15

Real-Time Dispatching for Truckload Motor Carriers

15.1 Introduction **15**-323
15.2 A Basic Load-Matching Model **15**-325
15.3 Variations and Extensions **15**-327
More Complex Operational Problems • Looking into the Future
15.4 Forecasting Demand **15**-329
Elementary Forecasting • The Challenge of Forecasting Daily Demand • Handling Pre-Booked Loads
15.5 Capacity Forecasting **15**-331
Simulating a Myopic Policy • An Approximate Dynamic Programming Solution • Getting Drivers Home
15.6 Demand Management **15**-334
15.7 Implementation **15**-335
Computer Integration • The Problem of Data • Measuring Compliance
15.8 Case Study—Burlington Motor Carriers **15**-338
Operations Research Models—Round I • The Real-Time Dispatch System—Round II
References **15**-342

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

15.1 Introduction

Truckload trucking may be the simplest operational problem in freight transportation. Shippers use truckload motor carriers to move large quantities of freight which require hiring an entire truck to move a load of goods from one location to another. Similar to taxi operations for passengers, a shipper will call a carrier with information about a load of freight that needs to be moved from one city to another. If the carrier agrees to move the load, he sends out a driver with a tractor (and possibly a trailer) who then picks up a trailer loaded with freight. When the driver delivers the freight at the origin, it is now the responsibility of the company to figure out what to do with the driver next. Although there are many one-man trucking companies, our focus is on operations that manage fleets that may range from several dozen to over 10,000 drivers.

At the heart of real-time operations in truckload trucking is a disarmingly simple problem: given a set of drivers and loads, which driver should be assigned to which load? We could address the problem

from the perspective of a particular driver (what is the best load?) or a particular load (what is the best driver?), but the real problem requires juggling the needs of multiple drivers and loads. A company with 100 drivers, faced with the decision of which driver to assign to each of 100 loads, can choose from among $100 \times 99 \times 98 \times \cdots \times 2 \times 1 \approx 10^{158}$ possible solutions.

In the 1970s, as part of a consulting project with a young trucking company called Schneider National (today, the largest truckload carrier in the United States), Dr. Richard Murphy, then a faculty member at the University of Cincinnati, recognized that the load-matching problem was a special type of linear programming problem known as a pure network. These are best visualized using the network in Figure 15.1. In this representation, drivers are represented as nodes on the left, with a flow of 1 unit entering each driver node. Links join driver nodes to load nodes, with an additional "load link" from each load node to a "supersink" from which all the flow leaves the network. An upper bound on each load link prevents more than one driver from covering each load. The mathematical structure of the problem guarantees that we would never assign a fraction of a driver to a load.

At that time, this observation meant that it was possible to solve very large problems with exceptionally powerful algorithms that were extremely fast (even on computers of that era). Just as important, the model provided for a surprisingly high level of detail in how the costs were calculated. The cost $c_{r\ell}$ of assigning driver r to load ℓ could include the cost of driving empty from the current location of the driver to where the load had to be picked up, along with a variety of other factors. For example, we could add artificial penalties for assigning drivers who would pick up the load after its pickup appointment, or if the driver would arrive so early that he would have to sit for several hours. We could put a bonus (negative cost) for desirable assignments such as putting sleeper teams (pairs of drivers who swap driving so that the truck does not sit idle while a driver sleeps) on long loads (where the team gets higher utilization). In real applications, the list of such issues can be quite long, and yet this simple model can handle a broad range of these operational goals and constraints.

Linear programming models and the associated algorithms looked like the perfect match of a new technology with an industrial application. They offered to overcome what appeared to be a major limitation of human dispatchers—the ability to consider all the drivers and loads at the same time when making a decision. Humans tend to break problems down into small pieces. What is the best driver to move a particular load? What load should I assign this available driver to? Juggling the assignment of multiple drivers and loads at the same time is beyond the problem solving skills of most people.

News of this model spread quickly, and suddenly other carriers wanted their own optimization models for solving the load-matching problems. A small cottage industry of consulting firms popped up in the late 1980s and 1990s to sell this technology. The promise of the technology closely matched the

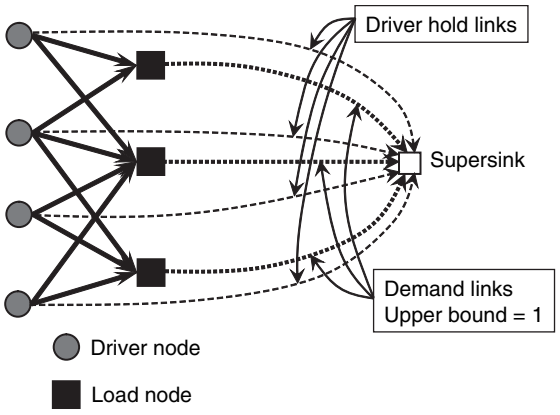

FIGURE 15.1 Network model for driver assignment problem.

early promises of robots building cars in the 1980s. What we learned is that the technology is promising, but the problem has proven to be far more difficult than anyone realized.

15.2 A Basic Load-Matching Model

The network model shown in Figure 15.1 is easily modeled as an assignment problem involving the assignment of driver r (we think of drivers as the resources we are managing) to load l. Such models would define c_{rl} to be the cost of assigning driver r to load l, and we would define a decision variable x_{rl} where $x_{rl} = 1$ means we have decided to assign driver r to load l. In this section, we adopt a somewhat different notation that will prove to be much more general, allowing us to easily represent other issues that are not captured by this basic model. We describe a driver using a vector of attributes we denote by a, such as:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \text{Location} \\ \text{ETA} \\ \text{Home domicile} \\ \text{Equipment} \\ \text{Team?} \\ \text{Days from home} \\ \text{DOT hours} \\ \vdots \\ a_n \end{pmatrix}$$

Here, location might represent his exact current location, his last reported location, or the location to which he is headed (it would be his current location if the driver is sitting still). If the driver is enroute, "estimated time of arrival " (ETA) represents when he is expected to arrive at his destination. Locations can be represented at a number of different levels of aggregation. Equipment might capture the type of trailer (and even tractor) he is pulling. "Team?" is an indicator variable that tells us whether it is a single driver or a pair of drivers who trade off between driving and sleeping. The attribute a_7,"DOT hours" is actually a vector that tells us how long the driver has been driving today, how long he has been on-duty, and how many hours he was on-duty for each of the last seven days. These attributes are used to enforce federal Department of Transportation rules on how much a driver can work in a given day.

Similarly, we let $b = (b_1,b_2,...,b_m)$ be a vector of attributes describing a load. Attributes might include origin, destination, pickup and delivery time windows, equipment characteristics, shipper (or shipper priority), and any other information needed to describe the load.

We need to represent how many drivers and loads we have of each type. We let $\boldsymbol{A}$ be the set of all possible attribute vectors for drivers, $\boldsymbol{B}$ be the set of all possible load attributes. We then define:

$$\begin{aligned}
R_{ta}^D &= \text{Number of drivers of type } a \text{ at time } t. \\
R_t^D &= \left(R_{ta}^D\right)_{a\in A} = \text{Driver resource vector.} \\
R_{tb}^L &= \text{The number of loads of type } b \text{ at time } t. \\
R_t^L &= \left(R_{tb}^L\right)_{b\in B} = \text{Load resource vector.} \\
R_t &= \left(R_t^D, R_t^L\right) = \text{System resource vector.}
\end{aligned}$$

Throughout our discussion, R_t describes the state of all our drivers and loads at time t. We need to emphasize that in a software implementation, we would never explicitly store the entire vector R_t^D or R_t^L.

