Dynamic Control of Multicommodity Fleet Management Problems

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-96-04

September 25, 1996

Abstract

Dynamic fleet management problems with multiple equipment types and limited substitution can be modeled as dynamic, multicommodity network flow problems. These problems are further complicated by the presence of time windows on task arcs (a task, or load, can be handled at different points in time) and the need for integer solutions. In this paper, we formulate the problem as a dynamic control problem, and show that we can produce solutions within four to five percent of a linear relaxation. In addition, we can solve the ultra-large problems that arise in certain applications; these problems are beyond the capabilities of state-of-the-art linear programming solvers.

Introduction

Freight transportation companies face the problem of matching a fleet of vehicles to current and future customer demands. Prior research has most commonly formulated the problem as a linear network (White and Bomberault [24], White [23], Herren [10, 11], Turnquist [22] and Joborn [13]) while other researchers have focused on incorporating uncertainty in forecasts of future demands (Jordan and Turnquist [15], Powell [18], Frantzeskakis and Powell [8], Cheung and Powell [2]). All of this work has assumed a homogeneous fleet of vehicles. Most problems in practice have multiple equipment types with limited substitution (a load might be served by two or more vehicle types). These problems are most commonly formulated as multicommodity network flow problems (Shan [21], Chih [3], Magnanti and Simpson [16], Hane *et al.* [12]) but can also be formulated as set partitioning problems (Desrosiers *et al.* [6, 5]).

Integer, multicommodity network flow problems remain a difficult challenge and active area of research. The hardest classes of problems (without assuming stochastic demands) are those that combine relatively wide time windows that describe when a task needs to be completed, with relatively long time horizons. In this paper, we propose a flexible, fast solution approach based on dynamic control of logistics queueing networks (LQN). This modeling approach was first suggested by Powell and Carvalho [19]. In contrast with prior modeling approaches, which formulate this class of problems as a multicommodity network flow problem, the LQN framework views the system as a network of double-ended queues, comprised of a queue of vehicles waiting to serve customers, and a queue of customers waiting to be served by a vehicle. When a vehicle moves a customer from i to j, it is removed from the queue of vehicles at i, and added to the queue at j at some point in the future. It then waits in the queue at j until it is assigned to a waiting customer. A customer, at the same time, might sit in a queue of other customers waiting to be assigned to a vehicle. However, while vehicles are expected to wait indefinitely, it is expected that a customer will leave the system at the end of a prespecified time window.

The prior work with this technique ([19]) assumed a homogeneous fleet of vehicles.

In this paper, we extend the methodology to handle heterogeneous fleets of vehicles with flexible substitution of different vehicle types for different types of customers. We assume there are A types of vehicles and B types of loads. We use Υ as the *matching matrix* such that $\Upsilon_{a,b} = 1$ if vehicle type a can be assigned to transport load type b, and $\Upsilon_{a,b} = 0$ otherwise. Thus, we can capture relatively general rules for the substitution of one type of equipment for another.

As an example, let us look at the simple problem of two different types of vehicles and two different types of loads. In case either type of load is fit for either type of vehicle, $\Upsilon_{a,b} = 1 \ \forall a, b$. If indeed the two types of vehicles are interchangeable with no cost penalty, the problem reduces to a *single* commodity fleet management problem. If there can be no substitution, as for example in the case that the matching matrix is an identity matrix, the problem decomposes into two independent single commodity fleet management problems. Therefore, true multicommodity fleet management problems fall in between these two extremes that are much simpler. Figure 1 shows a snapshot of a network of transportation services with two types of vehicles and two types of loads and the matching matrix for it.

In practice, substitution of equipment is seldom a clearcut issue. Resources and tasks can be divided into types but there can be exceptions. Our model is developed from the beginning assuming that loads have to be handled individually due to the time windows. Thus the number of load types does not impact the complexity of the problem. The number of vehicle types does affect complexity and every effort should be made by the user to reduce that parameter to the least possible.

Linear programs resulting from multicommodity network problems lend themselves to decomposition techniques due to their staircase structure. The way the problem is formulated plays an important role in the efficiency these problems are solved (see Jones *et al.* [14]). Decomposition techniques lead to the possibility of using parallel computation, and this has been explored extensively in the literature (for example, Pinar and Zenios [17] and Shultz and Meyer [20]). Interior point methods have recently been shown to be good tools for solving multicommodity problems (Choi and Goldfarb

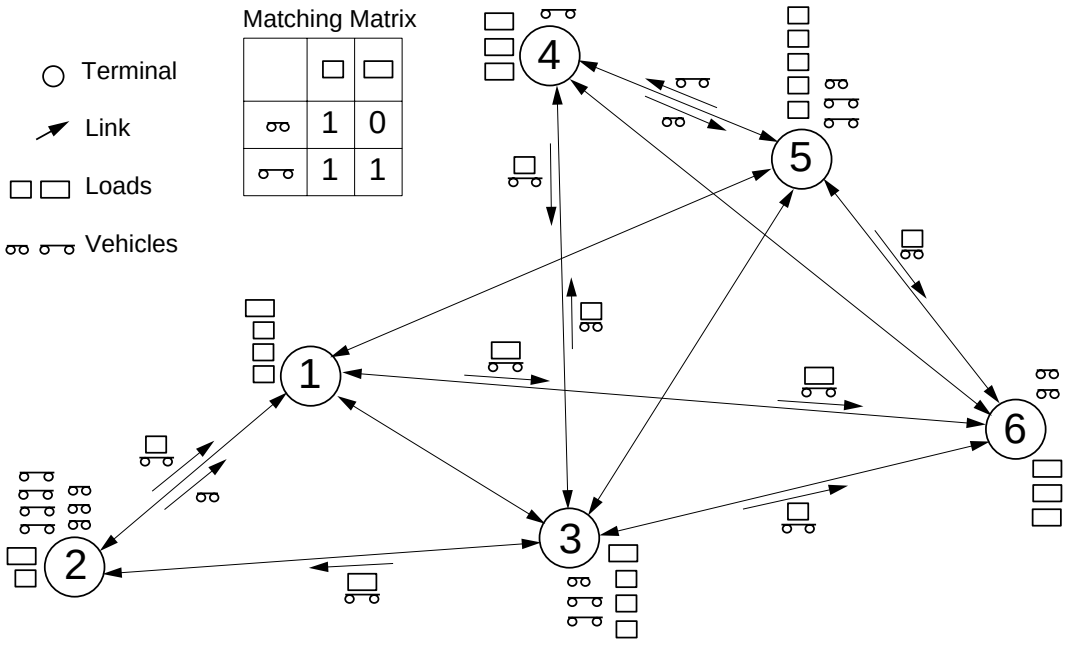

Figure 1: Network of transportation services with two types of loads and two types of vehicles.

[4]).

The logistics queueing network approach considers explicitly the dynamic structure of the problem, and is similar in philosophy to the approach first suggested in Jordan and Turnquist [15]. The concept involves solving large, dynamic resource allocation problems through a series of subproblems that is solved at each point in space time. At the level of each subproblem, complex equipment substitution rules are easily incorporated.

The contribution of this paper is as follows. 1) We show how the logistics queueing network concept can be extended to handle heterogeneous fleets of vehicles. 2) We present two algorithms for solving the problem as a dynamic control problem. 3) We demonstrate experimentally that the technique produces high quality solutions for problems that are small enough to be solved using a commercial linear programming solver, as well as problems that are too large to be solved in this way.

This paper is divided in five sections. Section 1 presents a mixed–integer program-

ming formulation for the problem. Section 2 introduces the basic equations for the methodology. Section 3 discusses the algorithms and implementation issues. Section 4 presents the results of the computational experiments. Finally, section 5 presents the conclusion and directions for further research.

1 Linear Programming Model

We assume that all the loads and vehicles that become available within the planning horizon are known. The planning horizon is split into *time periods* so that time increments by discrete integer steps.

To formulate this problem as a linear program, we define the following:

Network:

- $\mathcal{C}$ is the set of terminals i in the network.
- $\mathcal{A}$ is the set of vehicle types.
- τ_{aij} is the travel time for vehicle type $a \in \mathcal{A}$ 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.

