

Dynamic Fleet Management as a Logistics Queueing Network

Warren B. Powell
Tassio A. Carvalho
Gregory A. Godfrey
Hugo P. Simao

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

Statistics and Operations Research
Technical Report SOR-94-18

June 20, 1995

Abstract

This paper introduces a new framework for modeling and solving dynamic fleet management problems, which we call the Logistics Queueing Network (LQN). A variety of problems in logistics involve the combined problem of moving freight from origin to destination while simultaneously managing the capacity required to move this freight. Standard formulations for real-world problems usually lead to intractably large linear programs. The LQN approach can take into account more real world detail and is considerably faster than classical LP formulations. The solutions generated using the LQN approach are shown to be within a few percentage points of the LP optimal solutions depending on the size of the capacity fleets.

We consider the problem of managing a fleet of vehicles over time to service a known set of loads, each of which must be serviced within a specified time window (that can be fairly wide). These problems have been modeled in the past as linear programs with GUB constraints, or as set partitioning problems solved with column generation methods. Our interest is in solving problems with thousands, or even hundreds of thousands, of vehicles. As a rule, these problems result in intractably large linear programs which do little to take advantage of the dynamic structure of the problem.

A new approach is presented here based on a discrete event dynamic system (DEDS), where demands are queued at terminals while waiting for available capacity. We call it a logistics queueing network (LQN). In contrast with classical queueing networks, which track the movement of customers through a series of servers, an LQN is characterized by the management of containers that handle moving demands from one location to the next. When a container arrives at its destination, it becomes empty and available for its next movement. Since the flow of demands into a location might not equal the flow out of it, it is necessary to reposition containers empty from one location to another.

Companies that move freight have some degree of flexibility as to when a demand is shipped. Demands arise in a process very much like a Poisson process. In addition to utilizing resources optimally, there is a need to anticipate when and where demands will appear so that available equipment can be better positioned to satisfy customer needs. Each demand can be described by a list of attributes. The following attributes precisely describe a demand for our purposes in this paper.

- *Call-in time* is the time the transportation company is informed about the existence of a shipment to be moved from one terminal to another.
- *Earliest departure time* is the time the shipment is available at the origin terminal for pickup.
- *Latest departure time* is the latest time the demand can leave the origin and get to the destination by the due time. The *start time window* is the time

interval between the earliest and latest departure times. The movement can start at any time within this time window.

- *Due time* is the time by which the shipment must have been delivered at the destination terminal.
- *Revenue* is the return gained by transporting a shipment. If the demand cannot be satisfied within the time window, then the demand is considered lost and no revenue is received.

Capacity is the term that describes the equipment or collection of resources enabling a movement between two terminals. Capacity must be assigned to satisfy demands. We assume throughout this paper that there is no consolidation - two units of demand cannot share the same unit of capacity at the same time.

At any moment there might be shipments available to be transported. Other shipments that have been called in will become available in the future. Using historical data, it is possible to forecast shipments that will be called in later. Therefore, as we plan to move available shipments, we must also anticipate moving shipments that will become available in the future.

In addition to satisfying customer demands, profits must also be maximized. Taking into account future events, a set of decisions assigning available capacity to available demands must be made for the near future. Capacity can be moved empty from one terminal to another in order to satisfy demands which are on hold. Capacity must be relocated in anticipation of demands that have not yet appeared. However, there is a cost associated with relocating capacity. Therefore, the benefit from satisfying demands that would be lost otherwise must surpass the cost of moving capacity empty.

Fleet management problems are usually depicted using a dynamic network such as that shown in figure 1. Links represent the possibility of flowing capacity across time, either as inventory or as a move from one terminal to another. Only some of the possible links are represented in the figure. The length of the planning horizon, T , must be chosen

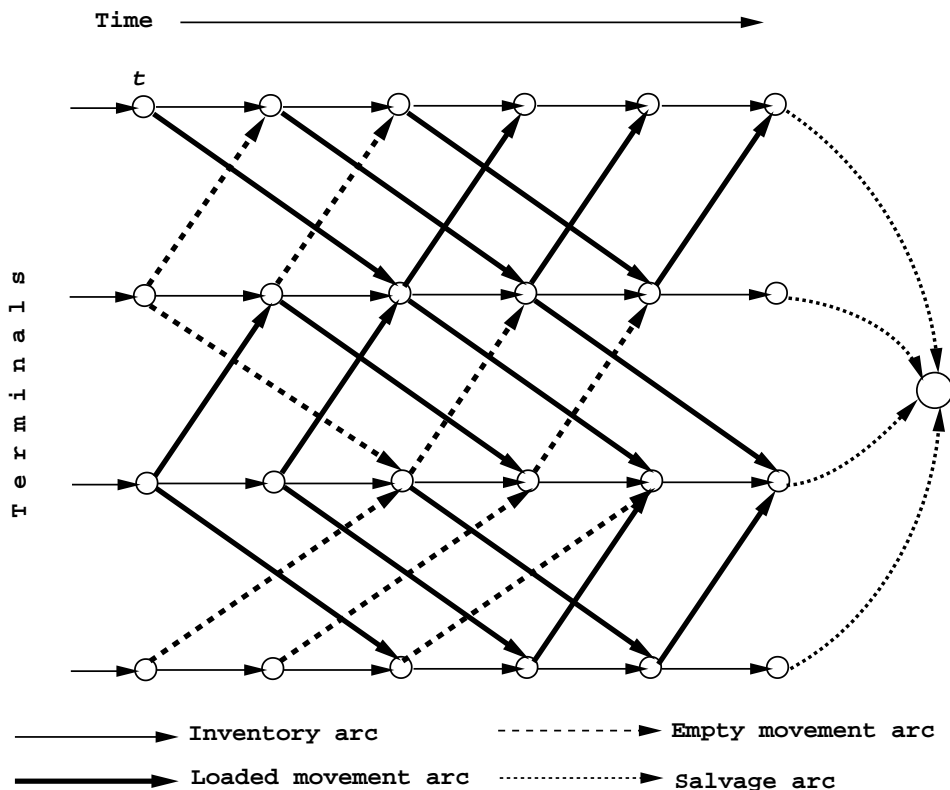


Figure 1: Illustration of dynamic network

long enough to ensure that the downstream effects of decisions made now are properly accounted for. In many short-haul routing problems, a time period of one day is typical. However, truckload trucking might require a horizon of at least six to 10 days, rail might require 10 to 20 days, while some problems in ocean shipping might need as long as 10 to 12 weeks.

This problem is most commonly formulated as a linear program over a dynamic network, where time windows can be handled using GUB constraints (see, for example, Magnanti and Simpson [10]). Multiple equipment types can be handled using a multicommodity flow formulation, introducing a different set of GUB constraints (see, for example, Jones *et al.* [8] and Powell *et al.* [11]). Our problem is also equivalent to

classical crew scheduling problems, that are often formulated as large set partitioning problems (see, for example, Desrosiers *et al.* [3] and [4]). These formulations, however, are not well suited to fleet management problems with wide time windows, and suffer from a high level of degeneracy that arises due to the lack of the complex work rules and tight time windows that characterize most crew scheduling problems. It is also difficult to obtain integer solutions from multicommodity formulations without the use of rounding heuristics (see, for example, Chih [2]). Set partitioning formulations provide integer solutions more readily, but the column generation step is very slow unless the problem is tightly constrained.

A separate line of research has focused on the classical dynamic vehicle allocation problem. This literature normally assumes that loads must be moved at a specific point in time. The problem was originally formulated by White [14] as a pure network. Jordan and Turnquist [9] presented the first formulation as a stochastic problem (also allowing demands to queue forward in time). Powell [12] presents a series of alternative models, including a formulation as a dynamic network with random arc capacities. Frantzeskakis and Powell [6] and Cheung and Powell [1] give specialized algorithms for this class of problems. The drawback of this research, however, is that it is designed for fleet management problems with single vehicle types, and no time windows. While these issues can be handled in an approximate way using rolling horizon methods (see, for example, Powell [13]), they cannot be captured accurately when forecasting into the future.

