Dynamic Models of Transportation Operations

Warren B. Powell

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

Prepared for

Handbooks in Operations Research
Volume on Supply Chain Management

October 9, 2002

Contents

A manufacturing supply chain can be viewed as a sequence of steps consisting of the modification of a resource at a point (manufacturing) followed by the transfer of the product over space (transportation). Transportation arises because of the spatial distribution of resources, skill sets and customers. The challenge we face is completing this component of the supply chain efficiently, reliably and, in the case of common carriers, profitably.

It is useful to contrast "transportation planning" as it is practiced in the context of moving people versus freight. Airlines, passenger trains and bus companies typically run fixed schedules over fixed routes that are planned months, if not a year, in advance. People are typically able to adjust their travel plans around a fixed schedule, and it is extremely important that the provision of the transportation service be almost perfectly predictable. By contrast, freight operations are highly dynamic, responding to the demands of the marketplace and the production processes that serve this market. This is not to say that planning problems are not important. Freight companies have to plan the location of terminals, and they will plan operations to a degree, although these tend to be modified on a day to day basis.

Our presentation focuses on the issues that arise in the dynamics of real-time operations. We do this in part because dynamic information processes are a key characteristic of freight transportation systems, and also because the literature on static models is relatively much more mature. For a recent and thorough review of planning models for freight transportation and logistics, an excellent reference is Crainic & Laporte (1997). Other important references include Bodin et al. (1983), Fisher (1995) and Desrosiers et al. (1995) for vehicle routing; Haghani (1989), Glickman & Sherali (1985), Crainic et al. (1984) for rail transportation; Brown et al. (1987) for ocean transportation; and Crainic & Roy (1992) and Powell (1986*a*) for less-than-truckload trucking. General discussions of modeling freight transportation systems can be found in Crainic & Roy (1988) and Crainic & Rousseau (1988).

Transportation companies are controlled by three key classes of decisions: physical (how to move the product), financial (how to price it) and informational (what information should be provided to manage the system). Of course, the greatest complexity in transportation and logistics is the complexity of the physical processes, which as a result occupies most of

our attention. We can use these three dimensions to briefly summarize the characteristics of transportation that make it hard:

1) Physical: The objects that we are managing:
   - Reusable resources: Classical models of the transportation function, when done from the perspective of the shipper, simply have a cost for moving product from one location to another. From the perspective of the transportation company, this activity is done with reusable resources: drivers, tractors and trailers, for example. Thus, serving a customer request (to move freight from one location to the next) has the effect of changing the state of the system.
   - Resource layering: Serving a customer request may require one or more resource classes, which are combined to get the job done. For example, moving a truckload of freight requires a driver, tractor and trailer. Combining different resource classes is called *layering* and it has the effect of creating complex interactions between resource classes.
2) Financial: In this dimension, we focus purely on pricing:
   - Contract pricing: Given the challenges of the physical process, it is necessary to price a transportation service correctly. The pricing of transportation services is complicated by network effects (sharing resources among different markets), consolidation (sharing space on the same vehicle), and the practice of paying only for the service they receive, while expecting the resources to be available on demand.
   - Static pricing: These would be standard prices a carrier would use for freight moving between a pair of regions (sometimes called traffic lanes). These are market rates (that is, they are not specific to a contract) but which are set in advance. These are generally the highest rates a carrier will quote.
   - Spot pricing: In some cases, a customer is willing to pay for a service when requested. A carrier has to be able to quote the right price for this request. Spot pricing needs to account for the state of the system as it now exists, and the impact the activity will have on the system (the cost of the decision).
3) Informational: A critical dimension of a modern transportation system is the flow of information.

- Customer demands: Customers place demands on the system randomly over time, with varying degrees of advance information.
- Resource availability: The availability of people (drivers and crews), complex equipment (locomotives, aircraft and even tractors) and containers (trailers, boxcars, intermodal containers) is often governed in part by exogenous factors.
- Spatially distributed information: Often (although this is changing in today's information age) there is a lot of information about the system that is not available centrally. As a result, many decisions are made locally based in large part on the "head knowledge" of local or regional managers and dispatchers.

In a short chapter such as this, it is not possible to discuss all the different variations of transportation operations, nor to discuss the most interesting variations in anything approaching completeness. In the face of such richness, the question arises: how do we discuss such a broad problem class without resorting to a series of anecdotes? Our response is to focus on fundamentals, with enough examples and illustrations so that the reader can tackle variations that we are not able to cover.

The modeling of dynamic systems has a long history, and yet in many ways remains an extremely young field. The earliest dynamic models in freight transportation addressed the issue of managing fleets of containers for rail or ocean operations (Leddon & Wrathall (1967), White (1972), Misra (1972), Herren (1977), Turnquist (1986), Mendiratta & Turnquist (1982), Crainic et al. (1993)). These earliest models captured the time staging of physical activities, but not the time staging of information (in other words, they were deterministic models). The first explicit stochastic model of the car distribution problem for rail is presented in Jordan & Turnquist (1983), which assumed a) that a car that was moved empty once could not be moved empty again, and b) a car assigned to a demand did not reappear. This line of research was continued in the context of truckload trucking in Powell (1986*b*), Powell (1987), Frantzeskakis & Powell (1990), Cheung & Powell (1996) and Powell (1996). A significant breakthrough came with the introduction of adaptive estimation techniques. Powell & Carvalho (1998) introduced the use of linear functional approximations to capture the impact of decisions made now on the future. These techniques then led to the use of nonlinear functional approximations which, while somewhat more difficult to use, produce

much higher solution quality (see Godfrey & Powell (to appear), Godfrey & Powell (2000)), as well as more stable solutions.

One of the oldest problems in transportation and logistics is the vehicle routing problem. The dynamic version of this problem has been recognized for many years (see, for example, Wilson (1969)) but received little attention in the research literature (some early references include Stein (1978), Jaw et al. (1986)). Psaraftis (1988) is an important early reference which discussed some of the issues arising in dynamic routing (for an update of this discussion, see Psaraftis (1995)). The large majority of the literature on dynamic vehicle routing as of this writing focuses on simulating myopic heuristics, and the computational issues that arise in this setting (Gendreau et al. (1999), Regan et al. (1998)). A literature has emerged on the so-called stochastic vehicle routing problem, which is really a static vehicle routing problem where the tours have to be designed to anticipate "route failures" which arise when the vehicle picks up more goods than it can hold, and has to return to the depot to empty out before resuming its tour (see Stewart & Golden (1983), Laporte & Louveaux (1990), Dror (1993), Dror et al. (1989)).

Research into routing and scheduling algorithms which explicitly capture the impact of the future on decisions made now is extremely young. A fairly complete review of this literature prior to 1995 is contained in Powell et al. (1995), which includes a review of the literature of probabilistic vehicle routing and stochastic fleet management. Powell (1996) appears to be the first paper to formulate and solve a dynamic routing and scheduling problem which uses an explicit stochastic model of future events. The problem involved the matching of drivers to loads for truckload motor carriers, which is considerably simpler than problems involving multiple pickups and deliveries. Secomandi (2000) and Secomandi (2001) consider the case of routing a single vehicle dynamically through time using neuro-dynamic programming techniques (see Bertsekas & Tsitsiklis (1996)). The single vehicle case avoids the explosive growth in the size of the state space that even neuro-dynamic programming methods are sensitive to.

The most significant advances in the modeling of problems in transportation and logistics in the presence of dynamic information processes have been made in the arena of fleet management (single and multicommodity flow problems). This work has led to a general

approach for using approximate dynamic programming methods for solving resource allocation problems (summarized in Powell et al. (2000*a*) which forms the methodological basis for portions of this chapter). One of the most significant technical challenges that arises in the use of these techniques for dynamic resource management problems is the size of the state space describing the attributes of a single resource. Both single and multi-commodity flow problems have relatively small attribute spaces. Flow problems involving more complex resources (people, locomotives, ships) can be modeled as *heterogeneous resource allocation problems* (Powell et al. (2000*b*)) which typically involve attribute spaces that are too large to enumerate. Even harder are multi-stop pickup and delivery problems, which not only exhibit a large state space but are characterized by a difficult mixture of known and unknown information (the easiest problems are those where we know everything or nothing; it is the ones in between that are the hardest).

Given the breadth and complexity of problems that we are trying to address, this chapter is going to focus on the following goals:

a) It provides an overview of the different types of problems arising in transportation and logistics, focusing on operational problems where dynamic information processes play a significant role. In contrast to other presentations, these problems are addressed from three perspectives, which we refer to as the physical, financial and informational views. The physical view focuses on the objects being managed; the financial view focuses on pricing; and the informational view discusses challenges from the perspective of designing information architectures that can be used to run an operational system.

b) We provide a notational framework that is extremely general, encompassing issues such as multiattribute resources, resource layering, complex system dynamics, and the organization and flow of information and decisions. This framework allows us to tackle a broad range of problems in transportation and logistics, without having to introduce new notational systems for each new problem class.

c) We introduce four major classes of information that may be used to solve a dynamic problem, and describe the types of algorithms that arise from using different classes

of information. These classes encompass all the major algorithmic strategies in use today, but include some new ideas that are not commonly used.

d) A relatively new class of approximation strategies is outlined based on dynamic programming. These strategies allow us to design practical algorithms that are more than just myopic or rolling horizon procedures.

e) A series of basic problems are described using the notational framework which illustrate, using problems of increasing complexity, how dynamic problems in transportation and logistics can be solved.

We begin our presentation in section 1 with a discussion of operational problems that arise in a range of industries that perform transportation functions. This review summarizes the key issues, helping to set the stage for the formulation of models. Then, section 2 presents a general modeling framework for dynamic problems, giving us a modeling vocabulary with which we can address a range of problem classes. Section 3 reviews general algorithmic strategies that arise in the context of dynamic systems, focusing in particular on the modeling of both the physical and informational dimensions of the problem. Section 4 then presents specific models for some of the major problem classes. Section 5 provides brief remarks on the issue of data quality when implementing operational models, and section 6 makes some closing remarks.

Due to space limitations, our mathematical modeling focuses on representing physical processes in the presence of dynamic information processes. We consider both single and multi-agent control structures, thereby capturing both the organization as well as the flow of information. Space constraints prevent a treatment of pricing problems (see Muriel & Simchi-Levi (to appear) for a treatment of pricing from a shipper perspective), which remains a surprisingly young field in freight transportation. Even less mature is the explicit modeling of the information infrastructure, which is the true means by which most systems are controlled. By explicitly modeling both pricing and information availability in our representation of the problems, we hope to set the stage for these emerging dimensions of research.

1. Operational challenges in transportation

Each of the subsections below addresses a different industry segment that serves the transportation function. These industries form in response to the characteristics of the market that each is serving. These characteristics include:

- Consolidation: Markets can be divided between small package (less than 150 pounds), less-than-truckload (150 to 10,000 pounds), full truckload, car- or container-load, and bulk (requiring many carloads or tankers).
- Distance: Delivery from a regional warehouse to local customers represents the shortest distances, which is work that is typically handled by pickup or delivery fleets. Medium distances might be 100 to 750 miles (approximately 150 to 1200 km), which might be handled by regional LTL or shorthaul truckload carriers. Long distances include moves over 750 miles (1200 km) within the continent, or intercontinental movements.
- Control: Private fleets are owned and operated by the customer. Common carriers represent outsiders. In railroads, freight cars may be owned by the railroad or the customer. Ownership primarily arises when service is an issue, but the opportunity to consolidate is also a major factor. A company will only want to own its own trucks when it feels that it can use them effectively. Also, the ability to place advertising on the side of a company-owned truck is a factor.
- Cost: Commodity products require the lowest possible price; higher margin products can absorb higher transportation prices for better service. The same truckload carrier will charge different prices to different customers in the same market for the same service, reflecting the nature of the product being moved. Private fleets will be used to provide more customized service but with lower utilization. This service can only be justified for products which command the margins to cover the cost.
- Service: Service is typically measured as a function of speed and reliability. Of course, this ignores the many other dimensions of service that a transportation company can provide (packaging, setup, tracking and billing). Everyone wants fast, reliable service, but not everyone is able to pay for it.

Each industry reviewed below services customers that can be characterized along at least some of the dimensions listed above. Most of the industries have specialized companies that further segment the market. Thus, railroads and waterways dominate low cost, bulk commodities, but compete aggressively (with mixed success) with trucking companies for merchandise freight.

Our discussion of different service types is organized in a very specific way. After giving a brief overview of the industry, we review the resource classes and the decision classes as a way of giving a feeling of what is being managed and how we are managing it. We focus on *active* resource classes, representing the resources we are actively managing. Given our emphasis on dynamic systems, our resources tend to consist of people and equipment over fixed facilities (which are dynamic over longer horizons). Our definition of a resource (taken from Powell et al. (2001)) is a general one, and includes, as a "resource class" the customer demand itself. This may not seem customary, but as we evolve to more complex operations, we have to manage the customer's order just as we would manage the "resources" (such as drivers, tractors and locomotives) that belong to a carrier.

We then summarize decisions, organized into three key classses: 1) physical (decisions that act on physical resources), 2) financial (pricing and incentives), and 3) informational (decisions that determine the availability and flow of information). Each of these classes can be organized into two types: dynamic (which depend on the physical state of the system) and static (which do not). Our presentation highlights the importance of all three dimensions, as opposed to more classical presentations which focus primarily on the management of physical resources.

1.1. **Truckload trucking.** On the surface, truckload trucking sounds deceptively simple. A customer requests an entire trailer to move freight from one location to another. He may call in the request for pickup the same day, but most requests are made between one and three days in advance (longer when a weekend is involved). The trucking company has to decide what driver will pick up the load, and when. Once the load is picked up, the driver may take the load directly to the destination, or drop it off at an intermediate relay so that another driver can complete the delivery. There are over 10,000 companies consisting of a single truck, and several companies consisting of over 10,000 trucks.

1.1.1. *Resource classes.* There are four primary resource classes in truckload trucking: the driver, the tractor, the trailer and the load itself. Issues associated with each resource class include:

- Drivers: The choice of the driver to cover a load has to consider factors such as the destination of the load and the home domicile of the driver. The load may have to cross borders into Canada or Mexico, and not all drivers have experience doing this. Or, the load may require the use of a sleeper team to reach the destination in time to make the delivery appointment. Drivers are typically on the road for two or more weeks at a stretch, so getting drivers home is a major challenge for truckload carriers in the presence of the highly random demands they have to serve.
- Tractors: Tractors need refueling, routine maintenance and, from time to time, major maintenance at a maintenance facility. As these major maintenance intervals arise, it can be necessary to route the tractor toward such a facility.
- Trailers: Trailers can be vans (boxes) or flatbeds. Vans may be refrigerated or "dry." Most freight moves in dry vans, which may be 45, 48 or 53 feet in length (48 feet is the most common). Trailers are typically called "semi-trailers" because there is a set of wheels on only one end of the trailer (since the tractor holds the other half of the trailer).
- Loads: The basic customer request is to move a load of freight from an origin to a destination, with specified constraints for pickup and delivery. A load may allow very little time between pickup and delivery, possibly requiring the use of sleeper teams (which can drive continuously). At the other extreme are loads which allow so much time between pickup and delivery that it is necessary to park the trailer for several days to avoid arriving before the delivery appointment. In some cases, the request may involve making a sequence of stops to deliver portions of the load (less frequently, the request may require making a sequences of pickups with a single destination).

