THE SINGLE-NODE DYNAMIC SERVICE SCHEDULING AND DISPATCHING PROBLEM

Leonardo Campo Dall'Orto
Vale do Rio Doce
and
Department of Industrial Engineering
Pontifícia Universidade Católica do Rio de Janeiro

Teodor Gabriel Crainic
Department of Management and Technology
École des sciences de la gestion
Université du Québec à Montréal
and
Centre for Research on Transportation, Montréal

José Eugenio Leal
Department of Industrial Engineering
Pontifícia Universidade Católica do Rio de Janeiro

Warren B. Powell
Department of Operations Research and Financial Engineering
Princeton University

June 10, 2004

Abstract

In this paper, we focus on a particular version of the Dynamic Service Network Design (DSND) problem, namely the case of a single-terminal that dispatches services to a number of customers and other terminals. We present a time-dependent, stochastic formulation that aims to optimize the problem over a given planning horizon, and propose a solution approach based on dynamic programming principles. We also present a static, single-period, formulation of the single-node problem that appears as a subproblem when addressing the time-dependent version and general service network design cases. Despite its apparent simplicity, it is still a network design problem and exact solution methods are not sufficiently fast. We therefore propose two tabu search meta-heuristics based on the ejection-chain concept. We also introduce a learning mechanism that takes advantage of experience gathered in repeated executions. Experiments with problem instances derived from real cases indicate that the proposed solution methods are efficient and yield good solutions.

Keywords: dynamic service network design, stochastic formulation, approximate dynamic programming, tabu search, ejection chains.

Résumé

Dans cet article, nous examinons un cas particulier du problème de conception dynamique du réseau de service : le cas d'un terminal qui envoie des services de transport vers des clients ou d'autres terminaux. Nous présentons une formulation dynamique et stochastique qui vise à trouver une solution optimale au problème pour l'ensemble de l'horizon de planification, ainsi qu'une approche de résolution basée sur des principes de programmation dynamique. Nous examinons également la formulation statique du problème, définie pour une seule période, qui apparaît comme sous-problème dans le modèle dynamique, ainsi que lors de la résolution de problèmes généraux de conception de réseaux dynamiques. En dépit de son apparente simplicité, ce dernier problème appartient à la classe de formulations de conception de réseau et les méthodes exactes de résolution n'offrent pas de performances satisfaisantes. Nous proposons donc des méta-heuristiques de recherche avec tabous, basées sur des chaînes d'éjection. Nous proposons également un mécanisme d'apprentissage qui accumule l'information au cours d'exécutions répétées. Les résultats d'expérimentations à l'aide de cas dérivés de données réelles indiquent que les méthodes proposées sont efficaces et fournissent des solutions de bonne qualité.

Mots clefs : conception de réseau de service dynamique, formulation stochastique, programmation dynamique, recherches avec tabous, chaînes d'éjection.

INTRODUCTION

Less-than-truckload (LTL) motor carriers, railways, and ocean container lines handle the flows of small shipments by consolidating them on common vehicles that must be dispatched over time. In the case of LTL carriers (and to a lesser extent rail), it is necessary to dynamically dispatch vehicles to move shipments that arrive according to a potentially random, and nonstationary, process. The decision to dispatch a vehicle must trade off the cost of holding freight that has already arrived against the cost of dispatching a vehicle that is not quite full. When these dispatches occur in the context of larger networks, we must also consider the cost of moving shipments from the destination of the vehicle to the final destination of each shipment.

Service Network Design (*SND*) is often associated with the class of problems that address the issue of determining the set of services and characteristics that optimize these goals. *Dynamic*, or time-dependent, *Service Network Design* (*DSND*) arises when the schedule is a function of time. SND problems are NP-Hard and are generally represented as mixed-integer multi-commodity, capacitated, network optimization formulations (Magnanti and Wong 1986; Minoux 1986; Balachrishnan, Magnanti, and Mirchandani 1997; Crainic 2000). Crainic (2000, 2003; see also Crainic and Laporte 1997 and Crainic and Kim 2004) provides a general view of these problems and surveys the main methodological contributions to the area. Cordeau, Toth, and Vigo (1998) present a review of the most recent contributions dealing with train routing and scheduling with regard to both freight and passenger transportation. Christiansen *et al.* (2004) address the same issues for maritime transportation. These problems exhibit very weak LP relaxations, which cripple any algorithms that depend on branch and bound. As a result, the field is dominated by various heuristics (e.g., Armacost, Barnhart, and Ware 2002, Barnhart, Jin, and Vance 2000; Barnhart and Schneur 1996; Kim *et al*. 1999; Crainic, Ferland, and Rousseau 1984; Crainic and Rousseau 1986; Crainic and Roy 1988; Grünert and Sebastian 2000; Powell 1986; Powell and Sheffi, 1989). Dynamic network design problems are even harder, and as a result have received even less attention (e.g., Haghani 1989; Farvolden and Powell 1994; Equi *et al*. 1997; Gorman 1998). To our knowledge, stochastic versions of the network design problem have not been addressed in the literature.

There have been separate lines of investigation into special cases of the network design problem. There is a history of research into the single-link problem, which involves modeling the control of a batch dispatch process of a vehicle from one node to another. Kosten (1967, 1973) was the first to consider the control of a single server stochastic, batch service problem in steady state. Ignall and Kolesar (1972), Deb and Serfozo (1973), and Powell and Humblet (1986) made further contributions to this problem class. Speranza and Ukovich (1992, 1994, and 1996) present deterministic formulations of the single link problem. Burns *et al.* (1984), Blumenfeld *et al.* (1985), and Daganzo (1991) investigate properties of the single link problem in the context of strategies in shipper logistics. Bertazzi and Speranza (1999) use approximate dynamic programming methods to solve a discrete state version of the stochastic problem with a small number of product types. Bertazzi and Speranza (2002) present shipping strategies for several products on a capacitated link, such that the sum of transportation and inventory costs is minimized; The discrete and continuous transport frequencies cases are derived from the general analysis framework. Papadaki and Powell (2002) exploit the monotone structure of the value function to estimate a discrete version of the value function for a single commodity. Papadaki and Powell (2003) introduce linear approximations and nonlinear, concave approximations and show that they produce much faster convergence and can be easily applied to problems with a large number of shipment types.

We propose to extend the basic strategy used in Papadaki and Powell (2003) to the problem of a single node, with vehicles departing over a set of outbound links to different terminals. We assume that there is a set of shipments waiting to be dispatched to various destinations that might serve as intermediate transfer points. Thus, the destinations of the vehicles will generally be a small subset of the set of destinations of shipments. This also introduces the dimension that we have to decide which outbound vehicle a shipment should be assigned to, recognizing that there are both immediate and downstream costs to be considered in such a decision. Our strategy is to formulate a precise version of the problem faced at a single node, using approximations of downstream costs. This view is a realistic model of the dispatching process of large less-than-truckload networks, where terminal managers make decisions using a high level of information about their own terminal (at the point in time that a decision is made), with approximations of downstream impacts (the cost of sending a shipment with final destination k on a truck whose destination is j). This paper directly extends this line of research to single-node problems, and represents, we believe, a potential stepping stone to a strategy that can be applied to large-scale networks.

A realistic dispatch problem from a single node might involve dispatching trucks to several dozen destinations. Given the relative ineffectiveness of LP relaxations in branch and bound algorithms for this problem class, it is computationally intractable to formulate even deterministic versions of this problem over multiple time periods. Our strategy is to solve the single-node, multiple time period problem using approximate dynamic programming methods, where the impact of decisions now on the future is approximated using linear approximations. The use of continuous approximations for value functions has been studied for years (see, for example, Tsitsiklis and van Roy 1997; Pflug 1996) but only recently has been recognized for its ability to allow dynamic programming techniques to be applied to high dimensional problems (Powell and Carvalho 1997, 1998; Powell and Topaloglu 2003; Powell and van Roy, 2004), most recently in this specific problem class (Papadaki and Powell 2003).

Even with these techniques, the single-period problem (which must be solved very quickly) remains computationally challenging. A single node at a single point in time may face several dozen outbound links, creating an integer programming problem with several dozen integer variables. Without a reasonable bound, branch and bound would virtually enumerate the entire tree. Our approximation strategy requires that many of these problems be solved, so speed is essential. We therefore propose a tabu search meta-heuristic (Glover 1989; Glover and Laguna 1997) based on the concept of ejection chains. We also introduce a learning mechanism that goes beyond the usual within-the-search-trajectory memories to take advantage of experiences gathered in repeated executions. Experiments with problem instances derived from real cases indicate that the proposed solution methods are efficient and yield good solutions.

The paper is organized as follows: Section 1 presents the problem and introduces the time-dependent formulations and the solution algorithm. Section 2 describes the single-period formulations, the tabu search meta-heuristics, and the long-term learning mechanism. Section 3 presents the experimental setting and the result analysis. We conclude in Section 4.

1 THE SINGLE-NODE MULTIPERIOD SERVICE NETWORK DESIGN PROBLEM

In the single-node-service design problem, shipments must be sent from a terminal to customers using the services available at the terminal. Services could be direct or not. Direct services link the origin directly to the destination, while the others use intermediate terminals to consolidate cargo. A "*fixed*" cost must be paid to operate a given service (to dispatch a vehicle), independently of the actual load. Each direct service has a fixed capacity. Transportation costs proportional to the ship-

ments carried may also be involved. Shipments can be kept at the terminal, incurring a holding cost that reflects the service requirements of each individual shipment. We must make the decision at time *t* to determine whether or not to dispatch a vehicle to each of a set of intermediate transfer points, and which shipments should go on each vehicle. A shipment that is put on a vehicle that takes it to an intermediate transfer terminal also incurs a cost to move the shipment from the intermediate transfer terminal to the final destination. For this paper, this downstream cost is assumed known. Our goal is to minimize the sum of dispatch costs, holding costs at the origin terminal, and the downstream costs for shipments arriving at an intermediate terminal. Costs are minimized over a finite planning horizon.

When the scheduling (or the dispatching) of services is contemplated, a *time* dimension must be introduced in the DSND formulations. This is usually achieved by representing the operations over a certain number of time periods by using a *space-time* network. This increases the size of the problem and also makes it more computationally complex, since the model must incorporate the impact of decisions taken in one period on decisions taken in later periods.