Activity variables:

- $\mathcal{B}$ is the set of load types.
- Υ is the indicator matrix of feasible load/vehicle pairs, such that if vehicle type a can be assigned to load type b then $\Upsilon_{ab} = 1$, otherwise $\Upsilon_{ab} = 0$.
- $\mathcal{L}$ is the set of loads l of all types available within the planning horizon, T.
- $\mathcal{L}^b$ is the set of loads l of type b, for each type $b \in \mathcal{B}$, available within the planning horizon, T.

- $\mathcal{T}_l$ is the set of feasible departure times for satisfying load $l \in \mathcal{L}$, otherwise known as the *departure time window*.
- $\mathcal{L}^b_{ijt}$ is the set of loads $l \in \mathcal{L}^b$ with origin i and destination j having t as a feasible departure time.
- R_{ait} is the net inflow ($R_{ait} > 0$) or outflow ($R_{ait} < 0$) of vehicles of type a at terminal i at time t.
- r_{alt} is the profit generated by choosing vehicle type a and time t to satisfy load l.
- c_{aij} is the cost of repositioning one vehicle type a over link (i, j, t).

Decision variables:

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

Whenever we refer to x, y, z and w, we assume they compose a feasible solution.

The objective function we are maximizing is stated as:

$$F(x, y) = \sum_{t=0}^{T} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \sum_{a \in \mathcal{A}} \left(\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}^b_{ijt}} r_{alt} x_{alt} - c_{aij} y_{aijt} \right) \tag{1}$$

This problem can be formulated as:

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

subject to:

$$\sum_{t \in \mathcal{T}_l} \sum_{a \in \mathcal{A}} \Upsilon_{ab} x_{alt} + z_l = 1 \quad \forall\, l \in \mathcal{L}^b \;\; \forall\, b \in \mathcal{B} \tag{3}$$

$$\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}^b_{ijt}} \Upsilon_{ab} x_{alt} + y_{aijt} - w_{aijt} = 0 \quad \forall\, i,\, j \in \mathcal{C},\, \forall\, t \;\leq T\,,\, \forall\, a \in \mathcal{A} \tag{4}$$

$$\sum_{j \in \mathcal{C}} w_{aijt} - \sum_{j \in \mathcal{C}} w_{ajit-\tau_{aij}} = R_{ait} \;\; \forall\, (i,t) \in \mathcal{N},\;\; \forall\, a \in \mathcal{A} \tag{5}$$

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

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

Our problem consists of maximizing profits by assigning up to one vehicle to each load, enforced by constraints (3), and having flow conservation of each type of vehicle at each node, enforced by constraints (4) and (5). This formulation can be regarded as a network problem with GUB (generalized upper bounding) constraints.

Typical multicommodity network problems consist of a similar set of bundling constraints plus flow conservation. In the multicommodity LQN, however, the bundling involves not only the choice of resource assignment (equation (4)), but also the choice of departure time (equation (3)). The purpose of this linear program is to evaluate the quality of the solutions obtained by using the LQN approach. In practice, large multicommodity network problems are not usually solved directly, but decomposition strategies are employed. 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, which is an upper bound to the optimal solution of the problem. Another way to obtain an upper bound to the optimal value of the integer program is by ignoring the vehicle and load types and solving the problem as if it were single commodity, i.e., $\Upsilon_{ab} = 1\ \forall a, b$. We refer to this relaxation as the *commodity relaxation* and we resort to it when the linear relaxation is deemed too large.

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 loads that use a sequence of links

through intermediate terminals from origin to destination.

2 Equations for the Multicommodity LQN

The previous section presented a somewhat standard formulation of the fleet assignment problem as a large scale linear program. These problems become notoriously difficult to solve as the planning horizon grows. In our approach, we view the problem in the context of optimal control of queues. Consider a queue of vehicles and loads waiting to be assigned to each other. In most large problems, we could assign vehicles to loads using a simple prioritization srategy (first-in, first-out, or prioritizing loads that are closest to the end of their time window). This simple strategy fails under two conditions: when there are too many vehicles, or too many loads. In the case of too many loads, we would like to prioritize loads in part based on the value of additional vehicles at the destination of the load. If there are too many vehicles, we may have to reposition some of them empty to other locations. Again, this would be based on the estimated value of the vehicles at those destination.

In the LQN approach, we use a concept called the *spatial potential function*, denoted by the vector:

$$\xi_t = (\ldots, \xi_{a,i,t}, \ldots)$$

where

$$\xi_{a,i,t} = \text{the value of an additional vehicle of type } a \text{ at location } i \text{ at time } t \quad .$$

Below, we show how to estimate ξ_t using gradients of an approximation of a value function derived using dynamic programming. The vector ξ_t can be viewed as an estimate of the slope of the value function, giving the marginal value of additional capacity of each vehicle type at each location at a point in time. Thus, if there is a vehicle of type a waiting at node (i,t) and we might assign it to a load l going to node $(j, t+\tau_{aij})$, then the value of this assignment would be the net revenue from the load, $r_{a,l,t}$, plus the value of the vehicle at the destination, $\xi_{a,j,t+\tau_{aij}}$. Technically, the destination value $\xi_{a,j,t+\tau_{aij}}$

is a function of all other decisions being made. The efficiency of the LQN approach derives from the approximation being made whereby $\xi_{a,j,t+\tau_{aij}}$ is held constant while the assignment problem at time t is being solved.

This section is divided in three parts. In the first part we present the LQN formulation for multicommodity problems. The second part contains equations to approximate the gradients. The third part consists of the equations to update the gradients between iterations based on the LAMA (Linear Approximation, Multiplier Adjustment) algorithm first presented in Powell and Carvalho [1]. The LAMA algorithm given in [1], in contrast with the subgradient algorithm given in [19], updates each individual element of the vector ξ_t based on a careful calculation of the exact impact of the change on overall solution quality. The LAMA algorithm is slower and more complicated than the subgradient algorithm given in [19] but gives higher quality results.

2.1 The Decomposition

In this section, we pose the problem in the format of a dynamic program. We then replace the value function in the dynamic program with a linear approximation, which in turn decomposes the problem into a series of double-ended queues. First define:

- $\mathcal{L}_{it}$ is the set of loads l with origin i having t as a feasible departure time.
- $\bar{\mathcal{L}}_{ijt}$ is the set of loads l with origin i and destination j, which are available to move at time t but 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 $\mathcal{T}_l$.

- $\mathcal{L}_{it}^{f}$ is the set of loads l with origin i, where t is the end of the time window $\mathcal{T}_d$.

The recursive form for the objective function is represented by:

$$g_{i,t}(x, y, V_{i,t}, \mathcal{L}_{i,t}) = \sum_{j \in \mathcal{C}} \sum_{a \in \mathcal{A}} \left(\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}_{ijt}^{b}} r_{alt} x_{alt} - c_{aij} y_{aijt} \right) \quad (8)$$

$$G_t(V_t, \mathcal{L}_t) = \sum_{i \in \mathcal{C}} g_{i,t}(x, y, V_{i,t}, \mathcal{L}_{i,t}) + G_{t+1}(V_{t+1}, \mathcal{L}_{t+1}) \quad (9)$$

where $g_{i,t}$ is the contribution to the objective function of the decisions taken at time t at terminal i and G_t is the contribution to the objective function of the decisions taken from time t to the end of the planning horizon.

We solve this problem by replacing the value function $G_{t+1}(V_{t+1}, \mathcal{L}_{t+1})$ in equation (9) with the linear function $\xi_t V_t$. Since a linear approximation can produce unstable results, we introduce a control variable in the form of an upper bound on the movement of empty vehicles between locations. These upper bounds are decision variables, not constraints, and the LQN algorithm iteratively adjusts these upper bounds to improve solution quality.

Let $V_{a,i,t}$ be the number of vehicles of type a available at node (i, t). With the approximation, equation (9) becomes

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

The flows of empties are then constrained by:

$$y_{aijt} \leq u_{aijt} \quad \forall a,\ i,\ j,\ t \quad (11)$$

where u_t is iteratively adjusted from one iteration to the next.