Instead, it makes more sense to define a set R_t^D where $r \in R_t^D$ is a particular driver with attribute vector a_r. However, the notation we have adopted will prove more convenient as we progress.

Rather than assigning a particular driver to a particular load, we adopt the convention that we are acting on a driver (or resource) with attribute a using a decision of type d chosen from a set of possible decision types, given by the set D. There are different classes of decisions, which we define using

D^L = Decisions to move a type of load. Each element of D^L corresponds to an element in the set of load attributes B.
D^M = Decisions to move empty to another location (perhaps in anticipation of loads that might become available in the future).
D^H = Decision to "go home" and go off duty for a period of time.
d^ϕ = The decision to "do nothing" (sit and wait)
$D = D^L \cup D^M \cup D^H \cup d^\phi$

This notation is useful since it allows us to easily add new decision classes (e.g., repair or clean a trailer, maintain a tractor) without fundamentally changing our model. The set D is all of our types of decisions. We then define

x_{tad} = Number of times we cat on a driver of type a with a decision of type d at time t.
$x_t = (x_{tad})_{a \in A, d \in D}$ = The decision vector.

As a rule, x_{tad} will be 0 or 1, but our notation allows us to aggregate drivers. $x_{tad} = 1$ when $d \in D^L$ means that we have assigned a driver to a load of type b_d. We also note that $x_{tad} = 1$ does not mean that we are moving a driver at time t. It only means we are making the decision at time t. We may be preassigning a driver due to arrive later in the afternoon to a load, but we are making the decision in the morning.

The effect of a decision is captured using the modify function, which we write as follows:

$$M(a,d) = (a', c, \tau)$$

The modify function is a set of rules that specifies that if we act on a driver with attribute vector a with a decision of type d, we produce a modified driver with attribute vector a' and generate a contribution (cost if we are minimizing) c, where the time required to complete the action is given by τ. The completion time τ is also captured by one of the attributes of a' (the "estimated time of arrival" field). It is also useful to define the

$a^M(a,d)$ = The terminal attribute function, which is the attribute a' produced by decision d.

$$\delta_{a'} = \begin{cases} 1 & \text{If } a^M(a,d) = a' \\ 0 & \text{Otherwise} \end{cases}$$

c_{ad} = Contribution generated by acting on a driver of type a with a decision of type d.

Which gives the attribute a' produced by decision d.

We can now state our basic problem (depicted in Fig. 15.1) as the following mathematical model:

$$\max_x \sum_{a \in A} \sum_{d \in D} c_{ad}\, x_{tad} \tag{15.1}$$

This is solved subject to the following constraints

$$\sum_{d \in \mathrm{D}} x_{tad} = R^D_{ta} \tag{15.2}$$

$$\sum_{a \in \mathrm{A}} x_{tad} \leq R^L_{tb_d} \qquad d \in \mathrm{D}^L \tag{15.3}$$

$$x_{tad} \leq 0 \qquad \text{(and integer)} \tag{15.4}$$

Equation 15.1 is our objective function, which we have written in the form of maximizing total contribution. Equation 15.2 is conservation of flow for drivers. Note that because "doing nothing" is an explicit decision, this must hold with equality. Equation 15.3 is conservation of flow for loads (we cannot move loads we do not have). Equation 15.4 states that flows must be nonnegative integers.

The optimization problem in 15.1–15.4 is powerful in part because it provides for a very high level of detail in the representation of drivers and loads, but it also has another extremely useful property. In practice, it is very common for issues not captured by the computer to prevent the assignment of a particular driver to a particular load. In commercial implementations, it is standard practice to produce a ranked list of options by using dual variables. Let v^L_a be the dual variable for each driver constraint 15.2 and let v^D_b be the dual variable for each load constraint 15.3. We can now compute the reduced cost of each decision using

$$\bar{c}_{tad} = c_{tad} + (v^L_{b_d} - v^D_a)$$

where b_d is the attribute of the load corresponding to decision $d \in D^L$. If $x_{tad} > 0$, then $\bar{c}_{tad} = 0$. If $x_{tad} = 0$, then $\bar{c}_{tad} \leq 0$, which means that contributions are reduced if we increase flow on these links. It is not unusual for $\bar{c}_{tad}$ to be zero (or very close to zero) for decisions that the model does not recommend, indicating that these are very close to being optimal. Errors in costs or missing data can easily change these recommendations, so we normally provide dispatchers with a ranked list of recommendations. In fact, it is common practice to choose a value, say \$10, where we would say that if the dispatcher chooses a decision where $\bar{c}_{tad} \leq \$10$, they we would say that the dispatcher "agrees with" the model.

15.3 Variations and Extensions

The initial appeal of the load-matching problem is quickly tempered by the realities of actual operations. In this section, we briefly review a number of operational issues that our basic model does not consider. We divide our discussion between more complex operational problems in Section 15.3.1, followed by a discussion of the challenge of working with forecasted demand.

15.3.1 More Complex Operational Problems

This section briefly describes some operational issues that arise in real applications that cannot be handled by our basic load-matching model.

15.3.1.1 Short-Haul Loads

The load-matching problem was first solved in the context of a long-haul truckload carrier. In this setting, loads typically require a day or more to complete. Since we cannot accurately predict what will happen a day from now, it makes sense to assign a driver to at most one load, and then wait until he is at least close to completing the load before assigning him to another load. Many trucking companies pull

a significant amount of short-haul movements, and even long-haul carriers have to execute a number of short movements. Since short movements can be completed quickly, we have the ability to plan a sequence of movements for a single driver through several loads. Figure 15.2 illustrates a problem with four drivers and six loads, three of which are quite short. If we use a load-matching model, where a driver can be assigned to at most one load, we obtain the solution in Figure 15.2a. If we plan a tour using all the loads we know about, we might obtain the solution in Figure 15.2b. The difference is significant, and remains a source of complaint about commercial load-matching systems.

15.3.1.2 Managing Trailers

It is common to assume that we are assigning "drivers" to "loads," but exactly what do we mean by a driver? We generally assume that a driver also has a tractor, but what about a trailer? A truckload carrier might have twice as many trailers as drivers. It is common, for example, for a driver to drop a loaded trailer in a shipper's lot, and then drive just his tractor to another shipper where he picks up a trailer that has just been loaded. At a later time, the first trailer will be unloaded, and either added to the shippers pool of trailers, or someone will have to pull the trailer out empty.

In the language of resource management, modeling just the drivers and loads represents a two-layer problem. If we explicitly model trailer inventories, we have a three-layer problem. Computationally, this can be much more difficult. It is fairly easy to model trailers in a simple way. For example, if a driver does not have a trailer, but the load requires that the driver bring a trailer to pick up the load, then the cost of assigning a driver to a load requires that we consider routing the driver through a trailer pool. But a real model of trailers would also make recommendations to move trailers from one pool to the next in order to manage trailer inventories.