The general approach is as follows. The representation of the physical network, of terminals in particular, is replicated in each period. Services are generated on temporal links. Starting from its origin in a given period, a service arrives (and leaves, in the case of intermediary stops) later at other terminals. Each service leg thus represents a service link between two different terminals at different time periods. Links between nodes representing the same terminal at two consecutive periods are used to model holding decisions. Figure 1 illustrates this concept through a network with four terminals and several services offered in each terminal. Services S1, S2, and S3 originate at terminal A. Services S1 and S2 are direct, while S3 uses terminal C as an intermediate terminal for consolidation operations. Similarly, service S4 initiates at terminal B, S5 from C, and S6 from D. Since terminals and services are replicated over time, the departing period is indicated (e.g., service S1,1 stands for S1 leaving terminal A at time 1 and arriving at terminal D at period 2). The option of holding freight at the terminal is illustrated by the links that connect the same terminal in subsequent periods.

Two types of decision variables are usually defined. Integer design variables are associated with each service. Restricted to {0, 1} values, these variables indicate whether or not the service will be offered at the specified time. When several departures may take place in the same period, either general (nonnegative) integer variables are used or the length of the period is redefined. Continuous variables are used to represent the distribution of freight flows through this service network. Additional decision variables may be introduced to represent the flow of empty vehicles in order to ensure the balance of vehicle flows at terminals. The resulting mixed-integer formulation minimizes the sum of service fixed costs, holding costs, and downstream transportation costs. It considers constraints such as flow conservation at nodes, capacity on service links, and general routing and frequency constraints. The formulations we present in this paper follow these general principles.

The single-node multiperiod service network design problem may be viewed as a special case of the general dynamic case with only one source node. Although this setting appears to simplify the formulation and the solution methods, the problem still exhibits strong combinatorial features and most difficulties related to solving network design problems are also encountered in this case. To simplify the presentation, we describe the problem and models in terms of less-than-truckload motor carrier transportation. The developments are general, however, and may be applied to any consolidation transportation case.

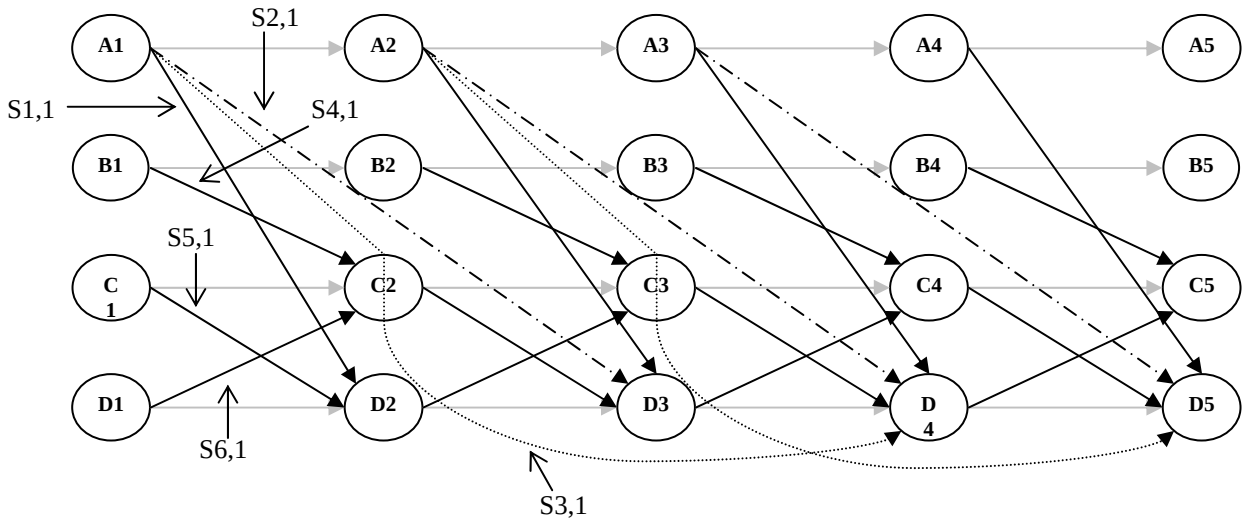

Figure 1 — A Dynamic Service Network

The problem represents the decision of a dispatcher of a truck terminal at a given moment. The dispatcher has to determine the best strategy to move freight from the terminal to its destination, selecting between i) sending in the current period, either directly or by using some intermediate terminals, or ii) holding the freight in inventory for shipment at a later period. A fixed cost is incurred when a service is selected (following a dispatching decision). Each service has a finite capacity representing the maximum load the service can carry. Holding costs apply to freight kept for future periods.

We present both a deterministic and a stochastic formulation of the problem. An algorithm is also described for the latter. The two formulations are experimentally contrasted in Section 3.

1.1 The deterministic model

The problem is illustrated in Figure 2 for periods $t = 1,\ldots,T$. All shipments are waiting at a single source node (which is not indexed to simplify notation). Let:

K_t = The set of destination nodes for shipments waiting at time *t*.

$\hat{R}_{kt}$ = The number of new arrivals of shipments with destination k arriving to the system at time *t*.

$\hat{R}_t = \left(\hat{R}_{kt}\right)_{k \in K_t}$

We define our state variable using:

R_{kt}^x = The number of shipments with destination k remaining in the system at the end of period *t*, after the vehicles have departed.

$R_t^x = \left(R_{kt}^x\right)_{k \in K_t}$

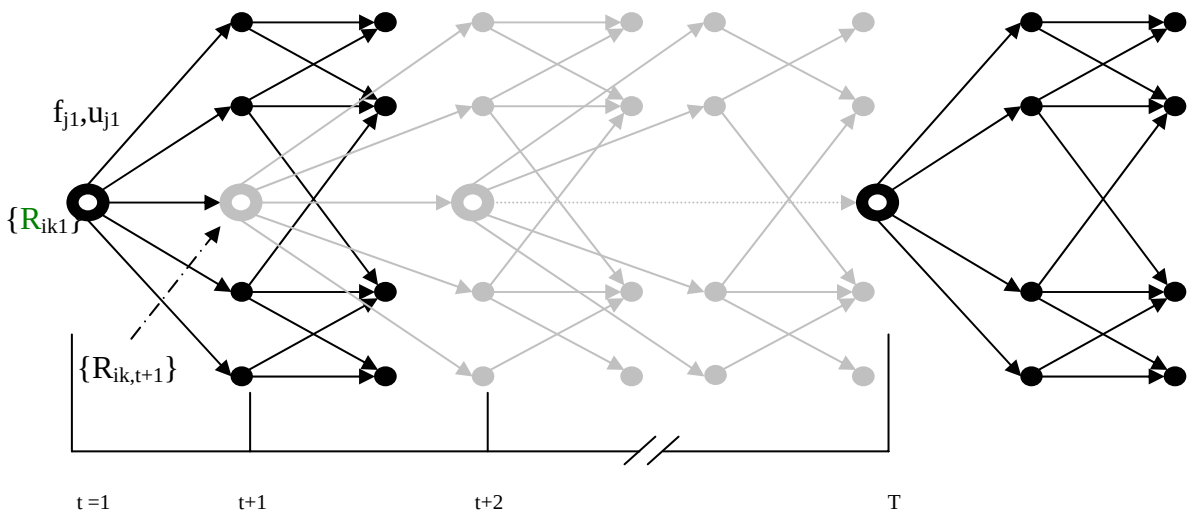

Figure 2 - The Single-Node Multiperiod Service Network Design Diagram

The vector R_t^x is known as the *post-decision* state variable (the state of the system after decisions have been made, hence the superscript), which plays an important role in the development of approximations for stochastic versions of the problem. R_t^x is also the vector of shipments available at the beginning of time *t+1* before new arrivals are added in. We index the variable by *t* (instead of *t+1*) because, in the stochastic version, it indicates the *information content* of the variable. We adopt this version of the state variable here to maintain consistency. The total number of shipments that need to be moved in period *t* is given by the *pre-decision* state variable, which we write $R_t = R_{t-1}^x + \hat{R}_t$.

Shipments may be moved either directly to their final destinations or to an intermediary terminal, which will then be responsible for the next dispatch. A *service* is a direct movement of a vehicle from the source to another terminal. Let:

J_t = The set of destinations to which vehicles may be sent at time *t*.

The decision variables are given by:

$$y_{jt} = \begin{cases} 1 & \text{If a vehicle is dispatched from the source node to node j, departing at time t;} \\ 0 & \text{Otherwise.} \end{cases}$$

x_{jkt} = The number of shipments with final destination k to be moved on a vehicle going to terminal j at time *t*.

Once a shipment has arrived at the transfer terminal *j*, we then approximate the cost of the remainder of the trip by a linear function. Shipments that are not dispatched to a terminal in J_t are held. The costs we wish to minimize are given by:

f_{jt} = The fixed cost of sending a vehicle to destination j at time *t*;
g_{jkt} = The unit cost of moving a shipment from intermediate terminal j to final destination k when it is dispatched to j at time *t*;
h_{kt} = The unit cost of holding at origin at time t a shipment with ultimate destination *k*.

