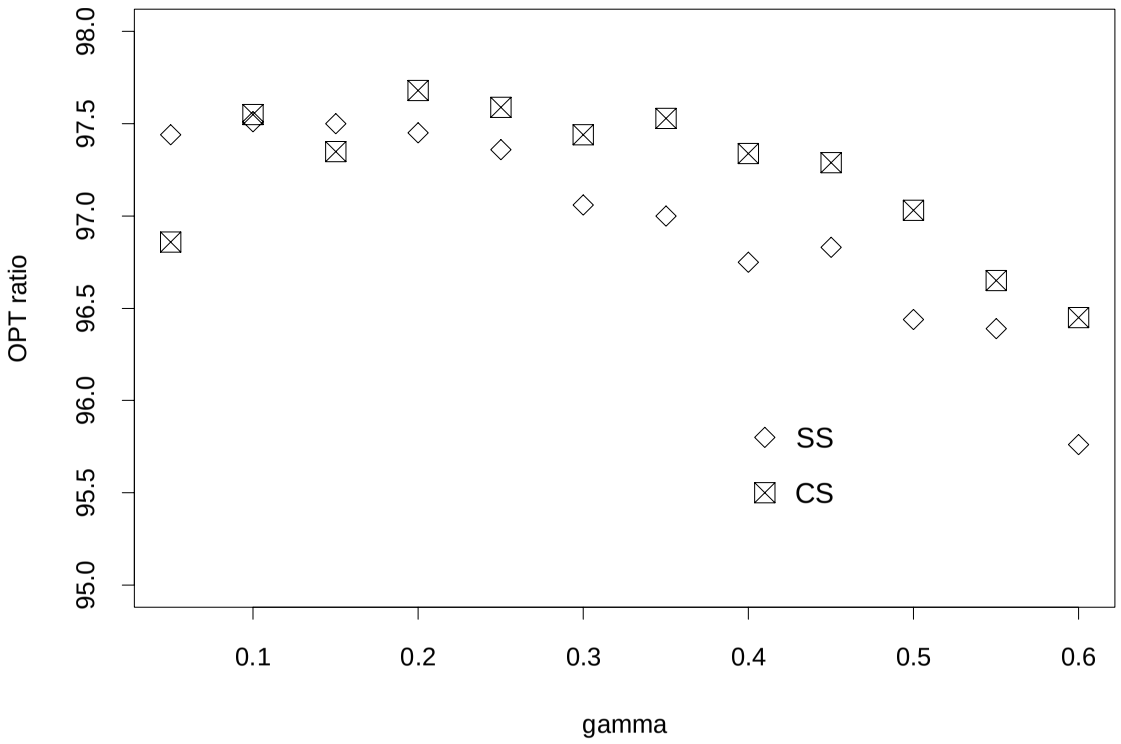

Figure 8: Percentage of optimal solution as a function of γ^n for control adjustment through coordinate search (CS) and gradient step (SS).

Method	SS+CS		CS		SS	
Data set	CPU ratio	OPT ratio	CPU ratio	OPT ratio	CPU ratio	OPT ratio
1.a	18.9	97.4	5.4	97.6	4.8	97.4
1.b	20.2	97.5	4.7	97.5	5.3	97.4
1.c	19.4	97.4	4.4	97.4	5.6	97.2
1.d	19.5	97.4	5.4	97.5	5.5	97.2
1.e	27.9	97.3	8.2	97.4	7.2	97.2

Table 3: CPU ratio and OPT ratio for data sets similar to the standard data set, obtained by running the hybrid *SS+SC* procedure (corresponding to 150 iterations), and running the SS and CS procedures for 600 iterations each.

6.3 Performance Evaluation

We now concentrate on the research questions related to the characteristics of the data sets. Basically, we have to look at how the following affect the OPT ratio, the CPU ratio, the convergence of the hybrid *SS+CS* method and the optimal range of γ^n:

1. Number of loads per time period.

2. Ratio of number of vehicles per load per day in the system.

3. Average size of the departure time window.

4. Size of the time period.

5. Size of the planning horizon.

The first three factors are determined by the application. The size of the time period and the planning horizon must be chosen by the modeler.