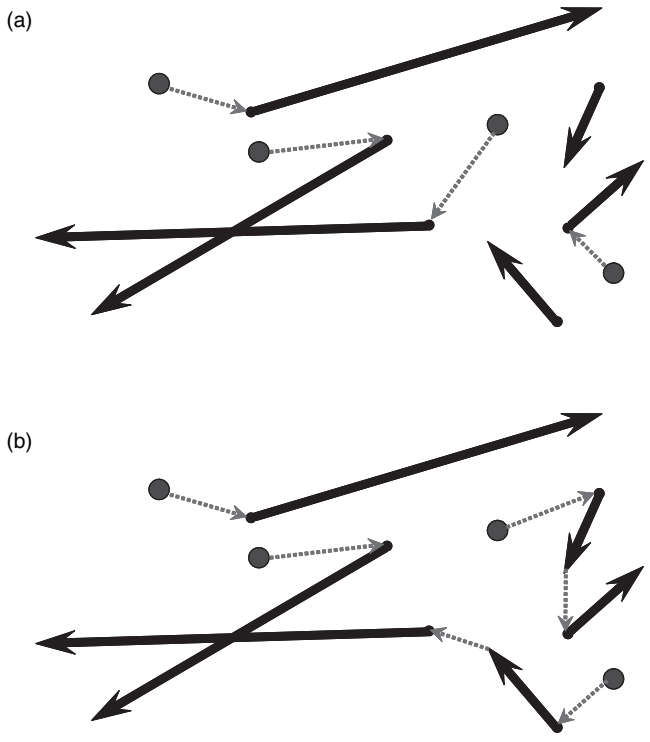

FIGURE 15.2 Assigning each driver to one load.

15.3.2 Looking into the Future

A second complication arises when decisions now have to consider what might happen in the future. There are three issues that we need to consider that require us to look into the future:

1. Should we commit to move this load? We often have the ability to decline loads (when they are first offered), and we have to think about whether we will have too many drivers at the destination of the load to use them productively.
2. Is this the right type of driver? Teams like long loads. Regional drivers like short loads that keep the driver close to home. A driver pulling a refrigerated trailer would like to move to locations where this is needed. Not every region has the same mix of freight.
3. Will we be able to get the driver home? A long-haul carrier might keep a driver away from home for several weeks. The carrier might like to get a driver home every week or two (less often for some drivers). What is the likelihood that we can get a driver to his home if we assign him to a particular load?

These issues require that we think about loads that we do not yet know about. In other words, we have to forecast future demands, a problem we address in Section 15.4. But it is not enough to forecast demands; we also have to think about how we might manage the drivers in the future. This problem is addressed in Section 15.5.

15.4 Forecasting Demand

Looking into the future requires that we estimate customer requests before they become known. This section provides a brief overview of issues that arise when forecasting for a truckload operation.

15.4.1 Elementary Forecasting

Let $\hat{D}_{tij}$ be a random variable representing the number of loads that will need to be moved from location i to location j on day t. $\hat{D}_{tij}$ is a random variable that we might assume takes the form

$$\hat{D}_{tij} = \mu_{tij} + \varepsilon_{tij} \tag{15.5}$$

where μ_{tij} is the mean of the random variable and ε_{tij} is a random error around the mean which we assume has zero mean and some variance. We will never be able to guess ε_{tij}, but we would like to try to estimate μ_{tij}. There is a vast array of forecasting techniques, but the simplest and most widely used is exponential smoothing. Let $\bar{\mu}_{t-1,ij}$ be our estimate of the mean after day *t-1*, and let $\hat{D}_{tij}$ be our observation of the actual demand on day t. Using exponential smoothing, we would update the mean using

$$\bar{\mu}_{t,ij} = (1-\alpha)\bar{\mu}_{t-1,ij} + \alpha\hat{D}_{tij} \tag{15.6}$$

where α, a parameter between 0 and 1, is variously known as the smoothing parameter, learning rate or stepsize.

If only forecasting were this easy. One challenge is that it is very common to have forecasts (for the flow from a particular origin to destination) to be a fraction less than one. Carriers will often forecast the total loads out of an origin, but this means they have no idea where the loads are going. This problem arises because of the common misconception that to forecast demand means to estimate μ_{tij}, which is known as a point forecast. More modern tools forecast the actual distribution of demand (e.g., the probability that there will be 5 loads from i o j).

The following two sections address two important issues: (1) the challenge of forecasting daily demand and (2) methods for handling advance bookings.

15.4.2 The Challenge of Forecasting Daily Demand

In an operational model, it is necessary to forecast demand on a particular day. This introduces not only the problem of handling day-of-week effects, but also the more challenging problem of holidays and other "special days"—end of month, end of quarter, the Monday after Thanksgiving, the fifth of July when the fourth of July is on a Thursday, and so on.

It is popular in industry to use techniques such as averaging the last four Mondays to forecast the next Monday. This might capture day of week effects, but requires that we go back a month, which means we might be missing out on seasonal effects such as those that typically arise around the Christmas season. It also ignores holidays and special days. An alternative that we have found effective is to use a model of the form (dropping the indices *i* and *j*):

$$\bar{\mu}_t = b_t \theta_t^{dow} \theta_t^{wom} \theta_t^{sd} \tag{15.7}$$

where b is the baseline, θ_t^{dow} is a day-of-week adjustment factor, θ_t^{wom} is a week-of-month adjustment factor, and θ_t^{sd} is a factor for special days. For example, $\theta_t^{dow} = 1.07$ if the day-of-week effect for day t is 7% higher than normal. θ_t^{sd} (is particularly challenging to estimate, since we might observe a particular "special day" only once a year. [see Godfrey and Powell (2000) for methods to update this model]

15.4.3 Handling Pre-Booked Loads

A particular challenge in forecasting demand in truckload trucking is the fact that customers will book orders in advance. We refer to a demand process where there is a gap between when we know about the demand, and when we can act on it, as a lagged information process. This is modeled using the notation:

$\hat{D}_{tt'}$ = The number of new demands that first become known between t–1 and t that need to be served at time t'.
$f_{tt'}$ = Point forecast of $\hat{D}_{tt'}$, made before time t.
$F_{tt'}$ = Forecast of the total demand for time t' using the information available at time t.

Here, time t refers to both a day as well as time of day. If today is Tuesday, our forecast of loads on Thursday depends on whether it is 10:00 a.m. on Tuesday or 4:00 p.m. Because of the need to have a truly continuous time forecast, we have to view the time index t as being continuous. However, when we prepare a forecast of the total loads to be served on Thursday, time t' needs to be viewed as an entire day. We have generally found it best to first forecast assuming time t represents an entire day, and then forecast an hour-of-day distribution. Thus, if we forecast that 10 loads will be called in on Tuesday to be served on Thursday, we can use a separate hour-of-day distribution to determine how many of these loads would be known by 11:30 a.m. on Tuesday.

It is important to phase in known demands with your forecast, so that at a time t, we take advantage of what we know with what we do not yet know. The biggest mistake is to forecast demand, and then update the forecast by subtracting what is already known. To illustrate, we might forecast that we will pick up 40 loads in a region. Assume we already have 27 booked loads. It is tempting to update the forecast so that we assume that we have 27 known loads and 40 − 27 = 13 forecasted loads, giving us a total forecast of 27 + 13 = 40. We hope the error in this process is apparent.

There are two methods for phasing in known demands, and we generally have to use both. The first is primarily suited for phasing in demands from numerous small customers, while the second is particularly important when phasing in demands from a small number of large customers.

15.4.3.1 Forecasting Small Customers

When we forecast demand from numerous small customers, we assume they behave like a model known as a Poisson process. This model assumes that the number of calls made, say, before noon on Tuesday, is

completely independent of the number of calls made afternoon on Tuesday (or the number of calls yet to be made on Wednesday). For this discussion, assume that we are forecasting the total demand on *day* t' in the future, but that we are forecasting demand at hour t, since as a rule, we need to update forecasts continuously during a day.