Other resource classes include fuel and maintenance resources.

1.1.2. *Decisions.* The decisions that govern a truckload carrier include:

1) Physical: It is useful to roughly divide decisions impacting physical resources between operational decisions which impact operations on a day to day basis, and planning decisions which capture design decisions which affect operations over longer periods of time.
   a) Operational:
      - Load acceptance: When the shipper calls, should the carrier accept the responsibility to move the load?
      - Driver assignment: What driver should be assigned to pick up a load?
      - Load routing: Should the load be moved directly to the destination, or should it be relayed at an intermediate point? If so, where, and what driver should then move the load for its final leg. Intermediate relays allow different drivers to perform the original pickup versus the final delivery.
      - Trailer pool management: It is necessary to manage pools of trailers, sometimes at specific shippers, so that they have access to trailer capacity when it is needed. One of the challenges of moving loads is that it is also necessary to shuttle trailers into and out of pools. These activities are normally handled by idle drivers waiting for an assignment.
   b) Planning:
      - Fleet size and mix.
      - Number of drivers and their home domicile.
      - Terminals: Truckload carriers will use terminals for maintenance and storage of tractors and trailers. It is necessary to determine how many terminals to have and their size and location.
      - Customers: What customers should a carrier serve, and what commitments (for example, in terms of number of loads) should the carrier make to the shipper? A carrier might commit to move loads for a shipper in a particular traffic lane (origin/destination pair).
2) Financial: For our applications, "financial" decisions focus on pricing, as opposed to other classes of financial decisions such as borrowing and investments.
   - Contract pricing: What price (usually specified as a cost per mile or kilometer that a trailer has to be moved) should be charged for freight in each lane? In

a typical contract, a shipper will estimate how much freight will move in each lane, but the shipper is not held to these estimates. Prices depend on the lane because of imbalances in the level of freight.

- Spot pricing: A shipper may offer a particular load at a spot price. A carrier has to decide whether to accept the load at that price at that time.

3) Informational: Here we are focusing on decisions to acquire information by investing in specific information technologies.
   - Management information systems: Most truckload carriers start as a single truck driver. As the company grows, it has to make the transition from slips of paper and a notebook to computerized systems of increasing sophistication. Several vendors market MIS systems, but most companies use these as starting points and then customize. The choice of MIS system usually involves both hardware and software.
   - Communications: A major decision faced by truckload carriers is whether to invest in onboard communications, providing two-way (data and possibly voice) communication with the driver and his unit. Other forms of communication allow tracking the status of the tractor and the trailer.
   - Real-time communication: Communicating costs money, so a decision has to be made whether to communicate with the driver at any given time.
   - Driver assignment models: These have been available for over a decade, but very few companies use them. At the same time, a handful of companies have seen dramatic successes with real-time driver assignment packages. The adoption of this technology is a major decision today.
   - Demand management systems: Forecasting demand and determining which freight to book is a key decision for truckload motor carriers.

1.2. **Private fleet operations for collection and distribution.** The vast majority of private fleets are primarily for local distribution (and sometimes collection), although some shippers will use their own trucks to handle movements between facilities. Private fleets are most commonly used for local distribution since this component of the process offers the fewest opportunities for joint use with other customers, and also offers the highest possible exposure to customers (hence the advertising on the side of the truck). Private fleets are used

when the volume of deliveries to a regional area is high enough to use the fleet effectively. When this is not the case, companies typically fall back on LTL and small package carriers.

The most basic operation faced by the private fleet for local delivery is loading up at a central terminal or warehouse, and then delivering to a group of customers. These operations typically work on a daily cycle (tied to business hours). Tours may be fairly regular (particularly when delivering high volume products to retail outlets) or highly dynamic, as would occur when delivering custom orders.

1.2.1. *Resource classes.* There are different ways to model basic pickup and delivery problems. The most classical view is that of the vehicle routing problem where the resource being managed is a "vehicle" which is understood to consist of a driver and a truck (which may itself consist of a tractor and a trailer). For this "simple" problem, we would manage:

- Vehicles: This is the principle active resource. Vehicles may be homogeneous or heterogeneous, but generally do not reflect the characteristics of individual drivers.
- Customer demands: The product to be picked up or delivered.

At the other extreme is the situation faced in the delivery of cryogenic chemicals. These companies must deliver product to tanks before they run out. A customer may require one or two deliveries per month, or several deliveries per day. It is often the responsibility of the company to estimate when a delivery is needed (an instance of vendor managed inventory). For this complex problem, the resource classes include:

- Driver: A driver may be characterized by home domicile, total driving time in a day, experience, days away from home, language skills.
- Tractor: There are two types of tractors (for example, a longer tractor with a double axle set in the rear, and a shorter one with a single axle set), and they also have maintenance requirements.
- Trailer: A trailer can hold a certain type of chemical. Also, there are different sizes of trailers, and they also have the attribute of how full they are.
- Product: There are several types of product, and it may be necessary for the truck to go from one terminal to another in order to pick up product.

- Customer tank: The customer tank is a reusable resource just like a driver, tractor or trailer. Delivering product to a tank simply changes its characteristics (the inventory level), which determines when it must be refilled again (which may be as little as a few hours into the future, or several weeks).

1.2.2. *Decisions.* The decisions that govern private fleet operations include:

1) Physical:
   a) Operational:
      - Consolidation: What customer orders should be consolidated into a particular truck?
      - Driver assignment: What driver should handle a particular delivery tour? These decisions may be static or dynamic.
   b) Planning:
      - Distribution facilities: Size and location.
      - Fleet size and type.
      - Delivery zones - In some operations, a particular driver will cover deliveries in a particular region.
      - Customer commitments - These decisions determine which customers the carrier commits to serve over the course of a year.
2) Financial:
   - Contract pricing: What price should a company charge for a pickup or delivery? The price may vary as a function of the location (which will capture the distance from the terminal) as well as the size and weight.
   - Zone pricing - Orders that are not served under a contract are typically charged a price based on shipment characteristics (size and weight) and the geographic zone.
3) Informational:
   - Communication: Should the company use radio technologies to communicate real-time with the driver? Should bar code scanning systems be used? In the case of pickups, should the company collect information about the pickup centrally when the original call is made?

- Databases: Many operations still work with sheets of paper and people. The transition to a computer in this segment remains a key decision.
- Decision support systems: GIS systems, map databases and vehicle routing algorithms are rapidly maturing, but remain imperfect. The decision to make the transition to an automated system is a major one today.

1.3. **Less-than-truckload trucking.** LTL trucking moves shipments that are typically between 150 and 10,000 pounds. The shipments can vary widely in terms of density and shape (which affects the ability to stack shipments). In the United States, shipments are typically loaded on 28 foot trailers or 48 foot vans. Most of the time, a single driver will pull a single 48 foot van or two 28 foot trailers, but "triples" are allowed by some states on selected portions of the interstate highway network. The 28 foot trailers are popular partly because they allow a driver to pull 56 feet of trailers, but also because the LTL carrier will often load the trailers with freight to different destinations. LTL carriers struggle to fill trailers to some locations. It is easier to fill a 28 foot trailer to some destinations than a 48 foot trailer.

A single tractor pulling two 28 foot trailers will be pulling, on average, between 30,000 and 35,000 pounds (around 14,000 to 16,000 kg). An LTL shipment averages about 1,000 pounds (about 450 kg), so a driver will be moving 30 to 35 shipments in a single move. Achieving this consolidation requires a tremendous amount of infrastructure. An LTL carrier has to have local pickup and delivery operations, and a network of terminals to handle the consolidation of freight.

LTL carriers can be roughly divided into two broad classes. The regional carriers move shipments up to 1,000 miles, typically with overnight or two-day service. These carriers must deliver this service with very high reliability, and as a result they will often have to move a trailer that is not full just to maintain service. The long-haul carriers focus on longer lengths of haul (although as this chapter is being written, the borders between regional and long-haul carriers is blurring), serving markets that are typically between two and five days (international movements can take longer). Although service reliability is quite high, these carriers move freight that is especially price sensitive, and as a result they have to focus on maximizing load average (the number of pounds on each trailer), minimizing the total miles travelled, and minimizing the number of times a freight bill is transferred. They are forced,

then, to take advantage of day to day variations to fill trailers to different destinations as opportunities arise. In addition, they will not move a trailer that is, say, less than a third full just to make service. However, the carriers have become increasingly sophisticated in their ability to identify which shipments actually require high service.

The typical path of an LTL shipment starts at the shipper's dock where the carrier will pick up the shipment with a pickup and delivery truck. These trucks make most of their deliveries in the beginning of the day, and then focus on pickups. At the end of the day, these trucks come into an end of line terminal, where the freight is usually (but not always) unloaded onto linehaul trucks which handle the movement of freight between the various terminals. In a regional carrier, this truck might then take the freight directly to the destination end of line for delivery the next day, or it may be transferred through a single breakbulk or distribution center. For a long haul carrier, the standard path is to first take the shipment into an origin breakbulk, where it is transferred onto a trailer that takes it to the destination breakbulk. There it is transferred a second time before the final segment to the destination end of line. In the past, some carriers followed a strict policy of forcing all shipments through two breakbulks, producing a *transfer ratio* of 200 (meaning that shipments were transferred, or handled, on average twice). However, the best run national LTL carriers actually achieve transfer ratios below 100 (that is, shipments are transferred on average less than once). This ratio arises because breakbulks are typically located near major cities, which means that a large number of shipments originate at one breakbulk and terminate at another (producing zero transfers). The transfer ratio of 100 means that there are still quite a few shipments moving through two breakbulks.

The largest LTL carriers in the U.S. are primarily unionized, which has the effect of imposing a variety of rules on how drivers must be managed, and what a worker can and cannot do. For example, unlike truckload carriers that face a real challenge in getting drivers home in a timely way, the long-haul LTL carriers manage their single drivers in a way that ensures that they are home every night or every other night, with only occasional trips that take the driver out for two nights. Sleeper teams moving freight over longer distances may be away from home for three or four days, but are guaranteed to be home every week.

The LTL carriers are exceptionally competitive, and the large majority have gone out of business since the industry was deregulated in 1980. The survivors have learned how to strike the delicate balance between cost and service for a particular set of markets. For the long-haul carriers, the emphasis is on cost with very high service. For the regionals, the expectations on service are even higher. The biggest challenge faced by the regionals is that they have so little time to move a shipment that they have to make decisions quickly, and they are often forced to move trailers that are less than half full. The long-haul carriers have more time to work with a shipment, but their large networks offer many more options that can be considered.

The pickup and delivery process of LTL carriers has many elements in common with the description of pickup/delivery operations for private fleets. The biggest difference is that private fleets are usually doing pure pickup or pure delivery, whereas LTL carriers must handle both activities. Furthermore, it is common to separate the planning of pickup and delivery operations for LTL carriers from the planning of the linehaul operation (movements between terminals).

A closely related segment is the small package industry, dominated at this writing in the U.S. by the United Parcel Service and FedEx. There are a number of subtle yet important differences between moving LTL freight (over 150 pounds) and small packages (under 150 pounds). For example, a trailer may hold 20 to 30 LTL shipments, while a trailer may hold hundreds of small packages. One effect of small shipment sizes is that there is considerably less variability in the day to day flows. Another is that small shipments lend themselves much more readily to automation in the sorting facilities.

1.3.1. *Resource classes.* The principle resource classes are:

- Drivers: These are characterized by their home domicile, driving hours, whether they are a single driver or part of a sleeper team, their bid characteristics and the number of days they have been away from home.
- Tractors: The carrier must maintain appropriate pools of tractors at major terminals, and manage the maintenace requirements of tractors.
- Trailers: These are typically 28 foot “pups” and 48 foot “vans.”

- Shipments: Varying in size between 150 and 10,000 pounds, traveling distances between 100 and 3,000 miles.
- Terminals: Here we have to determine the number of terminals, and their size, type and location.
- Dock labor: These are the people who load and unload trailers. This determines the fraction of the physical capacity of the terminal that is actually used.

As in most problems involving multiple resource classes, it is common to work with one or two classes at a time. We illustrate this modeling strategy in section 4.

1.3.2. *Decisions.* Key decisions include:

1) Physical:
   a) Operational:
      - Service network design: From a terminal, to which destinations should we send a trailer direct? These decisions have to capture the ability to consolidate freight in a timely fashion.
      - Traffic assignment: What freight bills should go on a specific trailer? This can be a difficult problem for long-haul carriers who face a number of options. For regional LTL carriers, it is usually obvious.
      - Pup matching: A tractor will usually pull two, and sometimes three, of the 28 foot "pups." It is not always the case that the pups will have the same origin or same destination. Pups must be matched so that the combined weight is within legal limits. If pups are being matched which do not have the same destination, it is desirable to match them so that they can stay hooked together as long as possible. Finally, it is best if the two pups have freight with similar service requirements.
      - What driver should be used to pull a load? The choice of driver depends on domicile, how many hours he has been driving, how many days he has been away from home, whether the "driver" is a single individual or a team, and the type of bid (for union drivers).
      - Trailer management - Most of the time trailers remain balanced because of the need to balance drivers (a driver normally pulls empty trailers if there

are no loaded ones to move). But sometimes it is necessary for a driver to bobtail (move the tractor without a trailer) which creates an imbalance in the flows of trailers.

- When should the loads be moved? This is one of the hardest decisions, since the decision has to balance the service requirements of the shipments on the trailers, and the constraints on moving the driver and getting him home.

b) Planning:

- Terminals: Size, type and location.
- Dock labor: How many people should staff each terminal?
- Equipment pools (tractors and trailers): How large should the pools be?
- Physical transportation links: LTL carriers are called "regular route" carriers, and the decision to move trucks over a particular route joining two terminals is a planning decision.
- Contracts: This covers the agreement to serve major accounts, typically for a year. These commitments reflect expectations of the amount of freight that will be moved for the account (typically by traffic lane).

2) Financial:

- Contract pricing: What price should be charged for freight for a specific account? These prices will be a function of the weight of an individual shipment, and the traffic lane (origin and destination) in which it is moving. For LTL trucking, this is an exceptionally difficult problem. It has to reflect the cost of pickup and delivery (see notes on this under private fleet operations), transferring the freight at terminals, and moving the freight over the linehaul network. Transportation costs (linehaul costs) have to reflect the density of the freight and its "stackability" (the ability to stack the freight with other shipments in the same lane).
- General pricing: Same as contract pricing, but it is for freight offered to the carrier that is not covered by a contract. These prices are typically substantially higher than a contract price.

- Spot pricing: what price should an LTL carrier offer for a specific shipment on a specific day to help with network balance? Spot prices apply almost exclusively to truckload shipments.

3) Informational:

- Communication technologies: LTL carriers face an array of decisions regarding communication technologies. Some of these include:
  - Shipment bar code scanning equipment for shipment pickup: This allows the carrier to learn more about the characteristics of the shipment when it is picked up.
  - On-board driver communication: This allows the carrier to dispatch a driver on the street to pick up a shipment that has just been called in.
  - Bar-code scanning at the terminals. This allows the carrier to know exactly when a shipment was pulled from one trailer and loaded onto another.
  - On-board vehicle tracking: This would be the same technology used by truckload carriers. Its adoption for LTL is less obvious since the LTL trucks typically follow fixed routes.
- Databases and screens: Collecting, storing and displaying data to support decisions is a major, ongoing challenge with any company. These systems are expensive, however, and have to be cost-justified.
- Planning models: A range of models are evolving to help support LTL carriers, ranging from routing of pickup and delivery trucks, service network design, pup matching and driver management. As of this writing, these models are young and are seeing only the earliest adoption.