The linear program for the standard data set has around $28,000$ variables and $12,000$ constraints. The largest linear program amongst all the data sets is the one for data set 7. It has almost $100,000$ variables and $32,000$ constraints.

We have adapted the procedure for the standard data set the following way when the parameters of the problem vary. After running 50 iterations using SS as the control adjustment procedure, we still run 100 iterations using the CS procedure. However, the number of upper bounds adjusted per iteration using CS varies according to the number of loads in the system. For each additional 2000 loads in the system, we adjust one more upper bound per iteration when using the CS procedure. The LQN approach becomes considerably faster than the linear program solver when the number of loads increases. The solutions using the LQN approach also improve on the bigger problems.

If the number of vehicles in the network increases, the problems become less tightly constrained. This has an impact on the CPU times for both the LQN approach and the linear solver, as it can be seen in table 5. For the data sets in this table we also present the rejection rate of loads, i.e., the percentage of loads that were not assigned a vehicle at any time within the planning horizon. This is an indication of how the LQN approach would perform in practice on applications that have similar load rejection rates. The LQN approach fails to return solutions close to optimal when the number of vehicles decreases. However, very tightly constrained problems are unlikely to be found

Data set	Loads	M	LQN CPU time	CPU ratio	OPT ratio
1	2000	1	103	17.5	97.6
2	3000	2	175	24.4	97.7
3	4000	2	202	43.6	97.8
4	5000	3	244	38.4	97.8
5	6000	3	249	67.4	98.1
6	8000	4	307	83.8	98.0
7	10000	5	492	88.1	98.0

Table 4: Performance of problems with different number of loads.

Data set	Vehicles	LQN CPU time	CPU ratio	OPT ratio	Rejection
8	200	103	41.0	92.5	28.8%
9	300	122	35.8	95.7	17.9%
1	400	103	17.5	97.6	12.2%
10	500	101	8.6	98.1	10.0%
11	600	96	12.1	98.0	9.6%

Table 5: Performance of problems with different number of vehicles and the percentage of loads rejected in each one.

in practice.

The CPU time to get to the optimal solution using a linear solver heavily depends on the average size of the departure time window. The results in table 6 indicate that an increase in the time window size does not harm the quality of the solution obtained using the LQN approach. Data set 18 is a special case because it reduces to a pure network problem, as each load has only one possible departure time. Further investigation into the behavior of the OPT ratio on problems with zero width time windows was performed. It has indicated that the strategy we selected in section 6.2 using the standard data set is not the best LQN strategy to find a good solution for these problems. However, network algorithms can be used to find the optimal solution for this class of problems much faster than our suboptimal strategy.

Choosing an appropriate time period is usually a compromise between having the time coordinate discrete enough to make the model useful and keeping the problem within a reasonable size. Notice that by changing the size of the time period, the optimal solution of the linear program also changes. Too coarse a time discretization may not provide usable decisions. Halving the size of the time period implies having twice as many time periods in the problem. For a linear solver, that is

Data set	Time window (h)	LQN CPU	CPU ratio	OPT ratio
18	0	74	1.2	96.5
19	20	107	17.1	97.6
20	40	128	25.3	97.5
21	60	142	22.5	98.1

Table 6: Performance of problems with loads with different time windows.

Data set	Time period (h)	LQN CPU time	CPU ratio	OPT ratio
12	2	168	85.5	95.8
1	4	103	17.5	97.6
13	6	85	6.3	97.5
14	8	71	4.0	97.9

Table 7: Performance of problems with different time period sizes.

likely to increase the computation times substantially. For the LQN approach, there are twice as many local problems to solve and twice as many gradients to compute, and therefore computation time is likely to double, at the most. But as it can be seen in table 7, the CPU time increases at a sublinear rate, because each local problem becomes simpler to solve as the same number of loads dilutes across a larger number of local problems. This table also indicates that smaller time periods tend to worsen the solution quality.

The LQN algorithm is sensitive to the choice of the planning horizon. A poorly appreciated fact is that optimal solutions for time-staged problems grow dramatically with the length of the planning horizon. The speed of the LQN algorithm relative to the optimal solution increases dramatically as the planning horizon increases, as is shown in table 8. At the same time, the solution quality produced by the LQN algorithm degrades relative to the optimal solution. We suspect that in a problem with stochastic data (reflecting forecasting uncertainties) that the LQN methodology will produce better solutions relative to the deterministic optimal solution. However, on a deterministic problem with deterministic data, as we have in this paper, an optimal solution can "see" into the future more accurately than our linear approximation of the value function.