$f_{tt'}$ is our estimate of the total number of phone calls that will arrive between t-1 and t. At time t, our forecast of the total number of loads that have to be served at time t' is given by

$$F_{tt'} = \sum_{t'' \le t} D_{t'',t} + \sum_{t'' > t} f_{t'',t} \tag{15.8}$$

Thus, our forecast of t' combines what we know as of time t, plus a forecast of what is not yet known. Note that this only produces a point forecast. We can also produce estimates of the variance of the forecast.

15.4.3.2 Forecasting Big Customers

The method described in the previous section does not work for large customers who tend to make a single phone call at some point during the day. They may make their phone call early in the day, but other days the phone call may come in later. The way that Equation 15.8 merges known and forecasted demands does not work for this type of process. Instead, it is better to have a separate forecast for each of these big accounts. Since we know when the account has made its orders known, we can simply use the forecasted before the orders are entered, and use the actual after the orders are entered.

15.5 Capacity Forecasting

Now that we have a basic method for forecasting demand, we next have to forecast capacity, which is the movement of trucks in the future. This is important if we want to know, for example, if we have too many trucks in a region compared to the demand. It is also needed if we want to estimate our ability to get drivers home.

Capacity forecasting is subtle. Older models would project capacity into the future by solving a large space–time network such as that depicted in Figure 15.3, where nodes represent points in space and time, solid links represent the movement of loads (known or forecasted), and dashed lines representing either holding in a location, or moving empty. A point in space is typically one "region," where the continental United States might be divided into 100 regions. **Q1**

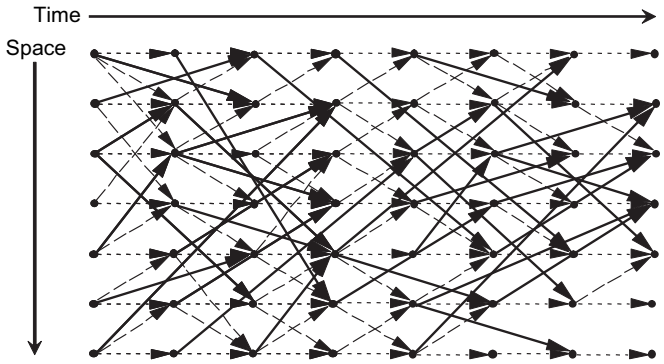

FIGURE 15.3 Illustrative space-time network.

The advantage of space-time networks is that they are easy to understand and communicate, and can be solved as linear programs using commercially available solvers. The problem is that they can do an extremely poor job of modeling the real system, seriously biasing the forecasts. One problem is that they completely ignore the uncertainty in demand forecasts, where the number of forecasted loads from one location to another is likely to be a fraction such as 0.17. The second and perhaps more serious error is that the network models the flow of trucks, not drivers, and is unable to capture limits on how much a driver can move, or any other issues such as getting a driver home. We have found that such models can dramatically overestimate the capacity of a fleet, disguising capacity problems in the future.

In the sections which follow, we briefly outline a strategy that is able to forecast capacity into the future, without any loss of detail in how drivers are represented. Section 15.5.1 begins by describing a simple, myopic simulator. Section 15.5.2 shows how we can take this basic simulator and produce a solution that looks into the future.

15.5.1 Simulating a Myopic Policy

The easiest way to forecast capacity in the future is to simply simulate the dispatch process, which we can do by applying the model we first introduced in Section 15.2 iteratively into the future. To describe this more formally (which we need for our discussion in Section 15.5.2), let $X_t^\pi(R_t)$ be a function that represents solving the load-matching optimization problem given by Equations 15.1 through 15.4 at time t, producing a vector x_t that satisfies the constraints 15.2 through 15.4. The problem is solved given the resource state vector R_t that tells us the status of all drivers and loads at time t. The superscript π is an index in a set Π so that we can represent the fact that this is not one function, but a family of functions from which we can choose (often referred to as policies). The model in Section 15.2 is one of these models which we represent using $\pi = \mathrm{M}$, where M denotes a myopic policy (i.e., a rule for making decisions that ignores the future).

In a real-time system, we would solve Equations 15.1 through 15.4 where $t = 0$. In this case, $R_0 = (R_0^D, R_0^L)$ tells us the status of all the drivers and loads that we know about right now. We again emphasize that this does not mean all drivers and loads that can be assigned right now. The driver's ETA attribute, and the pickup window of a load, may specify that a driver may not arrive for two days, and the load has to be picked up a week from now. Solving $X_0^\pi(R_0)$ returns a decision vector x_0. We can now combine this information with our modify function. If $x_{0ad} = 1$ (which means that $R_{0a} > 0$), then we now have a driver with attribute $a' = a^M(a,d)$. We write the effect of these decisions on our resource state vector using

$$R_{0a'}^{D,x} = \sum_{a\in\mathrm{A}}\sum_{d\in\mathrm{D}} x_{0ad}\,\delta_{a'}(a,d) \tag{15.9}$$

We refer to $R_0^{D,x} = (R_{0a}^{D,x})_{a\in A}$ as the post-decision resource vector for drivers. Equation 15.9 describes how our decisions impact drivers. We also have to model the effect of decisions on loads, which is quite simple. If we assign a driver to a load, the load leaves the system; otherwise it remains in the system, a process that is written as

$$R_{0b_d}^{L,x} = R_{0b_d}^{L} - \sum_{a\in\mathrm{A}} x_{0ad} \qquad d \in D^L \tag{15.10}$$

Now we are going to make the transition from $t = 0$ to $t = 1$ which might represent, for example, a point in time 4 hr later. During this time, we might have a set of phone calls. Earlier, we represented new demands using $\hat{D}_t \cdot$ Here, we slightly revise this notation so that our model can easily handle phone calls that provide updates to demands (new orders, changes in orders) as well as updates to drivers (drivers

being added to the system, drivers leaving the system, and updates to existing drivers, such as delays in arrival times). These updates are represented as the following random variables:

$\hat{R}^D_{ta}$ = Change in the number of drivers with attribute a due to exogenous information that arrived between $t-1$ and t.
$\hat{R}^L_{ta}$ = Change in the number of loads with attribute b due to exogenous information that arrived between $t-1$ and t.
$\hat{R}_t = \left(\hat{R}^D_{ta}, \hat{R}^L_{ta}\right)$ = Exogenous information arriving between $t-1$ and t.

We note that $\hat{R}_t$ is a function that depends on the drivers and loads already in the system, as well as exogenous information that arrives to tell us how the system is changing. With this notation, we can describe how our system evolves forward in time:

$$R^{\mathrm{D}}_{t+1,a} = R^{D,x}_{ta} + R_{t+1,a} \tag{15.11}$$

$$R^{L}_{t+1,a} = R^{L,x}_{ta} + R_{t+1,a} \tag{15.12}$$

Equations 15.9 through 15.11, combined with 15.11 through 15.12, tell us how the resource state vector R_t evolves from time t to time $t+1$, given a decision x_t and the exogenous information $\hat{R}_{t+1}$.