The holding costs h_{kt} can be used to capture differences in the value of the shipments. The inventory equation can now be written as

$$R^x_{kt} = R^x_{k,t-1} + \hat{R}_{k,t} - \sum_{j \in J_t} x_{jkt} = R_{kt} - \sum_{j \in J_t} x_{jkt} .$$

Let u_{jt} represent the service capacity to destination j departing at time t. The deterministic multiperiod node service network design formulation may then be written as follows:

$$\text{Minimize } Z(x,y) = \sum_{t=1}^{T} \left[\sum_{j \in J_t} (f_{jt} y_{jt} + \sum_{k \in K_t} g_{jkt} x_{jkt}) + \sum_{k \in K_t} h_{kt} (R_{kt} - \sum_{j \in J_t} x_{jkt}) \right] \qquad (1)$$

$$\text{Subject to: } \sum_{k \in K_t} x_{jkt} \le y_{jt} u_{jt} \qquad \forall j \in J_t, t = 1, \ldots, T \qquad (2)$$

$$\sum_{j \in J_t} x_{jkt} \le R^x_{k,t-1} + \hat{R}_{kt} \qquad \forall k \in K_t, t = 1, \ldots, T \qquad (3)$$

$$R^x_{kt} = R^x_{k,t-1} + \hat{R}_{kt} - \sum_{j \in J_t} x_{jkt} \qquad \forall k \in K_t, t = 2, \ldots, T \qquad (4)$$

$$x_{jkt} \ge 0 \qquad \forall j \in J_t, k \in K_t, t = 1, \ldots, T \qquad (5)$$

$$y_{jt} = \{0,1\} \qquad \forall j \in J_t, t = 1, \ldots, T \qquad (6)$$

Equation (1) captures the cost of moving vehicles and shipments, as well as keeping shipments in storage, over the planning horizon. Equation (2) is the usual linking (or knapsack) restrictions on using only open services that also double as capacity constraints. Equation (3) is the demand constraints indicating the maximum flow possible out to a given terminal in any period, while equations (4) capture the dynamics of the system. Equations (5) and (6) are the usual non-negativity and integrality (on design decisions) constraints on decision variables.

The multi-period problem belongs to the family of fixed cost, capacitated, multicommodity network design problems that is known to be NP-hard and have proved difficult to solve for realistic problem instances. Consequently, heuristics are used in most cases and, as illustrated in Sections 2 and 3, this solution strategy is also used here.

Load arrivals are generally not deterministic, however. Consequently, we reformulate the problem to reflect the stochasticity of the demand and derive a solution strategy based on approximate dynamic programming techniques. The resulting procedure requires solving several instances of the deterministic single-period variant of formulation (1) – (6). Section 1.2 is dedicated to the presentation of this model and the corresponding solution method.

1.2 The stochastic problem

Compared to the previous deterministic model, the new formulation considers explicitly the arrival over time of new information. The most common form of new information is the customers arriving to be served, but we can also allow for new information about transportation costs and holding costs. In transportation, “holding costs” typically represent a penalty that measures the quality of service. For example, the carrier could get a request to rush a shipment, or learn that a shipment is actually

arriving early (which can be a problem in some situations). We solve the resulting stochastic model by formulating a stochastic optimization problem over time, which we can approach using dynamic programming. Our development follows and generalizes the work of Papadaki and Powell (2002, 2003).

We let W_t be the vector of random variables representing all the information arriving during time interval *t*. W_t includes information about new customer arrivals and all costs, and can be written as $W_t = \left(\hat{R}_t, f_t, g_t, h_t\right)$, which means that the arrivals $\hat{R}_t$ and costs $\left(f_t, g_t, h_t\right)$ first become known during time interval *t*. W_t represents exogenous information.

We now adopt the convention that any variable indexed by time *t* is a function of the information up through time *t*. This means that at time *t*, any variable indexed by *t* is deterministic, while any variable indexed by *t' > t* is random.

Let Ω be the set of elementary outcomes of the information process, with $\left(W_t(\omega)\right)_{t=1}^{T}$ representing a sample realization of the information. To define a formal probability space, we let $\mathcal{F}$ be the set of outcomes of W_t (more formally, the $\sigma-$algebra generated by W_t) and let $\mathcal{P}$ be a probability measure on $\left(\Omega, \mathcal{F}\right)$. Then, our probability space is $(\Omega, \mathcal{F}, \mathcal{P})$. We would then let $\mathcal{F}_t$ be the $\sigma-$algebra generated by the history $\left(W_s\right)_{s=1}^{t}$. By construction, all functions indexed by *t* are $\mathcal{F}_t$-measurable.

Our problem consists of determining the dispatch policy that minimizes the total expected costs over time, that is, the policy that achieves the best trade-off between the dispatch and the inventory costs.

Let $C_t(x_t, y_t \mid R_t)$ be the one-period cost function given the pre-decision state R_t :

$$C_t(x_t, y_t \mid R_t) = \sum_{j \in J_t}\left(f_{jt} y_{jt} + \sum_{k \in K_t} g_{jkt} x_{jkt} \right) + \sum_{k \in K_t} h_{kt} R^x_{kt} \tag{7}$$

Note that the cost function depends on the pre-decision state variable R_t, but the holding costs are assessed only on the shipments left over at the end of the time period, given by R^x_t.

Decisions (*x*,*y*) are made according to a *policy* π. We represent the *decision functions* using:

$X^{\pi}_t(R_t)$ = The flow decision function, which returns *x* at time *t* given state R_t;

$Y^{\pi}_t(R_t)$ = The dispatch decision function, which returns *y* at time *t* given state R_t.

Let Π be our family of decision functions. The objective function can now be formulated as:

$$F^{\pi}_t(R_t) = E\left\{ \sum_{t'=t}^{T} C_{t'}(X^{\pi}_{t'}, Y^{\pi}_{t'} \mid R_{t'}) \right\}$$

The optimization problem, now, is to find the best policy, which is to say that we wish to solve:

$$F^*_0 = \inf_{\pi \in \Pi} \left\{ F^{\pi}_0(R_0) \right\}.$$

Given the sheer complexity of solving this problem exactly, we resort to a search for "good" policies. We start by writing the problem in terms of the Bellman optimality equations. Given a pol-

icy π, each outcome $\omega \in \Omega$ produces a specific sequence of state variables $\left(R_t(\omega)\right)_{t=0}^{T}$. We next define:

$V_t^{\pi}(R_t^x)$ = The future expected costs given that we start in period t in state R_t^x and follow policy π until the end of the horizon.

Since we are using the post-decision state variable, the value functions $V_t^{\pi}(R_t^x)$ are defined using a nonstandard form of the Bellman equations:

$$V_{t-1}^{\pi}(R_{t-1}^x) = E\left\{\min_{y_t, x_t} C_t(y_t, x_t \mid R_t) + V_t^{\pi}\left(R_t^x(y_t, x_t)\right) \middle| R_{t-1}^x\right\} \tag{8}$$

The value function V_t^{π} is too complex, however, and cannot be computed exactly. The challenge is to find an approximation $\bar{V}_t(R_t^x)$ that is computationally tractable and still represents in a satisfactory way the impact of the decisions taken in the period of decision over time. If we are in state R_{t-1}^x, we can sample ω (which determines $\hat{R}_t$ as well as the costs in period *t*). Then, using an approximation $\bar{V}_t(R_t^x)$, we can find the decisions in period *t* using:

$$\left(Y_t^{\pi}(R_{t-1}^x, \hat{R}_t(\omega)), X_t^{\pi}(R_{t-1}^x, \hat{R}_t(\omega))\right) = \arg\min_{y_t, x_t} C_t\left(y_t, x_t \mid R_t(\omega)\right) + \bar{V}_t\left(R_t^x(y_t, x_t, \omega)\right) \tag{9}$$

A critical feature of (9) is that given $(R_{t-1}^x, \hat{R}_t(\omega))$, the post-decision state variable R_t^x is a deterministic function of x_t and y_t.

We see, then, that an approximation $\bar{V}_t(R_t^x)$ determines a policy. Papadaki and Powell (2002, 2003) evaluate three types of approximations when R_t^x is a scalar: discrete, linear, and nonlinear (concave). If R_t^x is a scalar, equation (8) may be solved optimally. The results show that the linear and nonlinear approximations converge much more quickly than the discrete approximation, with results within one to two percent of optimality. The nonlinear approximation outperforms the linear, but only after hundreds of iterations. The linear approximation works well because the function $\bar{V}_t(R_t^x)$ is monotone. Not surprisingly, the linear approximation works especially well when the underlying problem is stochastic. For our application, we will generally be able to run only a few dozen iterations, and hence the linear approximation offers not only simplicity, but also works the best among the approximations that have been tested.

Equation (9) also illustrates the importance of using the post-decision state variable for stochastic problems. When R_t^x is a scalar, the expectation in equation (8) is generally easy to compute. When R_t^x is a vector, the expectation is computationally intractable. In equation (9), we took a sample realization and computed $\left(Y_t^{\pi}(R_{t-1}^x, \hat{R}_t(\omega)), X_t^{\pi}(R_{t-1}^x, \hat{R}_t(\omega))\right)$, where the decisions are allowed to "see" $R_t = R_{t-1}^x + \hat{R}_t$. Had we used a pre-decision state variable, the same trick would have required finding the decisions in period *t* allowing the decisions to see R_{t+1}, which is a violation of the information constraint on the decision function. This trick of solving approximations of the Bellman equations by formulating the optimality equations around the post-decision state variable, combined with the use of appropriate functional forms for the value function, has been the basis of work on large-scale fleet management problems (Godfrey and Powell 2002a, 2002b) and multiproduct batch

service problems (Papadaki and Powell 2003). The general idea is summarized in Powell and Topaloglu 2003 and Powell and van Roy 2004.

Let $\overline{v}_t = \{\overline{v}_{kt}\}_k$ be the vector of smoothed estimates of the unit value of shipments for destination *k* at time *t* before new arrivals. A linear approximation of the value function can then be written

$$\overline{V}_t(R_{t-1}^x, \hat{R}_t(\omega)) = \overline{v}_t R_t^x = \sum_{k\in K_t} \overline{v}_{kt} R_{kt}^x = \sum_{k\in K_t} \overline{v}_{kt} \left(R_{k,t-1}^x + \hat{R}_{k,t}(\omega) - \sum_{j\in J_t} x_{jkt} \right) \tag{10}$$

Substituting (10) into (9) gives

$$\begin{aligned} &\left(Y_t^\pi(R_{t-1}^x, \hat{R}_t(\omega)), X_t^\pi(R_{t-1}^x, \hat{R}_t(\omega))\right) \\ &\quad = \arg\min_{y_t, x_t} C_t\left(y_t, x_t \mid R_t(\omega)\right) + \sum_{k\in K_t} \overline{v}_{kt} \left(R_{t-1}^x + \hat{R}_{k,t}(\omega) - \sum_{j\in J_t} x_{jkt} \right) \\ &\quad = \arg\min_{y_t, x_t} C_t\left(y_t, x_t \mid R_t(\omega)\right) - \sum_{k\in K_t} \overline{v}_{kt} \left(\sum_{j\in J_t} x_{jkt} \right) + \sum_{k\in K_t} \overline{v}_{kt} \left(R_{t-1}^x + \hat{R}_{k,t}(\omega) \right) \end{aligned} \tag{11}$$

The last term on the right hand side of (11) is a constant with respect to *x* and *y*, and hence can be ignored. The middle term is linear in x_t. Combining the definition of $C_t\left(y_t, x_t \mid R_t^x(\omega)\right)$ in equation (7) with equation (11), and dropping the constant term, gives:

$$\hat{V}_{t-1}(R_{t-1}^x, \hat{R}_t(\omega)) = \min_{y_t, x_t} \sum_{j\in J_t} \left(f_{jt} y_{jt} + \sum_{k\in K_t} (g_{jkt} - \overline{v}_{kt}) x_{jkt} \right) + \sum_{k\in K_t} h_{kt} R_{kt}^x \tag{12}$$

where we let $\hat{V}_{t-1}(R_{t-1}^x, \hat{R}_t(\omega))$ represent a sample estimate of the value function.