The primary contribution of this paper is to introduce and develop the concept that large fleet management problems, which are most frequently modeled as large linear programs, can be formulated and solved by optimizing the control of a simulation. Effectively, we are replacing one large linear program with a series of small network problems which are solved iteratively. This paper serves only to introduce the idea and provide an initial indication of promise. We have intentionally not considered many possible features of the methodology, such as its ability to handle multiple equipment types, forecasting uncertainties, and congestion at terminals. For comparison purposes,

we have restricted our attention to a problem that can be formulated relatively easily as a linear program, which then serves as a basis for comparison.

We begin in section 1 by presenting a mixed integer global optimization formulation for the problem. The LP model forces us to consider only highly simplified versions of the problem, but it also provides a basis for evaluating the quality of our algorithm, which is presented in section 2. Section 3 presents the results of experiments comparing the LQN to the linear programming formulation, demonstrating the promise of the new technique.

1 A global optimization formulation

Linear programming models for real world problems usually require simplifying assumptions to keep the problem size compatible with available technology. The problem of managing capacity has an underlying network structure. To formulate this problem compactly, we make the following assumptions:

- There is only one type of capacity. Any unit of capacity is compatible with any demand.
- There is no consolidation. Only one unit of demand can be transported by each unit of capacity at a time.
- Shipments from one terminal to another are transported directly. Intermediate stops are not possible.
- Travel times are equal to one time period. Even though this assumption does not reduce the complexity of the problem, it provides us with simpler equations.

This problem can be modeled as a network with side constraints. We define the following sets:

- \mathcal{C} is the set of terminals i in the network.
- \mathcal{D} is the set of demands d available within the planning horizon, T .
- \mathcal{T}_d is the set of feasible departure times for satisfying demand $d \in \mathcal{D}$, otherwise known as the *start time window*.
- \mathcal{D}_{it} is the set of demands d with origin i , which are available to move at time t but have not been moved at a time prior to time t during a particular simulation.
- \mathcal{D}_{ijt} is the set of demands $d \in \mathcal{D}$ with origin i and destination j having t as a feasible departure time.
- \mathcal{D}_{it}^0 is the set of demands d with origin i , where t is the beginning of the time window \mathcal{T}_d .
- \mathcal{D}_{it}^f is the set of demands d with origin i , where t is the end of the time window \mathcal{T}_d .
- \mathcal{N} is the set of nodes $(i, t), i \in \mathcal{C}, t \leq T$, in the dynamic network.

The decision variables are:

- $x_{d,t} = 1$ if demand d is served at time t . Figure 2 illustrates the time window for a given demand.
- $z_d = 1$ if demand d is never served (within the time window).
- $y_{i,j,t}$ is the amount of capacity being repositioned empty along link (i, j, t) . If $i = j$, $y_{i,i,t}$ represents the amount of capacity in inventory at terminal i from time t to time $t + 1$.
- $w_{i,j,t}$ is the total flow of capacity on the dynamic link (i, j, t) .

And we also define the following coefficients:

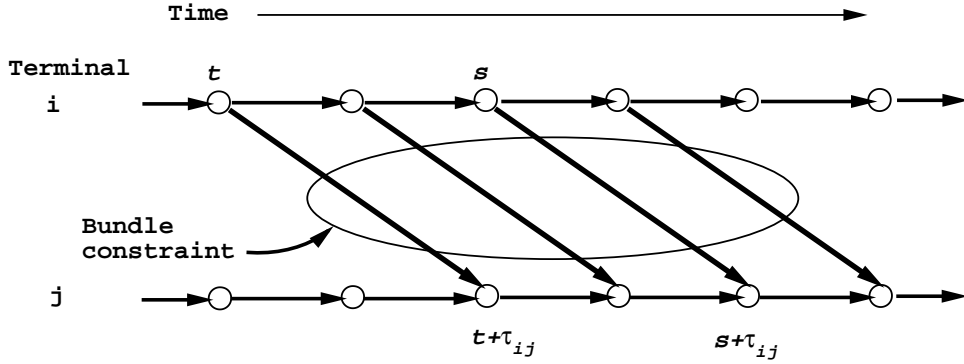


Figure 2: Feasible departure times for a demand with $\tau_{i,j}$ representing the travel time between i and j .

- R_{it} is the net inflow ($R_{it} > 0$) or outflow ($R_{it} < 0$) of capacity (vehicles) at city i at time t .
- $r_{d,t}$ is the revenue generated by choosing time t to satisfy demand d , i.e., the cost associated with $x_{d,t}$. Generally, $r_{d,t} = r_{d,t'}$ for $t, t' \in \mathcal{T}_d$, but we will allow the more general formulation.
- $c_{i,j}$ is the cost of repositioning capacity over link (i, j, t) , i.e., the cost associated with $y_{i,j,t}$. We make the assumption that there is no cost associated with keeping capacity in inventory, i.e., $c_{i,i} = 0$.

The amount of flow entering or leaving the network through each node (i, t) must also be specified. Typically, capacity enters the network at several terminals in the first time period. Capacity moving between two terminals during the first time period can be represented as flow into its destination at the time it arrives there. Capacity may also be added to or subtracted from specific nodes after the first time period for reasons such as maintenance.

This problem can be formulated as:

$$\max_{x,y} \sum_{t=0}^T \left(\sum_{d \in \mathcal{D}} r_{d,t} x_{d,t} - \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} c_{i,j} y_{i,j,t} \right) \quad (1)$$

subject to:

$$\sum_{t \in \mathcal{T}_d} x_{d,t} + z_d = 1 \quad \forall d \in \mathcal{D} \quad (2)$$

$$\sum_{d \in \mathcal{D}_{i,j,t}} x_{d,t} + y_{i,j,t} - w_{i,j,t} = 0 \quad \forall i, j \in \mathcal{C}, \forall t \leq T \quad (3)$$

$$\sum_{j \in \mathcal{C}} w_{i,j,t} - \sum_{j \in \mathcal{C}} w_{j,i,t-1} = R_{i,t} \quad \forall i \in \mathcal{N} \quad (4)$$

$$y_{i,j,t}, w_{i,j,t} \geq 0 \quad (5)$$

$$x_{d,t} = (0, 1) \quad (6)$$

Constraints (2) limit the amount of capacity used to satisfy each unit of demand. Constraints (3) are definitional constraints for the total flow w . Flow conservation at each node is enforced by (4). Equation (6) imposes integrality on the load assignment variables.

Without the GUB constraints in equation (2), this problem would be a pure network. With the constraints, we have a large, mixed integer program. Dropping the integrality constraint produces a linear program that is at least solvable, and provides a bound on the original problem (indeed, we expect this bound is fairly tight). We use this formulation as a basis of comparison to evaluate our LQN algorithm.

There are a number of limitations to this standard approach for modeling fleet management problems. First, the handling of time windows is clumsy and unnatural. Near integer solutions are unlikely, and a high level of degeneracy suggests that the dual variables will not be very useful for branching rules to determine departure times. Second, the model is difficult to extend to more realistic, and complex, operational settings which involve multiple equipment types, customer priorities, and terminal capacities. Finally, it is difficult to see how this model can be extended to capture the many sources of uncertainty that arise in these applications.

2 The Logistics Queueing Network Approach

In this section, we propose an entirely new approach to managing fleets of vehicles that can most easily be described as optimal control of a network of queues. In this view, each terminal is a double ended queue, comprised of a queue of customers (loads waiting to be served) and a queue of servers (units of capacity waiting to serve a customer). The problem is analogous to taxi stations where taxi cabs line up waiting for customers, and customers line up waiting for taxis. Since there does not appear to be a standard name for this in the literature, we refer to the system as a logistics queueing network.