1.4. **The railroads.** Railroads remain the primary mover of bulk cargo, including both dry (grain, coal) and liquid (although pipelines offer some competition here). In some areas, barges can handle bulk movements, but for land movements, rail is almost the only option. Bulk cargo, however, represents only a portion of rail business. A substantial amount of merchandise freight moves by rail, as well as container movements that are moving to or from international locations. In fact, although much is made of the competition between trucking companies and rail, all the major trucking companies (truckload and LTL) use rail extensively for long moves. Trucking companies in particular have a difficult time moving

freight between the midwest and the west coast in North America; the distance is long and it can be difficult finding and managing drivers over this long movement. It is cheaper and more convenient to take a trailer of freight and load it onto a flatcar to be moved by rail.

Railroads, however, struggle with certain operating characteristics that are fundamental to the nature of a railroad. First of course is the limited infrastructure. The massive majority of all non-bulk movements must begin and/or end on a truck. The process of picking up freight and taking it to a rail yard, or delivering freight from a railroad, is known as *drayage* (the drivers who handle this step are called *draymen*). Most drayage operations are relatively inefficient and can add a substantial amount of cost to the process.

A second challenge faced by railroads is that freight is moved in extremely large blocks. A single train will typically weigh between 2,000 and 5,000 tons, but some bulk trains can be as large as 15,000 tons. A typical truck, by contrast, is moving about 15 tons of freight, with a gross weight of about 30 tons. So, a single train can be equivalent to as many as 500 trucks moving down the track. The process of batching up enough freight to form these big blocks introduces a substantial amount of noise in the process. A train may be cancelled if there is not enough freight to justify its movement (the policy of canceling trains to save on crew and fuel costs is hotly debated in the industry, and is becoming less frequent), or because of problems finding enough power and a rested crew. When a train is delayed or cancelled (for any reason) the impact on the network can be fairly substantial.

A third challenge is the capacity of the track. High priority trains move faster than bulk trains. Passing a train going in the same direction requires that the slower train pull off on a siding. Since these sidings are located only at selected points along the track, it may be necessary to pull a train off the track for several hours until the pass can be completed. The same issue arises when trains have to move in opposite directions over a single-track route. One train will have to find an appropriate siding. Compounding the challenge of sitting at sidings is that the crew may run out of hours to complete the trip. Thus, an eight hour trip can turn into an 11 hour trip, violating maximum duty time rules for the crew. The railroad is forced to drive a new crew out to the train to finish the trip.

The major railroads all offer high priority service for certain classes of freight, where they try to compete with trucking companies. Given the limitations of the infrastructure and the

nature of rail operations, it is virtually impossible for rail to provide a higher level of service than truck. On the other hand, no other land-based mode can compete with its efficiency.

1.4.1. *Resource classes.* Railroads are characterized by the broad range of resources which must be managed to provide rail service. Some of these include:

- Freight cars: These come in a variety of styles, including boxcars, flatcars, and tanker cars, but with many variations of each type.
- The freight: Freight can exist by itself as an unsatisfied customer demand, or coupled with a boxcar (producing a loaded car). Freight is characterized by origin, destination, size and service characteristics.
- Locomotives: There are about a dozen major classes of locomotives, but viewed closely enough, locomotives are almost unique. Characteristics of a locomotive can include its horsepower, whether it is high or low adhesion (a feature that determines the ability of the train to get started from a standstill), features required to classify it as the lead unit on a train (which is where the crew rides), its maintenance status, and what other locomotives it is currently attached to (the process of connecting multiple locomotives to pull a single train involves a fairly elaborate set of connections which have to be tested before the train can move).
- Operators: The rules for moving crews are governed by federal regulations, and a dizzying array of union work rules, some of which date back centuries.
- Track: The track limits the ability of trains to move. Decisions to build new track or maintain existing track are some of the most important infrastructure decisions a railroad can make.
- Yards: As with track, the yards have capacity and limit the throughput of trains.
- Maintenance facilities: Locomotives represent complex pieces of equipment, with federally required maintenance intervals (in addition to those required to keep a locomotive in working order). Some maintenance equipment runs in the millions of dollars.

1.4.2. *Decisions.* As with the resources, there is a complex array of decisions required to manage these resources:

1) Physical:
    a) Operational:
        - Trip planning: How should a loaded freight car be routed through the network? Freight cars are allocated to *blocks* (a group of cars being routed over a common segment) which are moved by trains. Both trains and blocks have capacities, so it is necessary to plan the route of a car through a sequence of blocks while not violating either capacity constraints.
        - Blocking: What blocks should be formed? How should blocks be routed through the network?
        - Car distribution: To what yard or customer should an empty freight car be allocated to?
        - Locomotive management: What locomotive should be used to pull a train? How should power be repositioned from surplus to deficit locations?
        - Crew planning: What crew should be used to move a train?
        - Line capacity planning: How should trains be sequenced over a track (and the sidings)? When should trains be scheduled to depart?
    b) Planning:
        - How much track, how many sidings and their placement, and how well should they be maintained (which affects the speed at which trains can move over the track)?
        - Location and size of new yards, local stations and maintenance facilities.
        - Fleet size and mix, for locomotives as well as the freight cars.
        - Customer commitments, which set carrier expectations of the amount of freight a customer may tender (and therefore the resources required to serve the customer).
2) Financial:
    - What price should be charged for a contract? These prices will be a function of the origin and destination, shipment size, freight car requirements, and other service constraints.
    - What spot prices should the railroad accept?
3) Informational:

- Should the railroad invest in train tracking technology (which tells them the location of the train on the track)?
- Should the railroad invest in transponders that detect the presence of locomotives? freight cars?
- Should a train have voice communication while it is en route?
- What databases should be created to store and display information?
- What planning models should a railroad invest in? As of this writing, major railroads are taking noticeably different approaches toward the use of planning models; some are focusing on longer range planning models; others on short-term operational models, while others are limiting their use of models.

1.5. **Intermodal container operations.** Intermodal containers are boxes, typically 20 feet or 40 feet in length, which can themselves move by truck, rail or ocean container ships. Unlike trailers, containers can be stacked two levels high on a rail flatcar, in addition to being pushed against each other (trailers, with wheels, require special panels on the flatcar to ensure that they do not roll). As a result, they are a much more productive way to move freight over both road and rail. On the other hand, they are not as large as the 48 foot vans. The 20 foot containers are smaller than the 28 foot pups favored by LTL carriers, a difference that is compounded when a single driver can pull two 28 foot pups, a volume that is much larger than two 20 foot containers. It is not possible to pull two 40 foot containers using a single tractor.

If the freight has to move by vessel, the container is the only way to move merchandise. It is not possible to stack trailers, and stacking allows the largest container ships to hold thousands of containers.

In contrast with our previous examples, the management of intermodal containers consists only of the containers. Containers may be owned by shipping lines, or by other logistics organizations. For this reason, we do not address the dimensions of motive power (tractors, locomotives, ships) and operators (drivers, crews).

1.5.1. *Resource classes.*

- Containers: these come in two basic lengths (20 feet and 40 feet) but a variety of other features will distinguish one container from another, including refrigeration, height, and stacking capability.
- Customer orders: this is freight moving from an origin to destination (multiple stops are never permitted), with specific service requirements.

1.5.2. *Decisions.*

1) Physical:
   a) Operational:
      - What type of container should be allocated to an order?
      - How should containers be distributed in anticipation of forecasted orders? How many containers of each type should there be in a pool on any given day?
      - How should a loaded container be routed from origin to destination (this may involve a combination of ship, train and truck)?
      - Stacking and storage of containers in the port and on the ship itself: Where should a container be stacked and stored (both in the port, as well as on the ship) to minimize total handling of containers.
   b) Planning:
      - How many containers, and what type, should be owned?
      - What are the size and location of container pools?
      - What transportation contracts should be arranged? Container shipping typically requires arrangements with other transportation companies to move the containers.
2) Financial:
   - How should a contract be priced?
   - What should standard rates be for non-contract movements in a traffic lane?
   - How should the carrier spot price individual moves?
3) Informational:
   - What types of tracking technologies should a company use (especially for use in the ports)? There is a movement toward the use of satellite tracking of individual

trailers (in trucking) and containers (in shipping). To what extent should these technologies be used?

- Should the company invest in forecasting and optimization technologies?

2. A general modeling framework

The examples in the previous section illustrate the range of different industries that have evolved to meet segments of the freight transportation market. Each exhibits specific qualities in terms of cost and service, reflecting the nature of the market that is being served.

The next challenge is modeling these problems. Fortunately, all of these problems are characterized by certain physical processes. If we can develop models for the basic processes that characterize these problems, then we can build up more complex models from these building blocks.

To begin, we need a notational system to describe our problem. Our notation (which is based on Powell et al. (2001)) builds on standard notational conventions, but most of the standard research in logistics avoids some key issues that arise in real applications. Examples include multiattribute resources and resource layering, dynamic information processes and multiagent control.

We need some general notation through the presentation. WE represent the geography of our problem using: For transportation applications, it is useful to define in addition a set of geographical locations:

$\mathcal{I} =$ A set of locations, indexed by i and j.

We generally model our problem over a set of discrete time periods:

$$\mathcal{T} = (0, 1, \ldots, T).$$

At times, we want to represent rolling horizon problems where at time $t \in \mathcal{T}$, we will optimize over a set of time periods that start at t and extend over a planning horizon, given by:

$$\mathcal{T}_t^{ph} = (t, t+1, t+2, \ldots, t+T^{planninghorizon}).$$

Our representation is divided along three primary dimensions: resources, processes and controls.

2.1. **Resources.** Resources are comprised of three subdimensions:

a) Resource classes.
b) The attributes of each resource class.
c) The resource layering, which represents how resources can be coupled together to perform work.

For this, we define:

$\mathcal{C}^R =$ The set of resource classes (e.g. drivers, tractors, trailers, locomotives, loads).
$\mathcal{R}_t^c =$ The set of resources in class c at time t.
$a_r =$ The attributes of resource r.
$\mathcal{A}^c =$ The space of attributes for resources in class c, where $a_r \in \mathcal{A}^c$ for $r \in \mathcal{R}^c$.

In some cases, it is easiest to track individual resources, which means that $(a_r, r \in \mathcal{R}_t^c)$ would capture the state of the resources in class c at time t. This representation is often most useful when resources are relatively complex, such as people, aircraft and locomotives. When the resources are simpler, it is more useful to use vector notation:

$R_{ta} =$ The number of resources with attribute a at time t, before new arrivals in time t have been added in.
$R_t = (R_{ta})_{a \in \mathcal{A}}$.

An important but fairly subtle issue that arises purely in the context of dynamic problems is referred to as the *time lagging* of information. Specifically, there may be resources that we know about at time t, but which cannot be acted on until time t'. In this setting, t refers to the time at which the resource becomes known, whereas t' is when it becomes actionable. Thus, we may know about a customer order now, but we do not have to satisfy it until later. Or we may know about a boxcar that will become available in the future. We handle this concept by defining:

$R_{t,at'}$ = The number of resources with attribute a that we know about at time t but which do not become actionable until time t'.

$R_{tt'} = (R_{t,at'})_{a \in \mathcal{A}}$

$R_t = (R_{tt'})_{t' \geq t}$.

We call R_t the *resource state vector*. Not uncommonly, this vector is defined with respect to an aggregation function:

$$G : \mathcal{A} \mapsto \mathcal{A}^G$$

where $\mathcal{A}^G$ is a more compact space of attributes. For our purposes, we use R_t exclusively as our resource state vector, recognizing that the discrete representation $\mathcal{R}_t$ is more appropriate for complex resources.

The use of the attribute vector a is very convenient. For the simplest problems, we might be modeling the flows of a common type of trailer between locations $i \in \mathcal{I}$. In this case, the location i represents the state of the resource, and we would have $a = (i)$. We might have different types of trailers or containers $k \in \mathcal{K}$, as would commonly arise in multicommodity flow problems. In this case, k is the commodity and i is the state of the resource. The attribute vector would then be $a = (k, i)$. As we move to more complex resources, the attribute vector would grow. By using a common attribute vector a, our notational system responds easily to different types of resources.

One of the most difficult dimensions of resource management arises in the presence of resource layering. Consider, for example, the problem of moving a load in trucking. We need a driver and a trailer to pull a load of freight. We start with an idle driver (in a tractor). The first step is to find a trailer, at which point we have a driver with an empty trailer. Next we have to move to the customer and pick up the load of freight. Now we have a driver with a trailer and a load of freight. At this point we have to decide whether the driver should move directly to the destination to deliver the load, or to move the trailer to a relay point where he would drop off the trailer (of course, still full of freight). The driver/trailer, driver/trailer/load, or the trailer/load, represent instances of layered resources.

We represent layered resources by first defining a *layering*, $\mathcal{L}$. This is most easily described by example. Let $\mathcal{C} = (D, T, L)$ represent our three resource classes. For our example, a

layering would be:

$$\mathcal{L} = (\text{D}|\text{T}|\text{L}, \text{T}|\text{L})$$

If we use this as our layering, we would call the first layer the *driver layer*. It consists of a driver, trailer and load. The attributes of a driver layer consist of the attributes of a driver, and then the attributes of a trailer and a load that may be *coupled* to the driver. In general, if $\ell \in \mathcal{L}$ represents a particular layer, we let $a^{(\ell)}$ represent the attributes of layer ℓ, while a^c would be used to represent the attributes of a particular resource class $c \in \mathcal{C}^R$. We may refer to a specific layer, such as the driver layer, using:

$$\begin{aligned} a^{(D)} &= \text{The attributes of a driver layer.} \\ &= a^D|a^T|a^L. \end{aligned}$$

If a driver is not coupled with a trailer or load, his primitive attribute vector would be a^D, but his layered attribute vector would be $a^{(D)} = a^D|\phi^T|\phi^L$, where ϕ^T and ϕ^L are null vectors with the same dimensions as a^T and a^L, respectively. Thus, the attributes of a driver layer are not determined until we decide which resources (trailer and load) to couple the driver with. We also have a trailer layer, which again can consist of a single trailer, or a trailer and a load. We identify a layer by its lead resource class.

Layering is a an important concept for modeling more complex operations, but it can sometimes be avoided. Consider, for example, the case of truckload trucking, but assume that once a driver picks up a load, that he always drives it directly to the destination. Thus, the decision to assign a driver to a load produces a driver at the destination of the load and a (presumably) happy customer (the load has been delivered). At no time did we have to explicitly capture the state that the driver had the load. Layering arises when there is a specific set of decisions that we have to choose from given the attributes of a layered resource.

Resource layering is critical when we have to capture the state of two resources coupled together, at which point there is a new set of decisions that apply to the characteristics of the layered resource. More examples of resource layering are given in section 4.

2.2. **Processes.** There are three dimensions of processes:

a) Dynamic information processes.

b) System dynamics.

c) Constraints.

There are two types of information processes: exogenous information (outside of our control) and endogenous information, more commonly known as decisions (a good working definition of a decision is an endogenously controllable information class that changes the state of the system). At this stage, we use general models of both (specific illustrations are given in section 4). For exogenous information processes, we let:

W_t = A random variable representing a family of random variables describing new information arriving at time t.