Data set	Planning horizon (h)	LQN CPU time	CPU ratio	OPT ratio
15	60	50	2.0	97.5
1	120	103	17.5	97.6
16	240	301	98.4	95.5
17	360	363	231.0	93.4

Table 8: Performance of problems with different planning horizons.

7 Discussion

We have produced a strategy that generates integer solutions that are within 2.5% of optimality for dynamic fleet management problems typically found in practice. Furthermore, our strategy is able to generate these solutions in a fraction of the time a linear programming solver takes to find the optimal solution of linear relaxation of the problem.

Besides the fact that it generates integer solutions, there are clear advantages to the LQN approach. It allows for considering several real-world details that cannot be modeled into a linear program, such as labor regulations and load priorities. These constraints can be taken into account when solving the local problem, adding little overhead to solving the problems with the simplifying assumptions we have used throughout this paper.

It is possible to use the LQN approach to add a component of global control to local decisions. The actual decisions can be implemented exactly the way the forward pass has recommended or the gradients computed in the backward pass can be used to implement decisions slightly differently. Transportation networks are often set as decentralized decision environments and there exist constraints only known by the actual decision maker at each terminal. The LQN approach mimics a decentralized decision environment.

The LQN approach has not performed very well on problems that reduce to a pure network. These problems can be solved with specialized algorithms at a fraction of time our gradient procedure would take. However, these problems tend to appear in fleet management only when assumptions have over-simplified the real problem.

Rather than defining values for the smoothing factor γ^n and setting a definite answer on what is the best strategy for all ranges of problems, Section 6 must be regarded as a guide to calibrating

the methodology for those applying it to a specific environment. The resulting strategy may vary according to several parameters of the problem, such as the ratio between profit on a load and cost of moving empty. It also may vary depending on how imbalanced or dense the network is.

Further research must concentrate on improving the quality of the solutions obtained using the LQN approach. The performance of the technique must also be evaluated on a rolling horizon basis in order to validate its implementation in a real time environment. This approach can be adapted to solve other dynamic resource problems like for example machine scheduling and air traffic control.

References

Dantzig, G. & Fulkerson, D. (1954), 'Minimizing the number of tankers to meet a fixed schedule', *Naval Research Logistics Quarterly* **1**, 217–222.

Ermoliev, Y. (1988), Stochastic quasigradient methods, *in* Y. Ermoliev & R. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag.

Frantzeskakis, L. & Powell, W. (1990), 'A successive linear approximation procedure for stochastic dynamic vehicle allocation problems', *Transportation Science* **24**(1), 40–57.

Gupal, A. M. & Bazhenov, L. G. (1972), 'A stochastic method of linearization', *Cybernetics* pp. 482–484.

Hane, C., Barnhart, C., Johnson, E., Marsten, R., Nemhauser, G. & Sigismondi, G. (1994), A fleet assignment problem: Solving a large-scale integer program, Technical report, Georgia Institute of Technology, School of Industrial and Systems Engineering. Report Series 92-04.

Jordan, W. & Turnquist, M. (1983), 'A stochastic dynamic network model for railroad car distribution', *Transportation Science* **17**, 123–145.

Magnanti, T. & Simpson, R. (1978), Transportation network analysis and decomposition methods, Report no. dot-tsc-rspd-78-6, U.S. Department of Transportation.

Powell, W. (1988), A comparative review of alternative algorithms for the dynamic vehicle allocation problem, *in* B. Golden & A. Assad, eds, 'Vehicle Routing: Methods and Studies', North Holland, New York, pp. 249–292.

Powell, W. (1989), 'A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy', *Transportation Science* **23**(4), 231–243.

Powell, W. (1996), 'A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers', *Transportation Science* **30**(3), 195–219.

Powell, W. & Cheung, R. (1994), 'A network recourse decomposition method for dynamic networks with random arc capacities', *Networks* **24**, 369–384.