The primary difference between our solution approach, and the linear programming formulation described above, is that we do not attempt to solve the entire problem globally. Instead, we propose to solve a sequence of small subproblems. Each subproblem consists of assigning units of capacity to waiting customers, at a single terminal, at a given point in time. This strategy closely approximates many applications, such as freight car management, container management, as well as our taxi example.

The challenge of this decomposition is making the local decision making smart enough to match a global optimization. We achieve this through two mechanisms. First, since the decision to move capacity depends on the value of additional capacity at each location, we calculate a marginal value of additional capacity, at each location, at each point in time. Call this value ν_{it} . If we are sitting at terminal i and we are trying to decide if we should send a unit of capacity to terminal j , arriving at time t' , then we would consider the cost of the move, plus the value $\nu_{j,t'}$ of the capacity at the destination.

Of course, another terminal k , considering a move to terminal j , will see the same value $\nu_{j,t'}$. If this value is too high, we run the risk of flooding the location with capacity. If the value is too low, we may starve it. Finding the precise value can be very tricky. So we propose a second mechanism, which is a control variable u_{ijt} which is a limit on the amount of capacity we can send from i to j at time t .

The challenge of our LQN algorithm is to determine the best values of $\nu_{i,t}$ and

u_{ijt} so that local decision making produces a set of flows that approximate a globally optimal solution. We propose that this approach will be much better able to handle time windows and will naturally produce integer solutions. Most importantly, the ease with which the local optimization problems can be solved will allow us, in real applications, to introduce a number of other issues and constraints that are difficult to incorporate in a global optimization model.

2.1 Basic Equations

The LQN method is best illustrated by starting with the global formulation (1) and restating it recursively. Let:

$$f_t(x_t, y_t) = r_t x_t - c_t y_t \quad (7)$$

$$F_{T+1}(S_{T+1}, D_{T+1}) = 0 \quad (8)$$

We wish to solve:

$$F_t(S_t, D_t) = \max_{x_t, y_t, S_{t+1}} (f_t(x_t, y_t) + F_{t+1}(S_{t+1}, D_{t+1})) \quad (9)$$

subject to:

$$x_{d,t} \leq 1 \quad \forall d \in \mathcal{D}_{i,j,t} \quad \forall i, j \in \mathcal{C} \quad (10)$$

$$y_{i,j,t} \leq u_{i,j,t} \quad \forall i, j \in \mathcal{C} \quad (11)$$

$$\sum_{j \in \mathcal{C}} \sum_{d \in \mathcal{D}_{i,j,t}} x_{d,t} + \sum_j y_{i,j,t} = S_{i,t} \quad \forall i \in \mathcal{C} \quad (12)$$

$$S_{i,t+1} - \sum_k w_{k,i,t} = 0 \quad \forall i \in \mathcal{C} \quad (13)$$

$$\sum_{d \in \mathcal{D}_{i,j,t}} x_{d,t} + y_{i,j,t} - w_{i,j,t} = 0 \quad \forall i, j \in \mathcal{C} \quad (14)$$

where the decision variables $x_{d,t}$, $y_{i,j,t}$ and $w_{i,j,t}$ are nonnegative integers.

Our original optimization problem can then be stated as:

$$\max_{x_0, y_0, S_1} F_0(S_0) \quad (15)$$

Following Powell *et al.* [11], we refer to equation (1) as the *simultaneous* formulation, while (9) is the *recursive* formulation. The combination of S_t and D_t capture the state

of the system, where S_t gives the set of capacity inventories, and the set D_t is the unsatisfied demands at time t .

Equation (9) is computationally intractable, so we propose to solve it by replacing $F_{t+1}(S_{t+1}, D_{t+1})$ with a linear approximation. Let:

$$\nu_{i,t} = \left(\frac{\partial F_t(S_t, D_t)}{\partial S_{i,t}} \right)$$

denote a subgradient of $F_t(S, D)$ with respect to the capacity inventory S_{it} . Replacing $F_{t+1}(S, D)$ with a linear approximation, however, is likely to be unstable, so we add the following control variable:

$$u_{i,j,t} = \text{limit on the number of empties allowed to move from } i \text{ to } j \text{ at time } t.$$

Now define:

$$\hat{F}_t(S_t, D_t, u_t) = \max_{x_t, y_t} (r_t x_t - c_t y_t + \hat{\nu}_{t+1} S_{t+1}) \quad (16)$$

where:

$$\hat{\nu}_{i,t} = \left(\frac{\partial \hat{F}_t(S_t, D_t)}{\partial S_{i,t}} \right)$$

which is quite easy to calculate, rather than the original ν . Finally, let $x_t(u_t), y_t(u_t)$ represent the optimal solution of (16).

The objective function (16) is separable for each terminal $i \in \mathcal{C}$. To see this, the product $\hat{\nu}_{t+1} S_{t+1}$ can be represented as:

$$\begin{aligned} \sum_j \hat{\nu}_{j,t+1} S_{j,t+1} &= \sum_j \hat{\nu}_{j,t+1} \sum_k w_{kjt} \\ &= \sum_j \hat{\nu}_{j,t+1} \sum_k \left(\sum_{d \in \mathcal{D}_{kjt}} x_{dt} + y_{kjt} \right) \end{aligned}$$

Replacing this result in (16),

$$r_t y_t - c_t w_t + \hat{\nu}_{t+1} S_{t+1} = \sum_i \sum_j \left(\sum_{d \in \mathcal{D}_{i,j,t}} r_{d,t} x_{d,t} - c_{i,j,t} y_{i,j,t} \right) + \sum_j \hat{\nu}_{j,t+1} S_{j,t+1} \quad (17)$$

$$= \sum_i \sum_j \left(\sum_{d \in \mathcal{D}_{i,j,t}} (r_{d,t} + \hat{\nu}_{j,t+1}) x_{d,t} + (-c_{i,j,t} + \hat{\nu}_{j,t+1}) y_{i,j,t} \right) \quad (18)$$

Clearly, the objective function decomposes into independent components for each terminal i . Equations (13)–(14) have been incorporated into the objective function, and therefore are automatically satisfied. As a result, equation (16) can be solved using a simple sort. Given n units of capacity, we rank the available loads $d \in \mathcal{D}_{it}$ according to the values $(r_{d,t} + \hat{v}_{j,t+1})$, and the possible empties according to $(-c_{i,j,t} + \hat{v}_{j,t+1})$ and choose the n best. Only one unit of capacity can be assigned to any particular load, and at most u_{ijt} units of capacity can be assigned to move empty from i to j at time t .

If the optimal solution of (16) is $(x_t(u_t), y_t(u_t))$, then we can write the global objective as:

$$G(u) = \sum_{t=0}^T (r_t x_t(u_t) - c_t y_t(u_t)) \quad (19)$$

The problem of finding the right control limits, then, involves solving the following problem:

$$\max_u G(u) \quad (20)$$

subject to:

$$u_t \geq 0 \quad \forall t$$

We are now ready to provide the steps of the algorithm in greater detail.

2.2 The LQN algorithm

With the basic equations stated, we can now summarize the LQN algorithm more formally.

The LQN ALgorithm

Step 1 Initialization

- Set $k = 0$
- Set $\hat{v}_{it} = 0 \quad \forall i, t$
- Set $u_{i,j,t} = 0 \quad \forall i, j, t$

Step 2 Forward Pass

- Solve $\hat{F}_t(S_t, D_t, u_t)$ for $t = 0, 1, \dots, T$.