In complex problems, there can be a number of exogenous information processes. For us, we use W_t to represent all of these. We let $\omega \in \Omega$ represent an elementary outcome of the sequence $(W_t)_{t\in\mathcal{T}}$, and we let $\omega_t = W_t(\omega)$ be a realization of the information arriving in time period t. Following standard conventions in probability theory, we let $\mathcal{F}_t$ be the σ-algebra generated by $(W_{t'})_{t'=0}^{t}$. For our problem, there are two special types of information that arrive. The first is information about new resources that are arriving such as new customer demands, or new units of capacity entering the system from outside sources (for example, a boxcar being released empty to the network). We represent these by:

$\hat{R}_{t,a't'}$ = The number of resources with attribute a' that first become known at time t that can be acted on at time t'.

$\hat{R}_t = (\hat{R}_{t,a't'})_{a'\in\mathcal{A},t'\geq t}$.

The second class of information represents parameters that govern the dynamics of the system (described shortly). For example, we might get new information about the speed of a train, the cost of a movement, or the price of fuel. We capture these parameters using:

ρ_t = A vector of parameters that impact the dynamics of the system.

$\hat{\rho}_t$ = New information about these parameters arriving in time t.

An element of ρ_t might be the estimate of the transit time between two points, or the average number of pounds that a trailer normally holds.

Endogenous information processes represent our decisions. For the moment, we let:

$$\begin{aligned}\mathcal{D} &= \text{The set of possible decisions that can be used to act on the resources.}\\ x_{tad} &= \text{The number of resources with attribute } a \text{ that decision } d \in \mathcal{D} \text{ is applied to at time } t.\end{aligned}$$

System dynamics governs how the system changes in response to new information. The effect of new resources is captured simply using:

$$\begin{aligned}R_t^+ &= \text{Set of resources at time } t \text{ including new arrivals in time period } t.\\ &= R_t + \hat{R}_t.\end{aligned}$$

In section 3 we demonstrate the special roles of R_t and R_t^+ (and the reason for this particular notational style).

We represent the updating of system parameters using the general notation:

$$\rho_t \leftarrow U^{\rho}(\rho_{t-1}, \hat{\rho}_t)$$

For example, if ρ_t is an estimate of the travel time, and $\hat{\rho}_t$ is a recent observation of a travel time, we might think of $U^{\rho}(\rho_{t-1}, \hat{\rho}_t)$ as an equation that performs exponential smoothing, as in $\rho_t = (1-\alpha)\rho_{t-1} + \alpha\hat{\rho}_t$, where $0 < \alpha < 1$ is a smoothing factor.

More interesting is modeling the impact of a decision on the system. We use the concept of a *modify* function, which performs the mapping:

$$M(t, a, d) \rightarrow (a', c, \tau) \tag{1}$$

Where a is the attribute of a resource (or resource layer) being acted on by decision d, where t represents what we know when the decision is made (or implemented). a' is the attribute of the modified resource, c is the contribution (or cost, if we are minimizing) generated by the decision, and τ is the time required to complete the action. The modify function is useful conceptually and in software, but for algebraic purposes, it is useful to define:

$$\delta_{t'a'}(t, a, d) = \begin{cases} 1 & if M(t, a, d) = (a', c, t' - t) \\ 0 & \text{otherwise} \end{cases}$$

The modify function plays the role of a transfer function in dynamic systems, but it is expressed at the level of a single decision acting on a single (type of) resource. Sometimes it

is useful to refer specifically to the attribute of a transformed resource, or the cost or time required to complete the decision. For this purpose, we introduce the notation:

$$M(t,a,d) \rightarrow (a^M(t,a,d), c^M(t,a,d), \tau^M(t,a,d)) \tag{2}$$

We call a^M_{tad} the *terminal attribute function*, where the superscript "M" is used to help identify the difference between the attribute vector a and the terminal attribute function a^M_{tad}. More often, we use the vector notation $c_{tad} = c^M(t,a,d)$ and $\tau_{tad} = \tau^M(t,a,d)$ to represent costs and times. Our representation assumes that (a', c, τ) are all deterministic functions of (t,a,d). This assumption serves the purposes of our presentation here, but the reader should understand the richness of dynamic problems. For example, it is very common that transit times are not deterministic functions of (t,a,d); for some areas (intermodal container transportation, rail transportation, and even the large truckload and LTL carriers), the randomness of the transit time is of central concern to some shippers where precise delivery dates are essential.

Our first use of the delta function is to express the evolution of the resource vector:

$$R_{t+1,a't'} = R_{t,a't'} + \hat{R}_{t,a't'} + \sum_{a\in\mathcal{A}}\sum_{d\in\mathcal{D}} \delta_{t'a'}(t,a,d)x_{tad} \ \forall a' \in \mathcal{A}, t' \geq t \tag{3}$$

Finally, the evolution of the system is restricted by constraints. For our purposes, it is sufficient to represent flow conservation constraints:

$$\sum_{d\in\mathcal{D}} x_{tad} = R_{ta}$$

and rate of process transformation constraints:

$$x_{tad} \leq u_{tad}$$

where u_{tad} is an upper bound (normally some sort of physical constraint) on the flow. In practice, upper bounds apply to aggregations of flows.

2.3. **Controls.** Controls are characterized by five dimensions:

a) The types of controls.
b) The organization of controls.
c) The information available to a decision maker.
d) The decision function.

e) Measurement and evaluation.

We describe the types of controls using:

$\mathcal{C}^D =$ The set of decision classes.
$\mathcal{D}^c =$ The set of decisions in class $c \in \mathcal{C}^D$.

In many problems, it does not make sense to define a single general set of decisions. For example, the decision to "send a truck empty to Chicago" does not make sense if the truck is in Miami. Any practical implementation requires being able to specify the set of decisions given the attributes of the resource being acted on (typically, we would use an aggregation of the attribute vector). Thus, we would define:

$\mathcal{D}_a =$ The set of decisions that can be applied to a resource with attribute vector a.

In practice, the decision class we are working with is understood (or there is only one class), allowing us to avoid the explicit modeling of decision classes. Just the same, it is important to recognize the presence of multiple decision classes. Reading the academic literature could easily lead a student to think that the only thing a transportation company does is move something from one location to another. Companies have to buy and sell, maintain, paint and clean, and refuel. The notation we provide here allows us to write a basic formulation of the problem which will remain valid even if we add decision classes later.

For our discussion, we restrict our attention to classes of decisions that directly impact resources. Thus, these are the classical decisions of routing drivers and freight, as well as purchasing/selling new equipment, hiring new drivers, or choosing which customer demands to serve in the spot market. Other classes of decisions include pricing (both contract and spot), and decisions about the information infrastructure.

Decision classes can be divided into three major groups: couple, uncouple and modify. The couple and uncouple classes arise only when we are modeling resource layers. A couple decision brings two (or more) resources together. For example, a driver pulling a load, or a pilot flying a plane. In this case, a is the attribute of the active resource, while d is the decision to augment a with the attributes of a secondary resource. For a modify decision, d

simply modifies the attribute vector a. Most problems feature "one to one" coupling (one driver, one load; one pilot, one aircraft; one boxcar, one demand). More than one locomotive is needed to move a single train, which is an instance of "several to one" coupling. A single truck may move dozens (or hundreds, in the case of packages) of shipments, and this is an instance of "many to one" coupling.

It is useful to start by listing only the *primitive* decisions, each of which is a single, elementary action. For example, the decision to "assign a driver to a load" in truckload trucking can consist of the primitives: move to the load, couple with the load, move the load, and uncouple from the load. Once the primitives are in place, it is often useful to create *tactics* which represent sequences of decisions, as in our example to "assign a driver to a load." Had we formulated the problem purely in terms of the primitives, we would have to capture the layered state "driver coupled with a load." If we are not modeling driver relays, a model based purely on primitive decisions would be unnecessarily complex. But, planning driver relays may be important, in which case it is useful to work in terms of the primitive decisions.

For complex operations (railroads, trucking companies) it is important to model the organization of information. Large companies are managed by a series of decision makers which the modeling community calls *agents*. Let:

$\mathcal{Q} =$ A set of agents which control the system.

$\mathcal{D}_q =$ The set of decisions controlled by agent $q \in \mathcal{Q}$, which implicitly includes when a decision will be implemented.

$\mathcal{A}_q =$ The attributes of resources that are controlled by agent q, which we also assume includes the time at which the resource is available to be acted on.

$\mathcal{T}_q =$ The set of time periods over which the decisions in $\mathcal{D}_q$ apply.

$x_q = (x_{tad})_{t \in \mathcal{T}_q, a \in \mathcal{A}_q, d \in \mathcal{D}_q}$.

For notational simplicity, we assume that an agent implies an interval of time. Often, we will find ourselves modeling a single controller at a point in time, in which case we can simply replace the index q with a time index t. Our "agent" notation, where time is implicit in the

definition of the agent, gives us a simple notational mechanism for modeling more general informational decompositions with no additional complexity in notation.

The sets $\mathcal{Q}, \mathcal{D}_q, \mathcal{A}_q$ and $\mathcal{T}_q$ define the organization of control in the operation. It is assumed that an agent q will make decisions within $\mathcal{D}_q$ that are coordinated (for example, if the decision maker is assigning drivers to loads, he will not assign the same driver to two loads at the same time). It is also necessary to understand the impact of agent q on other agents (which may exist within the same organization, or in other organizations). For this purpose, we need to define:

$\vec{\mathcal{M}}_q$= The set of agents $q' \in \mathcal{Q}$ who are directly impacted by decisions made by agent q.

$R_{q,aq'}$ = The number of resources of attribute a that are sent from agent q to q'.

$R_{qq'} = (R_{q,aq'})_{a \in \mathcal{A}_q}$.

We next have to model the organization of information. We let:

I_q = The information elements available to decision maker q.

There are four classes of information that may be used in the set I_q:

K_q = Knowledge, which is the exogenous data that is accessible to q. Knowledge contains data in databases as well as other informal sources that are present as "head knowledge."

Ω_q = Forecasts of exogenous information which would come as updates to K_q. Normally, the set Ω_q will consist of a single element representing a point forecast, but it might include different elements, representing different scenarios that we wish to model in the future.

x_q^p = Plans for the future, which can be thought of as forecasts of future decisions.

$V_{qq'}(R_{qq'})$ = Value functions which capture the impact of decisions by q on $q' \in \vec{\mathcal{M}}_q$.

The value functions can be thought of as forecasts of dual variables. A simple example of these functions arises when purchasing parts from a supplier. The decision to place an order has an impact of requesting parts from the supplier ($R_{qq'}$ becomes the number of

orders that q is transferring to q'). The supplier then charges a price (say, $p_{q'}$), so our value function is simply $p_{q'}R_{qq'}$. When the value function is linear, it is possible to show that $V_{qq'}(R_{qq'}) = V_{q'}(R_{q'})$.

It is important to understand that when designing the set I_q, the goal is not to create the ultimate information set, but rather to model the information that is actually available. Many decisions are made purely using K_q (the vast majority of simulation models fall in this category). Optimization models that use deterministic forecasts would use the set $I_q = (K_q, \Omega_q)$ where the set Ω_q usually consists of a single point forecast. Models based on this information set are called rolling horizon procedures.

Given the information set, the next problem is to actually make a decision. Let:

$\mathcal{X}_q =$ The feasible region for agent q.

The process of actually making decisions is then given by:

$X_q^\pi(I_q) =$ The vector of decisions produced by information set I_q. Thus, we compute decisions using $x_q = X_q^\pi(I_q)$. We let:

$\Pi =$ The family of policies (literally, different decision functions, each of which constitutes a method for translating information into decisions).

A policy represents any means of finding a decision given a state, which we also call a decision function. Our problem is one of finding the best decision function. But, it is also going to be important to build functions that use information that is actually available. In this way, we are attempting to model the organization and flow of information just as we model the flows of physical resources.

Finally we have the dimension called measurement and evaluation. For our purposes, this is the objective function. We assume that we can define a contribution function:

$C_q(x_q, K_q) =$ The contribution from decision x_q given our knowledge K_q.

Remembering that each agent q implicitly defines a time interval over which his/her decisions apply, our objective function can now be stated:

$$\max_{\pi \in \Pi} E \left\{ \sum_{q \in \mathcal{Q}} C_q(X_q^\pi(I_q), K_q) \right\}$$

This optimization problem takes on more meaning when we define specific classes of functions X_q^π.

3. Algorithmic strategies

Now that we have a specific modeling framework, we have to address the challenge of designing an algorithmic strategy. We start in section 3.1 by presenting strategies for solving time-staged problems under uncertainty using a new class of dynamic programming approximations. Section 3.2 discusses the issues that arise when we combine nonlinear value functions with multiperiod travel times. This concept is then extended in section 3.3 to solve multiagent problems using the same framework. These two sections establish the fundamentals of solving the problems when information is staged over time, and when information is organized among different decision makers. These presentations then lay the groundwork for section 3.4 which provides a general framework for building different classes of decision functions for a variety of complex problems.

By the end of this section, we will have the foundation we need to address a fairly broad range of complex operational problems.

3.1. Strategies for dynamic problems.

Our first challenge is solving problems when information is staged over time. This is the classical problem of stochastic, dynamic problems. These can be solved approximately in a variety of ways that are discussed in section 3.4. Here, we demonstrate how to use dynamic programming approximations effectively to solve time-staged problems.

Our presentation is divided into two stages. First, we have to address a subtle but critical problem in how we model the evolution of information over time and the definition of the state variable. In particular, we do not use the classical definition of a state variable as it is presented in dynamic programming. Instead, we introduce the concept of an *incomplete*

state variable which will prove computationally far more tractable. After this discussion in section 3.1.1, section 3.1.2 discusses specific strategies for approximating value functions in dynamic programs.

3.1.1. *Setting up the optimality recursion.* We start by describing the evolution of information in our system. As we noted before, we have exogenous and endogenous information processes that can be represented using:

$$(W_0, X_0^\pi, W_1, X_1^\pi, \ldots, W_t, X_t^\pi, \ldots)$$

We need to capture what we know at each point in time. This can be measured immediately after we have new exogenous information, and after we make a decision. We let S_t^+ be the state after new information has arrived, and we let S_t the state after we make a decision, giving us the sequence.

$$(W_0, S_0^+, X_0^\pi, S_1, W_1, S_1^+, X_1^\pi, S_2, \ldots, S_t, W_t, S_t^+, X_t^\pi, S_{t+1} \ldots)$$

We refer to S_t^+ as the *complete* state variable, because it captures all the information needed to make a decision at time t. S_t is called the *incomplete* state variable, specifically because it does not include all the information needed to make a decision. The importance of this distinction will become clear shortly.

Our goal is to solve the problem:

$$\max_{\pi \in \Pi} E \left\{ \sum_{t \in \mathcal{T}} C_t(X_t^\pi, S_t^+) \right\} \tag{4}$$

Equation (4) can be formulated in general using the optimality recursion:

$$V_t^+(S_t^+) \quad = \quad max_{x \in \mathcal{X}} C_t(x, S_t^+) + E\{V_{t+1}^+(S_{t+1}^+) | S_t^+\} \tag{5}$$

Here and throughout this section, we use x as the variable we are optimizing over, and let x_t represent the solution of (5).