After replacing the approximation for the value function and expanding equation (10), we arrive at the following equation:

$$\hat{G}_t(V_t, \mathcal{L}_t) = \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \sum_{a \in \mathcal{A}} \left(\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}_{ijt}^{b}} (r_{alt} + \xi_{a,j,t+\tau_{aij}}) x_{alt} + (-c_{aij} + \xi_{a,j,t+\tau_{aij}}) y_{aijt} \right) \quad (12)$$

This problem decomposes into *local problems*, one for each terminal. In order to assure the feasibility of each local problem, we replace the linear approximation for the pricing of inventory links by a piecewise linear approximation (see [19]). We can then state the local problem at node (i,t):

$$\max_{x_t,y_t} \sum_{j\in\mathcal{C}} \sum_{a\in\mathcal{A}} \left(\sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}^b_{ijt}} (r_{alt} + \xi_{a,j,t+\tau_{aij}})x_{alt} + (-c_{aij} + \xi_{a,j,t+\tau_{aij}})y_{aijt} \right) \tag{13}$$

subject to:

$$\sum_{a\in\mathcal{A}} x_{alt} \leq 1 \qquad \forall l \in \bar{\mathcal{L}}_{it} \tag{14}$$

$$y_{aijt} \leq u_{aijt} \quad \forall j \in \mathcal{C},\ \forall a \in \mathcal{A} \tag{15}$$

$$\sum_{l\in\bar{\mathcal{L}}_{it}} \sum_{a\in\mathcal{A}} x_{alt} + \sum_{j\in\mathcal{C}} y_{aijt} \leq V_{a,i,t} \quad \forall a \in \mathcal{A} \tag{16}$$

$$x_{alt}\ ,\ y_{aijt} \geq 0 \tag{17}$$

In the single commodity case, the subproblem reduces to a simple sort. However, when there exist several types of vehicles, the local problem reduces to an assignment problem. The assignment problem can be solved using especialized algorithms or greedy heuristics to find a near optimal solution. After all the local problems for time t are solved, the set of available loads and the vector of vehicles available in the next time period are computed by:

$$\bar{\mathcal{L}}_{k,t+1} = \{\bar{\mathcal{L}_{kt}} \setminus \{\mathcal{L}^s_{kt} \cup \mathcal{L}^f_{kt}\}\} \cup \mathcal{L}^0_{k,t+1} \qquad \forall k \in \mathcal{C} \tag{18}$$

$$V_{a,k,t+1} = V_{a,k,t} - \sum_j w_{akjt} + \sum_i w_{aikt+1-\tau_{aij}} + R_{ait+1} \ \ \forall k \in \mathcal{C},\ \forall a \in \mathcal{A} \tag{19}$$

As the values of the decisions at node (i,t) depend on the coefficients of the local problem, we regard these decisions as having the following functional dependence:

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

$$y_{aijt} = y_{aijt}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \tag{21}$$

The contributions in the objective function of each local problem are added up so that

$$G(x,y) = \sum_{t=0}^{T} \sum_{i\in\mathcal{C}} g_{i,t}(x, y, V_{i,t}\,, \mathcal{L}_{i,t}\,) \tag{22}$$

An issue that greatly affects the quality of the solutions obtained using this approximation is the estimation of the spatial potential function ξ. For the single commmodity case, two strategies for the value of the potential function were explored. Let us first define the gradient of G_t with respect to $V_{a,i,t}$:

$$\nu_{a,i,t} = \frac{\partial \hat{G}_t}{\partial V_{a,i,t}} \tag{23}$$

A gradient approximation strategy is explored in [19]. In this strategy, ξ is defined as an average ($\bar{\nu}$) of the gradient approximations through iterations. For iteration $n+1$:

$$\bar{\nu}^{n+1}_{a,i,t} = \gamma \nu^{n}_{a,i,t} + (1-\gamma)\bar{\nu}^{n}_{a,i,t} \tag{24}$$

where $0 < \gamma \leq 1$ is the smoothing factor.

A multiplier adjustment strategy for ξ is investigated in [1], where the values of ξ are adjusted by computing finite differences $\Delta\xi$ and the impact of adjusting ξ by $\Delta\xi$ in the objective function, $\Delta G(\Delta\xi)$. The multiplier adjustment strategy was shown to yield values for the optimal solution that are within 2.0% of the optimal value of the linear relaxation, at the expense of more computation time.

In this paper, we chose to investigate the performance of both procedures for multicommodity LQNs. Depending on the features of the problem, the gradient approximation approach might be more desirable for being faster, even though it delivers results that might be slightly worse.

2.2 Gradients of the Objective Function

In this section we present the equations to compute the gradients of G_t with respect to V and u. By definition,

$$\nu_{a,i,t} = \frac{\partial \hat{G}_t}{\partial V_{a,i,t}} \tag{25}$$

We also define

$$\eta_{aijt} = \frac{\partial \hat{G}_t}{\partial u_{aijt}} \tag{26}$$

The equations to compute the gradients of G with respect to V and u for the single commodity case have been rigorously derived in [19]. The derivation for the multicommodity case is similar but very tedious. We present the equation for each gradient and the intuition behind it.

2.2.1 Supply Gradients

Using the recursive relation (9) we arrive at

$$\nu_{a,i,t} = \frac{\partial g_{i,t}}{\partial V_{a,i,t}} + \frac{\partial \hat{G}_{t+1}(V_{t+1}, \mathcal{L}_{t+1})}{\partial V_{a,i,t}} \tag{27}$$

In the single commodity case, an increase in the supply of vehicles at node (i,t) may produce the following changes: the movement of a load l from (i,t) to its destination node $(j, t+\tau_{ij})$, the increase in the supply of vehicles at the destination node, and the removal of load l from the queue of loads at node $(i,t+1)$ (it might be that load l was not moved until some time $t+k$, or it may never been moved at all.

In the multicommodity case, the outcome of increasing the supply of vehicle type a at node (i,t) may be far more varied due to substitution possibilities. For example, suppose there is only one load l_0 at node (i,t), which is being carried by vehicle type a_1 in the current solution. If vehicle type a_2 is more valuable at the destination of l_0 than a_1, then $\nu_{a_2,i,t}$ must reflect that substitution possibility. As there may be a chain of substitutions, we use the results from [19] to approximate the gradient by a sum of the individual contributions of perturbations in the solution for each task out of node (i,t).

We first show how to compute the right gradient. In order to represent the perturbation on the vector of vehicles available at node (i,t), we define the following notation:

$$\tilde{V}_{i,t}(a) = V_{i,t} + e_a \tag{28}$$

where e_a is a unit vector with the only positive component in position a. We compute the following indicator variables:

$$\frac{\partial x_{a'lt}}{\partial V^+_{a,i,t}} = X^+_{a'lt} = x_{a'lt}(\tilde{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - x_{a'lt}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \tag{29}$$

$$\frac{\partial y_{a'ijt}}{\partial V^+_{a,i,t}} = Y^+_{a'ijt} = y_{a'ijt}(\tilde{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - y_{a'ijt}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \quad (30)$$

$$\frac{\partial \tilde{y}_{a'iit}}{\partial V^+_{a,i,t}} = Z^+_{a'iit} = \tilde{y}_{a'iit}(\tilde{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - \tilde{y}_{a'iit}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \quad (31)$$

where $\tilde{y}_{a'iit}$ represents the unconstrained inventory of vehicles type a' at terminal i from time t to $t+1$.

Consistent with [19], there may be four types of perturbations in the current solution out a node. The first two types relate to the load variables x. In the first type, an additional load is covered. In the second type, a load that is satisfied at a later time period in the current solution, is pushed to an earlier time. The third type of perturbation is performing an additional empty move. Finally, the fourth type is to have the additional supply in inventory. In practice, the sum of the individual effects has shown to be a good approximation of the overall impact.