Step 3 Backward Pass

- Find \hat{v}_{it} for $t = T, T - 1, T - 2, \dots, 2, 1, 0$.

Step 4 Global Update

- Adjust the control variables u_{ijt} and update the values \hat{v}_{it} .

The Forward Pass successively solves linear approximations of the value function, effectively simulating the solution of the problem. The Backward Pass is a gradient pass, from which the approximate subgradients \hat{v}_{it} are computed. Finally, the Global Update adjusts the control variables, and updates the subgradients.

Each of these steps are now described in greater detail.

2.3 Lower Level Optimization (Forward Pass)

For each time period t , the subproblem can be solved if the capacity available at that time period, S_t , is known. Since initially only S_0 is known, we need to use a forward simulation starting at the first time period to create a feasible solution. Equation (18) shows how to rank activities considering that all travel times are equal to one time period. A very similar result holds for the more general case of travel times of different lengths.

For this case, the objective function is still separable, and the constraints represented by (10)–(12) also decompose by terminal i . Decomposing the problem, we arrive at the following integer program for each node (i, t) :

$$\max_{x_i, y_i} \sum_j \left(\sum_{d \in \mathcal{D}_{i,j,t}} (r_{d,t} + \hat{v}_{j,t+\tau_{i,j}}) x_{d,t} + (-c_{i,j,t} + \hat{v}_{j,t+\tau_{i,j}}) y_{i,j,t} \right) \quad (21)$$

subject to, for each $i \in \mathcal{C}$:

$$x_{d,t} \leq 1 \quad \forall d \in \mathcal{D}_{i,t} \quad (22)$$

$$y_{i,j,t} \leq u_{i,j,t} \quad \forall j \quad (23)$$

$$\sum_{j \in \mathcal{C}} \sum_{d \in \mathcal{D}_{i,j,t}} x_{d,t} + \sum_j y_{i,j,t} = S_{i,t} \quad (24)$$

$$x_{d,t}, y_{i,j,t} \geq 0 \quad (25)$$

Finding an optimal integer solution for each of these subproblems consists of ranking the decision variables in the objective by their coefficients, and assigning up to $S_{i,t}$ units to the higher valued variables, subject to the upper bounds on $x_{d,t}$ and $y_{i,j,t}$.

Once decisions at every terminal for time period t are made, the value of each component of the vector S_{t+1} is determined by:

$$S_{k,t+1} = S_{k,t} - \sum_j w_{k,j,t} + \sum_i w_{i,k,t-\tau_{ik}} \quad \forall k \in \mathcal{C} \quad (26)$$

Finally, we have to update the set of unsatisfied demands. Let

\mathcal{D}_{it}^s = the set of demands with origin location i which were satisfied at time t , $\mathcal{D}_{it}^s \subset \mathcal{D}_{it}$.

We can now write:

$$\mathcal{D}_{i,t+1} = \{\mathcal{D}_{it} \setminus \{\mathcal{D}_{it}^s \cup \mathcal{D}_{it}^f\}\} \cup \mathcal{D}_{i,t+1}^0$$

So, the set of uncovered tasks at time $t+1$ is the set of uncovered tasks at time t , minus the union of covered tasks and tasks for which time t is the end of the time window, plus the set of tasks for which time $t+1$ is the beginning of the time window.

The forward pass, then, involves solving a simple sort problem for each location i for each time period t , starting with $t=0$ until $t=T$. Such a simple optimization problem mimics actual decision making in some applications, but is unlikely to provide a good global solution. The challenge is to find gradients \hat{v} and control variables u so the result of solving a series of simple optimization problems is to produce a solution that closely approximates a global optimum. Needless to say, it is possible in real applications to insert a number of additional rules and constraints when solving the subproblem for each terminal. Herein lies the flexibility of this approach.

2.4 Multipliers update (Backward Pass)

The multipliers can be recursively computed starting at time $t = T$. We must look at the flow augmenting/reducing paths from each node (i, t) in respect to the current feasible solution obtained in the forward pass. We begin by noting:

$$\begin{aligned}
\hat{v}_{i,t} &= \frac{\partial \hat{F}_t(S_t)}{\partial S_{i,t}} \\
&= \frac{\partial f_t(x_d, y_{ijt})}{\partial S_{i,t}} + \frac{\partial \hat{F}_{t+1}}{\partial S_{i,t}} \\
&= \sum_{d \in \mathcal{D}_{it}} \left(\frac{\partial f_t(x_d, y_{ijt})}{\partial x_{dt}} \frac{\partial x_{dt}}{\partial S_{i,t}} \right) + \sum_{j \in \mathcal{C}} \left(\frac{\partial f_t(x_d, y_{ijt})}{\partial y_{ijt}} \frac{\partial y_{ijt}}{\partial S_{i,t}} \right) + \frac{\partial \hat{F}_{t+1}}{\partial S_{i,t}} \quad (27)
\end{aligned}$$

Since the solution of equation (21) is simply a sort, incrementing S_{it} by one means putting a unit of flow on the next item in the sorted list of options. Let:

$$\begin{aligned}
I_{dt}^x(k) &= \begin{cases} 1 & \text{if the } k^{\text{th}} \text{ unit of flow will result in changing } x_{dt} \text{ from 0 to 1} \\ 0 & \text{otherwise} \end{cases} \\
I_{ijt}^y(k) &= \begin{cases} 1 & \text{if the } k^{\text{th}} \text{ unit of flow will result in increasing } y_{ijt} \text{ by one unit} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Since S_{it} changes in integer units, we are interested in finding both left and right derivatives of \hat{F} with respect to S_{it} . The right derivative, which gives the value of increasing S_{it} by one, is given by:

$$\hat{v}_{i,t}^+ = \sum_{d \in \mathcal{D}_{it}} r_{dt} I_{dt}^x(S_{it} + 1) + \sum_{j \in \mathcal{C}} c_{ij} I_{ijt}^y(S_{it} + 1) + \hat{v}_{j,t+1}^+ \quad (28)$$

while the left derivative is:

$$\hat{v}_{i,t}^- = \sum_{d \in \mathcal{D}_{it}} r_{dt} I_{dt}^x(S_{it}) + \sum_{j \in \mathcal{C}} c_{ij} I_{ijt}^y(S_{it}) + \hat{v}_{j,t+1}^- \quad (29)$$

Finally, for $t = T$, we have:

$$\hat{v}_{i,T+1} = 0 \quad \forall i \in \mathcal{C}$$

Equations (28) and (29) form the basis of a backward recursion. Below we provide the detailed mechanics of calculating these derivatives.

Computing $\hat{\nu}^+$: To compute $\hat{\nu}_{i,t}^+$ one has to look at the flow augmenting path from node (i, t) . From equation (27), the computation can be divided into two parts: local impact and downstream impact. Equation (21) must be maximized, this time subject to

$$\sum_{j \in \mathcal{L}} \sum_{d \in \mathcal{D}_{i,j,t}} x_{d,t} + \sum_j y_{i,j,t} = S_{i,t} + 1$$

and constraints (22)–(23). The solution of this problem should be compared to the original solution in the forward pass. From (21) and (27) we can identify the following four cases. For each case m , we present a calculation ν^m , and a condition that must be met for the case to apply. For all cases, we assume we are working on terminal i at time t .

Case 1: The extra unit of capacity at node (i, t) would be kept in inventory.

$$\hat{\nu}^1 = c_{i,i} + \hat{\nu}_{i,t+1}^+$$

Assuming no upper bound on inventories, this option can always be considered.