The field of dynamic programming is typically expressed in terms of discrete states and actions (decisions), with algorithms that assume that you can loop over all possible states and actions. This approach suffers from the classic "curse of dimensionality" which means that when the state variable is multidimensional, the state space becomes intractably large.

For this reason, dynamic programming has seen few applications in transportation and logistics. Not surprisingly, this is partly to blame for the dependence on myopic models and deterministic approximations found in transportation.

It turns out that the situation is even worse than we thought. Equation (5) actually suffers from three curses of dimensionality: the state space, the outcome space, and the action space. To avoid this problem, we adopt a new approach for approximating dynamic programming. As a first step, we could replace the value function with an approximation, producing a recursion that looks like:

$$\tilde{V}_t^+(S_t^+) = max_{x\in\mathcal{X}} C_t(x, S_t^+) + E\{\hat{V}_{t+1}^+(S_{t+1}^+)|S_t^+\} \tag{6}$$

On the right hand side of (6), we have an approximation $\hat{V}_{t+1}^+(S_{t+1}^+)$. On the left hand side, we use a placeholder that we call $\tilde{V}_t^+(S_t^+)$.

For the next step, we assume that $\hat{V}_t^+(S_t^+) = \hat{V}_t^+(R_t^+)$, which is to say that our approximation is purely a function of the resource state variable, and not the full information state. In fact, it is sometimes important to write the function in terms of an aggregated form of the resource state variable, which we could write $\hat{V}_t^{G+}(G(R_t))$. For the rest of our discussion, we do not include the aggregation function $G()$ explicitly, but the reader should understand that we can use this device at any time. Now, we have:

$$\tilde{V}_t^+(S_t^+) = max_{x\in\mathcal{X}} C_t(x, S_t^+) + E\{\hat{V}_{t+1}^+(R_{t+1}^+)|R_t^+\} \tag{7}$$

Our next problem is the expectation. For real problems, this is computationally intractable. We could approximate the expectation using a sample, as in:

$$\tilde{V}_t^+(S_t^+) = max_{x\in\mathcal{X}} C_t(x, S_t^+) + \sum_{\omega\in\hat{\Omega}} \hat{p}(\omega)\hat{V}_{t+1}^+(R_{t+1}^+(\omega)) \tag{8}$$

where $\hat{\Omega}$ is a sample from Ω and $\hat{p}(\omega)$ is probability of outcome $\omega \in \hat{\Omega}$.

Equation (8) can itself be quite hard, even when the sample $\hat{\Omega}$ is relatively small. In transportation problems, the basic one-period optimization model could represent a resource allocation problem with thousands of variables, or a difficult integer programming problem arising in vehicle routing or network design. We would prefer to use a single sample:

$$\tilde{V}_t^+(S_t^+, \omega) = max_{x\in\mathcal{X}} C_t(x, S_t^+, \omega) + \hat{V}_{t+1}^+(R_{t+1}^+(\omega)) \tag{9}$$

Now, we have created a decision function where x_t is allowed to "see" $R_{t+1}^+(\omega)$, which violates a basic information constraint. We avoid this problem by formulating our recursion in terms of our incomplete state variable:

$$V_t(S_t) = E\left\{max_{x\in\mathcal{X}} C_t(x, S_t^+) + V_{t+1}(S_{t+1})|S_t\right\} \tag{10}$$

Since S_t is incomplete, the decision x_t is a random variable, and as a result we have to pull the expectation outside of the max operator. Following the same path as before, we obtain the approximation:

$$\widetilde{V}_t\,(S_t, \omega) = max_{x\in\mathcal{X}} C_t(x, S_t, \omega) + \hat{V}_t(R_t(\omega)) \tag{11}$$

Note that we index $\hat{V}_t(R_t(\omega))$ by t instead of $t+1$ because it is a function of the information in time t. We have to devise an updating strategy that revises the estimates from one iteration to another. If n is our iteration counter, then we can just use the representation:

$$\hat{V}_t^n \leftarrow U^V(\hat{V}_t^{n-1}, \widetilde{V}_t^{\,n}, R_t^n) \tag{12}$$

The updating function $U^V(\cdot)$ could be nothing more than the use of exponential smoothing on a constant (this would be the case when we are using linear approximations) or a strategy for updating nonlinear approximations (specific examples are given in the next section).

We now have a general approximation strategy for dynamic programs, with two "hot spots." The first is that we have to devise an approximation scheme $\hat{V}_t(R_t)$. The second is that we have to exploit the structure of the resulting approximation to solve what is typically an integer program.

3.1.2. *Approximating the value function.* We propose using two classes of approximations for $\hat{V}_t$: linear, and nonlinear, separable. For problems where integer solutions are required (which is common in logistics problems), we would use a piecewise linear function instead of a continuously differentiable function (which might be attractive because of the low number of parameters needed to characterize it).

Linear functions are always the easiest to implement and use, but they can be unstable. Just the same, they serve as a useful illustration. Assume that the basic problem $\max_{x\in\mathcal{X}} C_t(x, S_t)$ is computationally tractable. Then,

$$\widetilde{V}_t^{\,n}\,(S_t, \omega_t) = \max_{x\in\mathcal{X}} C_t(x, S_t) + \hat{v}_{t+1}^n R_{t+1} \tag{13}$$

Adaptive dynamic programming algorithm

Step 1 Initialize all $\hat{V}_t(R_t)$ for all t. Set $n = 1$.

Step 2 Generate an outcome $\omega = (\omega_0, \omega_1, \ldots, \omega_{T-1})$.

Step 3 For $t = 0, 1, \ldots, T-1$, find:

$$x_t^n(S_t, \omega) = \arg\min_{x \in \mathcal{X}} \left\{ c_t(S_t, \omega_t, x) + \hat{V}_t^{n-1}(S_{t+1}(\omega, x)) \right\}$$

Update S_t using the system dynamics.

Step 4 For $t = T-1, T-2, \ldots, 1, 0$, update $\hat{V}_t^n$ for all t using the update function:

$$\hat{V}_t^n = U^V \left(\hat{V}_t^{n-1}, \widetilde{V}_t^{\,n}\,(R_t^n), R_t^n \right)$$

where:

$$\widetilde{V}_t^{\,n}\,(R_t) = \min_{x \in \mathcal{X}} \left\{ c_t(S_t, \omega, x) + \hat{V}_t^n(S_t(\omega, x)) \right\}.$$

Step 5 Let $n := n + 1$, and go to step 2.

FIGURE 1. Prototype of an adaptive dynamic programming algorithm

subject to:

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{ta} \tag{14}$$

$$x_{tad} \leq u_{tad} \tag{15}$$

should also be computationally tractable. If the problem is a continuous linear program, then we can use the dual variable for constraint (14) to help us estimate our linear approximation. Let $\tilde{v}_{ta}^n$ be the dual variable of equation (14) at iteration n. We may then estimate a linear approximation using:

$$\hat{v}_{ta}^n = (1 - \alpha^n)\hat{v}_{ta}^{n-1} + \alpha^n \tilde{v}_{ta}^n \tag{16}$$

Linear approximations can work well, but for the types of resource allocation problems that arise in fleet management, (separable) nonlinear approximations have proven to work the best. Although a number of strategies can be used to estimate nonlinear functions, the interest in obtaining integer solutions has led to the development of piecewise linear approximations. Thus, we can write our nonlinear approximation in the form:

$$\hat{V}_t(R_t) = \sum_{a \in \mathcal{A}} \hat{V}_{ta}(R_{ta})$$

Many problems in transportation and logistics require integer solutions. When this is the case, it is easiest to build piecewise linear approximations. Piecewise-linear concave value function approximation components are characterized by a series of break points

$\{u_0, u_1, u_2, \ldots, u_n\}$ and slopes v_l on the portion $[u_l, u_{l+1}]$ with $v_0 \geq v_1 \geq \ldots \geq v_n$. Then (dropping the subscripts and superscripts for state and time):

$$\hat{V}(R) = \sum_{l=0}^{m-1} \hat{v}_l(u_{l+1} - u_l) + \hat{v}_m(R - u_m) \tag{17}$$

where $m = \max\{l : u_l \leq R\}$. We can update $\hat{V}(R)$ using sample gradients. Let $\tilde{v}^n$ be a sample estimate of the dual variable of the resource constraint (14). When the underlying problem is a network, it is possible to get left and right gradients using flow augmenting paths (see Powell (1989)). When this is possible, let $\tilde{v}^+$ and $\tilde{v}^-$ be the right and left gradients, respectively (in the discussion below, if these are not available, simply let $\tilde{v}^+ = \tilde{v}^- = \tilde{v}$). We now want to use these gradients to update our slopes for $\hat{V}$. The idea is to use this information to update the function locally, while retaining the basic concavity of the function at all times.

This process is illustrated in figure 2. In figure 2(a), we have a concave estimate of the value function, along with new slopes at a particular point. Figure 2(b) shows that if we smoothed these new estimates of slopes into the immediate area of the estimate, we would obtain a nonconcave approximation. Figure 2(c) shows that if we expand the range over which we are smoothing the slopes, then the resulting updated function remains concave.

More formally, let u_l^n and $\hat{v}_l^n$ denote the breakpoints and slopes of the function at iteration n. To maintain concavity, we update the function over the range (l^-, l^+), given by:

$$\begin{aligned} l^+ &= \min\{l : u_l \geq R_{ta}^n, (1-\alpha^n)\hat{v}_l^n + \alpha^n\tilde{v}^{n+} \geq \hat{v}_{l+1}^n\} \\ l^- &= \max\{l : u_l \leq R_{ta}^n, (1-\alpha^n)\hat{v}_l^n + \alpha^n\tilde{v}^{n-} \leq \hat{v}_{l-1}^n\} \end{aligned}$$

Then for all $l \in [l^-, l^+]$ we update the slopes as:

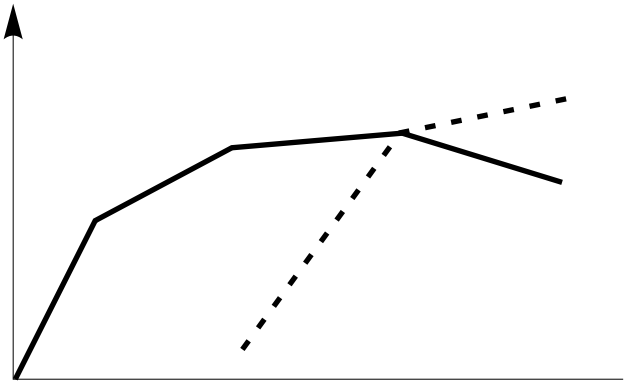
$$v_l^{n+1} = \begin{cases} (1-\alpha^n)v_l^n + \alpha^n\tilde{v}^{n-} & l < R^n \\ (1-\alpha^n)v_l^n + \alpha^n\tilde{v}^{n+} & l \geq R^n \end{cases}$$

to obtain the value function approximation at iteration $n+1$.

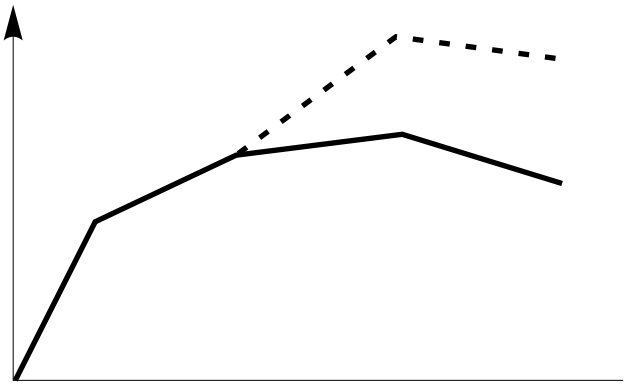
A somewhat simpler way of estimating a nonlinear function is via the SHAPE algorithm (Cheung & Powell (2000)). Here, the basic updating equation is given by:

$$\hat{V}^n(R) = \hat{V}^{n-1}(R) + \alpha^n\left(\tilde{v}^n - \nabla\hat{V}^{n-1}(R^n)\right) \cdot R \; R \geq 0 \tag{18}$$

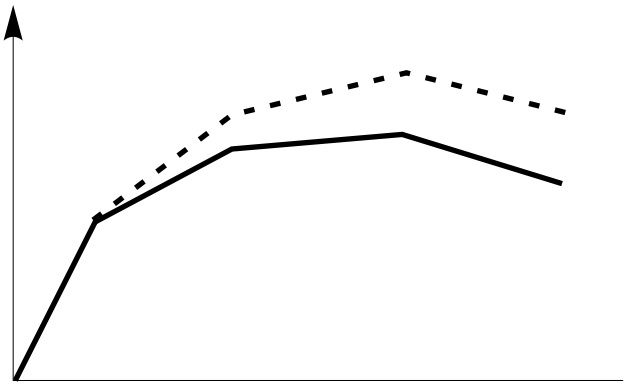
The basic idea is that we start with an initial approximation $\hat{V}^0$, and then successively "tilt" the function using the linear slope term $\left(\tilde{v}^n - \nabla\hat{V}^n(R^n)\right)R$, which serves as a correction



(a) Initial function with unsmoothed update



(b) Smoothing creates nonooncave functional approximation



(c) Expanding the smoothing range maintains concavity

FIGURE 2. Illustration of updating over a smoothing interval to maintain concavity

term by adding the difference between the current *estimate* of the slope of the function and the actual slope of the approximation. Since we want to maintain concavity, we should use a concave function, such as:

$$\begin{aligned}\hat{V}^0(R) &= \rho_0\left(1-e^{-\rho_1 R}\right)\\ \hat{V}^0(R) &= \ln(R+1)\\ \hat{V}^0(R) &= -\rho_0(x-\rho_1)^2\end{aligned}$$

If we need a piecewise linear function, any of these examples can be modeled as piecewise linear with breakpoints at each integer. SHAPE is provably convergent for continuously differentiable functions. If piecewise linear functions are used, it appears to provide very good results based on experimental testing. If we are solving sequences of network problems and have access to left and right gradients, we can use a two-sided version of SHAPE given by:

$$\hat{V}^{n+1}(R) = \begin{cases}\hat{V}^n(R)+\alpha^n\left(\tilde{v}^{n-}-\hat{V}^n(R^n)\right)R & R\le R^n\\ \hat{V}^n(R)+\alpha^n\left(\tilde{v}^{n+}-\hat{V}^n(R^n)\right)R & R\ge R^n\end{cases} \tag{19}$$

3.2. **Nonlinear value functions and multiperiod travel times.** Special care has to be used when adopting nonlinear functions. One issue that arises is in the context of multiperiod travel times. Consider two locations i and j sending vehicles to location k (see figure (3)). Assume that the travel time τ_{jk} from j to k is greater than that from i to k. If we use a nonlinear value function approximation, location j will "see" this function first, before the arrivals from i have been planned. As a result, location j will underestimate the total flow into the location, and therefore use the higher estimate of the slope of the function (the solid part of the function in figure (3)). By overestimating the value of resources at this location, the model is encouraged to move them a longer distance than might be necessary.