Powell, W. B., Jaillet, P. & Odoni, A. (1995*a*), 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.

Powell, W., Carvalho, T., Godfrey, G. & Simao, H. (1995*b*), 'Dynamic fleet management as a logistics queueing network', *Annals of Operations Research* **61**, 165–188.

White, W. (1972), 'Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers', *Networks* **2**(3), 211–236.

White, W. & Bomberault, A. (1969), 'A network algorithm for empty freight car allocation', *IBM Systems Journal* **8**(2), 147–171.

Appendix A Computing the gradients

We develop the gradients in two stages. First, we compute the task gradients, represented by μ_t. These gradients are then used in section A.2 to compute the supply gradients ν_t. Finally, section A.3 develops the gradients needed for upper bound adjustment.

A.1 Task Gradients

In order to represent the fact that a load l is satisfied or not at a given time t we define:

$$Q_{lt} = 1 - \sum_{t'<t} x_{lt'}$$

While the choice might not seem obvious, this results in $Q_{lt} = 0$ when the departure time for load l precedes t, and $Q_{lt} = 1$ otherwise. Thus, $Q_{lt} = 1$ can be read as "load l is in the queue at time t".

As the left derivative of G_t with respect to Q shows in the derivation of ν, we also define:

$$\mu_{lt'} = \frac{\partial G_t(x, y)}{\partial Q^-_{lt'}} \tag{41}$$

which can be easily read as "the change in G_t resulting from from removing load l from the queue at time t". These gradients must be computed because varying the supply of vehicles available at a node perturbs the queue of loads. They are computed as a function of the vector ν.

Let x, y and w represent the current solution from which the gradients are to be computed. Two cases arise, depending on whether load l is assigned a vehicle or not in the current solution. Let $\mu^1_{lt'}$ be the gradient computation for case 1 and $\mu^2_{lt'}$ for case 2.

Case 1: If for all $t \geq t'$, $x_{lt} = 0$, it follows from the definition of Q_{lt} that

$$\mu^1_{l,t'} = \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial Q^-_{lt'}} = 0 \tag{42}$$

In other words, removing the load to the set of tasks at time t' has no impact on the system, because load l is never served.

Case 2: There exists some time $t'' \geq t'$ such that $x_{lt''} = 1$. The removal of load l from the queue results in this variable dropping to *zero*. Let i be the origin and j the destination terminal for load l. Then:

$$\frac{\partial x_{lt''}}{\partial Q^-_{lt'}} = 1$$

and:

$$\begin{aligned} \mu^2_{l,t'} = \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial Q^-_{lt'}} &= \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial x_{lt''}} \frac{\partial x_{lt''}}{\partial Q^-_{lt'}} \\ &= \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial x^-_{lt''}} \end{aligned} \tag{43}$$

The decrease in $x_{l,t''}$ affects the current solution only at time period t'' or at later time periods. Therefore:

$$\mu^2_{lt'} = \frac{\partial G_{t''}(V_{t''}, \mathcal{L}_{t''})}{\partial x^-_{lt''}}$$

Based on arguments presented in section 3, we can develop the following expression:

$$\frac{\partial G_{t''}(V_{t''}, \mathcal{L}_{t''})}{\partial x^-_{lt''}} = r_{lt''} - \nu^+_{it''} + \nu^-_{j,t''+1}$$

Section 4 illustrated that the expression is neither an upper or lower bound on the actual gradient. We propose, however, to use it as an approximation.

Finally, we arrive at:

$$\mu_{l,t'} \begin{cases} = 0 & \text{if } x_{lt} = 0 \ \ \forall t \geq t' \\ \simeq r_{l,t''} - \nu^+_{i,t''} + \nu^-_{j,t''+1} & \text{if } \exists\ t'' \geq t' \text{ such that } x_{lt''} = 1 \end{cases} \tag{44}$$

where i and j represent the origin and destination terminals for load l.

A.2 Supply Gradients

We can now derive the equations for ν. By definition,

$$\begin{aligned}
\nu_{it'} &= \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial V_{it'}} \\
&= \sum_{k \in \mathcal{C}} \frac{\partial g_{k,t'}}{\partial V_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V_{it'}} \\
&= \frac{\partial g_{it'}}{\partial V_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V_{it'}} \qquad (45)
\end{aligned}$$

V_{it} varies in integer units and $G_t(V_{it}, \mathcal{L})$ is a piecewise linear function. Therefore we can compute the right and left directional derivatives of G_t by using a finite difference. We begin by first computing the right directional derivative, $\nu^+_{it'}$.