Case 2: The extra unit of capacity at (i, t) would be repositioned to terminal j .

$$\hat{\nu}_j^2 = \begin{cases} -c_{i,j} + \hat{\nu}_{j,t+\tau_{i,j}}^+ & y_{ij,t} < u_{ij,t} \\ -\infty & \text{otherwise} \end{cases}$$

Case 3: The extra unit of capacity at (i, t) would transport a demand d from i to j , which had not been dispatched in the forward pass.

$$\hat{\nu}_d^3 = \begin{cases} r_{d,t} + \hat{\nu}_{j,t+\tau_{i,j}}^+ & \sum_{t' \in \mathcal{T}_d} x_{d,t'} = 0, d \in \mathcal{D}_{it} \\ -\infty & \text{otherwise} \end{cases}$$

Case 4: The extra unit of capacity at (i, t) would transport a demand which had been moved at a later time period t' during the forward pass. In this case, the movement of this demand would be moved to an earlier time period. The sliding of demand d in time directly affects three nodes in the dynamic network (see figure 3). By not moving the load at time t' , we effectively create an extra vehicle at node (i, t') , and eliminate a

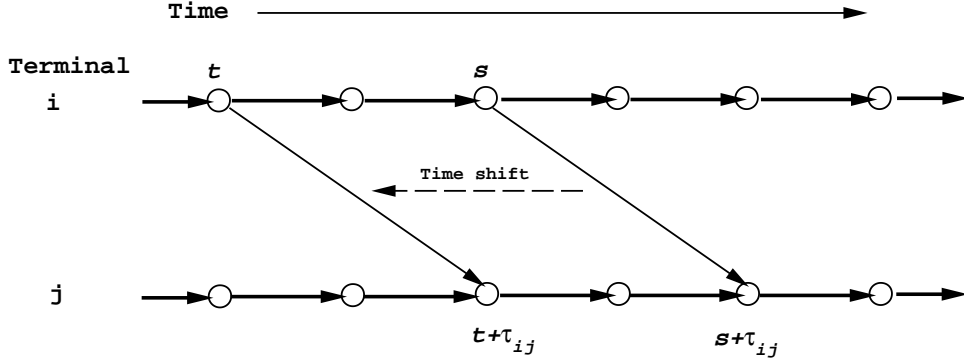


Figure 3: Case 4 for computing $\hat{\nu}^+$ when the load was served later, but an additional unit of capacity allowed the demand to be served earlier

vehicle at node $(j, t' + \tau_{ij})$. Then, by moving the load from (i, t) to $(j, t + \tau_{ij})$, we create an additional vehicle at node $(j, t + \tau_{ij})$. It is helpful to define:

$$h_d^+(t, s) = (r_{d,t} + \hat{\nu}_{j,t+\tau_{ij}}^+) + (-r_{d,s} + \hat{\nu}_{i,s}^+ - \hat{\nu}_{j,s+\tau_{ij}}^-) \quad (30)$$

which gives the value of adding a unit of flow to (i, t) and serving demand $d \in \mathcal{D}_{it}$ which was otherwise served at time $s > t$. Equation (30) is a valid subgradient, but its accuracy can be improved under certain conditions. Assume the flow augmenting path out of node $(j, t + \tau_{ij})$ passes through node $(s + \tau_{ij})$. In this case, there is no net change at node $(s + \tau_{ij})$, and we can replace (30) with:

$$h_d^+(t, s) = r_{d,t} - r_{d,s} + \hat{\nu}_{i,s}^+ \quad (31)$$

We assume that h^+ will be defined by (31) when appropriate.

We can now write the value of sliding the service of this load from s to t as:

$$\hat{\nu}_d^4 = \begin{cases} h_d^+(t, s) & \text{if } \sum_{t' \in \mathcal{T}_d, t' > t} x_{d,t'} = 1 \text{ and } x_{d,s} = 1 \\ -\infty & \text{if } \sum_{t' \in \mathcal{T}_d, t' > t} x_{d,t'} = 0 \end{cases} \quad (32)$$

The last step is calculating $\hat{\nu}^+$. This is done by taking the maximum over all the

cases calculated above. That is:

$$\hat{v}_{it}^+ = \max\{\hat{v}^1, (\hat{v}_{ij}^2, j \in \mathcal{C}), (\hat{v}_d^3, d \in \mathcal{D}_{it}), (\hat{v}_d^4, d \in \mathcal{D}_{it})\}$$

Computing \hat{v}^- : The problem of finding \hat{v}^- closely mirrors that of finding \hat{v}^+ . Whereas before we have to consider what to do with one more unit of capacity, now we have to find out how to eliminate a unit of capacity. This can be represented with the following cases:

Case 1: We would have to eliminate an empty movement:

$$\hat{v}_j^1 = \begin{cases} -c_{i,j} - \hat{v}_{j,t+\tau_{i,j}}^- & y_{ijt} > 0 \\ -\infty & \text{otherwise} \end{cases}$$

We include in this case the inventory link where $i = j$.

Case 2: The loss of a unit of capacity would mean that a demand d served at time period (i, t) can no longer be served. In this case, we have to find out whether this load would be served in the future. The first step is to find t'_d , the first time period in which demand d could be dispatched. For convenience, define the function $h_d^-(t, s)$ as the value of moving load $d \in \mathcal{D}_{ijt}$ from time period t to time period $s > t$:

$$h_d^-(t, s) = (-r_{d,t} - \hat{v}_{j,t+\tau_{i,j}}^-) + (r_{d,s} - \hat{v}_{i,s}^- + \hat{v}_{j,s+\tau_{i,j}}^+) \quad (33)$$

This function captures the value of first refusing load d served in time period t , and then serving the same load starting in time period s . As before it is possible that the flow decrementing path out of node $(j, t + \tau_{i,j})$ passes through node $(j, s + \tau_{i,j})$. In this case, the net change in flow at node $(j, s + \tau_{i,j})$ is zero.

$$h_d^-(t, s) = -r_{d,t} + r_{d,s} - \hat{v}_{i,s}^-$$

Let \mathcal{S} be the set of valid departure times for the load, defined as:

$$\mathcal{S}_d = \{s \mid h_d(t, s) > 0, s \in \mathcal{T}_d, s > t\}$$

Now let:

$$t' = \min\{s \in \mathcal{S}_d\} \quad (34)$$

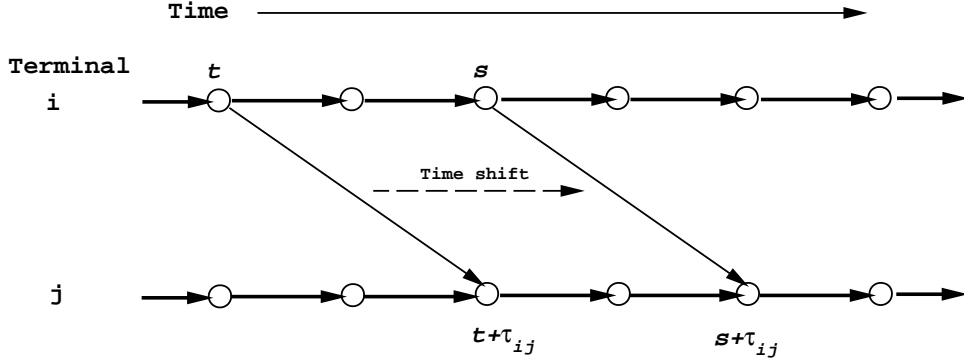


Figure 4: Case 3 for computing $\hat{\nu}^-$ when the load slides to a later time period

be the earliest time period in \mathcal{S} , representing the first time period that the load might satisfy the conditions for being dispatched. Figure 4 illustrates the case of a load sliding to a later time period. If \mathcal{S} is empty, then the load will never be dispatched, with a loss in revenue. So we can now define:

$$\hat{\nu}_d^2 = \begin{cases} h_d^-(d, t') & \text{if } \mathcal{S}_d \neq \emptyset \\ -r_{d,t} - \hat{\nu}_{j,t+\tau_{i,j}}^- & \text{otherwise} \end{cases} \quad (35)$$

where t' is given by (34).