If we use linear value function approximations, both i and j see the same value of vehicles downstream, since the slope of a linear function is independent of the flow into the location. Presumably, our updating strategy will eventually find the right price (or slope) for resources in the future which will result in a solution that uses resources from j rather than i. But, when we use nonlinear value functions, this will not generally be the case. Location j will see the function first, and will price resources at the steepest part of the curve (since it is concave). If location j sends vehicles to k, location i will then see this decision (which at

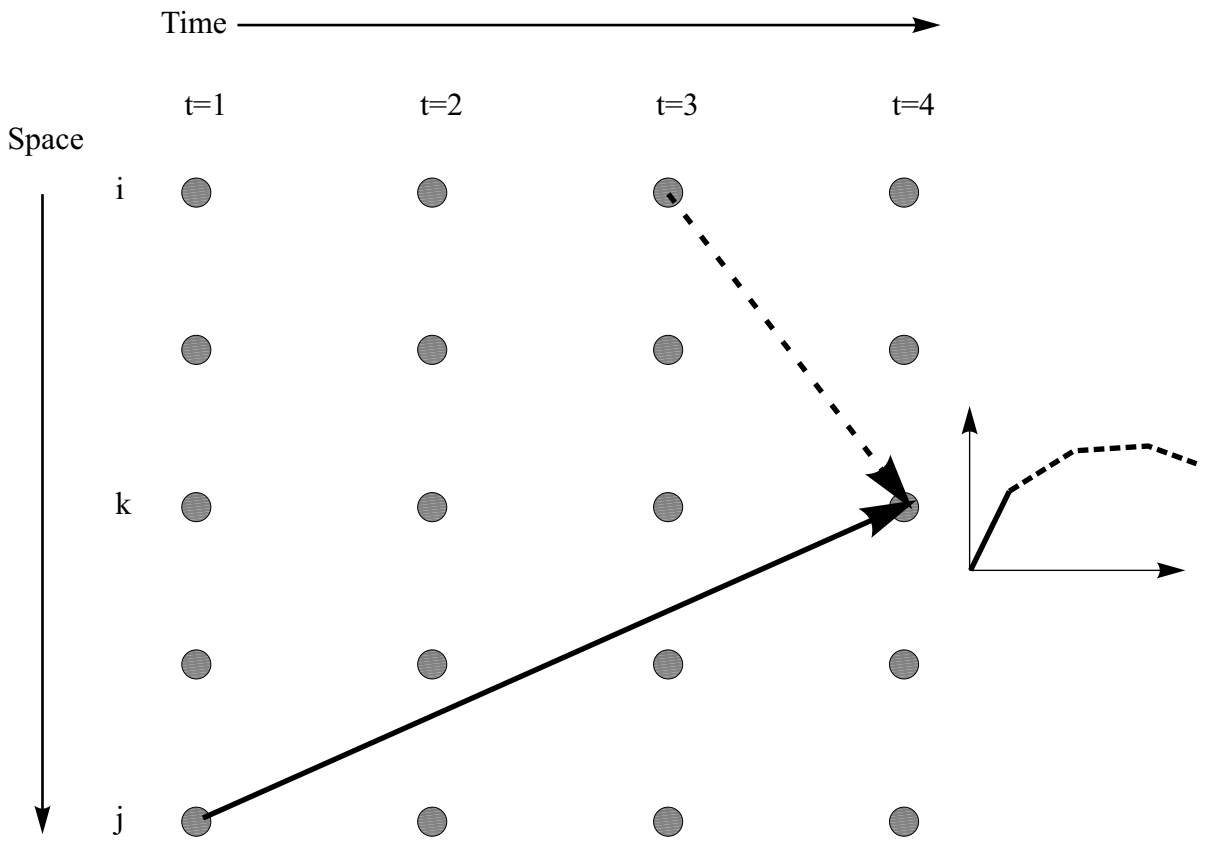


FIGURE 3. The challenge of using nonlinear value functions with multiperiod travel times

time t' has already been made) and subsequently value additional resources at a smaller slope.

Our solution to this problem is relatively simple. First define:

$R_{tt'}$ = The number of resources that we know about at time t that can be used (acted on) at time t'.

In a particular subproblem, we may act on resources in R_{tt}, whereas resources captured by $R_{tt'}, t' > t$ would represent resources that are en route and will not arrive until some point in the future.

We use a value function approximation that is separable over time:

$$\hat{V}_t(R_t) = \sum_{t' \geq t} \hat{V}_{tt'}(R_{tt'})$$

Let:

$x_{tt'}$ = The vector of decisions made at time t producing resources that will become available at time t'.

$x_t = (x_{tt'})_{t' \geq t}$

$\bar{R}_{tt'}(x_t) = A_{tt'}x_t$, where $A_{tt'}$ is a matrix that sums the elements in x_t that arrive to locations in time t'.

Finally, we would like to define the cumulative number of resources at time t' that we know about at time t, including decisions made before time t:

$R_{tt'}$ = The cumulative number of resources that will become available at time t' made *before* time t.

$= \sum_{\bar{t}<t} A_{\bar{t}t'}x_{\bar{t}t'}$,

$= R_{tt')+\bar{R}_t t'}$.

Thus, $R_{tt'} + \bar{R}_{tt'}(x_{tt'})$ is the total number of resources that will be available at time t' that we know about at time t, including the effect of decisions made at time t.

Our basic approximation strategy involves solving problems of the form:

$$\widetilde{V}_t\left(R_t, \omega_t\right) = max_{x \in \mathcal{X}} C_t(x, S_t) + \sum_{t'>t} \hat{V}_{tt'}(R_{tt'} + \bar{R}_{tt'}(x_t, \omega_t)) \tag{20}$$

which is solved subject to:

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{t,at} + \hat{R}_{t,at} \tag{21}$$

$$\sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} x_{tad}\delta_{t'a'}(t, a, d) - R_{t,a't'} = \hat{R}_{t,a't'} \tag{22}$$

Let $\tilde{v}_{t,at}$ be the dual variable with respect to equation (21) and let $\tilde{v}_{t,a't'}$ be the dual variable for equation (22). Equation (22), then, captures the impact of a decision made before time t on problem t by creating resources that become actionable at time t'. This issue did not arise with single period travel times, or with linear approximations. If possible, we will try to find the value of one more and one less resource. In this case, the duals are denoted $\tilde{v}^+$ and $\tilde{v}^-$, respectively.

The updating strategy is basically the same as before:

$$\hat{V}_{tt'}^{n+1} \leftarrow U^V\left(\hat{V}_{tt'}^n, \tilde{v}_{tt'}^{n-}, \tilde{v}_{tt'}^{n+}, (R_{tt'}^n + \bar{R}_{tt'}^n)\right) \tag{23}$$

We are updating the slopes around the point $(R^n_{tt'} + \bar{R}^n_{tt'})$, since we are effectively approximating the value function as a function of the number of resources that we know about at time t.

Note that we only solve a single problem at time t, and yet we approximate functions of the form $\hat{V}_{tt'}$. In the case of problems that can be formulated as multistage linear programs (which covers most of the problems that arise in this setting), we would use the dual variables for the resource constraints, $\tilde{v}^n_{tt'}$ to update separable nonlinear approximations using either SHAPE or CAVE.

One step that can dramatically accelerate the rate of convergence (especially when some travel times are quite long, measured in units of time periods) works as follows. Instead of using the dual variable $\tilde{v}_{tt'}$, we instead use:

$$\bar{v}^-_{t,at'} = \min_{t \le \bar{t} \le t'} \{\tilde{v}^-_{\bar{t}at'}\} \tag{24}$$

$$\bar{v}^+_{t,at'} = \max_{t \le \bar{t} \le t'} \{\tilde{v}^+_{\bar{t}at'}\}. \tag{25}$$

Equation (24) uses the best dual variable of all the subproblems that are sending resources to state a at time t'. This has the effect of quickly finding the best location that should send resources arriving at time t' and then building the value of this location into the value function approximation for time t. We would then use $\bar{v}^-$ and $\bar{v}^+$ instead of $\tilde{v}^-$ and $\tilde{v}^+$.

3.3. **An algorithmic metastrategy for multiagent problems.** Large, complex systems such as trucking companies, railroads, and intermodal operations are almost always characterized by a number of decision makers (or agents) solving different parts of the problem, each with their own information. In section 2.3 we introduced the basic notation required to handle multiagent thinking.

The challenge of multiagent problems, of course, is trying to devise a strategy that allows each agent to behave independently, but using information that encourages the agents to behave in a coordinated way. We are going to accomplish this using a relatively minor adjustment to our basic dynamic programming recursion. In fact, we are going to find that the presentation in sections 3.1 and 3.2 is most of what we need. The handling of multiple agents is more a change in perspective than an entirely new class of techniques.

The transition to multiagent thinking involves making the transition from stepping through time, to one of stepping through areas of control as well as time. In a time-staged environment, we made decisions at each time period $t \in \mathcal{T}$. We can think of each of these decision epochs as a decision with a different set of information. We can formulate a kind of dynamic programming recursion using:

$$\widetilde{V}_{qq}\,(R_{qq}) \;=\; max_{x_q} C_q(x_q) + \sum_{q' \in \vec{\mathcal{M}}_q} \hat{V}_{qq'}(R_{qq'} + \bar{R}_{qq'}(x_{qq'})) \tag{26}$$

Note the similarities between equation (23) and (26). In fact, if we assume that the agent subproblems are solved in sequence, then we can view the multiagent problem in a manner identical to a time staged formulation by simply using q as the time variable. Of course, there is no assurance that such a sequencing would occur.

We can solve equation (26) using the same approximation techniques that we used for the time-staged problem, and the same updating schemes. In fact, the same issue arises when we decide to use nonlinear value functions as arises in the context of multiperiod travel times.

$$\hat{v}^{-}_{q,aq'} \;=\; \min_{q \le \bar{q} \le q'} \{\tilde{v}^{-}_{\bar{q}aq'}\} \tag{27}$$

$$\hat{v}^{+}_{q,aq'} \;=\; \max_{q \le \bar{q} \le q'} \{\tilde{v}^{+}_{\bar{q}aq'}\}. \tag{28}$$

We then use $\hat{v}^{-}_{q,aq'}$ and $\hat{v}^{+}_{q,aq'}$ to update $\hat{V}_{q,aq'}$.

If we use a linear approximation for $\hat{V}$, we do not need the double indexing (qq'). In fact, linear approximations fall in the general strategy of pricing systems in multi-agent systems. Nonlinear approximations do not seem to have been considered. But, as we have discussed (primarily in the context of multiperiod travel times) they offer special challenges that need to be addressed.

3.4. **Classes of decision functions.** We now have the foundation to introduce a very general class of decision functions. We return to our four classes of information: knowledge (K_q), forecasts of exogenous processes (Ω_q), plans (x^p) and values $(V_{qq'})$. We illustrate each of these classes of information by briefly describing a decision function based on knowledge alone, or knowledge paired with each of the other three classes of information by themselves, creating four combinations of information sets. Each of these combinations produces a decision function that falls within a major class of algorithms.

Information set	Function class	Designation
K_q	Myopic policies	Π^M
K_q, Ω_q	Rolling horizon policies	Π^{RH}
K_q, x^p	Proximal point algorithms	Π^{PP}
$K_q, V_{\vec{\mathcal{M}}_q}$	Dynamic programming	Π^V

TABLE 1. Summary of elementary classes of decision functions

This discussion is intended to emphasize that optimizing dynamic systems can come in a variety of forms. It is very common in the transportation and logistics community to assume the use of myopic policies, or rolling horizon policies based on deterministic forecasts of future activities. Both are valid approximations which can work well in certain situations. But, they overlook the other two classes of decision functions, or the possibility of mixing information to form a hybrid strategy.

We now discuss each class of decision function.

3.4.1. *Myopic policies* (Π^M). We start with knowledge alone. These decision functions know the state of the system, but do not make any forecast of the future. These represent myopic policies, which we designate by Π^M. The information set for myopic policies is represented by: $I_q^M = (K_q)$.

Myopic policies are the most widely used in practice (humans predominantly use myopic policies). LTL companies use myopic policies to determine when trucks should be dispatched and the routing of freight through the network. The most basic dispatch rule is a control limit policy. Let $X_{tij} = 1$ if a truck should be dispatched from i to j at time t, and 0 otherwise. Let R_t be the amount of freight weight to be dispatched. Then a basic dispatch rule is simply:

$$X_t^\pi = \begin{cases} 1 & R_t \geq \bar{d}_t \\ 0 & \text{Otherwise} \end{cases}$$

Here, $\bar{d}_t$ is a dispatch rule. If the amount of freight is at least $\bar{d}_t$, then we dispatch the truck. Otherwise, we hold. In LTL carriers, the basic rule will typically be "send the truck if full until the end of the freight cycle; if it is the last dispatch of the night, send the truck if it has at least a certain amount of freight." Such a policy would be used if there is a strong daily cycle to the freight, as would happen if the freight is arriving from the city trucks coming off the street. Dispatchers know when they are filling up the last truck of the night. If there are

only a few shipments on the truck, the carrier will typically hold the freight until the next day, resulting in a service failure (with some insight, the carrier has held a few noncritical freight bills to the side).

Another example of a myopic policy is a dynamic assignment problem where we are assigning drivers to loads. Let $\mathcal{R}_t$ be the set of drivers available to be assigned at time t, and $\mathcal{L}_t$ the set of loads. We may optimize the assignment of drivers to loads using a simple assignment problem:

$$min_x \sum_{r\in\mathcal{R}_0}\sum_{l\in\mathcal{L}_0} c_{0rl}x_{0rl} \tag{29}$$

subject to:

$$\sum_{r\in\mathcal{R}_0} x_{0rl} \leq 1 \ \forall l \in \mathcal{L}_0 \tag{30}$$

$$\sum_{l\in\mathcal{L}_0} x_{0rl} \leq 1 \ \forall r \in \mathcal{R}_0 \tag{31}$$

Again, we are using only the information we know at time t.

3.4.2. *Rolling horizon policies* (Π^{RH}). Rolling horizon policies combine what we know now (our knowledge base) with forecasts of the future over a planning horizon. We let $\mathcal{T}_t^{ph}$ be the set of points in time in our planning horizon given that we are planning a system at time t. Our information set for a rolling horizon policy, then, would be expressed by:

$$I_t = (K_t, \Omega_t)$$

where $\Omega_t = (\Omega_{tt'})_{t'\in\mathcal{T}_t^{ph}}$ is the set of events that we have forecasted in the future given what we know at time t. In practice, Ω_t contains a single outcome representing a *point forecast*, and we are going to assume that we are using a point forecast here. For example, if we are trying to allocate containers to meet future demand, we would normally forecast what we would *expect* would happen. The biggest challenge of using distributional forecasts ($|\Omega_t| > 1$) is the lack of effective tools for solving problems under multiple future scenarios (by contrast, we do not have any difficulty using distributional forecasts when we use value functions).