We know how to find x_t (by solving the Equation 15.1 through 15.4. How do we actually obtain $\hat{R}_{t+1}$? We do this by forecasting future updates to the system, and then randomly sampling from this forecast. In a basic model, we might ignore any random events happening to drivers, and simulate only the random arrival of new customer orders. If we have forecasted, say, 0.20 orders will arrive to move from Dallas to New York this Thursday, then we would generate a random integer whose mean is 0.20. For example, we might generate a random variable between 0 and 1; if the random variable is less than 0.20, then we would set the random demand to 1, and otherwise set it to 0. To allow for means greater than 1, we might treat the forecast as the mean of a Poisson random variable and sample from this. A number of popular simulation textbooks describe this process.

Since we have to sample randomly, it is generally a good idea to perform repeated sample realizations and average any statistic that is desired from the model. Let $\hat{R}^n_t$ be the nth sample of the random information that arrived between $t-1$ and t. We refer to the sequence $\left(\hat{R}^n_1, \hat{R}^n_2, \ldots, \hat{R}^n_t, \ldots, \hat{R}^n_T\right)$ as a sample path, which is to say a single set of realizations over all the time periods we are interested in.

We now have a process for simulating our system as far into the future as we would like. The only weakness is that our decision function $X^\pi_t(R_t)$ always ignores the impact of decisions now on the future. The following section addresses this problem.

15.5.2 An Approximate Dynamic Programming Solution

There is a simple way to make our myopic decision function much more sophisticated. Instead of solving the objective function given by Equation 15.1, assume instead that we solve the problem

$$\max \sum_{a\in \mathrm{A}} \sum_{d\in \mathbf{D}} c_{ad} x_{tad} + \bar{V}_t(R^x_t(R_t, x_t)) \tag{15.13}$$

where $\bar{V}_t\left(R^x_t(R_t, x_t)\right)$ approximates the value of being in resource state $R^x_t(R_t, x_t)$ which depends, of course, on R_t and x_t. For this class of problems, a reasonable approximation is a linear function, given by

$$\bar{V}_t(R^x_t(R_t, x_t)) = \sum_{a'} R^{D,x}_{ta'} \bar{v}_{ta'} \tag{15.14}$$

where $\overline{v}_{ta'}$ can be thought of as the marginal value of drivers with attribute a'. Combining 15.9 and 15.14, and using the definition of our terminal attribute function $a^M(a,d)$, allows us to write

$$\begin{aligned}\overline{V}_t(R_t^x(R_t,x_t)) &= \sum_{a'\in A}\left(\sum_{a\in A}\sum_{d\in D} x_{tad}\delta_{a'}(a,d)\overline{v}_{ta'}\right)\\ &= \sum_{a\in A}\sum_{d\in D} x_{tad}\overline{v}_{t,a^M(a,d)}\end{aligned} \tag{15.15}$$

Combining 15.13, 15.14, and 15.15, with a bit of algebra, produces

$$\max \sum_{a\in A}\sum_{d\in D} c\left(c_{ad} + \overline{v}_{t,a^M(a,d)}\right)x_{tad} \tag{15.16}$$

which we solve subject to the flow conservation constraints 15.2 through 15.4. The only question we have not answered is: how do we get the values $\overline{v}_{ta'}$?

Fortunately, this is the easy part. When we solve problem (15.1)–(15.4), or problem (15.16) subject to (15.2)–(15.4) using any commercial linear programming package, we also obtain a dual variable that for the flow conservation constraint on drivers Equation 15.2, that tells us the marginal value of a driver with attribute a. We have to keep in mind that we are solving these problems iteratively, where at iteration n we use the sample realization $\hat{R}_t^n$. Let $\overline{v}_{t-1,a}^{n-1}$ be our estimates of the marginal values after iteration $n-1$. Let $\hat{V}_{ta}^n$ be the dual variable we obtained during the nth iteration of our simulation. We then apply exponential smoothing to obtain our value function approximation, using

$$\overline{v}_{t-1,a}^n = (1-\alpha)\overline{v}_{t-1,a}^{n-1} + \alpha v_{ta}^n \tag{15.17}$$

We note that $\hat{v}_{ta}^n$ is used to update $\overline{v}_{t-1,a}^{n-1}$. For further background on this subtle bit of modeling, see the discussion in Powell et al. (2006).

Recall that the attribute a can be quite complicated. Although we do not write it explicitly, while $\hat{v}_{ta}^n$ will depend on the full attribute vector, our value function approximation $\overline{v}_{t,a}^n$ depends only on a subset of attributes such as location, the domicile of the driver and perhaps his equipment type.

15.5.3 Getting Drivers Home

One of the real challenges of load-matching models is getting drivers home. While this can be quite difficult, our dynamic programming approximation of the previous section already accomplishes this for us, as long as we retain driver domicile as one of the attributes of the value function. In addition, it is also necessary to include logic in the load-matching problem that recognizes that a driver may be close to his home, or is assigned to a load that allows a driver to pick up a load, move to his home, spend a day or two and still deliver on time. The model must include rewards for getting drivers home, or penalties for keeping drivers away from home.

15.6 Demand Management

The focus of real-time dispatch systems is typically on what we should do with a driver, but it is usually the case that we can have a much bigger impact on the company by controlling which loads are accepted. This is particularly true since many orders are booked several days in advance.

Carriers typically accept or reject loads based on issues such as, (1) Is this a major account? (2) Is the rate being offered (often expressed in units of dollars per mile) above a minimum? (3) Do I have enough

capacity (relative to demand) out of the region where the load originates? and (4) If I accept the load, will this create a situation where I have too many drivers at the destination of the load?

The capacity forecasting logic described in Section 15.5 can produce estimates of the number of drivers (and loads) out of an origin or into a destination several days into the future. In addition, it can even account for the fact that while we may be sending too many drivers into New Jersey, we have a shortage of drivers in nearby eastern Pennsylvania, which means that we may have more capacity in a region than would be forecasted if we simply add up the number of loads terminating in a region.

15.7 Implementation

The use of computers to not only store information about drivers and loads, but also to recommend how drivers should be managed, seems like it should be a major application of operations research. A number of issues have resulted in extremely slow adoption. One problem is that current commercial packages do not handle problems such as trailers, routing drivers through a sequence of several loads, and the uncertainty of forecasts in the future.

15.7.1 Computer Integration

A real-time dispatch system requires having up-to-date information about drivers and loads, information that carriers enter into the computer. There are a number of commercial management information systems designed specifically for truckload motor carriers, and as of this writing, none include an automated driver assignment module. The reason for this is simple. There are thousands of trucking companies with less than 50 drivers, and this is the market when a company starts to use computers. It is only when a company gets a fleet of at least 200 drivers that an automated dispatch system starts to make sense.

Since the 1980s, a small cottage industry emerged to install real-time load-matching systems for truckload carriers (two of these companies, Princeton Transportation Consulting Group and Transport Dynamics, were founded by students under the supervision of this author). Without question, the Achilles heel of these systems was the interface between the optimization model and the dispatch system. To be successful, this interface has to be seamless, with rapid transmission of information, something that has been achieved only in highly customized applications (and very high cost).

15.7.2 The Problem of Data

Not surprisingly, automated dispatch systems depend on quality data. But data "errors" can be quite subtle. Consider the problem of assigning five drivers to five loads as depicted in Figure 15.4. Take a minute to solve the problem in your mind before turning the page.