The challenge now is to devise a strategy to update $\overline{v}_t$ in such a way that the formulation for the dynamic problem can be solved satisfactorily. Let $\hat{v}_t$ be a sample estimate of the slope of V_t in a particular iteration. If the value function were differentiable, we could write:

$$\hat{v}_{kt} = \frac{\partial V_t}{\partial R_{kt}} \tag{13}$$

We use the notation that $\overline{v}_t$ is a *smoothed* estimate of the slope, which combines estimates over multiple iterations, while $\hat{v}_t$ is a sample estimate obtained at a particular iteration. For our setting, we use a finite difference approximation of the derivative:

$$\hat{v}_{kt} \cong [\hat{V}_t(R_t^x + \Delta R_{kt}) - \hat{V}_t(R_t^x)] / \Delta R_{kt} \tag{14}$$

The $\hat{v}_{kt}$ slope captures the impact on the value of the objective function of having in inventory at the beginning of period *t* (before the arrival of the new shipments) one extra unit of flow with destination *k*. By using the definition of the value function $\overline{V}_t$, one obtains

$$\hat{v}_{kt} = \frac{1}{\Delta R_{kt}} \left(\begin{array}{l} \sum_{j \in J_t} \left(f_{jt} \Delta y_{jt} + \sum_{k \in K_t} (g_{jkt} - \overline{v}_{kt}) \Delta x_{jkt} \right) \\ \quad + \sum_{k \in K_t} h_{kt} \left\{ \left(R^x_{t-1} + \Delta R_{kt} + \hat{R}_{kt} - \sum_{j \in J_t} \left(x_{jkt} + \Delta x_{jkt} \right) \right) - \left(R^x_{t-1} + \hat{R}_{kt} - \sum_{j \in J_t} x_{jkt} \right) \right\} \end{array} \right) \qquad (15)$$

where

$$\Delta y_{jt} = y_{jt}(R_{kt} + \Delta R_{kt}) - y_{jt}(R_{kt})$$

and

$$\Delta x_{jkt} = x_{jkt}(R_{kt} + \Delta R_{kt}) - x_{jkt}(R_{kt}).$$

The solution algorithm is schematically displayed in Figure 3. The initialization phase sets slope estimates and indices to 0 and takes a single sample of demands (shipments) at all periods of the planning horizon. The algorithm then proceeds iteratively in a series of alternating *forward* and *backward passes* until some stopping criteria are satisfied. Each iteration is composed of a forward pass and a backward pass. In the forward pass, the deterministic multiperiod service selection and flow assignment problem is solved from time $t = 0$ to the end of the planning horizon ($t = T$), using the evaluations of the shipment values at nodes updated during the previous backward pass. The problem is solved using the method presented in Section 3, which is based on the meta-heuristic strategy detailed in the next section. The backward pass proceeds from the end of the planning horizon to its beginning. It computes new shipment value approximations at nodes (slope values) using expression (15) for a perturbation of one unit of the inventory available at each node and time period for each destination ($\Delta R_{kt} = 1$) and the updated dispatching and routing decisions yielded by the previous forward step. New sets of flows (y_{jt}, x_{jkt}) and shipment value approximations $\hat{v}_{kt}$ are thus obtained at each iteration n. Since the values $\hat{v}_{kt}$ are sample estimates, we approximate their expectation by performing the standard smoothing operation:

$$\overline{v}^n_{kt} = (1 - \gamma^n)\overline{v}^{n-1}_{kt} + \gamma^n \hat{v}^n_{kt}, \qquad 0 \le \gamma^n \le 1. \qquad (16)$$

2 DETERMINISTIC SINGLE-PERIOD, SINGLE-NODE PROBLEM

The general strategy used to address the multiperiod formulations of the previous section decomposes the problem by period and solves a sequence of deterministic single-period, single-terminal network design problems. The problem belongs to the class of capacitated, multicommodity network design problem and is known to be NP-Hard. Moreover, it has to be solved repeatedly and a very efficient solution method is required. In fact, good solutions obtained very quickly are of greater interest than optimal ones obtained at greater time cost. Optimality in this sense is a bonus, not an objective *per se*. We propose therefore an efficient tabu search meta-heuristic procedure. We first state the mathematical formulation, followed by the description of the ejection-chain neighborhood we propose, and, finally, we introduce the complete tabu search meta-heuristics.

Step 0 – Initialization

Set $\bar{v}_{kt}^{0} = 0$, for all k and t. Let $n = 1$, $t = 0$.

Step 1 – *Forward Pass*

Draw a sample realization ω that determines the outcome of the information process at each time period.

For $t \leftarrow 1$ to T

Solve the single-node, single-period deterministic service selection and demand routing problem using the method of Section 2.3. This yields new values for (y_{jt}, x_{jkt}).

Step 2 – *Backward Pass*

For $t \leftarrow$ T to 1

For $k \leftarrow 1$ to K

Compute a new sample gradient $\hat{v}_{kt}^{n}$ (using (14) and (15)).

Use (16) to smooth the $\hat{v}_{kt}^{n}$ values to obtain updated estimates of $\bar{v}_{kt}$.

Step 3 – Stop

If the stopping criterion is reached STOP, otherwise, set n ← n + 1 and return to 1.

Figure 3 – Algorithm for the Stochastic Multiperiod Problem

2.1 The single-period formulation

The network of the static, single-period problem is rooted at node (terminal) i, in period t, and iteration n. It is denoted by $G_{it}^{n} = \left(N_{it}^{n}, A_{it}^{n}\right)$ and represents the decision of a carrier, given a moment in time and the availability of a certain type of information. Figure 4 illustrates the concept. To simplify the notation in this section, we drop the indices n, t, and i except when necessary to avoid ambiguities.

Following the notation of the previous section, demand is defined as the volume (number of shipments) R_k to be sent from the origin node to a destination $k \in K$. Nodes in K may be terminals or conceptual nodes created by aggregation. Let $R = \sum_{k \in K} R_k$ be the total demand, in number of shipments, at node i.

To reach a destination, a first movement (dispatch) is performed to one of a set of adjacent nodes $j \in J \subseteq N$. The root terminal is connected to the terminals in J by a set of service design arcs (i,j) that represent the possible services at the terminal. From each $j \in J$, one can reach a destination $k \in K$ by using a transportation arc (j,k) that represents a direct link or a path linking the terminals. Transportation links capture the "future periods" effect of the multiperiod formulation. Holding arcs (i,k) directly link the root terminal to each destination k and capture the decisions to hold shipments for dispatch in later periods. Thus, $N = \{i, J, K\}$ and $A = \{(i, j), (j, k), (i, k) \,|\, j \in J, k \in K\}$.

A fixed cost f_j and a capacity u_j are associated to each design link. A cost g_{jk} is associated with transportation links, which represents the cost of sending one unit of flow to a destination k through an intermediate terminal j. When the intermediate node is the final destination, $g_{jj} = 0$. There is always the option of not moving the freight at this moment and this option implies a holding cost h_k. The decision variables represent whether or not a truck is to be dispatched to j (or, equivalently, the utilization of the intermediate terminals), and the routing of freight to each destination k:

- Design (dispatching) variables: y_j (y^n_{ijt}), $y_j = 1$ if node j is used (a vehicle is dispatched for i to j) and 0 otherwise. We assume in this paper that the length of the time period is set in such a way that only one departure towards each terminal j is possible at each period.
- Freight flows: volumes sent from i to k through j: x_{jk} (x^n_{ijkt}).

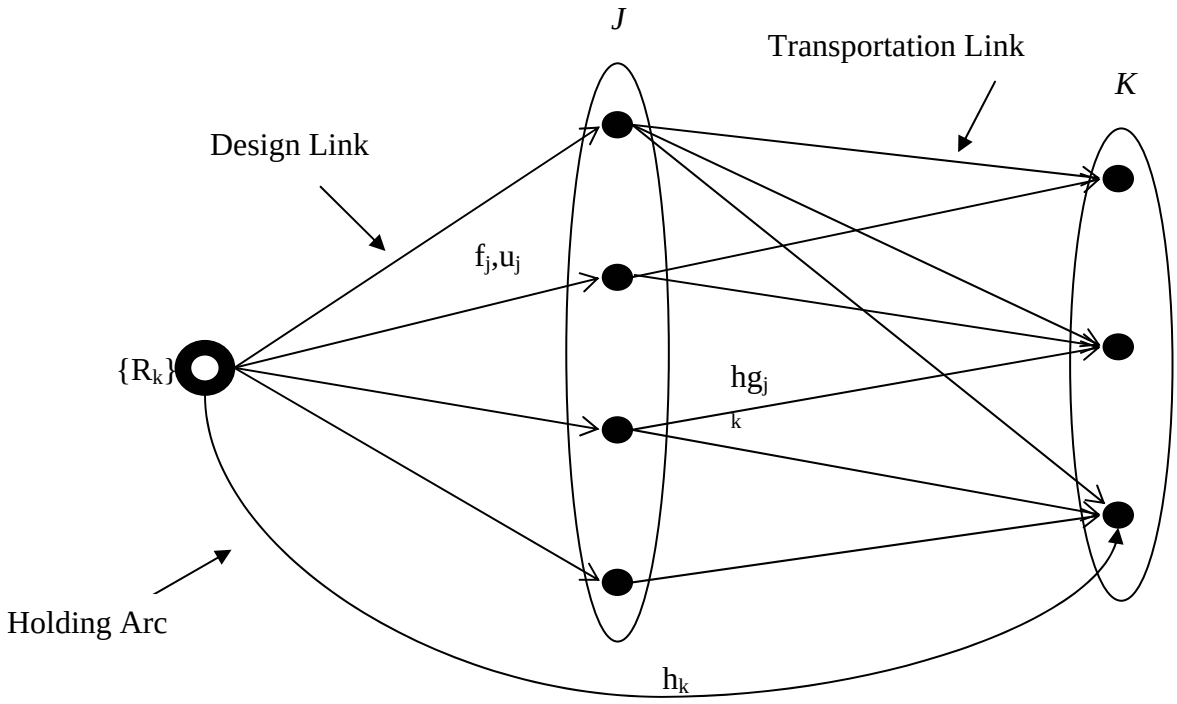

Figure 4 - Static, Single-Period Single-Node Service Network Design Problem

The general formulation of Section 1 then becomes

$$\text{Minimize } Z(x,y) = \sum_{j \in J} (f_j y_j + \sum_{k \in K} g_{jk} x_{jk}) + \sum_{k \in K} h_k (R_k - \sum_{j \in J} x_{jk}) \qquad (17)$$