Consider, for example, the basic assignment problem we formulated in (29) - (30). Assume we can generate a forecast of resources and tasks in the future. Thus, $\omega \in \Omega$ would correspond

to $(\hat{\mathcal{R}}_t, \hat{\mathcal{L}}_t)_{t\in\mathcal{T}^{ph}}$. We might want to allow a resource at time t to be assigned to a task at time $t' > t$, so we let:

$$\begin{aligned}
\mathcal{R}_t &= \text{The cumulative set of all resources available at time } t \text{ or some time in the future.} \\
&= \mathcal{R}_t \bigcup_{t'\geq t} \hat{\mathcal{R}}_{t'} \\
\mathcal{L}_t &= \text{The cumulative set of all tasks available at time } t \text{ or some time in the future.} \\
&= \mathcal{L}_t \bigcup_{t'\geq t} \hat{\mathcal{L}}_{t'}
\end{aligned}$$

Under this forecast, we would solve the following problem:

$$min_x \sum_{t\in\mathcal{T}^{ph}} c_t x_t$$

subject to:

$$\sum_{l\in\mathcal{L}_t} x_{trl} \leq 1 \ \forall r \in \mathcal{R}_t$$
$$\sum_{r\in\mathcal{R}_t} x_{trl} \leq 1 \ \forall l \in \mathcal{L}_t$$

The myopic version of the assignment problem can be criticized because we might take a driver and assign it to a less valuable load now, when we could have used it on a more valuable load later. By contrast, when we use a deterministic forecast, the rolling horizon procedure could have us holding a driver now, even though there is a load available, for a load in the future that may never materialize.

Myopic policies, and rolling horizon procedures, are the most widely used techniques in practice for solving dynamic problems in transportation and logistics. Myopic policies tend to work well in situations that are either highly dynamic, and when rules can be devised which reflect the outcomes that might happen in the future. For example, in our assignment problem, we might have a basic rule that we will not assign a driver to a load shorter than 500 miles (since it probably pays too little). Thus, if the only load we have available to us is only 200 miles, we will refuse the assignment, knowing that there is a good likelihood that a longer load will become available shortly. Thus, a good myopic policy can work quite well.

3.4.3. *Proximal point algorithms* (Π^{PP}). Often overlooked in the design of algorithms is the value of making decisions that reflect either a forward looking plan, or past patterns of behavior. We claim that both of these represent instances of planning, and should be reflected in decisions made now.

Assume we are managing the flows of intermodal containers on a global level. A separate planning process has made a projection of the number of containers which should move from one location to another on a week by week basis for the next 10 weeks. We can represent this plan using the basic form:

$$x^p_{tad} = \text{The number of containers with attribute } a \text{ to which we will apply decision } d \text{ at time } t.$$

A plan is almost always expressed at some level of aggregation. Thus, we may have 30 types of containers, but we may plan for only the five major groups. Similarly, it may be necessary to send containers to specific locations, but our plan may express decisions only on a regional level. For simplicity, we may let $\hat{a}$ represent an aggregation of the attribute vector, and $\hat{d}$ an aggregation of a decision (such as, the decision to send to a region instead of a specific location). Similarly, we may aggregate time as well (total flow over a week instead of on a particular day). Our vector x^p, then, is expressed at a fairly aggregate level.

In a number of operations, there is not an explicit plan, but there is a *pattern* of activity. In this setting, we may define x^p_{tad} as the average flow which satisfies the pattern (a, d). When looking at past history, we would aggregate time into a period such as a day of week. As with planning, averaging past history is usually done at a more aggregate level.

Now we wish to solve a problem which we might express as:

$$\min_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} c_t x_t \tag{32}$$

where $\mathcal{T}$ is the set of time periods in the planning horizon. It is intuitively reasonable to make decisions that do not deviate from a plan by too much. At the same time, if x^p represents a summary of past patterns of behavior, we can also argue that our optimization model should not deviate too much from past patterns. We can achieve this by modifying

our basic optimization problem (32) as follows:

$$\min_{x\in\mathcal{X}} \sum_{t\in\mathcal{T}} c_t x_t + \rho||G(x) - x^p|| \tag{33}$$

Here, $G(x)$ is an aggregation function that maps our decision variable x (which presumably is fairly detailed) back into the more aggregated space that we are using to plan. The term $\rho||G(x) - x^p||$ is precisely the term used in Rockafellar's proximal point algorithm, which solves sequences of problems of the form:

$$x^{n+1} = \arg\min_{x\in\mathcal{X}} \sum_{t\in\mathcal{T}} c_t x_t + \rho||x - \bar{x}^n||$$

where:

$$\bar{x}^{n+1} = (1-\alpha^n)\bar{x}^n + \alpha^n x^{n+1}$$

3.4.4. *Dynamic programming* (Π^V). Our last information class is $I_q^{DP} = (K_q, V_q)$. Here, we want to make decisions that reflect what we know, and the impact of our decisions on other parts of the problem. The conceptual framework is precisely that of dynamic programming, which we have already covered in earlier sections. Return to our illustrative assignment problem, but now let's try to solve it over time, with multiple potential outcomes in the future. This would be formulated as:

$$min_{\pi\in\Pi} E\{\sum_{t\in\mathcal{T}} c_t X_t^\pi\}$$

We can formulate this using a basic dynamic program:

$$V_t^+(S_t^+) = max_{x\in\mathcal{X}} C_t(x, S_t^+) + E\{V_{t+1}^+(S_{t+1}^+)|S_t^+\}$$

but these are rarely solvable. Instead, we resort to our approximation strategy:

$$\widetilde{V}_t\,(S_t, \omega_t) = \max_{x\in\mathcal{X}} C_t(x, S_t^+) + \hat{V}_{t+1}(R_{t+1}(\omega_t))$$

where the goal is to devise a version of $\hat{V}$ which will produce a near-optimal solution. We can think of this as a function:

$$X_t^\pi(S_t, \hat{V}_{t+1}) = \arg\max_{x\in\mathcal{X}} C_t(x, S_t^+) + \hat{V}_{t+1}(R_{t+1}(\omega_t))$$

This expression uses a state variable which, in our information-theoretic vocabulary, represents our knowledge base.

It is significant that a dynamic-programming based approach, which uses value functions to capture the impact of decisions made now on the future, incorporates uncertainty relatively

easily. The effect of different possible outcomes is captured in the value function $\hat{V}$, which is much simpler than solving a problem at time t with an explicit set of multiple scenarios in Ω. Since the function $\hat{V}$ is estimated over a number of iterations, it is useful to use the notation $V(\Omega)$ to represent the information content of a value function. Specifically, a decision function which uses value functions is implicitly using a forecast of exogenous outcomes, expressed through the value functions.

Adding value functions to a decision function is equivalent to using a forecast of the impact of a decision on another agent. Companies do this all the time when the decision is to purchase supplies, and the agent is a supplier. The value function, then, is usually a linear function that is the price of the product times the quantity. A car distribution manager for a railroad might implicitly use a value function when he looks at a region and recognizes that there is a surplus (marginal value of additional equipment is small) or a deficit (marginal value of additional equipment is large). The distribution manager is implicitly using a nonlinear value function if he is also thinking "this region needs 20 additional cars."

As a rule, humans have difficulty with value functions because it explicitly requires using costs to make decisions. Human decision making is based on the concept of state/action pairs: if the system is in this state, then take this action. Recognition of this fact is the basis for artificial intelligence. The application of AI to complex problems have typically failed simply because the state variable is far too complex. The power of the brain to sort through patterns to identify the relevant portion of the state variable has not been matched on the computer. Cost-based optimization models, on the other hand, have little difficulty with very complex state variables. Computers are good at adding up costs to make a decision, which is the reason that math programming-based models have proven to be so popular. Needless to say, value functions appear to be most useful to computer-based models and algorithms. If you tell a human that you are going to give him a value function to help him make a decision, the response is generally going to be disappointing.

3.5. **A hybrid model.** We have seen that four information classes each produce a different class of algorithms that have been widely studied. This raises a natural question of whether we can combine all four classes. We propose to do this by incorporating forecasts (Ω) through the value function as we did in dynamic programming. Thus, our information set is given

by:

$$I_t \quad = \quad (K_t, x_t^p, V_{t+1}(\Omega))$$

Such a decision function would look like:

$$X_t^\pi(S_t, \hat{V}_{t+1}) \quad = \quad \arg\max_{x_q \in \mathcal{X}_q} C_q(x_q, S_t^+) - \rho||G(x) - x^p|| + \sum_{q' \in \vec{\mathcal{M}}_q} \hat{V}_{qq'}(R_{qq'} + \bar{R}_{qq'}(x_{qq'}, \omega))$$

(34)

We offer equation (34) as a relatively general function which is scalable to very large problems such as railroads and trucking companies. Not only does it incorporate all four information classes, it also handles the multiagent structure common to complex operations. At the same time, it is important to realize that it is not necessary to use the ultimate decision function, since value can be obtained using much simpler functions, and all the more basic decision functions, including myopic policies, can be very effective.

4. Modeling operational problems

The next step is to apply our framework to specific operational problems in transportation and logistics. An effective way to classify operational problems is to begin by organizing them on the basis of how resources interact. There are three fundamental ways to change a resource:

1) Couple - Combine two resources to create a layered resource consisting of two or more resources.
2) Uncouple - Break down a composite resource into its primitive components (or simply decouple one resource from a set of layered resources).
3) Modify - Major classes of modify include: a) move (from one location to the next), b) entry (such as purchasing a resource), c) exit (a resource leaves the system), and d) do nothing. Other examples might include: perform maintenance on an engine, clean out a trailer, have a driver go on vacation.

Different problem classes can often be created based on the type of coupling they entail. Special classes of interest in transportation include:

1) One to one - such as one driver and one load, one pilot and one plane, one box car and one customer demand.
2) Several to one - several locomotives pull one train, two drivers may create a sleeper team to drive a tractor, several customers can fit in one vehicle.
3) Many to one - many freight bills or packages may fit in one trailer, many boxcars fit in one train.

The several-to-one class has some important variations in transportation. The first is the bundling of several resources with a common location (multiple locomotives at a location being assigned to the same train; two drivers being assigned to move the same tractor). The second is bundling resources with different locations (clustering), such as occurs in the vehicle routing problem.

Our discussion proceeds in stages. We start with resource allocation problems which are all in the class of one-to-one coupling problems. These are described starting with single layer problems (section 4.1), two-layer problems (section 4.2) and finally multi-layer problems (section 4.3). Finally, we turn to problems that involve bundling (section 4.4).

4.1. **Single layer resource allocation.** Fundamental to operational problems is the coupling of two layers (product with customer, driver with load, vehicle with delivery). We might say that the "energy" derived from coupling two resource layers together is what keeps the process moving. So, how can we even have a single-layer problem? The answer is simple: any time we have demands that must be satisfied at a particular point in time. In production problems, this means no backlogging of demand. In transportation and logistics, it often means that there are "tight time windows." For example, we would have a one layer problem if we were assigning locomotives to trains, where the trains had to be moved at a point in time. The same would be true if we are moving box cars to serve demands that have to be served at a particular point in time.

We are interested in problems where we are managing a set of reusable resources. These problems arise when we are managing sets of containers (trailers, boxcars, intermodal containers), vehicles (tractors, locomotives, aircraft) and people (drivers, pilots, crews). In this

section, we are going to focus on problems where the number of resources being managed is relatively large, which means that it is typically not useful to track each individual resource.

Most of the time, representing these problems as single-layer resource allocation problems can be justified only as simplifications of real-world problems. But, the one layer problem serves not only as a useful pedagogical tool, but it is also practical for some problem classes.

Throughout our discussion, our solution strategy is assumed to follow the framework described in section 3. For the most part, we are primarily concerned with how to solve the basic problem:

$$\max_{x \in \mathcal{X}} c_t x_t$$

which means, "what do we do at time t?" Note that what we do at time t may consist of a series of steps that extend into the future. Recall that $x_t = (x_{tt'})_{t' \geq t}$, meaning a vector of decisions over time using information that we know at time t. Thus, $c_t x_t$ is equivalent to $\sum_{t' \geq t} c_{t'} x_{tt'}$. As we proceed, it is important to be clear whether we are solving a problem at time t with actions strictly at time t, or whether the actions may extend into the future using the information at time t.

We proceed with the expectation that including plans or value functions would not destroy the fundamental structure. There are different ways to incorporate the effect of plans, with the use of the term $||G(x) - x^p||$ only one of them. If we include value functions, we note that linear value functions will never destroy structure, but nonlinear functions (even separable nonlinear functions) must be handled with care.

Our discussion of resource allocation proceeds in a progression from single commodity (section 4.1.1) to multicommodity (section 4.1.2) to heterogeneous resources (section 4.1.3). In all three of these sections our subproblems consist of a single set of actions initiated at time t.

4.1.1. *Single commodity.* Single commodity problems arise when a) the attribute vector a consists only of a scalar state variable (which in transportation problems usually represents a geographical location), and b) when a resource must be in the same state as a task to serve the task. In transportation applications, it is very common for the "state" of a resource to

be a geographical location. If we have only one type of resource, we would use $a = (i)$. For this section, we use the index i instead of the attribute vector a to emphasize the structure of the problem. Our purpose in switching to a different notation can be explained by the desire to exploit structure that arises only in the context of single commodity problems. We are going to continue to use this specialized notation when we discuss multicommodity problems, which also exhibit special structure.

We have two types of decisions for this problem class:

$\mathcal{D}^s =$ Decisions to serve a task. The set $\mathcal{D}^s$ may be a set of specific tasks, or a set of task types. We let $\mathcal{D}_i^s$ be the set of tasks that can be served by a resource in state i.

$\mathcal{D}^r =$ Decisions to reposition a resource from one state to another.

$u_{tid} =$ Upper bound on the number of times that decision $d \in \mathcal{D}_i$ may be executed. We assume that u_{tid} is bounded for $d \in \mathcal{D}^s$, and unbounded for $d \in \mathcal{D}^r$.

Similarly, we assume that:

$$M_t(t, i, d) \quad = \quad (i^M_{tid}, c_{tid}, \tau_{tid})$$

We use the notation i^M_{tid} as our terminal attribute function instead of a^M_{tid} to be consistent with our adoption of the simple state notation i instead of the more general attribute vector notation a for single commodity flow problems.

A myopic version of the problem (at time t) is given by:

$$\max \sum_{i,j \in \mathcal{I}} \sum_{d \in \mathcal{D}} c_{tid} x_{tid} \tag{35}$$

subject to:

$$\sum_{d \in \mathcal{D}_i} x_{tid} \quad = \quad R_{ti} + \hat{R}_{ti} \tag{36}$$

$$x_{tid} \quad \leq \quad u_{tid} \tag{37}$$

Such a formulation would never work because we would never reposition resources from where we need them to where we want them. Virtually all transportation companies which solve resource allocation problems require some sort of mechanism (typically, a central planning

group) which looks into the future and makes decisions about repositioning. The simplest model which looks into the future is based on a deterministic forecast over a planning horizon. We may be using a forecast of new resources ($\hat{R}_t$), upper bounds (u_t), times (τ_{tid}) and costs (c_{tid}):

$$\max \sum_{t' \in \mathcal{T}_t^{ph}} \sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}} c_{t'id} x_{t'id} \tag{38}$$

subject to, for $t' \in \mathcal{T}_t^{ph}$:

$$\sum_{d \in \mathcal{D}_i} x_{t'id} = R_{t'i} + \hat{R}_{t'i} \; \forall j \in \mathcal{I} \tag{39}$$

$$\sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}_i} x_{t'-\tau_{tid},id} \delta_{t'j}(t' - \tau_{tid}, i, d) = R_{t'j} \; \forall j \in \mathcal{I} \tag{40}$$

$$x_{t'id} \leq u_{t'id} \tag{41}$$

$$x_{t'id} \geq 0 \tag{42}$$

Equations (39) and (40) can be combined to create classical flow conservation constraints. We retain this form since it creates a more natural transition with stochastic models. The problem (38) - (42) is a pure network and is easily solved as a general linear program or with more specialized solvers.