Let us look at the local problem at node (i, t') when $V_{it'}$ is increased. We can write equation (3) considering $\tilde{y}_{iit'}$ as the slack variable and then find its derivative with respect to $V_{it'}$:

$$\sum_{l \in \mathcal{L}_{it'}} \frac{\partial x_{lt'}}{\partial V^+_{it'}} + \sum_{j \in \mathcal{C}} \frac{\partial y_{i,j,t'}}{\partial V^+_{it'}} + \frac{\partial \tilde{y}_{iit'}}{\partial V^+_{it'}} = 1 \qquad (46)$$

These partial derivatives can be computed by:

$$\frac{\partial x_{lt'}}{\partial V^+_{it'}} = X^+_{lt'} = x_{lt'}(V_{it'}+1, \nu_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - x_{lt'}(V_{it'}, \nu_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \qquad (47)$$

$$\frac{\partial y_{i,j,t'}}{\partial V^+_{it'}} = Y^+_{i,j',t'} = y_{i,j,t'}(V_{it'}+1, \nu_{t'+1}, u_{it'}, \mathcal{L}_{it'}) - y_{i,j,t'}(V_{it'}, \nu_{t'+1}, u_{it'}, \mathcal{L}_{it'}) \qquad (48)$$

$$\frac{\partial \tilde{y}_{iit'}}{\partial V^+_{it'}} = Z^+_{iit'} = 1 - \sum_{l \in \mathcal{L}_{it'}} X^+_{lt'} - \sum_{j \in \mathcal{C}} Y^+_{i,j,t'} \qquad (49)$$

Due to the non-negativity and integer constraints, only one of the derivatives in (46) can be positive. Four cases arise. For each case n we present a computation for the right directional derivative $\nu^{n+}_{it'}$. Depending on which partial derivative in (46) is positive, one of the following cases apply.

Case 1: There exists a terminal $j' \in \mathcal{C}$ such that

$$Y^+_{i,j',t'} = 1 \qquad (50)$$

It follows from equation (22) applied to node $(j', t'+1)$ and differentiated with respect to $V_{it'}$ that:

$$\begin{aligned}\frac{\partial V_{j',t'+1}}{\partial V^+_{it'}} &= \frac{\partial V_{j',t'}}{\partial V^+_{it'}} - \sum_k \frac{\partial w_{j,k,t'}}{\partial V^+_{it'}} + \sum_i \frac{\partial w_{i,j,t'}}{\partial V^+_{it'}} \\ &= \frac{\partial w_{i,j',t'}}{\partial V^+_{it'}} = \frac{\partial y_{i,j',t'}}{\partial V^+_{it'}} = Y^+_{i,j',t'} = 1\end{aligned}$$

We can then evaluate $\nu^{1+}_{it'}$ for this case. From equation (45):

$$\begin{aligned}\nu^{1+}_{it'} &= \sum_{l \in \mathcal{L}_{it'}} r_{lt'} \frac{\partial x_{lt'}}{\partial V^+_{it'}} - \sum_{j \in \mathcal{C}} c_{ij} \frac{\partial y_{i,j,t'}}{\partial V^+_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V^+_{it'}} \\ &= -c_{i,j'} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V_{j',t'+1}} \frac{\partial V_{j',t'+1}}{\partial V^+_{it'}} \\ &= -c_{i,j'} + \nu^+_{j',t'+1} \qquad (51)\end{aligned}$$

where $j' \in \mathcal{C}$ is such that it satisfies condition (50).