$$\text{Subject to: } \sum_{k \in K} x_{jk} \le y_j u_j \qquad \forall j \in J \qquad (18)$$

$$\sum_{j \in J} x_{jk} \le R_k \qquad \forall k \in K \qquad (19)$$

$$x_{jk} \geq 0 \qquad \forall j \in J, k \in K \qquad (20)$$

$$y_j = \{0,1\} \qquad \forall j \in J \qquad (21)$$

where constraints (18) are the usual linking (or knapsack) restrictions on using only available services, while equations (19) are flow (or demand) constraints.

Notice that for each destination *k*, one can explicitly enumerate all the paths from *i* to *k* through nodes *j*. These p_{ijk} paths are all made out of two arcs: $p_{ijk} = \{(i,j),(j,k)\}$. The cost of sending a full truck on each of the paths is given by $C_{p_{ijk}} = f_j + g_{jk}u_j$. It is then clear that for each destination *k*, there is a preferred route that ensures the smaller cost for moving a full truck. Therefore, when $R_k \geq u_j$, $\lfloor R_k / u_j \rfloor$ trucks will be dispatched on the preferred route and only routes for the remaining $R_k - \lfloor R_k / u_j \rfloor$ truck-equivalents will have to be determined. In the rest of this paper, we assume therefore that R_k represents this remaining quantity (and thus $R_k < u_j$).

Notice also that the objective function can be recast as:

$$C(x,y) = \sum_{j \in J} c_j y_j + \sum_{k \in K} \sum_{j \in J} (g_{jk} - h_k) x_{jk} + \sum_{k \in K} h_k R_k \; .$$

It is then clear that for all *k* such that $g_{jk} > h_k$, $C(x,y)$ will be minimized for $x_{jk} = 0$. Thus, one has only to consider paths for which the transportation cost is less than the cost of "do nothing" (hold) option for the particular destination. It is also interesting to note that the path-based and the arc-based formulations of the network design model are the same, and that the problem may also be stated as a capacitated, multicommodity node design (location) model (Daskin 1995; Labbé and Louveaux 1997).

Three heuristics have been developed to address this problem and are compared in Section 3. All three are iterative improvement procedures based on moving through a search space defined by the continuous transportation variables. The first heuristic accepts only changes that improve the objective function. The other two are tabu search meta-heuristics. These strategies are described next, following the presentation of the ejection-chain neighborhood used by all procedures.

2.2 Ejection chain neighborhood and greedy descent procedure

The search strategies we examine proceed in the space of the continuous flow variables according to a neighborhood defined by *ejection chains* (Glover and Laguna 1997; Rego and Roucairol 1996). An ejection chain combines simple moves to produce a composed movement. The simple move generally modifies the attributes of a single element of a solution. If the resulting solution is feasible, the procedure stops. Otherwise, the attributes of another solution are modified to accommodate the previous modification. And so on, until a certain stopping criterion is met. Initial applications were dedicated to vehicle routing problems where the simple move consisted in moving one customer to a different route and, eventually, ejecting another customer from the receiving route to make room for it.

The application of the ejection-chain idea to the single-node service network design problem is to consider the shipments as the individual solution elements and their assignment to services as the attribute to modify. Thus, the chain is initiated by selecting a load and moving it to another service. If the capacity restriction of the new service is still verified, the procedure stops. Otherwise, a load previously on the receiving service is selected and moved (it is ejected) to another service. And so on, in a sequence of "move a load from one service to another and, eventually, eject another load

to make room for it". A limit is imposed on the maximum size of an ejection chain. When this limit is reached without reaching feasibility, the load is moved to the holding arc.

Figure 5 illustrates an ejection chain. The upper left figure represents a feasible solution. The figures in the sequence show the reaction caused by a move of a load from one service to another. The values within square parentheses represent the size of the shipments, while values within parentheses represent arc capacities. In this example, the chain stops by moving a load to the holding arc.

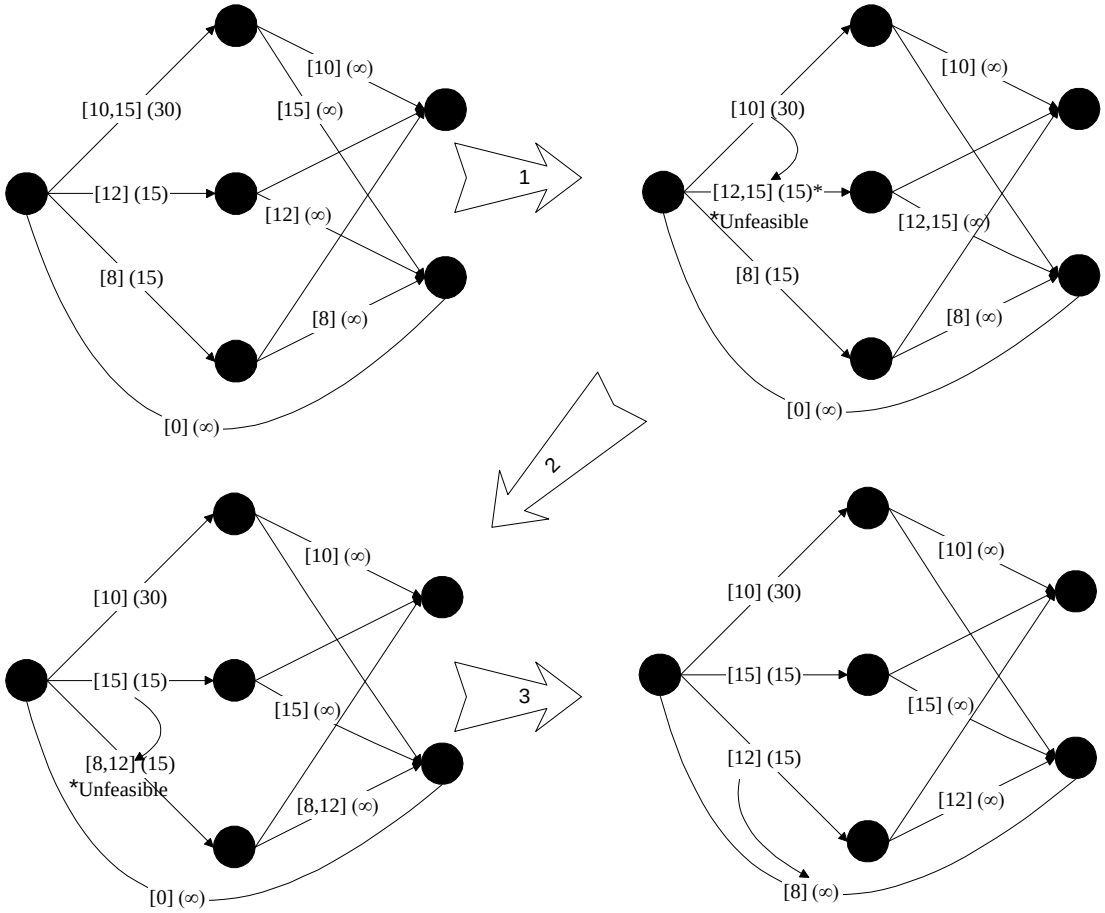

Figure 5 - Illustration of an Ejection Chain

A cost-based greedy procedure is used to determine an ejection chain. The first move in the chain corresponds to a load on the path with the highest unit cost. The load is moved to its minimum unit cost path. Because one of the goals of the procedure is to reduce the number of services used (empty some and load as much as possible the others), the initial load has the highest volume among those on the service, while the loads ejected in subsequent moves have the lowest volume among those on the ejecting service. Figure 6 displays the main steps of this greedy strategy.

Step 1 – Sort paths in decreasing order of unit cost

Step 2 – Select the maximum volume load on the path with the highest unit cost
Eject this shipment and identify it as the first of the chain

Step 3 – While (chain not completed) **or** (maximum chain size not reached)) **Do**
Find the minimum unit cost path for this shipment
If (residual capacity of the service ≥ size of shipment) **Then**
The chain is *completed*
Else
Identify the lowest volume shipment on the receiving service
Eject this load and add it to the chain

Step 4 – If (chain reached the maximum size) **Then**
Put the last ejected shipment on its inventory arc

End

Figure 6 – Greedy Algorithm to Determine an Ejection Chain

A *Simple Heuristic* (*SH*) that greedily accepts an ejection chain that improves the current solution can be described as follows. Given a feasible solution, determine an ejection chain using the procedure described above. If the new solution is better than the current solution, implement it; otherwise, determine a new ejection chain (starting with the next load in decreasing order of volumes on the paths ordered by decreasing unit costs). Continue until an improving solution is found (or a time limit is reached). It is clear that the procedure will stop with the first local optimum encountered, irrespective of the quality of the solution. The tabu search meta-heuristics described next attempt to overcome this shortcoming.