The mathematically guaranteed, optimal solution provided by the computer is given in Figure 15.5. This may surprise the reader, but it is because the reader has not been provided all the information. The problem is first noticed by the company (which cannot look at the entire solution) when driver B calls in and the dispatcher sees that he has been assigned to load 1 instead of load 2, which seems closer. The problem, actually, is with the data associated with load 3. When this load was first called in, the traffic manager asked "Is it possible to pick the load up before noon? We get busy in the afternoon." Dispatch systems allow the specification of pickup time windows, but these are hard constraints; failure to pick the load up within the window is viewed as a service failure which can lead to the loss of the contract. Only driver A in this group is arriving early enough to meet this constraint. As we can see, however, the request to pick it up before noon was only a preference.

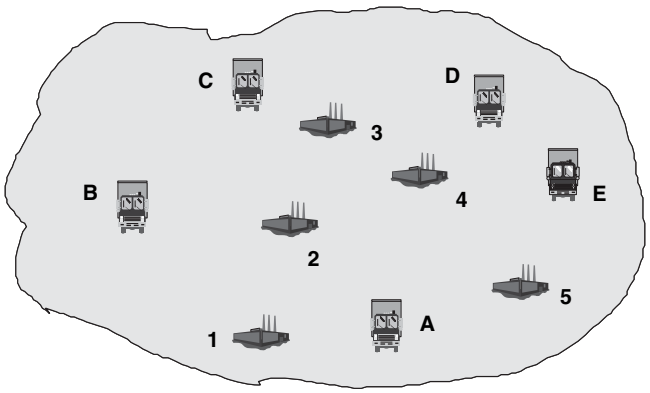

FIGURE 15.4 A driver assignment problem.

15.7.3 Measuring Compliance

Dispatch systems are successful only if people actually do what the systems tell them to do. Not surprisingly, it is common to measure compliance, which is a statistic that describes the number of driver assignments where the decision made by the dispatcher agrees with the model. Unsuccessful implementations (where the company runs the model but no-one uses the recommendations) tend to observe around 30–35% compliance. Carefully calibrated models with strong management support might average 80–85% compliance (with some users over 90%). However, users can manipulate these numbers if they feel they are being judged by them.

The issue of user noncompliance has received relatively little attention from the academic community since it is an issue that only arises in a field implementation. Powell et al. (2000) studied the effect of user noncompliance. After solving the myopic optimization model defined by equations 15.1 through 15.4, the value of assigning a driver of type a to a load of type b is given by a formula based on the reduced cost:

$$\bar{c}_{ab_d} = c_{ab_d} + \theta\,(v_a^D - v_{b_d}^L)$$

where c_{ab_d} is the direct contribution of assigning a driver of type a to a load of type b_d, v_a^D is the dual variable for the driver node of type a (the dual for Equation 15.2), v_{bs}^L is the dual for the load node for a load

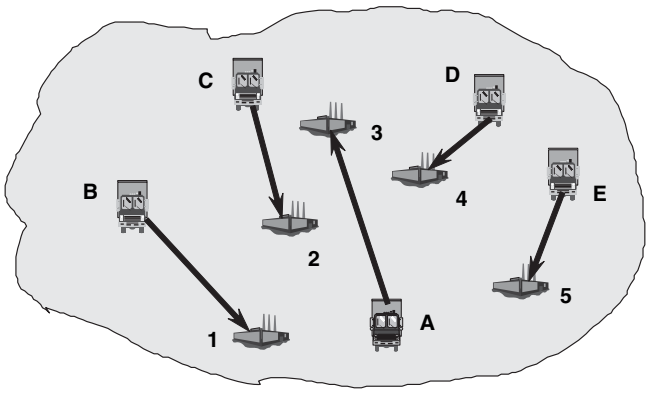

FIGURE 15.5 The computer generated solution.

of type b, and θ is a scaling factor. Our rule is to assign a driver to the load with attribute b_d with the highest value of $\bar{c}_{ab_d}$. We can model user compliance by randomly deciding whether the recommendation is "acceptable" to the dispatcher; if this assigned is (randomly) judged to be unacceptable, we go to the second-ranked load, and so on. If we are modeling perfect user compliance, and if $\theta = 1$, then we are implementing basically the same solution recommended by solving the global optimization model in Equations 15.1 through 15.4. If we use $\theta = 0$, then we are ignoring the dual variables, and implementing a greedy solution where we do the best for each driver. Intermediate policies are obtained using $0 < \theta < 1$. Figure 15.6 shows total profits as a function of the level of user compliance for $\theta = 0$, 0.75, and 1.00. Note that at an 80% compliance rate (considered quite good in actual applications), the difference between the globally optimal solution and the greedy solution is not large, but there is a noticeable improvement if we use $\theta = 0.75$.

The issue of user compliance is a serious one. We have to recognize that the computer simply does not have all the information needed to make perfect decisions. We have often found that senior management is attracted to models since it provides them some level of control over the decisions made on the dispatch floor. One vice president used the term "dispatcher savant" to describe talented people in operations who otherwise could not be controlled. Responding to the challenges of changing operating philosophies on the dispatch floor, another senior manager remarked "Sometimes, when you can't change the people, you have to change the people." The problem that managers face is identifying the best dispatchers. Each manager works with a different region of the country, or different groups of drivers, making direct comparisons impossible. Too much emphasis on user compliance produces behavior where dispatchers "game the system," manipulating the process to produce the best score. Despite these qualifications, automated systems can add real value in the following ways:

- While it is possible to put too much emphasis on "matching the computer," it is generally the case that the best dispatchers have the highest compliance, but it is very important that the model be of high quality, capturing most operational situations. The early commercial models, despite their tremendous promise, did possess serious limitations.
- Models provide a useful benchmark. Comparing user performance (empty miles, on-time service, getting drivers home) against model performance for the same region and/or the same

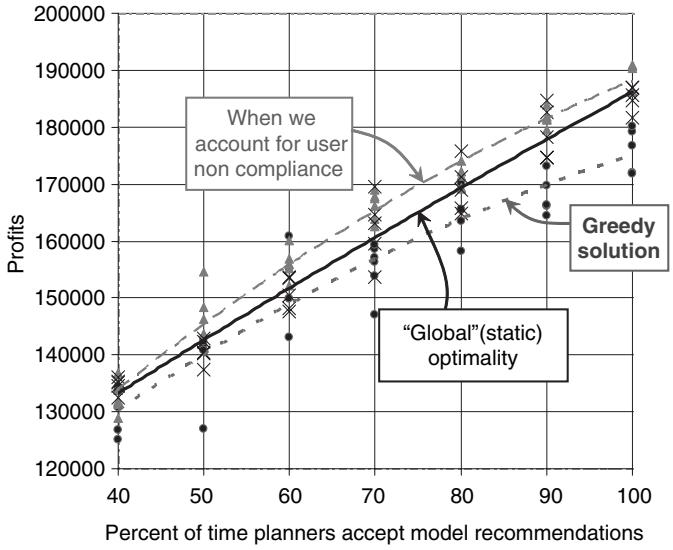

FIGURE 15.6 Value of global optimization in the presence of user noncompliance.