Case 2: There exists a load $l' \in \mathcal{L}_{it'}$ such that

$$X^+_{l',t'} = 1 \qquad (52)$$

and in the current solution load l' was never assigned a vehicle:

$$x_{l',t} = 0 \quad \forall\, t > t'$$

It follows from equation (42) that

$$\frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial Q^-_{l',t'+1}} = 0 \qquad (53)$$

Let j' be the destination of load l'. As in case 1, it follows from equation (22) applied to node $(j', t'+1)$ and differentiated with respect to $V_{it'}$ that:

$$\frac{\partial V_{j',t'+1}}{\partial V^+_{it'}} = \frac{\partial w_{i,j',t'}}{\partial V^+_{it'}} = \frac{\partial x_{l',t'}}{\partial V^+_{it'}} = 1 \qquad (54)$$

We can then evaluate $\nu^{2+}_{it'}$ for this case. From equation (45):

$$\begin{aligned}\nu^{2+}_{it'} &= \sum_{l\in\mathcal{L}_{it'}} r_{lt'}\frac{\partial x_{lt'}}{\partial V^+_{it'}} - \sum_{j\in\mathcal{C}} c_{ij}\frac{\partial y_{i,j,t'}}{\partial V^+_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1},\mathcal{L}_{t'+1})}{\partial V^+_{it'}} \\ &= r_{l',t'} + \frac{\partial G_{t'+1}(V_{t'+1},\mathcal{L}_{t'+1})}{\partial V^+_{it'}} \end{aligned} \tag{55}$$

In equation (55), it can be seen that the immediate consequence of having an additional vehicle at node (i,t') in this case is to collect the added contribution from load l'. This results in two downstream effects: there is an increase in the supply of vehicles at the destination node of load l', and l' is removed from the queue of loads at node (i,t'). Therefore:

$$\nu^{2+}_{it'} = r_{l',t'} + \frac{\partial G_{t'+1}(V_{t'+1},\mathcal{L}_{t'+1})}{\partial V_{j',t'+1}}\,\frac{\partial V_{j',t'+1}}{\partial V^+_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1},\mathcal{L}_{t'+1})}{\partial Q_{l',t'+1}}\frac{\partial Q_{l',t'+1}}{\partial V^+_{it'}}$$

By using the definition of Q, it can be shown that

$$\frac{\partial Q_{l',t'+1}}{\partial V^+_{it'}} = -1 \tag{56}$$

Using equations (53) , (54) and (56),

$$\nu^{2+}_{it'} = r_{l',t'} + \nu^+_{j',t'+1}$$

where j' represents the destination terminal for load l' and $l' \in \mathcal{L}_{it'}$ is such that it satisfies condition (52).

Case 3: There exists a load $l' \in \mathcal{L}_{it'}$ such that

$$X^+_{l',t'} = 1 \tag{57}$$

and there exists $t'' > t'$ such that:

$$x_{l',t''} = 1 \tag{58}$$

Let j' be the destination of load l'. It follows from equation (22) applied to node $(j',t'+1)$ and differentiated with respect to $V_{it'}$ that:

$$\frac{\partial V_{j',t'+1}}{\partial V^+_{it'}} = 1 \tag{59}$$

From equation (44) we find:

$$\mu_{l',t'+1} \simeq r_{l',t''} - \nu^{+}_{i,t''} + \nu^{-}_{j',t''+1} \tag{60}$$

From the definition of Q_{lt},

$$\frac{\partial Q_{l',t'+1}}{\partial V^{+}_{it'}} = -1 \tag{61}$$

Thus an increase in $V_{it'}$ has two effects on the state of the system at time $t'+1$. It increases $V_{j',t'+1}$ by one unit and decreases $Q_{l',t'+1}$ also by one unit. We can then evaluate $\nu^{3+}_{it'}$ for this case:

$$\begin{aligned}
\nu^{3+}_{it'} &= \sum_{l \in \mathcal{L}_{it'}} r_{lt'} \frac{\partial x_{lt'}}{\partial V^{+}_{it'}} - \sum_{j \in \mathcal{C}} c_{ij} \frac{\partial y_{i,j,t'}}{\partial V^{+}_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V^{+}_{it'}} \\
&= r_{l',t'} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V^{+}_{it'}} \\
&= r_{l',t'} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V_{j',t'+1}} \frac{\partial V_{j',t'+1}}{\partial V^{+}_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial Q_{l',t'+1}} \frac{\partial Q_{l',t'+1}}{\partial V^{+}_{it'}}
\end{aligned}$$

Using (59) and (61):

$$\begin{aligned}
\nu^{3+}_{it'} &= r_{l',t'} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V^{+}_{j',t'+1}} - \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial Q^{-}_{l',t'+1}} \\
&= r_{l',t'} + \nu^{+}_{j',t'+1} - \mu_{l',t'+1} \\
&\simeq r_{l',t'} - r_{l',t''} + \nu^{+}_{j',t'+1} + \nu^{+}_{i,t''} - \nu^{-}_{j',t''+1}
\end{aligned}$$

where $j' \in \mathcal{C}$ is the destination of load $l' \in \mathcal{L}_{it'}$ and l' satisfies condition (57) and t'' satisfies condition (58).