2.3 Tabu Search Meta-heuristics

Tabu search is a meta-heuristic that has been successfully used to address many hard, combinatorial optimization problems, including network design (Glover 1986; Glover and Laguna 1997; Crainic, Farvolden, and Gendreau 2000; Ghamelouche, Crainic, and Gendreau 2003). Tabu search is an iterative procedure that uses memories of solutions (or attributes thereof) already visited to learn about the solution space and to guide the search out of local optima and towards promising regions. One or several neighborhoods are defined, together with the corresponding moves. Then, at each iteration, solutions neighboring the current solution are identified and the "best", not necessarily improving, one is selected as the new current solution. A short-term memory, usually identified as the *tabu list*, is used to avoid cycling by forbidding moves to solutions recently visited or sharing attributes with such solutions. More advanced memory and neighborhood structures may be advantageously used to attempt to reach higher quality solutions. In the present case, however, speed is of essence and we do not implement these features. Experimental results show, however, that the simple tabu search strategy we propose still achieves good quality solutions.

The tabu search procedures we propose are based on the ejection-chain neighborhood defined previously (see Dall'Orto 2001 for detailed descriptions of these procedures). Solutions are evaluated using the objective function of the design problem, which is also used to define an aspiration criterion: a move to a tabu solution is accepted if the new solution has a lower objective function value than the current best. Once an ejection chain is implemented, it becomes tabu (one cannot use it again) for a given number of iterations. The induced neighborhood is huge, however, since in theory, one should construct all ejection chains before selecting the solution to move to. This is impractical in any circumstance. We thus adopt a well-known technique in meta-heuristics in general and tabu search in particular, and use a candidate list.

A *candidate list* is a collection of neighbors among which the selection of the next solution is performed. We define two tabu search procedures that differ in how the candidate list is built: *Search First Feasible Chain* (*SFFC*) and *Search First Improving Chain* (*SFIC*).

The SFFC strategy builds ejection chains starting from the current solution and accepts the first feasible solution that either is not tabu or fulfills the aspiration criterion. Notice that non improving solutions are accepted. Therefore, if after a certain number of iterations, neither the best solution nor the current solution are improved, the procedure returns to the current best solution and proceeds with the search. If the search returns to the same solution several consecutive times, the search is stopped.

A broader part of the neighborhood is explored by the SFIC procedure. Here, ejection chains are built until a feasible solution is found that improves the current solution. If no chain improves the current solution, the one that least degrades the solution is implemented. If, after a certain number of iterations the current solution is still not improved, the procedure returns to the best current solution and the search continues on the second best chain for that solution. The process continues until it returns a certain number of times to the same best solution.

2.3 Initial solution

Irrespective of the strategy selected, an initial feasible solution has to be identified. The heuristic we use starts by sorting all paths in increasing order of cost. It then proceeds iteratively through the shipments and attempts to send every load on its shortest (cheapest) path. The next path in the list is used when the capacity of the path is reached. This heuristic is easy to implement and fast. Moreover, as illustrated by the computational results of the next section, the tabu search procedures obtain good solutions in short computational times.

Shipments are assigned to and moved among services based on the path valuations that reflect both the fixed and the variable costs of the system. The fixed service cost has to be paid as soon as the service is selected. In order not to penalize the shipments that are assigned first to a previously unselected service, approximations are used. Thus, to compute the initial solution, we associate the fixed cost to the capacity of the service, which yields the following path cost:

$$c_{p_{ijk}} = \frac{f_j}{u_j} + g_{jk}$$

Once the initial solution is found, the unit cost of a path using selected services is computed in the usual way:

$$c_{p_{ijk}} = \frac{f_j}{x_j} + g_{jk}$$

On the other hand, unused arcs are "closed" ($y_{ij} = 0$) during the search. Then, to evaluate the cost of sending a load on a path that makes use of a not yet selected service, the corresponding arc is given a cost equal to the maximum cost value computed for that arc during the previous iterations, including those of the initial solution.

2.4 Long-term memory

As already noticed, the single-period, single-terminal problem has to be solved repeatedly when DSND problems are contemplated: at each time period and forward-backward iteration. This may be extremely time-consuming as illustrated in Section 3. Moreover, it is felt that the behavior of a terminal, the number of potential service departures, as well as the composition and amplitude of demand and, thus, the actual dispatch decisions, will not vary widely from period to period or from one iteration to the next. Some "order" exists in the randomness of demand arrivals and the operations of the system. Observations of operations of actual transportation systems tend to support this hypothesis. Then, if such a behavioral pattern could be identified for a terminal, it could be used to accelerate the search by, for example, identifying the preferred service for shipments departing to certain final destinations. This should help to reduce significantly the computational time.

Our strategy is based on long-term memories that record, in a *frequency matrix*, the frequencies of services used to move each demand to its final destination. This matrix is built during the execution of the algorithm. Once the static problem is solved for the current period, the matrix is updated by increasing the frequencies of the services used. After a few iterations, the dispatch pattern for each demand should start to emerge as the services most heavily used. We will use this pattern to bias the selection of services when initial solutions need to be computed. This approach is somewhat similar to the *target analysis* methodology proposed initially by Glover (see Glover and Laguna 1997).

3 EXPERIMENTAL RESULTS

The experimental phase of our work was planned to examine several issues:

1. What is the performance in terms of computing efficiency and solution quality of the heuristics for the static, single-period problem? Which one is better? Recall that this problem has to be solved repeatedly when the dynamic problem is addressed, and that the number of repetitions is expected to get huge when full dynamic service network design problems are contemplated.
2. How efficiently can dynamic problems be solved? Is the long-term memory mechanism (the matrix of frequencies) helpful?
3. What is gained by explicitly considering the dynamics of the system compared to simply solving a series of single-period static instances?

3.1 Experimental Design

We tested the models on problem instances based on data from a large U.S. LTL motor carrier, operating numerous breakbulk and end-of-line terminals, that serve the continental United States. We used this information to set the general parameters of our problem, but used randomly generated data to control the characteristics of individual problem instances.

Three classes of problems were generated, each with a different demand-to-capacity ratio. Class I problems have more demand than the capacity of the network, while the contrary is true for problems of Class III. Problem instances in Class II display a balance between demand and capacity. Problem instances were generated randomly, by using uniform distributions on the intervals presented in Table 1 for capacity and demand, as well as for service (fixed), transportation, and holding

costs. Problems of four different sizes, small, medium, large, and extra large, were generated as defined in Table 2. Dynamic problem instances were generated following a similar process but generating a different vector of demand for each period of a 7-day planning horizon.

Table 1 – General Characteristics of Test Problem Instances

Problem Class	Capacity	Demand	Fixed Cost	Holding Cost	Transportation Cost
Class I	[50,150]	[150,300]	[50,300]	[100,300]	[20,120]
Class II	[150,400]	[130,260]	[50,300]	[100,300]	[20,120]
Class III	[150,450]	[30,60]	[50,300]	[100,300]	[20,120]

Table 2 – Test Problem Instances Dimensions

Problem Size	# of Terminals	# of Intermediate Terminals	# of Intermediate for each Destination
Small	[10,20]	[3,8]	[2,5]
Medium	[21,100]	[10,50]	[4,12]
Large	[101,250]	[50,100]	[8,20]
XLarge	[260,450]	[100,200]	[16,50]

For the sake of conciseness and clarity, only aggregated results are included in the paper. Detailed results are presented in Dall'Orto (2001).

3.2 Static Experiments

As indicated previously, the solution procedure for the single-period, single-terminal service design problem must be very efficient. Of course, the quality of the solution is also of major importance. A number of parameters characterize the heuristics and impact their performance. These parameters must be calibrated prior to undertaking the performance analysis.

The calibration phase has been conducted using 250 problem instances (small, medium, and large of all types), on a PC equipped with a Pentium III processor with an 800 MHz clock. Lindo 6.1 was used to obtain the optimal solutions.

Five parameters characterize the heuristics. The values tested for each one of them were:

1. *Maximum number of iterations* the procedure was allowed to run (if not stopped otherwise). We tested values equal to 100, 200, 300, and 500 for HS, and 500, 1000, 1500, 2000, 2500, and 3500 for the SFFC and SFIC tabu search procedures.
2. *Maximum size of the ejection chain*. This parameter limits the number of load exchanges allowed when ejection chains are built. Values equal to 4, 6, 8, 10, and 12 were tested.
3. *Tabu list size* that indicates the number of iterations a solution is tagged tabu, e.g., one cannot return to it unless it satisfies the aspiration criteria (better than the current best). This parameter attempts to avoid cycling and to foster a broader exploration of the solution space. Tabu list length of 5, 10, 15, and 20 were tested.

4. *Maximum number of returns* to the same solution before stopping the procedure. When this limit is attained, it is an indication that, given the settings of the other parameters, the search has reached a local optimum from which it is incapable to escape. We tested 4, 6, 8, 10, and 12.
5. *Maximum number of successive iterations without improvement.* When the search reaches this limit, it returns to the current best solution to initiate a new search thread. We tested 3, 5, and 10.

The procedures and parameter settings were compared according to the percentage of deviation from the optimal solution and the computation time required. The general conclusion of these experiments is that the procedures are robust with respect to the parameter settings. For each heuristic, the results for different parameter settings are very similar, the calibration appearing more as fine tuning of parameters, rather than a discrimination procedure.

As expected, tabu search outperforms greedy descent in solution quality, the procedure using First Improvement (SFIC) reaching higher quality solutions than the one (SFFC) that moves to the First Feasible solution. Again, without surprise, the relation on computing times displays exactly the opposite characteristics. In fact, the number of iterations has no impact on the performance of the greedy HS heuristic, provided it is sufficiently long to allow the procedure to reach a local optimum. This is not true for the tabu search procedures that generally obtain better solutions when allowed to search longer. Yet, the rate of improvement slows down after a certain point that we fixed empirically to 2500 iterations.

We somehow expected to find a relationship between the length of the ejection chains and the problem dimensions. We did not find such a relationship in our experiments, however. We rather identified a relationship between the ejection chain length and the type of solution method.