So, instead of starting from equation (27), we compute each individual perturbation by the differences (29)– (31):

$$\nu^+_{a,i,t} \simeq \sum_{j\in\mathcal{C}} \sum_{a'\in\mathcal{A}} \Big(\sum_{l\in\bar{\mathcal{L}}_{ijt}} \frac{\partial \hat{G}_t}{\partial x_{a'lt}} \frac{\partial x_{a'lt}}{\partial V_{a',i,t}} + \frac{\partial \hat{G}_t}{\partial y_{a'i,j,t}} \frac{\partial y_{a'i,j,t}}{\partial V_{a',i,t}} \Big) \quad (32)$$

This results in

$$\begin{aligned} \nu^+_{a,i,t} \simeq & \sum_{j\in\mathcal{C}} \sum_{l\in\bar{\mathcal{L}}_{ijt}} \sum_{a'\in\mathcal{A}} X^+_{a'lt} \left(r_{a'lt} + \nu^+_{a',j,t+\tau_{a'ij}} + \sum_{t'>t} \sum_{a''\in\mathcal{A}} \hat{x}_{a''lt'}(-r_{a''lt'} + \nu^+_{a'',i,t'} - \nu^-_{a'',j,t'+\tau_{a''ij}}) \right) \\ & + \sum_{j\in\mathcal{C}} \sum_{a'\in\mathcal{A}} Y^+_{a'ijt} \left(-c_{a'ij} + \nu^+_{a',j,t+\tau_{a'ij}} \right) + \sum_{a'\in\mathcal{A}} Z^+_{a'iit}\, \nu^+_{a',i,t+1} \end{aligned} \quad (33)$$

where

$$\hat{x}_{a''lt'} = x_{a''lt'}(V_{i,t'}, \xi_{t'+1}, u_{i,t'}, \mathcal{L}_{it'}) \quad (34)$$

Equation (33) has three components. The first one captures the first two types of perturbations (load variables). The second captures perturbations from changes in the repositioning decisions and the third changes from inventory decisions.

The left gradient is computed in the same fashion, but it is only defined where $V_{a,i,t} > 0$. We define the following notation:

$$\bar{V}_{i,t}(a) = V_{i,t} - e_a \quad (35)$$

where e_a is a unit vector with the only positive component in position a. We compute the following indicator variables:

$$\frac{\partial x_{a'lt}}{\partial V^-_{a,i,t}} = X^-_{a'lt} = x_{a'lt}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - x_{a'lt}(\bar{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \quad (36)$$

$$\frac{\partial y_{a'ijt}}{\partial V^-_{a,i,t}} = Y^-_{a'ijt} = y_{a'ijt}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - y_{a'ijt}(\bar{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \quad (37)$$

$$\frac{\partial \tilde{y}_{a'iit}}{\partial V^-_{a,i,t}} = Z^-_{a'iit} = \tilde{y}_{a'iit}(V_{i,t}, \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) - \tilde{y}_{a'iit}(\bar{V}_{i,t}(a), \xi_{t+1}, u_{i,t}, \mathcal{L}_{it}) \quad (38)$$

The left gradient is approximated by

$$\nu^-_{a,i,t} \simeq \sum_{j\in\mathcal{C}} \sum_{l\in\bar{\mathcal{L}}_{ijt}} \sum_{a'\in\mathcal{A}} X^-_{a'lt} \left(r_{a'lt} + \nu^-_{a',j,t+\tau_{a'ij}} + \sum_{t'>t} \sum_{a''\in\mathcal{A}} \frac{\partial x_{a''lt'}}{\partial V_{a',i,t}} (-r_{a''lt'} + \nu^-_{a'',i,t'} - \nu^+_{a'',j,t'+\tau_{a''ij}}) \right) + \sum_{j\in\mathcal{C}} Y^-_{a'ijt} \left(-c_{a'ij} + \nu^-_{a',j,t+\tau_{a'ij}} \right) + Z^-_{a'iit}\, \nu^-_{a',i,t+1} \quad (39)$$

2.2.2 Upper Bound Gradients

Increasing the upper bound on a link may result in flow increase or not. If there is no flow increase, there is no impact in the current solution. But if raising the upper bound results in a flow increase, then the origin node of that link loses one vehicle and the destination node gains one vehicle. Through a similar derivation to the one for the single commodity case [19], we arrive at

$$\eta^+_{aijt'} \begin{cases} \simeq -c_{aij} + \nu^+_{a,j,t'+1} - \nu^-_{a,i,t'} & \text{if } -c_{aij} + \nu^+_{a,j,t'+1} - \nu^-_{a,i,t'} > 0 \\ = 0 & \text{otherwise} \end{cases} \quad (40)$$

The result is similar for decreasing an upper bound. If indeed there is a decrease in flow, one vehicle is added at the origin node of the link and one vehicle is removed from the destination node.

$$\eta^-_{aijt'} \begin{cases} \simeq c_{aij} - \nu^-_{a,j,t'+1} + \nu^+_{a,i,t'} & \text{if } c_{aij} - \nu^-_{a,j,t'+1} + \nu^+_{a,i,t'} > 0 \\ = 0 & \text{otherwise} \end{cases} \quad (41)$$

2.3 LAMA equations

In the Linear Approximation and Multiplier Ajustment (LAMA) method, the spatial potential function for vehicles is regarded as a control variable and adjusted according

to the impact in the objective function. For each node (i,t), we compute the minimum necessary perturbation on the value of $\xi_{a,i,t}$ for each vehicle type a. Increasing or decreasing $\xi_{a,i,t}$ may result in a change in the number of vehicles available at node (i,t). For a perturbation on $\xi_{a,i,t}$ we compute the corresponding change in the objective function. At any given iteration of the LAMA method, we adjust the component of ξ that results in the highest increase in the objective function. In this section we first present equations to increase ξ and decrease ξ and the computation of the corresponding perturbations in the objective function.

2.3.1 Increasing the Spatial Potential Function

Let us look at increasing the multiplier $\xi_{a,i,t}$. Whenever (i,t) is the destination node for a task from terminal k that can be assigned vehicle type a, $\xi_{a,i,t}$ must appear in the objective function of the local problem at node $(k, t-\tau_{aij})$. We need to find $\Delta\xi_{a,i,t}$, the smallest increase in $\xi_{a,i,t}$ that will push an additional unit of flow type a into node (i,t). In order to find $\Delta\xi_{a,i,t}$ we must compute δ^+_{akit} for each terminal k, which is the smallest increase in $\xi_{a,i,t}$ that will result in an additional unit of flow being pushed from node $(k, t-\tau_{aki})$ to node (i,t).

$$\Delta\xi^+_{a,i,t} = \min_{k\in\mathcal{C}}\{\delta^+_{akit}\} \tag{42}$$

Increasing flow on link $(k,i,t-\tau_{aki})$ can be achieved in three ways. First, a possible empty move from $(k,t-\tau_{aki})$ to (i,t) could be assigned flow. Let $\delta^+_1(a,k,i,t)$ be the necessary increase in $\xi_{a,i,t}$ for this case. Second, a load destined to (i,t) that would otherwise be held may be assigned flow. Let $\delta^+_2(a,k,i,t)$ be the necessary increase in $\xi_{a,i,t}$ for this to happen. Third, a load bound to (i,t) that had been assigned a different type of vehicle could be assigned vehicle type a. Let $\delta^+_3(a,k,i,t)$ be the necessary increase in $\xi_{a,i,t}$ for this case. Therefore

$$\delta^+_{akit} \;=\; \min\{\delta^+_1(a,k,i,t),\, \delta^+_2(a,k,i,t),\, \delta^+_3(a,k,i,t)\} \tag{43}$$

where

$$\delta_2^+(a,k,i,t) = \min_l\{\delta_{2l}^+(a,k,i,t)\ \forall\, l \in \bar{\mathcal{L}}_{kit-\tau_{aki}} \mid \sum_{a'} x_{a'lt-\tau_{kit}} = 0\} \quad (44)$$

$$\delta_3^+(a,k,i,t) = \min_l\{\delta_{3l}^+(a,k,i,t)\ \forall\, l \in \bar{\mathcal{L}}_{kit-\tau_{aki}} \mid x_{a'lt-\tau_{kit}} = 1\,,\ a' \neq a\} \quad (45)$$

The smallest increase in $\xi_{a,i,t}$ that would result in an additional empty move from $(k, t-\tau_{aki})$ to (i,t) is represented by $\delta_1^+(a,k,i,t)$. Therefore the empty move has to be more attractive, i.e., have a larger coefficient in the objective funtion, than the value of one less vehicle of type a at node $(k, t-\tau_{aki})$. It follows that

$$\delta_1^+(a,k,i,t) = \nu^-_{a,k,t-\tau_{aki}} + c_{aki} - \nu^+_{a,i,t} + 1 \quad (46)$$

The smallest increase in $\xi_{a,i,t}$ that would result in assigning flow to a load l at node $(k, t-\tau_{aki})$ that would otherwise go on hold is represented by $\delta_{2l}^+(a,k,i,t)$. The coefficient for $x_{al,t-\tau_{aki}}$ in the objective function of the local problem has to be larger than the value of one less vehicle of type a at node $(k, t-\tau_{aki})$.