We remind the reader that the result obtained for $\mu_{l',t'+1}$ in the previous section is an approximation. Therefore, the result we present for $\nu^{3+}_{it'}$ is also an approximation.

Case 4:

$$Z^{+}_{iit'} = 1$$

Thus, the additional vehicle is carried over in inventory to node $(i, t'+1)$, as it is clear from

finding the derivative of equation (22) with respect to $V_{it'}$.

$$\frac{\partial V_{i,t'+1}}{\partial V^+_{it'}} = 1$$

We can then evaluate $\nu^{4+}_{it'}$ for this case:

$$\begin{aligned}
\nu^{4+}_{it'} &= \sum_{l \in \mathcal{L}_{it'}} r_{lt'} \frac{\partial x_{lt'}}{\partial V^+_{it'}} - \sum_{j \in \mathcal{C}} c_{ij} \frac{\partial y_{i,j,t'}}{\partial V^+_{it'}} + \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V^+_{it'}} \\
&= \frac{\partial G_{t'+1}(V_{t'+1}, \mathcal{L}_{t'+1})}{\partial V_{i,t'+1}} \frac{\partial V_{i,t'+1}}{\partial V^+_{it'}} \\
&= \nu^+_{i,t'+1}
\end{aligned}$$

The equations for ν^- parallel those of ν^+ and are not presented.

A.3 Upper Bound Gradients

In order to update the control vector we must compute the gradients

$$\eta^+_{ijt'} = \frac{\partial G_t(x,y)}{\partial u^+_{i,j,t'}} = \frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial y_{i,j,t'}} \frac{\partial y_{i,j,t'}}{\partial u^+_{i,j,t'}}$$

Therefore it follows that

$$\eta^+_{ijt'} = \begin{cases} \frac{\partial G_{t'}}{\partial y^+_{i,j,t'}} & \text{if } \frac{\partial y_{i,j,t'}}{\partial u^+_{i,j,t'}} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{62}$$

Our problem is now to compute the right gradient of $G_{t'}$ with respect to $y_{i,j,t'}$. We use the following approximation for this gradient:

$$\frac{\partial G_{t'}(V_{t'}, \mathcal{L}_{t'})}{\partial y^+_{i,j,t'}} \simeq -c_{ij} - \nu^-_{it'} + \nu^+_{j,t'+1}$$

We have to find the condition to be satisfied for

$$\frac{\partial y_{i,j,t'}}{\partial u^+_{i,j,t'}} = 1$$

The upper bound increase in link (i,j,t') indeed corresponds to an additional empty move if the possible empty move is assigned a vehicle. In order for this to happen, the empty move has to be more valuable than the task that would have capacity diverted to it at node (i,t'):

$$-c_{ij} + \nu^{+}_{j,t'+1} > \nu^{-}_{it'}$$

It follows that

$$\eta^{+}_{ijt'} \begin{cases} \simeq -c_{ij} + \nu^{+}_{j,t'+1} - \nu^{-}_{it'} & \text{if } -c_{ij} + \nu^{+}_{j,t'+1} - \nu^{-}_{it'} > 0 \\ = 0 & \text{otherwise} \end{cases} \tag{63}$$

The derivation for $\eta^{-}_{ijt'}$ is similar and leads to

$$\eta^{-}_{ijt'} \begin{cases} \simeq c_{ij} - \nu^{-}_{j,t'+1} + \nu^{+}_{it'} & \text{if } c_{ij} - \nu^{-}_{j,t'+1} + \nu^{+}_{it'} > 0 \\ = 0 & \text{otherwise} \end{cases} \tag{64}$$