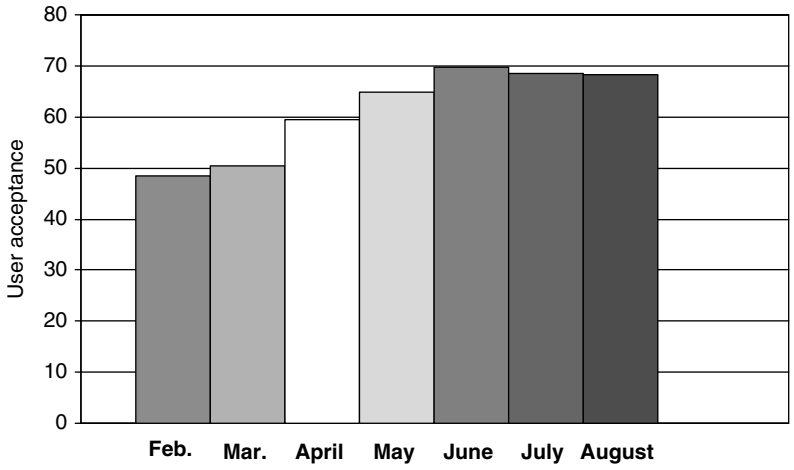

FIGURE 15.7 Evolution of system usage over first six months.

group of drivers, provides a benchmark that adapts to the unique situations faced by each dispatcher.

- Filling in for vacations and departures—Models can be of significant value during times such as when a dispatcher leaves or takes a vacation. Experienced dispatchers can often outperform a model, but the model can provide a genuine safety net for new people.

15.8 Case Study—Burlington Motor Carriers

I knew the Burlington Motor Carriers (BMC) from when they had four or five employees, until they finally closed their doors. The company was started in the mid-1980s by the Burlington Northern Railroad by Dr. Michael Lawrence, a Ph.D. economist with a dream of merging a series of truckload carriers to gain the economies of larger networks. Dr. Lawrence hired Michael Crowe from Schneider National, then (and now) the nation's largest truckload motor carrier, and one with a long history of innovation. Mike had a Master's in Operations Research, and had acquired during his years at Schneider a vision of how operations research could be used to help run a truckload operation.

The story of BMC unfolded in two acts, which we refer to as "Round I" and "Round II."

15.8.1 Operations Research Models—Round I

The first attempt to implement operations research models at BMC occurred while the company was first being formed, combining an entirely new information system with the purchase of four or five companies with established operations. Mike Crowe's vision of what models to use and how to use them demonstrated a deep understanding of the right way to use models within truckload trucking. Rather than focus on using models to assign drivers to loads (a technology pioneered by Richard Murphy at Schneider National), the vision was to focus more on demand management and capacity management. You can have a much greater impact by managing demand than managing capacity. This was aligned with some of our recent research at Princeton (see Powell, 1987), which focused more on looking into the future (while capturing uncertainty) and less on modeling individual drivers. The ideas were embodied in a software package we dubbed LOADMAP, which was also implemented at the same time at North American Van Lines [Powell et al. (1988)].

The system that Mike Crowe had designed was brilliant, and at least 10 years ahead of its time. Although the strategy emphasized network-wide profits rather than micro-level decisions, it was still important to know basic data such as where the driver was located, where he was headed to, and when he was likely to arrive. Today, many long-haul carriers use satellite systems to provide two-way communication with drivers, but this did not come until the late-1980s. The computers and communication technologies we take for granted today were just being invented in 1985. I realized the project was headed to failure when I met Mike after a conference call with field operations, and listened to his frustration getting basic data such as when a driver might arrive at the destination.

15.8.2 The Real-Time Dispatch System—Round II

In 1994, under new management, BMC again attempted a project to perform real-time dispatch. Before describing the details of the project, some background is needed.

15.8.2.1 A Bit of History

Our first efforts at fleet management focused on more aggregate level capacity measurements—how many drivers were in a region, how many loads were booked out of a region, and how many loads were booked into a region. Our model would make recommendations such as "move two drivers loaded from region A to region B." These instructions were met with complete mystery by the dispatchers, who would immediately respond "which two drivers?" For them, every assignment was unique. This driver needed to get home to Dallas, another driver was supposed to be available at 2:00 p.m. but was notoriously unreliable, a third driver needed to go on rest for the remainder of the day because he had hit the limit on the number of hours he could drive. Regions were good for planning, but it made a big difference if a driver was on the northern boundary of eastern Pennsylvania or was south of Philadelphia (all in the same region). A driver might be sitting in a yard staring at a trailer in the process of being loaded that had not yet been entered into the computer system. It may be that aggregate, network-level flows drives the economics of a carrier, but the devil is in the details, and our first effort completly ignored this. **Q2**

By the early 1990s, BMC had moved to its third headquarters, a low-rent office building in a cornfield north of Indianapolis. With a much lower cost structure, the company began to thrive, growing to a respectable $300 million annual revenue and a fleet of about 500 trucks. Under new management, I was contacted again, but this time to implement a real-time dispatch system. Knowing many of the uncertainties, I offered to take on the task as a research project through Princeton University, which introduced its own special issues. While negotiating the contract, the Princeton University grants office first insisted that BMC was welcome to use the results of our work in their research, but had to pay royalties if they wanted to actually use the system. After getting over the vision of a small trucking company writing journal publications, I had to explain that the project was field research. I wanted to observe the process of implementation, take measurements, and publish the results. This is exactly what happened, and the results were published in the prestigious journal *Operations Research* [Powell et al. (2002)].

15.8.2.2 The Dispatch System

Through the 1980s and 1990s, one of our most significant achievements was the development of a model which combined the real-time assignment of individual, rather than aggregated, drivers and loads which also looked into the future and captured the uncertainty of future demands. In a separate breakthrough (at the time), we also found a way to route drivers through a sequence of two or more loads, rather than restricting each driver to being assigned to at most one load [Powell et al. (2000)]. The challenge was solving these problems in real-time (updates could not take more than a second or two), using available computers and algorithms. Just as important—it was not enough just to tell dispatchers what load a driver should pick up, it was still necessary to provide a ranked list of options, a feature from the basic load-matching model that was critical to field implementation.

The project proceeded smoothly. BMC's capable vice president of information technology, Mr. Robert Lamere, handled all aspects of the interface between the corporate information system and the model. However, due to the nature of the technology, the resulting solution was hardly pretty. As with many truckload carriers, BMC used a very popular computer developed by IBM called an AS/400, a fantastic machine for processing data but which would not run languages such as C or C++. The preferred implementation platform for models at the time was Unix-based machines, in our case a Silicon Graphics workstation. Bob worked out a solution where data would be passed from the AS/400 to a PC which then talked to the Unix workstation. Ugly, but it worked. The system went into production about a year after the project started, and we began the painful process of gaining user acceptance.

15.8.2.3 Oh, but could You Help us with…

In the middle of the project, I received a call from senior management asking if I could take a look at their network profitability and pricing policies. I had developed a model for this purpose that had been applied to several other carriers, and I offered to apply it to their network. Normally the model would produce a profit-and-loss statement fairly close to actuals, but after several weeks of fiddling, I had to call to tell them that the model did not seem to be working. It was producing results where the operating ratio (total costs over total revenue) was 110 (i.e., costs were running 10% higher than revenues)! I was politely informed "that was about right" !!!.

I learned in that phone call that the reason the company had funded the dispatch project was a belief that the reason their costs were 10% higher than revenues was problems with the dispatch floor. By this time, we knew that dispatch systems could reduce operating costs by 1% or 2%. They do this by reducing the empty miles traveled, but these are typically only 10% of total miles traveled. A 1% or 2% reduction in total costs is a big number if your profit margin is only 3% or 4%. But they will never drop costs by 10%. The phrase "Houston, we have a problem" was appropriate here.

We had to put the dispatch project on the shelf for about six months to focus on their profitability problem. Truckload trucking is famous for lines such as "we lose money on every load but make it up in volume." Approximately 95% of the total costs of a truckload operation are what economists would call short-term variable costs (i.e., directly related to miles traveled and the size of the fleet). BMC did not have an operational problem. They had a pricing and marketing problem. They were carrying the wrong loads at the wrong price.