$$\delta_{2l}^+(a,k,i,t) = \nu^-_{a,k,t-\tau_{aki}} - r_{al,t-\tau_{aki}} - \nu^+_{a,i,t} + 1 \quad (47)$$

The smallest increase in $\xi_{a,i,t}$ that would result in load l switching from vehicle type a' to vehicle type a is represented by $\delta_{3l}^+(a,k,i,t)$. The coefficient for $x_{alt-\tau_{aki}}$ in the objective function of the local problem has to be larger than the coefficient for $x_{a'l,t-\tau_{aki}}$.

$$\delta_{3l}^+(a,k,i,t) = \nu^+_{a',i,t} + r_{a'l,t-\tau_{aki}} - r_{al,t-\tau_{aki}} - \nu^+_{a,i,t} + 1 \quad (48)$$

After computing the smallest increase in $\xi_{a,i,t}$ that perturbs the current solution, the next step is to find the impact in the objective function G of doing so, $\Delta G(\Delta\xi^+_{a,i,t})$. Let $\hat{x}$ and $\hat{y}$ be the current solution of the local problem at node $(k, t-\tau_{aki})$ and $\tilde{x}$ and $\tilde{y}$ represent the solution of the local problem at the same node with $\xi_{a,i,t}$ increased by $\Delta\xi^+_{a,i,t}$. We compute the differences:

$$\bar{X}^+_{alt-\tau_{aki}} = \tilde{x}_{alt-\tau_{aki}} - \hat{x}_{alt-\tau_{aki}} \quad \forall l \in \bar{\mathcal{L}}_{k,t-\tau_{aki}} \quad (49)$$

$$\bar{Y}^+_{akjt-\tau_{aki}} = \tilde{y}_{akjt-\tau_{aki}} - \hat{y}_{akjt-\tau_{aki}} \quad \forall j \in \mathcal{C} \quad (50)$$

$$\bar{Z}^+_{akkt-\tau_{aki}} = -\sum_{l \in \bar{\mathcal{L}}_{it-\tau_{aki}}} \bar{X}^+_{at-\tau_{aki}} - \sum_{j \in \mathcal{C}} \bar{Y}^+_{aijt-\tau_{aki}} \quad (51)$$

As the increase in ξ may result in other changes due to substitution possibilities, we approximate the total change in G by a sum of the individual perturbations. The equations to approximate single perturbations have been derived in [1].

$$\Delta G(\Delta\xi^+_{a,i,t}) \simeq \sum_{j\in\mathcal{C}} \sum_{l\in \bar{\mathcal{L}}_{kjt-\tau_{aki}}} \sum_{a'\in\mathcal{A}} X^+_{a'lt-\tau_{aki}} \Big(r_{a'lt-\tau_{aki}} + \nu^+_{a',j,t-\tau_{aki}+\tau_{a'kj}} + \sum_{t'>t-\tau_{aki}} \sum_{a''\in\mathcal{A}} \hat{x}_{a''lt'} \left(-r_{a''lt'} + \nu^+_{a'',k,t'} - \nu^-_{a'',j,t'+\tau_{a''kj}} \right) \Big) + \sum_{j\in\mathcal{C}} \sum_{a'\in\mathcal{A}} Y^+_{a'kjt-\tau_{aki}} (-c_{a'kj} + \nu^+_{a',j,t-\tau_{a'ki}+\tau_{a'kj}}) + \sum_{a'\in\mathcal{A}} Z^+_{a'kkt-\tau_{aki}} \nu^+_{a',k,t-\tau_{aki}+1} \quad (52)$$

2.3.2 Decreasing the Spatial Potential Function

By using the same approach, we can find $\Delta\xi^-_{a,i,t}$, the smallest amount by which we must decrease $\xi_{a,i,t}$ so that $V_{a,i,t}$ is reduced by one unit. We must then compute δ^-_{akit} for each k, which is the smallest decrease in $\xi_{a,i,t}$ that would result in one less unit of flow type a being pushed from node $(k, t-\tau_{aki})$ to node (i,t).

$$\Delta\xi^-_{a,i,t} = \min_{k\in\mathcal{C}} \{\delta^-_{akit}\} \quad (53)$$

The decreasing of flow in this link can happen in several ways. An empty move from $(k, t-\tau_{aki})$ to (i,t) could be called off. A load destined to (i,t) that had been assigned flow could either be put on hold or reassigned another type of vehicle. Therefore

$$\delta^-_{akit} = \min\{\delta^-_1(a,k,i,t),\, \delta^-_2(a,k,i,t)\} \quad (54)$$

where

$$\delta^-_2(a,k,i,t) = \min_l \{\delta^+_{2l}(a,k,i,t)\,,\ \forall\, l \in \bar{\mathcal{L}}_{kit-\tau_{aki}} | x_{alt-\tau_{kit}} = 1\} \quad (55)$$

The smallest decrease in $\xi_{a,i,t}$ that would result in one less empty move from $(k, t-\tau_{aki})$ to (i,t) is represented by $\delta^-_1(a,k,i,t)$. Therefore the empty move has to be less attractive, i.e., have a smaller coefficient in the objective funtion, than the value of one additional vehicle of type a at node $(k, t-\tau_{aki})$. It follows that

$$\delta^-_1(a,k,i,t) = -\nu^-_{a,k,t-\tau_{aki}} - c_{aki} + \nu^+_{a,i,t} + 1 \quad (56)$$

The smallest decrease in $\xi_{a,i,t}$ that would result in load l not being assigned flow type a at node $(k, t-\tau_{aki})$ is represented by $\delta^-_{2l}(a,k,i,t)$. The coefficient for $x_{al,t-\tau_{aki}}$ in the objective function of the local problem has to be smaller than the value of one additional vehicle of type a at node $(k, t-\tau_{aki})$.

$$\delta^-_{2l}(a,k,i,t) = -\nu^-_{a,k,t-\tau_{aki}} - r_{al,t-\tau_{aki}} + \nu^+_{a,i,t} + 1 \tag{57}$$

After computing the smallest decrease in $\xi_{a,i,t}$ that perturbs the current solution, we need to find the impact in the objective function G of doing so, $\Delta G(\Delta\xi^-_{a,i,t})$. Let $\hat{x}$ and $\hat{y}$ be the current solution of the local problem at node $(k, t-\tau_{a,k,i})$ and $\tilde{x}$ and $\tilde{y}$ represent the solution of the local problem at the same node with $\xi_{a,i,t}$ decreased by $\Delta\xi^-_{a,i,t}$. We compute the differences:

$$\bar{X}^-_{al,t-\tau_{aki}} = \tilde{x}_{al,t-\tau_{aki}} - \hat{x}_{a,l,t-\tau_{aki}} \qquad \forall l \in \bar{\mathcal{L}}_{k,i,t-\tau_{k,i}} \tag{58}$$

$$\bar{Y}^-_{akj,t-\tau_{aki}} = \tilde{y}_{akj,t-\tau_{aki}} - \hat{y}_{akj,t-\tau_{aki}} \qquad \forall j \in \mathcal{C} \tag{59}$$

$$\bar{Z}^-_{akk,t-\tau_{aki}} = -\sum_{l\in\bar{\mathcal{L}}_{it-\tau_{aki}}} \bar{X}^-_{al,t-\tau_{aki}} - \sum_{j\in\mathcal{C}} \bar{Y}^-_{aij,t-\tau_{aki}} \tag{60}$$

As in the case of increasing ξ, there might be several changes in the local problem out of node $(k, t-\tau_{aki})$. The decrease in G can be approximated by

$$\begin{aligned}\Delta G(\Delta\xi^-_{a,i,t}) \simeq & \sum_{j\in\mathcal{C}} \sum_{l\in\bar{\mathcal{L}}_{kjt-\tau_{aki}}} \sum_{a'\in\mathcal{A}} \bar{X}^-_{a'l,t-\tau_{aki}} \Big(r_{a'l,t-\tau_{aki}} + \nu^-_{a',j,t-\tau_{aki}+\tau_{a'kj}} + \\ & \sum_{t'>t-\tau_{aki}} \sum_{a''\in\mathcal{A}} \frac{\partial x_{a''lt'}}{\partial V_{a',i,t}} (-r_{a'',l,t'} + \nu^-_{a'',k,t'} - \nu^+_{a'',j,t'+\tau_{a''kj}}) \Big) + \\ & \sum_{j\in\mathcal{C}} \sum_{a'\in\mathcal{A}} \bar{Y}^-_{a'kj,t-\tau_{aki}} (-c_{a'kj} + \nu^-_{a',j,t-\tau_{aki}+\tau_{a'kj}}) + \sum_{a'\in\mathcal{A}} \bar{Z}^-_{a'kk,t-\tau_{aki}} \nu^-_{a'k,t-\tau_{aki}+1}\end{aligned} \tag{61}$$