The last step is calculating $\hat{\nu}^-$. This is done by taking the maximum over all the cases calculated above. That is:

$$\hat{\nu}_{i,t}^- = \max\{(\hat{\nu}_{i,j}^1, j \in \mathcal{C}), (\hat{\nu}_d^2, d \in \mathcal{D}_{it})\}$$

2.5 Global update

There are two issues we address in the global update. The first is updating the marginals ν^+ and ν^- . The second is updating the control limits on flows of empties.

Gradient Smoothing: The gradients $\hat{\nu}^+$ and $\hat{\nu}^-$ are valid subgradients of $\hat{F}(S, D)$, but when used to approximate the gradient of $G(u)$ as in equations (36) and (39), they

are only approximations. The reason is that when we revise $\hat{\nu}^+$, we change the optimal ordering of tasks implicit in the solution of the function \hat{F} . Consider the LQN algorithm from iteration k to iteration $k + 1$. Our solution changes in part because we update the control vector u , described below. The solution also changes because we revise our estimate of the duals $\hat{\nu}^+$, which changes the solution of \hat{F} . For example, when we start the algorithm, we initialize $(\hat{\nu}^+)^k = 0$ for $k = 0$. After solving the forward pass, we recalculate $\hat{\nu}^+$, which would then change the forward pass, even if the control vector remained unchanged. If we update u , we end up with a change in $G(u)$ partly because we have updated u , and partly because we have modified $\hat{\nu}^+$.

The net effect of changing the duals $\hat{\nu}^+$ from one iteration to the next, given that we use these during the forward pass, is to introduce a degree of randomness in the estimate of the gradient of $G(u)$. Our estimate of the effect of increasing the supply of capacity S_{it} at a node is disturbed from one iteration to the next by the change in the ordering of tasks, induced by a change in the vector $\hat{\nu}$. As a result, we propose to use a smoothing scheme identical to that used in stochastic linearization algorithms (see, for example, Ermoliev [5], Gupal and Bazhenov [7]). Let $\hat{\nu}^k$ be the duals calculated in the backward pass of the k^{th} iteration (where $\hat{\nu}$ can refer to either $\hat{\nu}^+$ or $\hat{\nu}^-$). Now calculate:

$$\bar{\nu}^k = \gamma \hat{\nu}^k + (1 - \gamma) \bar{\nu}^{k-1}$$

where $\gamma, 0 < \gamma \leq 1$, is the smoothing coefficient. For a value of γ that must be chosen experimentally, we propose to use $\bar{\nu}$ instead of $\hat{\nu}$ in both the forward pass (in equation (16) and in the global updates (below).

Updating the control vector After we complete the backward, gradient pass, we now have the problem of updating our control vector u . We wish to do so in a way that improves the function $G(u)$, defined by (19). We propose to do this by approximating the subgradient of $G(u)$ with respect to a control variable $u_{i,j,t}$ with the subgradient of $\hat{F}_i(S_i, D_i)$ with respect to the control variable $u_{ij,t}$. Thus, let:

$$\eta_{ij,t}^{g+} = \partial_{u_{ij,t}} G(u)$$

represent a subgradient of $G(u)$ with respect to increasing the control variable u_{ijt} . Let:

$$\eta_{ijt}^{f+} = \partial_{y_{ijt}} \hat{F}_t(S_t, D_t) \quad (36)$$

$$= -c_{i,j} + \bar{v}_{j,t+\tau_{i,j}}^+ - \bar{v}_{i,t}^- \quad (37)$$

We propose to use:

$$\eta_{ijt}^{g+} \approx \begin{cases} \eta_{ijt}^{f+} & \text{if } \eta_{ijt}^{f+} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (38)$$

Similarly, we estimate the value of decreasing $G(u)$, η_{ijt}^- using:

$$\eta_{ijt}^{g-} = c_{i,j} + \bar{v}_{j,t+\tau_{i,j}}^- - \bar{v}_{i,t}^+ \quad (39)$$

These equations are only an approximation because from one iteration to the next, we not only change the upper bound, but also we change the marginals \hat{v} that are used in equation (21) for sorting options (loaded or empty movements) in the forward pass. As a result, η_{ijt}^{g+} may be positive, indicating that we should increase u_{ijt} , but in actuality profits may decrease, not because we made the wrong control adjustment, but simply as a result of the randomness introduced by changes in the ranking of options. In the next section, we show that when the values \hat{v} are held constant in equation (21), we get an almost monotonic increase in the objective function, suggesting that equation (38) is a good approximation.

Given an approximation of η_{ijt}^{g+} and η_{ijt}^{g-} , we now have to update the control vector u . Clearly, we would like to keep u integer, so we will use the set of subgradients to determine whether we should increase or decrease u . Let $\mathcal{W} = \{w_1, w_2, \dots, w_N\}$ be the set of gradient values, where an element w_n represents a value of the vector η^{g+} or $-\eta^{g-}$. 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 $w_n = \eta_{ijt}^{g+}$. Now, let $\mathcal{K} = \{\ell_n, n = 1, 2, \dots, K, w_n > 0\}$ be the first K elements, provided that each has a positive derivative. We propose to increase or decrease the elements of u_{ijt} by one, depending on whether $\ell_n = (i, j, t)^+$ or $\ell_n = (i, j, t)^-$.

Another strategy would be to find an ‘‘equilibrium’’ set of duals \hat{v}^+ before changing the control vector. While this might improve the accuracy of the approximation in equation (38), it would be very expensive. Simply put, if (38) is only an approximation, it is a good approximation, and it is better to use it than to spend time refining it.

3 Computational Experiments

In this section, the results obtained from the queueing approach are compared to the optimal solution of the linear relaxation model. The main purpose of the computational experiments is to compare the quality of the solutions obtained by the two methods. The stopping criteria for the LQN method was to exit after it ran for 50 iterations without improvement in the objective function.

The research questions we seeked answers for were:

1. What is the best value for the smoothing factor? How does the smoothing factor affect the evolution of the objective function?
2. What is a good policy for adjusting the upper bounds?
3. How does the average time window for dispatching demands affect the quality of the solution and the CPU times?
4. How does the size of the problem affect the quality of the solution and the CPU times?

We chose two measures of performance. The first one, *OPT ratio*, is the percentage of the optimal value of the objective function reached by the LQN method. The second measure of performance is the *CPU ratio* – the ratio between the CPU time for CPLEX reaching optimality and the CPU time for running the LQN until the stopping criteria is satisfied.

The assumptions here are common to Sections 1 and 2. There are no penalty costs related to keeping capacity in inventory. Capacity becomes available immediately after it reaches a terminal. The set of terminals for the data sets is a real set of terminals spread across the United States. The demands and the initial distribution of capacity were randomly generated. The planning horizon of five days is divided into time periods two hours wide. For the queueing approach, the vector of upper bounds u is initialized as a null vector, meaning that no repositionings are allowed in the first iteration.

problem	demands	capacity	avg. time window in time periods	horizon in days
small1	1000	200	4.0	5
small2	1500	300	4.0	5
std0	2000	400	0.0	5
std4	2000	400	4.0	5
std5	2000	400	5.0	5
std6	2000	400	6.0	5
std7	2000	400	7.0	5
std8	2000	400	8.0	5
large	4000	400	4.0	10

Table 1: Summary of problem sets used for testing, where a time period is two hours

The features of the data sets chosen to compare the procedures are shown in table 1. While we expect real world problems to be larger than any of these data sets, they are large enough to provide an indication of how CPLEX and the LQN method would perform in practice. In addition, we wanted to work with problems that could be solved by CPLEX with reasonable run times. The reader must keep in mind that the LQN algorithm can solve problems that are well beyond the reach of a "vanilla" application of CPLEX.