Using my network planning model, BMC shrank the company by over 25%, reducing their operating ratio from 110 to below 100 (which is to say, profitable). This is extremely difficult in most companies, but is surprisingly easy to do in the truckload industry. It is not that hard to shrink a company to profitability as long as assets such as fleets are also reduced. Of course, corporate overhead has to be reasonable. With the big problem solved, we returned to the dispatch system.

15.8.2.4 Implementing the Dispatch System

The remainder of the project progressed smoothly. System compliance was tracked daily, and dispatchers were given bonuses for higher levels of compliance. The vice president of operations particularly enjoyed the sudden control he was given over the behavior of the dispatchers. A major frustration in managing a room full of dispatchers is that they are notoriously independent. It is important for a company to balance on-time service, operating costs (measured primarily through the miles a driver moves empty to pick up a load), equipment productivity (often miles per driver per week), and keep drivers happy (which translates to putting them on long loads and getting them home on time). Not surprisingly, these goals are often competing, and management may decide to shift emphasis from one goal to another as business conditions warrant. The model, however, allowed management to easily raise or lower penalties and bonuses to emphasize different management objectives, and Burlington Management took advantage of this feature.

This control, however, proved somewhat illusory. One week we found that system compliance had dropped noticeably. We found that some of the parameters controlling the importance the model puts

on these soft issues had been adjusted somewhat dramatically. We reset the parameters (we had direct access to the Silicon Graphics workstation where the model was run) and called the vice president of operations to explain that optimization models are a little like artificial hearts. You can use these machines to increase or decrease the speed of a patient's heart, but it is also a nice way to give the patient a stroke. Models work the same way. Just because you adjust the parameter does not mean that all the dispatchers adjust equally quickly.

15.8.2.5 Measuring the Impact

We carefully documented the impact of the model by collecting an extremely valuable dataset. Since the model ran in real-time, we not only received updates to drivers and loads, we were also given, in real-time, actual assignments of drivers to loads. We stored every transaction during a day for approximately 50 days out of a 6-month period. This also allowed us to make changes to the model, and then re-simulate an entire day using actual transactions. With this system, we could make certain measurements from actual decisions and compare them to what the model recommended at that point in time. We measured empty miles, on-time performance and our ability to get drivers home on time. Figure 15.8 reports reduction in empty miles and improvement in getting drivers home for each day that we captured. We found that the model consistently produced improved performance.

15.8.2.6 Prologue

At first, this seemed like a completely successful project in every measure. In addition, one of my graduate studies, Derek Gittoes, had just started a new consulting firm called Transport Dynamics to implement and maintain these systems. Although the project was quite successful, they did not want to maintain an ongoing research relationship, and a university was not the right organization to provide maintenance and support. By this time, the company was convinced that it was not possible to run a profitable truckload carrier without models to identify profitable accounts and to help the dispatchers. But facing continuing cost pressures, they decided to move forward with the load profitability and driver dispatch systems without maintenance.

A few years later (post year 2000) I received a call saying that the system had not survived the post-2000 transition. Our model was specifically designed to be Y2K compliant (i.e., we could handle dates in two-digit format, such as 00, 01, 02), but we suspected a problem either in the PC interface between the AS/400 and the workstation. But when I asked what their level of user compliance was, they reported that it had dropped to around 35%, a level that I understood to mean that they were no longer using the dispatch system. Knowing that they did not have a budget for maintenance, I recommended that they simply shut off the dispatch system, which they did.

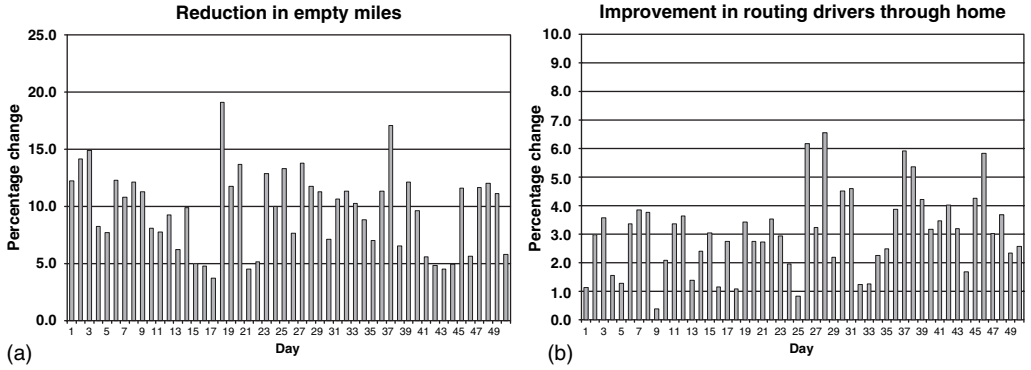

FIGURE 15.8 Day-by-day comparison of model recommendations against actual performance for (a) reducing empty miles and (b) getting drivers home.

Three months later, I received another phone call from the president asking how we could get the system running again. The problem, he explained, was that while the experienced dispatchers seem to do fine, they "got killed" each time one of them went on vacation or left the company. This was the first time I had solid evidence that a real-time dispatch system offered the kind of value that a manager could clearly and unambiguously recognize.

With a renewed commitment to the use of models, the company looked at bids from Transport Dynamics and a competitor. They had just hired a new manager from the competitor, and not surprisingly decided to go with the competitor. BMC closed its doors and liquidated two years later. Sigh.

References

Godfrey, G. and W.B. Powell, "Adaptive Estimation of Daily Demands with Complex Calendar Effects," Transportation Research, Vol. 34, No. 6, pp. 451–469 (2000).

Powell, W.B. "An Operational Planning Model for the Dynamic Vehicle Allocation Problem with Uncertain Demands," Transportation Research , Vol. 21B, No. 3, pp. 217–232 (1987).

Powell, W.B., Y. Sheffi, K. Nickerson, K. Butterbaugh, and S. Atherton, "Maximizing Profits for North American Van Lines' Truckload Division: A New Framework for Pricing and Operations," Interfaces, Vol. 18, No. 1, pp. 21–41 (1988).

Powell, W.B., "A Stochastic Formulation of the Dynamic Assignment Problem, with an Application to Truckload Motor Carriers," Transportation Science. Vol. 30, No. 3, pp. 195–219 (1996). **Q3**

Powell, W.B., W. Snow, and R. K.-M. Cheung, "Adaptive Labeling Algorithms for the Dynamic Assignment Problem," Transportation Science, Vol. 34, No. 1, pp. 67–85 (2000). **Q4**

Powell, W.B., M.T. Towns, and A. Marar, "On the Value of Globally Optimal Solutions for Dynamic Routing and Scheduling Problems," Transportation Science, Vol. 34, No. 1, pp. 50–66 (2000). **Q4**

Powell, W.B., A. Marar, J. Gelfand, and S. Bowers, "Implementing Operational Planning Models: A Case Application from the Motor Carrier Industry," Operations Research, Vol. 50, No. 4, pp. 571–581 (2002).

Powell, W.B., B. Bouzaiene-Ayari, and H.P. Simao, "Dynamic Models for Freight Transportation," Handbooks in Operations Research and Management Science: Transportation (G. Laporte and C. Barnhart, eds.), Elsevier, Amsterdam (2006). **Q5**