3 Algorithms

In this paper we compare two procedures for the multicommodity LQN. In the gradient approximation method, the spatial potential function for vehicles is defined as an average of the right gradient of G with respect to V (equation 24). In the LAMA method, the spatial potential function for vehicles is regarded as a control variable that must be

adjusted according to gains in the objective function. For both methods, the algorithm consists of three parts. The first part is the forward pass, consisting of solving the local problems at each terminal, starting at time $t = 0$. After the local problems for time period t are solved, the set of loads and the number of vehicles available for the next time period are computed according to equations (18) and (19). The second part consists of finding approximations for the gradients ν according to the equations presented in the previous section. As the gradients for a given time t depend on the gradients for later time periods, they are computed from the end to the beginning of the planning horizon (Backward Pass). The gradients are used in the third part to perform the control adjustment. The procedure is repeated iteratively to obtain values for the control variables that yield a solution close to optimal.

3.1 Algorithm for the Gradient Approximation Method

The objective function of the local problem is represented by equation (13). The gradient approximations are updated in every iteration and thus their presence in the objective function may lead to instability, as the gradients might change abruptly from iteration to iteration. To mitigate this effect we resort to a smoothing scheme similar to the one used in stochastic linearization algorithms (see, for example, Ermoliev [7], Gupal and Bazhenov [9]). For any iteration $n+1$ we weight the new value of ν with the one used in the previous iteration:

$$\bar{\nu}^{n+1}_{a,i,t} = \gamma \nu^{n+1}_{a,i,t} + (1-\gamma)\bar{\nu}^{n}_{a,i,t} \tag{62}$$

where γ, $0 < \gamma \leq 1$, is the smoothing factor. The effect of γ in the quality of the solutions for a range of single commodity problems was examined in [19]. It was found that $0.1 \leq \gamma \leq 0.3$ is the best range for γ values. We use $\bar{\nu}$ in the Forward Pass of the gradient method to price tasks. The objective function for the local problems in this case is then:

$$\max_{x_t, y_t} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \sum_{a \in \mathcal{A}} \left(\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}^b_{ijt}} (r_{alt} + \bar{\nu}^{+}_{a,j,t+\tau_{aij}}) x_{alt} + (-c_{aij} + \bar{\nu}^{+}_{a,j,t+\tau_{aij}}) y_{aijt} \right) \tag{63}$$

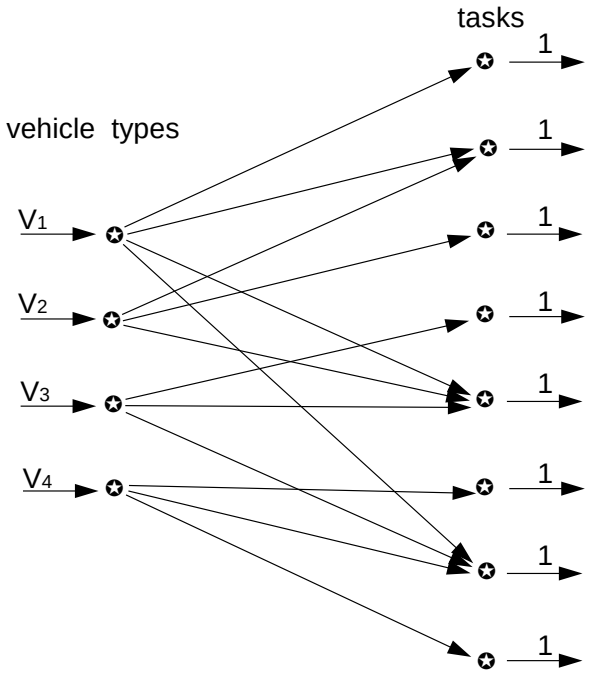

Figure 2: Local problem to be solved at each node.

In order to save CPU time we chose to employ a greedy heuristic, instead of solving each local problem to optimality. A list of feasible assignments is created and the higher valued assignments are implemented subject to the availability of vehicles.

The assignment that is solved at every local problem is illustrated in figure 2. This figure shows an assignment problem with the left side representing vehicles and the right side representing tasks available at terminal i at time t. There is one node for each vehicle type and the inflow is represented by the number of vehicles available at the node. There is one node for each task (loaded or empty move allowed by upper bound) that can be dispatched. The heuristic consists of computing the cost on each link of this assignment problem. Let l be the task and a' the vehicle type associated with the highest valued link. We validate this assignment, i.e., $x_{a',l,t} = 1$ and then update the assignment problem by removing the node associated to load l and reducing the number of vehicles of type a' available. Then we again find the highest valued link and continue until either all vehicles have been used or all tasks have been assigned vehicles.

We also use the smoothed gradients to perform the upper bound adjustment in the

gradient approximation method, so that the right gradient of G with respect to u is

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

and the left gradient is

$$\bar{\eta}^{-}_{aijt'} \begin{cases} \simeq c_{a,i,j} - \bar{\nu}^{-}_{a,j,t'+1} + \bar{\nu}^{+}_{a,i,t'} & \text{if } c_{a,i,j} - \bar{\nu}^{-}_{a,j,t'+1} + \bar{\nu}^{+}_{a,i,t'} > 0 \\ = 0 & \text{otherwise} \end{cases} \tag{65}$$

Two strategies can be used to perform the upper bound adjustment. One is to employ a gradient step. This procedure is used to start up the upper bound vector. It consists of taking a gradient step 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

$$u^{n+1} = u^n + s_n \omega_n \tag{66}$$

where the step size s_n is computed by

$$s_n = \frac{\lambda_n (G^u_n - G^l_n)}{\|\omega_n\|^2} \tag{67}$$

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. G^u_n and G^l_n are upper and lower bounds on the objective function. The best value for the objective function found up to iteration n is used as G^l_n. Whenever it is possible to solve the linear relaxation to optimality, we use the optimal value of the objective function as G^u_n. Otherwise, we use the optimal value of the commodity relaxation. Upper bounds on the objective function can also be obtained by solving a static network or simply adding up the profits of all loads in the system.

This procedure is employed in the first 50 iterations and leads to non-integer values for u. Therefore, the values obtained by using (66) are rounded to the closest integer to be used in the Forward Pass.

The other upper bound adjustment procedure is to employ a coordinate search. It consists of finding the highest value among all $\bar{\eta}^{+}_{aijt'}$ and $\bar{\eta}^{-}_{aijt'}$. In case the highest value is a member of $\bar{\eta}^{+}$, the corresponding upper bound is increased by one unit, otherwise it is decreased by one unit.

3.2 Algorithm for the LAMA Method

By solving the local problem at each node, starting from time $t = 0$ up to time $t = T$ we obtain a feasible solution to the problem. Again, as in the gradient approximation method, we do not solve each local problem to optimality, but employ a greedy heuristic. For the gradient method, the objective function of the local problem is

$$\max_{x_t, y_t} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \sum_{a \in \mathcal{A}} \left(\sum_{b \in \mathcal{B}} \sum_{l \in \mathcal{L}^b_{ijt}} (r_{alt} + \xi_{a,j,t+\tau_{aij}}) x_{alt} + (-c_{aij} + \xi_{a,j,t+\tau_{aij}}) y_{aijt} \right) \tag{68}$$

As the gradients do not appear in the objective function of the local problem in the LAMA method, there is no need to smooth the gradient approximations. The spatial potential function can be adjusted according to the equations presented in section 2.3. Initially, we adopt the same upper bound adjustment strategy used in the gradient approximation method, which is a gradient step towards the steepest ascent direction. After getting the initial values for the upper bounds, we adopt a very conservative strategy for the LAMA method in order to avoid sudden changes in the solution. At any given iteration, only one multiplier or one upper bound is adjusted.