The data set *std4* was used to choose a good value of the smoothing factor γ . Figure 5 shows the OPT ratio for different values of γ . From this plot we selected $\gamma = 0.4$ for the next several runs.

Figure 6 shows the evolution of the objective function for *std4* when the gradients $\hat{\nu}$

<i>problem</i>	OPT ratio	CPLEX CPU (s)	LQN CPU (s)	CPU ratio
small1	97.1	177	68	2.6
small2	97.0	513	132	3.9
std4	97.2	1673	241	7.0
large	96.0	39364	374	105.3

Table 2: Relative performance of the LQN for problems of different sizes.

in equation (21) are kept at a constant value so that the ranking of tasks at each node (determined by equation (21)). The near monotonic increase in the objective function indicates that equation (38) is providing a good approximation of the objective function when the prioritization of the loads is held constant. Note that this algorithm eventually stops short of the optimal solution. The figure also shows the objective function for the same problem when the values $\hat{\nu}$ in equation (21) are updated with each iteration. While this version of the algorithm produces a much more erratic progression, it also produces an overall higher objective function value than when the values $\hat{\nu}$ are held constant.

Table 2 presents the results of a set of runs to compare the general performance of LQN versus CPLEX for small, medium and large problems (defined relative to each other - many real problems are much larger than our "large" dataset). The table shows relative solution quality and run times. The results show that for these datasets, the logistics queueing network approach is considerably faster than the LP approach using CPLEX, but produces a slightly worse solution. Of course, CPLEX is producing a fractional solution. In addition, CPLEX, by virtue of finding a global optimum, is optimizing over data that in a real application would not be known with certainty. However, it does serve as an effective benchmark. These runs suggest that the LQN algorithm will be able to handle problems of realistic size, producing solutions that are near the best that can be achieved in a real application.

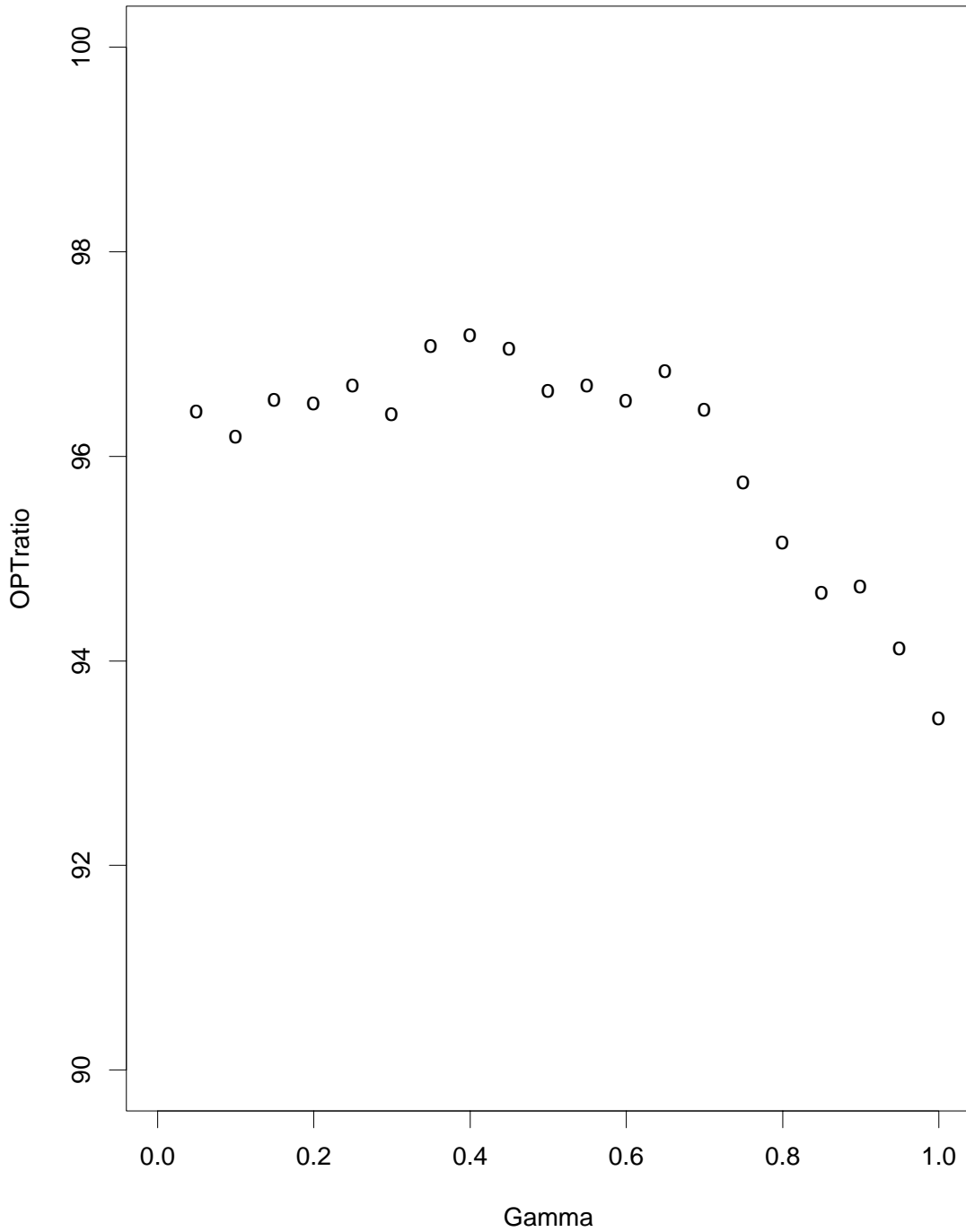


Figure 5: Percent of LP optimal solution produced by LQN algorithm as a function of the smoothing parameter γ .