The other three parameters do not apply to the HS procedure. For tabu search, the duration of tabu tags is generally an important parameter. This turned out to be the case here as well, the tabu list length impacting both the solution quality and the computational times. The last 2 parameters, the maximum number of returns to the same solution and the maximum number of successive iterations without improvement, did not impact the solution quality in any significant way. They did influence computing times, though. The final parameter settings appear in Table 3 (an X indicates that the parameter is not used by the procedure) and were used to perform the comparison analysis of the three heuristics presented next.

A new set of 100 problems of all types and dimensions were generated for the comparative analysis of the proposed procedures. We generated a larger number of Class II problems, where demand and capacity are balanced, because they appear more difficult to solve than Class I (the solution consists in using "all" services and holding the excess demand) and Class III (where the preferred service is used for almost all demands) problem instances.

Table 3 – The Parameters for the Solution Strategies

Parameter	SH	SFFC	SFIC
Max. iterations	200	2500	2500
Max. chain size	6	8	8
Tabu list size	X	15	10
Max. returns to the same solution	X	8	4
Max. iterations without improvement	X	10	10

Optimal solutions have been obtained by using CPLEX 7.1. All experiments have been performed on a SUN Workstation. The results are summarized in Tables 4, 5, and 6. We indicate average results for the percentage of optimality reached (a value of 100 indicates that optimal solutions

have been found), as well as for the computational time in seconds CPU (the figures are given within square brackets).

Two main conclusions emerge from the experimental results summarized in these tables. First, despite its apparent simplicity, the single-node, single-period service network design problem is difficult to solve optimally within a short computation time for most problems of interest. Even for the easy problem instances of Class I, where demand exceeds supply, the exact method required significantly more time than the heuristics. Second, the heuristics we propose achieve good solutions both in terms of deviation from the optimal solution and in computation effort required to achieve these results.

Table 4 – Results for Class I (Optimum Percentage and Time [sec])

Problem Size	CPLEX	SH	SFFC	SFIC
Small	100 [0.015]	100 [0.0325]	100 [0.0325]	100 [0.0325]
Medium	100[0.245]	100 [0.085]	100 [0.265]	100 [0.110]
Large	100[8.455]	100 [0.508]	100 [1.886]	100 [0.851]
XLarge	100[58.852]	99.98[0.587]	99.98[3.538]	99.98[2.658]

Table 5 – Results for Class II (Optimum Percentage and Time [sec])

Problem Size	CPLEX	SH	SFFC	SFIC
Small	100 [0.0233]	89.37 [0.04]	99.11 [0.04]	96.79 [0.04]
Medium	100 [0.48]	88.03 [0.0525]	92.20 [0.4025]	92.28 [0.2803]
Large	100 [12.05]	84.88 [0.85]	89.39 [2.25]	90.18 [3.30]
XLarge	100[68.75]	85.14[1.50]	90.54[5.05]	91.54[8.52]

Table 6 – Results for Class III (Optimum Percentage and Time [sec])

Problem Size	CPLEX	SH	SFFC	SFIC
Small	100 [0.01]	96.65 [0,1775]	98.61 [0.0425]	98.84 [0.0325]
Medium	100 [0.265]	93.94 [0.33]	96.71 [0.365]	97.89 [0.1083]
Large	100 [9.343]	88.31 [0.466]	92.49 [1.995]	95.04 [2.852]
XLarge	100 [64.26]	88.46 [0.978]	92.47 [3.678]	92.80 [4.162]

Comparing the performances of the three heuristics, it is again clear that the greedy heuristic is the fastest but yields the poorer results. Tabu search obtains the best solutions, the First Improvement strategy offering the best performances in this respect at the expense of slightly longer computational times. With respect to the type of instance, problems are easy when demand exceeds supply. The simple heuristic is sufficient in these cases. An excess of capacity (Class III) also makes problems easier to address, although not as easy as for Class I instances. It is for problems of Class II, representative of most actual cases where capacity "balances" demand, that the differences among heuristics and the advantages of the tabu search appear clearly.

3.3 – Dynamic Experiments

The only calibration required for the dynamic, multi-period algorithm concerns the parameters normally used to ensure convergence. Thus, for example, Powell and Carvalho (1998) use an upper bound on the empty movements and a smoothing factor to guarantee the convergence of their strat-

egy. We decided not to impose limits on the quantities kept in inventory to avoid making some problems infeasible. Thus, we use the smoothing factor only, even though it does not guarantee the convergence of the dynamic model. We tested using a smoothing factor varying from 0.2 to 0.8 (see Figure 7). The best behavior was observed using $\gamma = 0.8$: with this value the results oscillate around the best solution found.

This setting was then used to analyze the performance of the dynamic algorithm. Notice that the first forward step in the dynamic strategy uses slopes equal to zero. This is the same thing as solving the dynamic problem using the static strategy sequentially and independently for each decision period. Thus, the performance of the dynamic approach is measured as the improvement it induces compared to the initial, greedy-myopic solution. 84 new test problems have been generated for these experiments, 21 instances for each problem size.

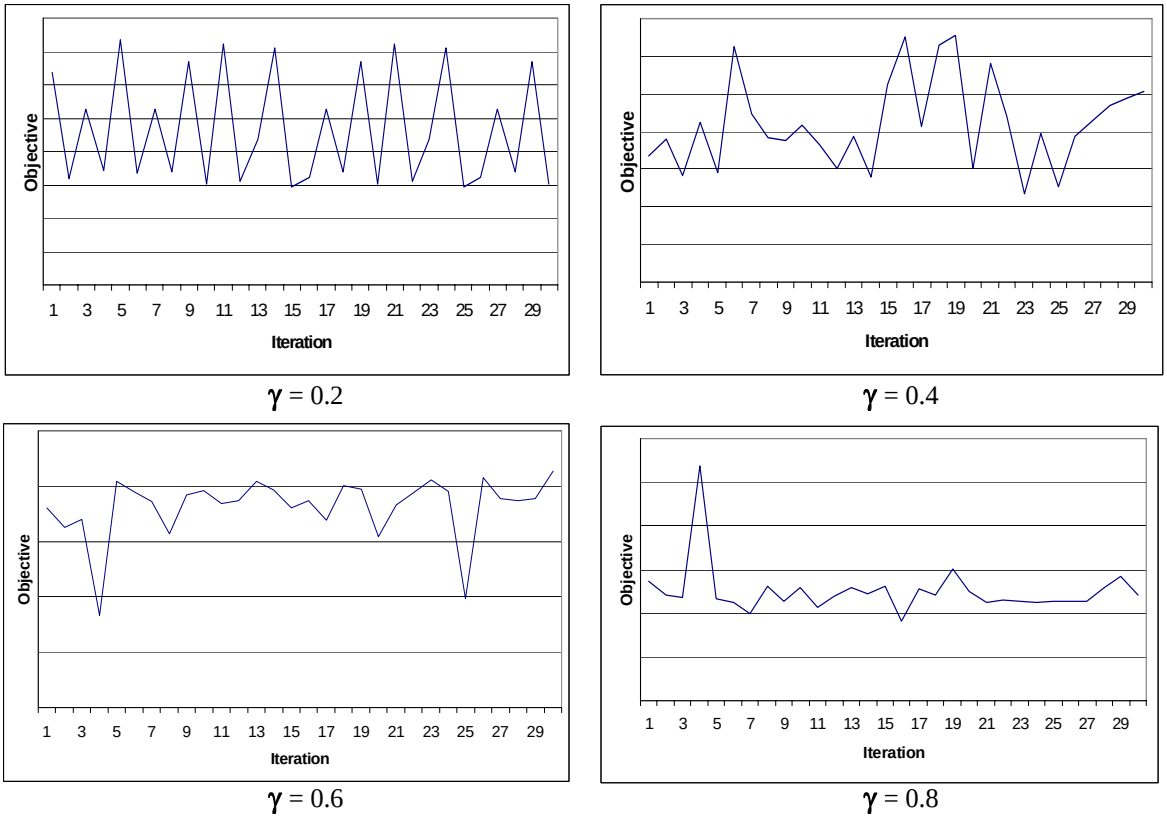

Figure 7 – Convergence Behavior According to the Smoothing Factor

Table 7 displays the average improvement (in percentage) obtained by using the dynamic algorithm for these test problems. The SH strategy does not improve a (good) greedy-myopic initial solution. It proves to be incapable of escaping the corresponding local optimum. The tabu search procedures obtain more significant improvements, especially for problems of class II. For classes I and III one observes, again, that there is little to improve when there is almost no demand or capacity to manage. For problems of class II, where capacity is sufficient but not in excess and good operating policies are important, the tabu search methods obtain very good results, with an average improvement of 15.55% and 17.19% for the SFFC and SFIC strategies, respectively. It is important to emphasize that class II problems are the most common in practice.

Table 7 – Improvement Obtained by the Dynamic Algorithm from a Greedy-Myopic Solution

Problem Class	SH	SFFC	SFIC
I	0	1.04	0.46
II	0.06	15.55	17.19
III	0	0.54	1.43

The computational time required by a straightforward application of the dynamic procedure appears too large, however, of the order of 1 to 3 minutes. The factor that most contributes to these times is the backward pass where the problem is solved several times to compute the marginal values. Most of these problems are very similar, differing by one unit of demand only. We therefore repeated the experiments using the long-term memory mechanism described at Section 2.4. Table 8 presents the aggregated results.

Table 8 – Impact of Long-Term Memory on the Performance the Dynamic Algorithm

	Greedy-Myopic	Dynamic SFFC	Dynamic SHIC
Total time "straight"	11.5	940.6	817.4
Total time "memory"	11.5	190.8	177.8
Gain	0	79.72%	78.25%

Each of the first two rows of Table 8 displays the total CPU time required to apply the algorithm indicated in each column to all 84 problem instances without ("straight") and with ("memory") the long-term memory. The last row indicates the gain (in percentage) obtained by using the frequency matrix memory. The results are given for the greedy-myopic method and for the two dynamic algorithms that correspond to using each of the two tabu search methods to solve the single-period problem. These results confirm the importance of the long-term memory for the dynamic solution method and opens up interesting perspectives for addressing full scale DSND problems.