Our preliminary runs have shown that after running many iterations it is very likely that no upper bound and no multiplier can be adjusted yielding an improvement in the objective function. This is due to our restrictive policy of small changes from iteration to iteration. We then added a perturbation step on the multipliers, arriving at the iterative procedure shown in figure 3.

The perturbation step on the multipliers consists of slightly altering their values by a small percentage of the values of the gradients ν. Whenever we get to iteration k and no upper bounds or multipliers can be adjusted improving the overall objective function, we update the multipliers for iteration $k + 1$ using:

$$\xi^{k+1}_{i,t} = (1 - \alpha)\xi^k_{i,t} + \alpha \nu^k_{i,t} \tag{69}$$

where α is the perturbation factor and we use $\alpha = 0.01$.

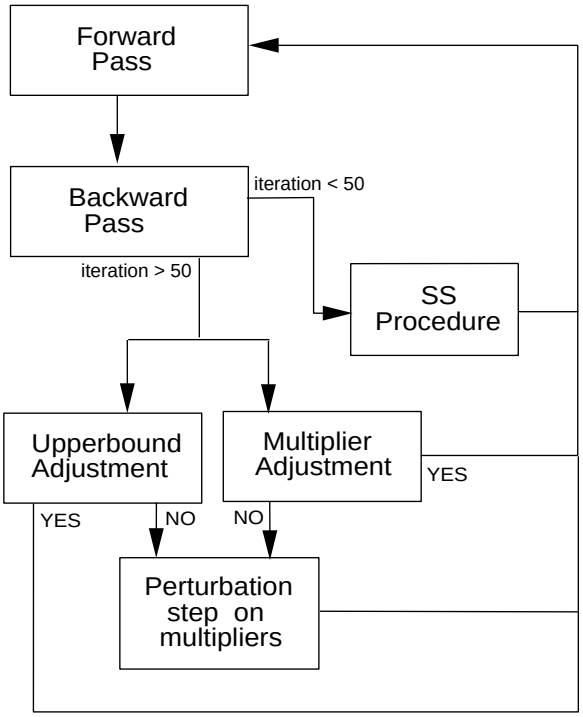

Figure 3: Iterative procedure for the LAMA method.

4 Numerical Experiments

We have produced data sets with features that resemble those found in practice. We chose a set of 40 terminals spread across the Eastern United States. The mileage between terminals was taken from a road mileage chart and the average speed of 50 mph was set regardless of vehicle type. We also set values for the empty cost per mile and the profit for each load that were independent of vehicle type.

We have initially looked at data sets that have the same number of load types and vehicle types. In order to generate the matching table, we first set all the diagonal elements $\Upsilon_{a,a} = 1$. The *substitution rate* ϕ is defined as the probability that any non–diagonal element $\Upsilon_{a,b} = 1$. Using the substitution rate we randomly generate each non–diagonal element of the matching matrix.

Each terminal has a different rate for loads being called in. In the real world certain load types tend to show up more often at a particular set of terminals. To mimic this feature, we have generated different load–type probabilities for each terminal. Let $p_b(i)$

be the probability that a load appearing at terminal i is of type b. These probabilities are generated using an exponential distribution. Some vehicle types are more useful than others. Vehicle types that satisfy types of loads that appear more often are more useful than those that satisfy types of loads that are not so common. Vehicle types that satisfy several load types are also more useful than those that do not. In order to take these two facts into account, we compute the probability for a vehicle being type a for each terminal i, $q_a(i)$, according to:

$$q_a(i) = \frac{\sum_b p_b(i)\Upsilon_{a,b}}{\sum_a \sum_b p_b(i)\Upsilon_{a,b}} \tag{70}$$

The data sets used for testing are described in table 1. Each data set has a planning horizon of five days. The departure time windows were generated according to a uniform distribution in the interval $[0, 40]$ hours. The total number of vehicles in each data set was adjusted so that the rate of loads rejected is between 85% and 90%. In order to reach the same rate of load rejection, more vehicles are needed when the substitution ratio decreases or the number of vehicle types increases.

We chose data set 1 as the standard data set. In practice, it is expected that the number of vehicle types will be larger than five. Also, real-world problems can have as many as 2000 loads per day. However, linear programming solvers take too long to solve larger problems. In order to find the optimal value of the linear relaxation of the problem, we used CPLEX with steepest edge pricing. The CPU ratio is the ratio between the CPU time for the linear programming solver and the CPU time necessary for running 500 iteration of the LQN approach. The OPT ratio is the ratio between the best objective function value obtained by using the LQN approach and the optimal value of the objective function for the linear relaxation.

Table 2 shows the influence of the substitution ratio in the OPT ratio and the CPU ratio for the gradient approximation method and for the LAMA method. As in the single commmodity case investigated in [1], the LAMA method performed slightly better at the expense of additional computation time. The better performance is due to the more stable nature of the solutions of the local problems. More computation time is the result

data set	load types	vehicle types	subtitution ratio	total loads	total vehicles
1	5	5	0.3	2000	600
2	5	5	0.1	2000	660
3	5	5	0.2	2000	630
4	5	5	0.4	2000	570
5	5	5	0.5	2000	540
6	5	5	0.6	2000	510
7	5	5	0.7	2000	480
8	5	5	0.8	2000	450
9	5	5	0.9	2000	420
10	2	2	0.3	2000	450
11	3	3	0.3	2000	500
12	4	4	0.3	2000	550
13	6	6	0.3	2000	590
14	7	7	0.3	2000	630
15	8	8	0.3	2000	670
16	9	9	0.3	2000	700
17	10	10	0.3	2000	730
18	10	5	0.3	2000	650
19	5	10	0.3	2000	650
20	12	12	0.3	2000	760
21	15	15	0.3	2000	780
22	20	20	0.3	2000	800
23	10	10	0.3	4000	1420
24	10	10	0.3	6000	2000
25	20	20	0.3	6000	2100

Table 1: Summary of problem sets used for testing.

		Gradient method		LAMA	
data set	ϕ	CPU ratio	OPT ratio	CPU ratio	OPT ratio
2	0.1	1.6	94.5	1.4	95.0
3	0.2	3.9	95.3	3.2	95.3
1	0.3	7.5	96.2	6.5	96.3
4	0.4	9.7	96.6	9.0	96.7
5	0.5	12.1	96.3	10.4	96.7
6	0.6	12.5	97.0	12.0	97.4
7	0.7	18.3	97.2	15.4	97.1
8	0.8	19.4	96.6	15.6	96.6
9	0.9	15.4	96.7	14.8	96.4

Table 2: CPU ratio and OPT ratio for different substitution ratios.

of the added procedures to compute the variations in the spatial potential functions, $\Delta\xi$, and the corresponding variation in the objective function, $\Delta G(\Delta\xi)$.

From table 2 we can also see how the substitution ratio affects the quality of the solution. The OPT ratio generally increases with the substitution ratio. Problems where there can be little substitution usually decompose into several problems with a few vehicle types in each. Therefore, better results for the cases that have low substitution ratio could be obtained by finding a decomposition of the matrix Υ.

While the solution time of the linear programming relaxation increases with the substitution ratio, the solution time using the LQN approach does not depend on it. High substitution rates lead to very degenerate linear programs. Thus the CPU ratio increases with the increase of the substitution ratio.

In real world problems we expect to find subtitution matrices that are not very dense. Therefore the substitution ratio was kept at $\phi = 0.3$ for the other data sets. In table 3 we compare problems that have square substitution matrices of different sizes. Once again the LAMA method is shown to yield slightly better solutions at the expense of more computation time. There is only a slight degradation in the quality of the solution obtained using the LAMA method when the number of vehicle and load types increases.

		Gradient method		LAMA	
data set	commodities	CPU ratio	OPT ratio	CPU ratio	OPT ratio
10	2	3.3	96.6	3.0	97.0
11	3	4.7	95.4	3.9	95.3
12	4	7.5	96.0	6.6	95.9
1	5	7.5	96.2	6.5	96.3
13	6	11.3	96.1	9.3	96.4
14	7	11.6	95.6	9.5	95.7
15	8	17.8	96.4	15.2	96.7
16	9	15.4	96.0	12.6	96.6
17	10	21.1	95.7	17.1	96.2

Table 3: CPU ratio and OPT ratio for problems with square substitution matrices of different sizes.

data set	1	18	19	17
vehicle types	5	5	10	10
load types	5	10	5	10
CPLEX cpu (s)	13,456	11,896	64,353	61,402
LAMA cpu (s)	2,061	2,165	3,519	3,586
CPU ratio	6.5	5.5	18.3	17.1
OPT ratio	96.3	95.5	96.6	96.2

Table 4: Comparison among data sets with different number of vehicle and load types.