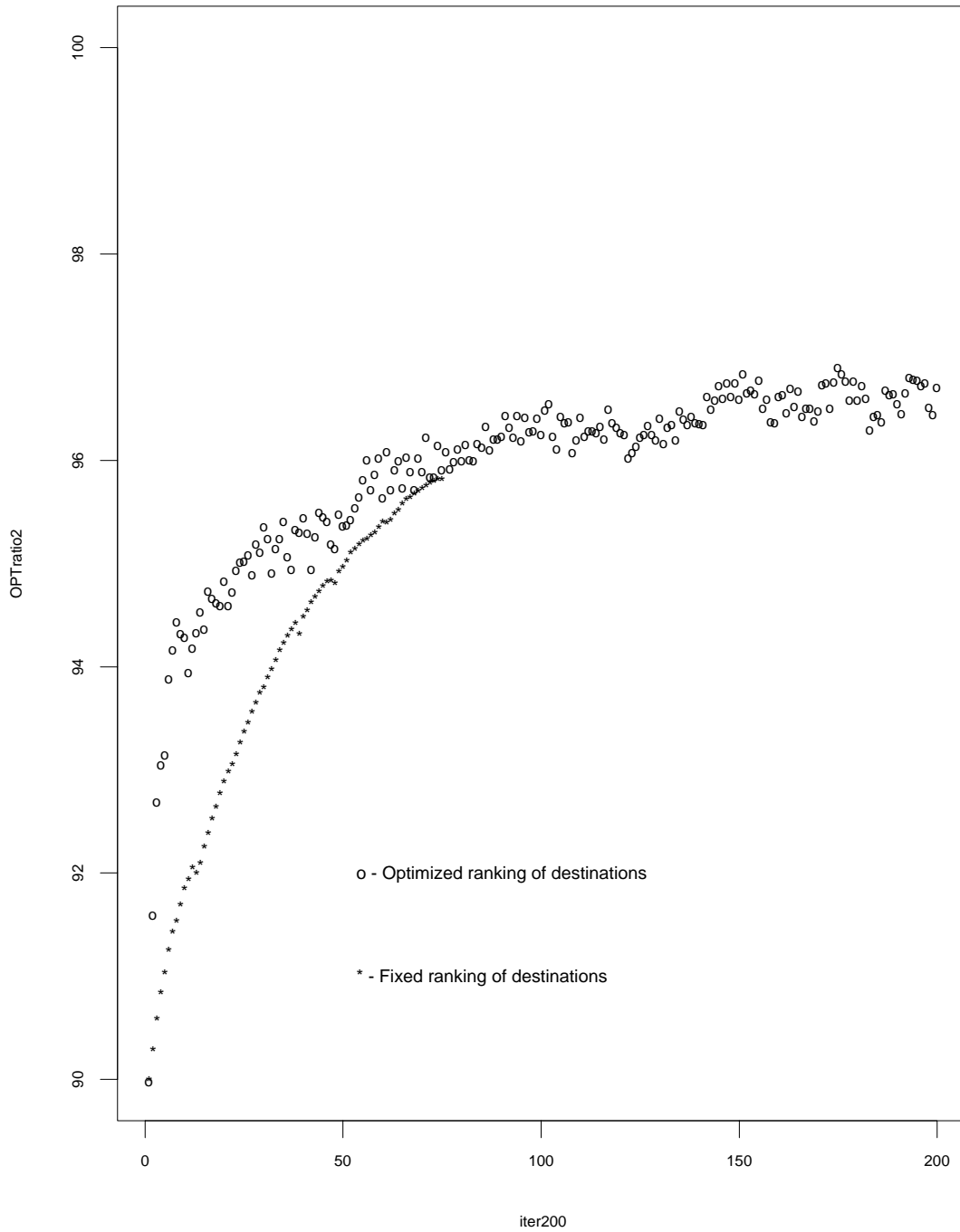


Figure 6: Comparison of objective function convergence when the rankings of destination are held fixed and optimized

problem	OPT ratio	CPU ratio
std0	97.9	0.2
std4	97.2	7.0
std5	97.1	11.3
std6	97.6	21.4
std7	97.4	27.0
std8	97.4	42.7

Table 3: Effect of average time window size on execution times

Finally, we compared data sets having different average time windows for dispatching demands, shown in table 3. The datasets have time windows that increase from zero to eight time periods. While the performance of the LQN method is independent of the time window size, the running times for CPLEX increase considerably with larger average time windows. In some applications, time windows are one-sided (the load must be picked up after a particular date) which would be especially difficult to model.

Data set *std0* is a special case as it has departure time windows of width zero, which can be used to provide a measure of the gap between the continuous solution and the optimal integer solution. When time windows are zero, LQN produces a solution within two percent of the CPLEX solution. Since the relative optimality produced by LQN degrades slightly with larger time windows, it is possible to speculate that this reduction is due to the gap between the continuous and integer optimal solutions, and not due to a reduction in the quality of the solution produced by LQN.

4 Conclusion

In the description of the algorithm, many simplifying assumptions were only made to enable a consistent comparison to the optimal value of the objective function in the linear program model. Lifting some of these restrictions adds little complexity to the LQN algorithm:

- There can be more than one type of capacity. The units of capacity available at a terminal must be matched to demands. In this case the subproblem represented by equations (21)–(24) results in an assignment problem where each combination capacity-activity has a specified cost. The subgradients must represent the value of one more or one less unit of each type of capacity at each node at each time period.
- Demands can be routed through intermediate stops. Each step can be carried out by a different unit of capacity. In this case demands are queued at the intermediate terminals waiting for capacity to carry out the next step.
- Consolidation can also be considered. However, the subproblem (21)–(24) turns out to be more complicated. Demands must be grouped together in loads, resulting in a bin-packing problem.
- Instead of expiring after the due time, demands can have soft time-window constraints. Late demands have a penalty function assigned to them.
- Problems where departures follow a fixed schedule can be successfully solved by the same strategy. Instead of considering discrete time and solving one assignment problem for each terminal at each time period, the assignment problem can be solved before every scheduled departure.
- The problem can be stochastic. During the forward pass, we may randomly sample demands, travel times, or other forecasted quantities.

- We may use other strategies to prioritize loads, such as first come, first served, customer importance, closeness to end of the time window, and so on.

The method presented here is a powerful simulation tool and can be used to answer a variety of questions, including those that arise in problems where departures follow a fixed schedule. For example, the effects of offering a new frequency can be identified. If an increase in demand is forecasted, several hypothetical schedules can be compared for efficiency. The fleet size can be adjusted for planning purposes.

A new approach was proposed for solving dynamic scheduling problems. This approach is flexible enough to include several real world details which would be overlooked in a straightforward LP formulation.

The quality of the solution was evaluated by comparing the optimal value of the objective function of the LP formulation to the value of the objective function achieved by the LQN method. Availability of capacity plays an important role determining how close to the optimal value one can get using the queueing network approach. For problems similar to those found in practice, the OPT ratio was around 97%. In practice, local optimality does not translate into optimality when real time comes into play. Our next step is to determine how these two techniques compare when real time updates and uncertainty are taken into account. Still, the LQN method is a much faster alternative to commercial simplex codes and provides integer solutions.

Acknowledgement

This research was supported in part by grant DDM-9102134 from the National Science Foundation, and by grant AFOSR-F49620-93-1-0098 from the Air Force Office of Scientific Research.

References

- [1] R.K-M. Cheung and W.B. Powell. An algorithm for multistage dynamic networks

- with random arc capacities, with an application to dynamic fleet management. *Operations Research*, 1994.
- [2] K.C. Chih. A real time dynamic optimal freight car management simulation model of the multiple railroad, multicommodity, temporal spatial network flow problem. Ph.D. Dissertation, Department of Civil Engineering and Operations Research, Princeton University, 1986.
 - [3] J. Desrosiers, Y. Dumas, and F. Soumis. The multiple vehicles many to many routing problem with time windows. Publication No. 362, Centre de Recherche sur les transports; Université de Montréal, 1984.
 - [4] J. Desrosiers, M. Solomon, and F. Soumis. Time constrained routing and scheduling. Technical report, Groupe d'études et de recherche en analyse des décisions, 1992.
 - [5] Y. Ermoliev. Stochastic quasigradient methods. In Y. Ermoliev and R. Wets, editors, *Numerical Methods in Stochastic Programming*. Springer-Verlag, 1988.
 - [6] L.F. Frantzeskakis and W.B. Powell. A successive linear approximation procedure for stochastic dynamic vehicle allocation problems. *Transportation Science*, 24(1):40–57, 1990.
 - [7] A. M. Gupal and L. G. Bazhenov. A stochastic method of linearization. *Cybernetics*, pages 482–484, 1972.
 - [8] K.L. Jones, I.J. Lustig, J.M. Farvolden, and W.B. Powell. Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming*, 62:95–117, 1993.
 - [9] W.C. Jordan and M.A. Turnquist. A stochastic dynamic network model for railroad car distribution. *Transportation Science*, 17:123–145, 1983.
 - [10] T.L. Magnanti and R.W. Simpson. Transportation network analysis and decomposition methods. Report no. dot-tsc-rspd-78-6, U.S. Department of Transportation, 1978.
 - [11] W. B. Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. Technical report, Princeton University, 1993. To appear in *Handbook in Operations Research and Management Science*, Volume on *Networks*.
 - [12] W.B. Powell. A comparative review of alternative algorithms for the dynamic vehicle allocation problem. *Vehicle Routing: Methods and Studies*, pages 249–292, 1988.
 - [13] W.B. Powell. A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers. Working paper–sor-94-03, Department of Civil Engineering and Operations Research, Princeton University, 1994.
 - [14] W.W. White. Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers. *Networks*, 2(3):211–236, 1972.