4 – CONCLUSIONS

We have examined the single-node dynamic service network design problem and presented two formulations: a time-dependent, stochastic formulation that aims to optimize the problem over a given planning horizon, and a static, single-period, model that appears as a subproblem when addressing both the time-dependent version and general DSND cases. Appropriate solution methods have been presented for both formulations. In particular, a tabu search framework based on ejection chains has been introduced for the single-period problem. Moreover, a long-term learning mechanism has been introduced as well, which allows significant gains in computing efficiency for the dynamic algorithm. Experimentations performed on a large number of problem instances derived from actual data indicate that the methods presented are both computationally very efficient and provide solutions close to the optimal ones. This opens very promising perspectives for the general dynamic service network design problem.

ACKNOWLEDGMENTS

Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada, the Fonds F.C.A.R. of the Province of Québec (Canada), and CNPq - The National Research Council of Brazil. The first author (Dall'Orto) thanks CAPES (Brazil) for the PhD

fellowship he received as well as for the financial support to the visit to Canada. The work of the last author (Powell) was partially supported by grant AFOSR-F49620-93-1-0098 from the Air Force Office of Scientific Research.

REFERENCES

Armacost , A.P., Barnhart, C., and Ware, K.A. (2002). Composite Variable Formulations for Express Shipment Service Network Design, *Transportation Science* 36(1):1-20.

Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. (1997). Network Design. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 311-334, John Wiley & Sons, New York, NY.

Barnhart, C., Jin, H., and Vance, P.H. (2000). Railroad Blocking: A Network Design Application. *Operations Research*, 48(4):603-614.

Barnhart, C. and Schneur, R.R. (1996). Network Design for Express Freight Service, *Operations Research*, 44(6):852-863

Bertazzi, L. and Speranza, M.G. (1999). Inventory Control on Sequences of Links with Given Transportation Frequencies. *International Journal of Production Economics*, 56:261-270.

Bertazzi, L. and Speranza, M.G. (2002). Continuous and Discrete Shipping Strategies for the Single Link Problem. *Transportation Science*, 36(3):314-325.

Blumenfeld, D.E., Burns, L.D., Diltz, J.D., and Daganzo, C.F. (1985). Analyzing Trade-Off Between Transportation, Inventory and Production Costs on Freight Networks. *Transportation Research B: Methodology*, 19B(5):361-380.

Burns, L.D., Hall, R.W., Blumenfeld, D.E., and Daganzo, C.F. (1984). Distributions Strategies that Minimize Transportation and Inventory Cost. *Operations Research*, 33(3):469-490.

Cordeau, J-F., Toth, P., and Vigo, D. (1998). A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*. 32(4):380-404.

Crainic, T. G. (2000). Network Design in Freight Transportation. *European Journal of Operational Research*, 122(2):272-288.

Crainic, T. G. (2003). Long-Haul Freight Transportation. In R. Hall, editor, *Handbook of Transportation Science*, 2nd Edition, pages 451-516, Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G. and Kim, D. (2004). Intermodal Transportation. In Barnhart, C. and Laporte, G., Editors, *Transportation* volume of *Handbooks in Operations Research and Management Science*. North- Holland, Amsterdam (forthcoming).

Crainic, T. G. and G. Laporte (1997). Planning Models for Freight Transportation. *European Journal of Operational Research*, 97(3):409-434.

Crainic, T. G. and J. Roy (1988). OR Tools for Tactical Freight Transportation Planning. *European Journal of Operational Research*. 33:290-297.

Crainic, T. G. and J-M. Rousseau (1986). Multicommodity, Multimode Freight Transportation: A General Modeling and Algorithmic Framework for the Service Network Design Prob-

lem. *Transportation Research B: Methodology*, 20(B):225-242.

Crainic, T. G., J-A. Ferland and J-M Rousseau (1984). A Tactical Planning Model for Rail Freight Transportation. *Transportation Science*, 18(2):165-184.

Crainic, T. G., Gendreau, M., and Farvolden, J. (2000). A Simplex-Based Tabu Search Method for Capacitated Network Design. *INFORMS Journal on Computing* 12(3): 223-236.

Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2004). Maritime Transportation. In Barnhart, C. and Laporte, G., Editors, *Transportation* volume of *Handbooks in Operations Research and Management Science*. North-Holland, Amsterdam (forthcoming).

Daganzo, C. F. (1991). *Logistics Systems Analysis*. Springer-Verlag, Berlin.

Dall'Orto, L.C. (2001). *Modelagem e algorithmos para o Problema do Projeto de Redes de Serviço Dinâmicas*. Ph.D. Thesis, Department of Industrial Engineering, Pontificia Universidade Catolica do Rio de Janeiro, Brazil.

Daskin, M.S. (1995). *Network and Discrete Location, Models, Algorithms and Applications*, Wiley-Interscience, New York.

Deb, R. and Serfozo, R. (1973), Optimal Control of Batch Service Queues, *Advances in Applied Probability*, 5:340-361.

Equi, L., Gallo, G., Marziale, S., and Weitraub, A. (1997). A Combined Transportation and Scheduling Problem. *European Journal of Operational Research*, 97(1):94-104.

Farvolden, J. and Powell, W.B. (1994). Subgradient Methods for the Service Network Design Problem. *Transportation Science*, 28(3):256-272.

Ghamlouche, I., Crainic, T. G., and Gendreau, M. (2003). Cycle-based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design, *Operations Research* 51(4): 655-667.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 13:533-549.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.

Godfrey, G.A. and Powell, W.B. (2002a). An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management I: Single Period Travel Times. *Transportation Science*, 36(1):40-54.

Godfrey, G.A. and Powell, W.B. (2002b). An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management II: Multi-Period Travel Times. *Transportation Science*, 36(1):21-39.

Gorman, M.F. (1998). An Application of Genetic and Tabu Search to the Freight Railroad Operating Plan Problem. *Annals of Operations Research*, 78:51-69.

Grünert, T. and Sebastian, H.-J. (2000). Planning Models for Long-Haul Operations of Postal and Express Shipment Companies. *European Journal of Operational Research*, 122:289-309.

Haghani, A.E. (1989). Formulation and Solution of Combined Train Routing and Makeup, and Empty Car Distribution Model. *Transportation Research B: Methodology*, 23B(6):433-

452.

Ignall, E. and Kolesar, P. (1972). Operating Characteristics of a Simple Shuttle under Local Dispatching Rules, *Operations Research,* 20:1077-1088.

Kim, D., Barnhart, C., Ware, K., and Reinhardt, G. (1999). Multimodal Express Package Delivery: A Service Network Design Application. *Transportation Science*, 33(4):391-407.

Kosten, L. (1967). The Custodian Problem. In *Queueing Theory: Recent Developments and Applications*, pages 65-70, English Press, Ltd., London.

Kosten, L. (1973). *Stochastic Theory of Service Systems*, Pergamon Press, New York.

Labbé, M. and Louveaux, F.V. (1997). Location Problems. In Dell'Amico, M., Maffioli, F., and Martello, S. Editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 261-281, Willey-Interscience, New York.

Magnanti, T.L. and Wong, R.T. (1986). Network Design and Transportation Planning: Models and Algorithms, *Transportation Science,* 18(1):1-55.

Minoux, M. (1986). Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. *Networks*, 19:313-360.

Papadaki, K. and Powell, W.B. (2002). Exploiting Structure in Adaptive Dynamic Programming Algorithms for a Stochastic Batch Service Problem. *European Journal of Operational Research,* 142(1):108-127.

Papadaki, K. and Powell, W.B. (2003). An Adaptive Dynamic Programming Algorithm for a Stochastic Multiproduct Batch Dispatch Problem, *Naval Research Logistics*, 50(7), 742-769.

Pflug, G. (1996). *Optimization of Stochastic Models: the Interface Between Simulation and Optimization*, Kluwer Academic Publishers, Norwell, MA.

Powell, W.B. (1986). A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks. *Transportation Science*, 20(4):246-357.

Powell, W.B. and Carvalho, T.A. (1997). Dynamic Control of Multicommodity Fleet Management Problems. *European Journal of Operational Research,* 98:522-541.

Powell, W.B. and Carvalho, T.A. (1998). Dynamic Control of Logistics Queueing Networks for Large-Scale Fleet Management. *Transportation Science,* 32(2):90-109.

Powell, W.B. and Humblet, P. (1986). The Bulk Service Queue with a General Control Strategy: Theoretical Analysis and a New Computational Procedure, *Operations Research*, 34:267-275.

Powell, W.B. and Sheffi, Y. (1989). Design and Implementation of an Interactive Optimization System for Network Design in the Motor Carrier Industry, *Operations Research*, 37(1):12-29 (1989).

Powell, W. B. and Topaloglu, H. (2003). Stochastic Programming in Transportation and Logistics. In A. Shapiro and A. Ruszczynski, Editors, *Stochastic Programming* volume of *Handbooks in Operations Research and Management Science*, pages 555-635, Elsevier, Amsterdam.

Powell, W.B. and van Roy, B. (2004). Approximate Dynamic Programming for High Dimensional Resource Allocation Problems, In J. Si, A. Barto, W.B. Powell and D. Wunsch, Editors, *Learning and Approximate Dynamic Programming: Scaling up to the Real World*, John-Wiley and Sons, New York.

Rego, C. and Roucairol, C. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the VRP. In Osman, I.H. and Kelly, J.P., Editors, *Meta-Heuristics: Theory & Applications*, pages 253-295, Kluwer Academic Publishers, Norwell, MA.

Speranza, M.G. and Ukovich, W. (1992). A Decision Support System for Materials Management. *International Journal of Computer Integrated Manufacturing*, 5:239-244.

Speranza, M.G. and Ukovich, W. (1994). Minimizing Transportation and Inventory Costs for Several Products on a Single Link, *Operations Research*, 42:879-894.

Speranza, M.G. and Ukovich, W. (1996). An Algorithm for Optimal Shipments with Given Frequencies. *Naval Research Logistics*, 46:399-417.

Tsitsiklis, J. and van Roy, B. (1997). An Analysis of Temporal-Difference Learning with Function Approximation, *IEEE Transactions on Control*, 42:674-690.