The platform used for the experiments did not have enough memory to store the linear programs generated for data sets larger than data set 17. In order to solve the linear relaxation for data set 17, more than 60,000 CPU seconds were necessary. For these two reasons, solving the linear relaxation for larger problems is not practical.

So far we have looked at data sets that have the same number of vehicle types and load types, i.e., with square matching matrices. In order to investigate how each dimension of the matching matrix is the dominant factor on the performance of the algorithms, we use data sets 18 and 19. Table 4 shows the OPT ratio and the CPU ratio for the LAMA method applied these data sets, and compared to data sets 1 and 17.

data set	OF linear relaxation	OF commodity relaxation	gap %
14	264,147	265,091	0.36
15	259,901	260,064	0.06
16	252,681	252,730	0.02
17	254,754	254,889	0.05

Table 5: Optimal objective function values (OF) for the linear relaxation and the commmodity relaxation.

The number of vehicle types is the dominant parameter on the amount of computation involved, not only for the LAMA method, but also for the linear program. This is in fact expected as both the LQN approach and the linear program treat each load individually with its specific time window, whereas vehicles are grouped into types. Also, in the LQN approach, we have different gradients for each vehicle type at a given node. By varying the number of load types, we do not increase the number of gradients that have to be computed. Thus, the number of vehicle types is the crucial parameter that affects the computational complexity of the problem.

For data sets with more than 10 commodity types, we resort to the commodity relaxation of the problem in order to find an upper bound for the objective function. In this relaxation, the matching restrictions are ignored and the problem is solved as if there were only one type of commodity. In order to probe the tightness of this bound, table 5 compares the bounds obtained by the linear relaxation and the commodity relaxation for some of the mid–size data sets. We define the *CR ratio* as the ratio between the objective function value obtained by the LQN approach and the optimal value for the objective function of the commodity relaxation. Three of the four datasets showed gaps of less than one tenth of a percent, while one of the datasets (number 14) produced a gap of 0.36 percent. All of these gaps are quite narrow, although we have no particular explanation why dataset 14 was noticeably larger than the others. The narrowness of the gap indicates that relaxing the constraints on the commodities produces a tight bound on the optimal objective function, which otherwise would not be computable for

data set	CR ratio	CPU time (s)
20	96.5	3838
21	96.2	4924
22	96.4	6271
23	97.7	5866
24	96.2	9639
25	96.9	18925

Table 6: CR ratio and CPU time for the LAMA method applied to large data sets.

the larger problems. For example, the linear program for the linear relaxation of data set 25 has around 30,000 constraints and 600,000 variables.

On average, we found an optimality gap of 3.5%. There are several reasons for this gap. First, the objective function value used in the denominators of the OPT ratio and the CR ratio are not the true optimal values but the result of relaxations. We expect these bounds to be reasonably tight. Linear relaxation gaps of up to 0.2% have been reported in [12] for similar problems of smaller size. We report in table 5 an additional gap of up to 0.3% for the CR ratio. Second, the local problems were not solved to optimality, but a feasible solution was found using a greedy heuristic. Third, some of the gap is due to the gradient approximations. The gradient estimation fails to capture acurately the interactions between perturbations that result of vehicle type substitution. Finally, we attribute most of the optimality gap to the coordinate search used to update the upper bounds. This procedure is limited in its quest to look for local improvement. However, the search for improvement trees can turn very complicated and time consuming, and thus we have not investigated this strategy.

5 Conclusion

We extend the logistics queueing network approach to multicommodity network flow problems. We obtain integer solutions that are on average within 3.5% of the optimal value of the linear relaxation of the problem. On larger problems where the linear

relaxation cannot be solved, we obtain integer solutions that are within 3.5% of the optimal value of the commodity relaxation of the problem.

Compared to a linear solver, the LQN approach becomes much faster for larger problems. Our implementation of the algorithms was tuned to perform well on problems with few commodity types and dense matching matrices. Improved computation times on data sets with many commodity types and sparse matching matrix can be obtained by using customized implementation. Another way to cut down the computation time is to do a parallel implementation of the algorithm. The planning horizon can be divided in several sections and one processor can be assigned to each section. The processors share information on the gradient approximations and vehicle supply but can work asynchronously.

The number of vehicle types is one of the key parameters that affect computational complexity and thus CPU times. From a modeller perspective, it is desirable to consolidate vehicle types as much as possible. The number of load types was shown not to affect computation requirements.

The LQN approach has several advantages over linear programming formulations. It provides integer solutions. It allows for considering real-world constraints inside each local problem that cannot be modeled as linear constraints. It provides solutions much faster for the large problems found in the real-world.

The multicommodity LQN approach can be used not only for real time dispatching but also to answer a variety of questions on strategical planning. One area of interest is intermodal operations where rail, sea lines and trucks form complementary networks and compete for service on some of the links. Given resource availability, the LQN approach can be used to guide answers on matters like fleet renewal and facility location. This can be achieved by running a variety of scenarios on a single model.

Future research can focus on problems involving consolidation. Such is the case when intermodal operations involve railcars that can transport more than one container. Also, the stochastic nature of fleet management problems can be explicity considered in the

model.

The assumption of multiple vehicle and load types adds considerable complexity to the problem of dynamic resource allocation. In spite of this, we have shown that the LQN approach can handle very big problems and provide near-optimal integer solutions.

References

[1] T. Carvalho and W.B. Powell. A multiplier adjustment method for dynamic resource allocation problems. Report 96-03, Department of Civil Engineering and Operations Research, Princeton University, 1995.

[2] 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*, (to appear), 1994.

[3] 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.

[4] I.C. Choi and D. Goldbarb. Solving multicommodity network flow problems by an interior point method. *SIAM Proceedings in Applied Math*, 46:58–69, 1990.

[5] J. Desrosiers, M. Solomon, and F. Soumis. Time constrained routing and scheduling. In C. Monma, T. Magnanti, and M. Ball, editors, Handbook in Operations Research and Management Science, *Volume on* Networks. North Holland, 1995.

[6] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14:545–565, 1984.

[7] Y. Ermoliev. Stochastic quasigradient methods. In Y. Ermoliev and R. Wets, editors, *Numerical Methods in Stochastic Programming*. Springer-Verlag, 1988.

[8] 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.

[9] A. M. Gupal and L. G. Bazhenov. A stochastic method of linearization. *Cybernetics*, pages 482–484, 1972.

[10] Herren H. The distribution of empty wagons by means of computer: An analytical model for the swiss federal railways (ssb). *Rail International*, 4(1):1005–1010, 1973.

[11] Herren H. Computer controlled empty wagon distribution on the ssb. *Rail International*, 8(1):25–32, 1977.

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

[13] M. Joborn. Empty freight car distribution at swedish railways - analysis and optimization modeling. Ph.d. thesis, Department of Mathematics, Linkoping University, Sweden, 1995.

[14] 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.

[15] W.C. Jordan and M.A. Turnquist. A stochastic dynamic network model for railroad car distribution. *Transportation Science*, 17:123–145, 1983.

[16] 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.

[17] M.C. Pinar and S.A. Zenios. Parallel decomposition of multicommodity network flows using smooth penalty functions. Technical report, University of Pennsylvania, The Warton School, 1990. Technical Report 90-12-06.

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

[19] W.B. Powell and T. Carvalho. Dynamic control of logistics queueing network for large-scale fleet management. Report 96-01, Department of Civil Engineering and Operations Research, Princeton University, 1996.

[20] G.L. Schultz and R.R. Meyer. An interior point method for block angular optimization. *SIAM Journal on Optimization*, 1(4), 1991.

[21] Y Shan. A dynamic multicommodity network flow model for real-time optimal rail freight car management. Ph.d. dissertation, Princeton University, 1985.

[22] M.A. Turnquist. Mov-em: A network optimization model for empty freight car distribution. School of Civil and Environmental Engineering, Cornell University, 1986.

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

[24] W.W. White and A.M. Bomberault. A network algorithm for empty freight car allocation. *IBM Systems Journal*, 8(2):147–171, 1969.