Solving rolling horizon problems using deterministic forecasts is popular and can be effective, but suffers from several limitations: it uses point forecasts of demands (which means it may not supply enough capacity to provide a high level of service), and it takes a problem where all you can do is determine what to do right now (since information will change in the future) and formulates a problem where you are making decisions over an extended planning horizon, which is inherently more difficult. We overcome these limitations by using our dynamic programming approximations and solve:

$$\widetilde{V}_t\left(R_t, \omega_t\right) = \max_{x \in \mathcal{X}} \sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}} c_{tid} x_{tid} + \sum_{t' > t} \sum_{j \in \mathcal{I}} \hat{V}_{t+1,jt'}(R_{t+1,jt'} + \bar{R}_{t+1,jt'}(x_t, \omega_t)) \tag{43}$$

If we use a linear approximation for $\hat{V}$, then equation (43) reduces to:

$$\widetilde{V}_t(R_t,\omega_t) = \max_{x\in\mathcal{X}} \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} c_{tid}x_{tid} + \sum_{t'>t}\sum_{j\in\mathcal{I}} \hat{v}_{t+1,jt'}\left(R_{t+1,jt'} + \bar{R}_{t+1,jt'}(x_t,\omega_t)\right) \tag{44}$$

$$= \max_{x\in\mathcal{X}} \left\{\sum_{i,j\in\mathcal{I}}\sum_{d\in\mathcal{D}} c_{tid}x_{tid}\right\} + \left\{\sum_{t'>t}\sum_{j\in\mathcal{I}} \hat{v}_{t+1,jt'}R_{t+1,jt'}\right\}$$

$$+ \left\{\sum_{t'>t}\sum_{j\in\mathcal{I}} \hat{v}_{t+1,jt'}\bar{R}_{t+1,jt'}(x_t,\omega_t)\right\} \tag{45}$$

The second term in brackets on the right side of (45) is not a function of x_t and hence can be ignored. The third term can be simplified by using:

$$\bar{R}_{t+1,jt'} = \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}_i} \delta_{t'j}(t,i,d)x_{tid} \tag{46}$$

Dropping the second term in brackets in equation (45) and substituting equation (46) into (45) gives:

$$\widetilde{V}_t(R_t,\omega_t) = \max_{x\in\mathcal{X}} \left\{\sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} c_{tid}x_{tid}\right\} + \left\{\sum_{t'>t}\sum_{j\in\mathcal{I}} \hat{v}_{t+1,jt'}\sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} \delta_{t'j}(t,i,d)x_{tid}\right\} \tag{47}$$

$$= \max_{x\in\mathcal{X}} \left\{\sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} c_{tid}x_{tid}\right\} + \left\{\sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}}\left(\sum_{t'>t}\sum_{j\in\mathcal{I}} \delta_{t'j}(t,i,d)\hat{v}_{t+1,jt'}x_{tid}\right)\right\} \tag{48}$$

We note that:

$$\sum_{t'>t}\sum_{j\in\mathcal{I}} \delta_{t'j}(t,i,d)\hat{v}_{t+1,jt'}x_{tid} = \hat{v}_{t+1,i^M_{tid},t+\tau_{tid}}x_{tid} \tag{49}$$

Equation (49) simply says that if we act on a resource in state i at time t with decision d and it produces a resource in state $j = i^M_{tid}$ at time t', then we can pick up the value of that resource. This allows us to reduce (48) to:

$$\widetilde{V}_t(R_t,\omega_t) = \max_{x\in\mathcal{X}} \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}}\left(c_{tid} + \hat{v}_{t+1,i^M_{tid},t+\tau_{tid}}\right)x_{tid} \tag{50}$$

Equation (50) shows us that using a linear approximation of the value function is equivalent to adding a price to each assignment that is the marginal value of the resource in the future. In fact, if we look at the updating equation for linear approximations, we quickly see that $\hat{v}_{tt'} = \hat{v}_{t'}$, allowing us to further simplify (50) to:

$$\widetilde{V}_t(R_t,\omega_t) = \max_{x\in\mathcal{X}} \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}}\left(c_{tid} + \hat{v}_{i^M_{tid},t+\tau_{tid}}\right)x_{tid} \tag{51}$$

Linear approximations introduce an additional simplification: problem (50) decomposes by location. Thus, we can solve (51) by solving a sequence of problems that look like:

$$\widetilde{V}_{ti}\left(R_{ti},\omega_t\right) = \max_{x\in\mathcal{X}} \sum_{d\in\mathcal{D}_i} \left(c_{tid} + \hat{v}_{i^M_{tid},t+\tau_{tid}}\right) x_{tid} \tag{52}$$

Furthermore, the solution of (52) involves nothing more than a sorting of decisions $d \in \mathcal{D}_i$ in order of $(c_{tid} + \hat{v}_{i^M_{tid},t+\tau_{tid}})$.

Linear approximations are especially appealing since they are so simple. In practice, they can be unstable. If the term $(c_{tid} + \hat{v}_{i^M_{tid},t+\tau_{tid}})$ is attractive, we end up with a large value for x_{tid}. If $d \in \mathcal{D}^s$, which means we are serving a task, then the number of tasks serves as a natural upper bound which stabilizes the solution. If $d \in \mathcal{D}^r$, then typically u_{dt} is unbounded, and we can get extreme flows.

There are three solutions to this behavior. One is to add an artificial upper bound, which we might call y_{tid}, where y is a decision variable. We would then solve the same problem with the added constraint $x_{tid} \leq y_{tid}$. We then have to introduce procedures for changing the artifical controls y. This approach was used in Powell & Carvalho (1998) with reasonable success. But, it does not generalize easily to multicommodity and heterogeneous resources (see below).

A second approach is to include a nonlinear stabilization term. One framework for including such a term is to use a proximal point algorithm, where at iteration n we would solve:

$$x^n = \arg\max_{x\in\mathcal{X}} \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} \left(c_{tid} + \hat{v}_{t+1,j,t+\tau_{tid}}\right) x_{tid} + \theta \sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} (x_{tid} - \bar{x}^n_{tid})^2 \tag{53}$$

with the updating scheme:

$$\bar{x}^{n1} = (1-\alpha^n)\bar{x}^{n-1} + \alpha^n x^n$$

The proximal term $(x_{tid} - \bar{x}^n_{tid})^2$ helps to stabilize the solution, and because the additional term is separable, it does not generally cause serious algorithmic headaches. If we are looking for integer solutions, then a piecewise linear penalty term should be used.

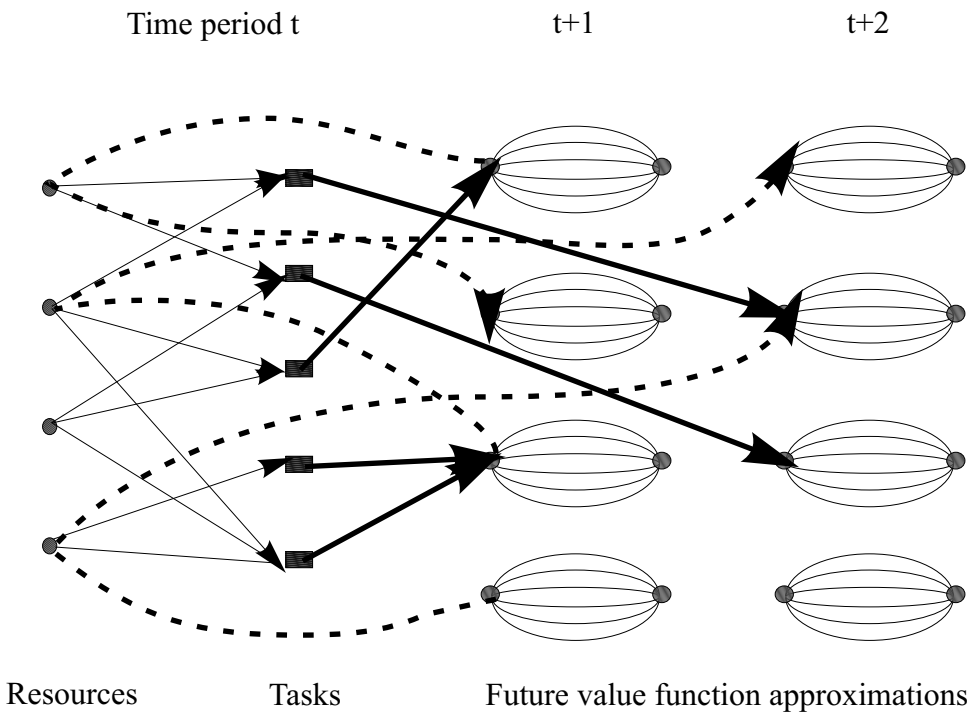


FIGURE 4. Illustration of a single commodity flow problem at time t with separable, nonlinear value function approximations. Thin, solid arcs represent assignment of resources to tasks. Thick solid arcs are tasks moving forward in time. Dashed arcs represent repositioning moves in response to future value function approximations.

A third approach is to use a nonlinear value function approximation. Separable functions of the form:

$$\hat{V}_{tt'}(R_{tt'}) = \sum_{i \in \mathcal{I}} \hat{V}_{t,it'}(R_{t,it'}) \tag{54}$$

are generally fairly easy to work with. We find ourselves having to solve equation (43) directly. Assume that we are interested in integer solutions, which leads us to use a piecewise linear form for $\hat{V}$, as given in equation (17). This would produce a network such as the one illustrated in figure 4. This problem is easily solved as a linear network, and it naturally returns integer solutions.

For this problem class, nonlinear functions appear to work extremely well. They are easy to estimate using the techniques of section 3.1; they are computationally quite easy to solve (sequences of pure networks) and produce high quality solutions. Table 2 compares the technique when applied to deterministic networks (something you would not want to do in

	Planning Horizon		
Locations	**15**	**30**	**60**
20	100.00%	100.00%	100.00%
40	100.00%	99.99%	100.00%
80	99.99%	100.00%	99.99%

TABLE 2. Percentage of integer optimal value obtained using CAVE for second set of deterministic experiments with single-period time windows (network problems), from Godfrey and Powell, 2001a.

		Percentage of Posterior Bound	
Number of Locations	**Number of Resources**	**Rolling horizon**	**Stochastic using CAVE**
20	100	92.2%	96.3%
20	200	96.3%	97.8%
20	400	96.6%	98.1%
40	100	81.0%	90.5%
40	200	90.7%	96.2%
40	400	92.6%	96.8%
80	100	66.3%	82.1%
80	200	81.4%	93.3%
80	400	84.8%	94.5%

TABLE 3. Comparison of nonlinear approximation using CAVE to a deterministic rolling horizon procedure, for stochastic problems with different numbers of locations and resources. Posterior bound is computed by finding optimal solution assuming all information is known (from Godfrey and Powell, 2001a).

practice, since specialized algorithms are extremely good), indicating near optimal performance. When compared against a rolling horizon procedure, we get the results shown in table 3. This table provides results as a function of the number of locations, and of the number of resources (holding the number of tasks fixed). Problems with a larger number of locations are harder to solve, in part because the problem becomes increasingly nonseparable. The number of resources is important since the problem becomes more difficult as the number of resources is decreased. The results indicate that a nonlinear value function approximation can significantly outperform a deterministic approximation based on rolling horizon simulations.

Of particular value is going to be our ability to take this general strategy and apply it to increasingly more general problems. We first illustrate its application to multicommodity

problems, followed by heterogeneous resource allocation problems. We then indicate how it can be applied to two-layer problems.

4.1.2. *Multicommodity.* Multicommodity flow problems arise whenever we have different types of resources and different types of tasks, and we are allowed to substitute the use of different resources, but where the cost of serving a task depends on the type of resource. This might arise when we are managing fleets of trailers, and there are different types of trailers with some substitution. It arises when managing fleets of boxcars and containers, as well as distributing different product types to consumers.

Multicommodity flow problems arise when the attribute of a resource can be described as $a = (k, i)$ where k represents a commodity class (or simply a commodity) while i remains our state variable. In any transformation:

$$M(t, a, d) \quad \rightarrow \quad (a', c, \tau)$$

we assume that if $a = (k, i)$ then $a' = (k, i')$. We let:

$\mathcal{K}$ = Set of commodity classes.
R_{it}^k = The number of resources of type k in state i.
x_{tid}^k = The number of times we act on a resource of type k in state i with decision d.

We note that we are following standard notational conventions of putting the commodity class as a superscript. This runs against the notational style that we have been following in this chapter, where all indices are expressed as subscripts. We violate our own notational conventions for reasons of consistency with the research literature. The reader is encouraged to contrast this presentation with our discussion of a more complex problem, heterogeneous resources, where our notation is actually simpler.

We can set up and solve the multicommodity version of the problem just as we did with the single commodity. Rolling horizon procedures are stated simply as:

$$\max \sum_{t' \in \mathcal{T}_t^{ph}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}} c_{t'id}^k x_{t'id}^k \tag{55}$$

subject to, for $t' \in \mathcal{T}_t^{ph}$:

$$\sum_{d \in \mathcal{D}_i} x_{t'id}^k = R_{t'i}^k + \hat{R}_{t'i}^k \ \forall j \in \mathcal{I}, k \in \mathcal{K} \tag{56}$$

$$\sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}_i} x_{id,t'-\tau_{tid}}^k \delta_{jt'}(t' - \tau_{tid}, i, d) = R_{jt'}^k \ \forall j \in \mathcal{I}, k \in \mathcal{K} \tag{57}$$

$$\sum_{k \in \mathcal{K}} x_{t'id}^k \leq u_{t'id} \tag{58}$$

$$x_{t'id}^k \geq 0 \tag{59}$$

The costs c_{tid}^k may incorporate the cost of assigning a resource of type k to a particular type of task, if $d \in \mathcal{D}^s$. We could, for example, divide the set $\mathcal{D}^s$ (representing decisions to serve a demand) into subsets $\mathcal{D}_k^s$.

The complicating constraint in this formulation is equation (58). If our problem is not too large, and we are not interested in integer solutions (or, we are willing to find a near-optimal solution), then commercial LP solvers should work fine here. More problematic is that we are again making the assumption that we know the future perfectly. Also, a multiperiod multicommodity flow problem can be relatively hard to solve.

We may incorporate uncertainty in our forecasts by using the same types of dynamic programming approximations described for single commodity formulations. Without repeating the algebra, it is not hard to show that the multicommodity version of equation (50) is:

$$\widetilde{V}_t\left(R_t, \omega_t\right) = \max_{x \in \mathcal{X}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} \sum_{d \in \mathcal{D}} \left(c_{tid}^k + \hat{v}_{t+\tau_{tid}, i_{tid}^M}^k\right) x_{tid}^k \tag{60}$$

The slopes $\hat{v}^k$ are updated using the sample gradients of the resource constraint (56) when solving subproblem t.

We earlier showed that the use of linear approximations for single commodity problems produced subproblems that involved nothing more than simple sorts. Multicommodity problems are a bit more complex. We still require that a resource be in state i to be acted on by a decision in $\mathcal{D}_i$, but we now have the behavior that different types of resources in state i can be acted on by decisions in $\mathcal{D}_i$. The problem reduces to a network which we illustrate in figure 5. Note that when we use linear approximations, we can take the slopes of the value function approximations and simply add these to the costs on the coupling arcs, along with any cost that might exist on the decoupling arcs. The resulting problem is a pure network.

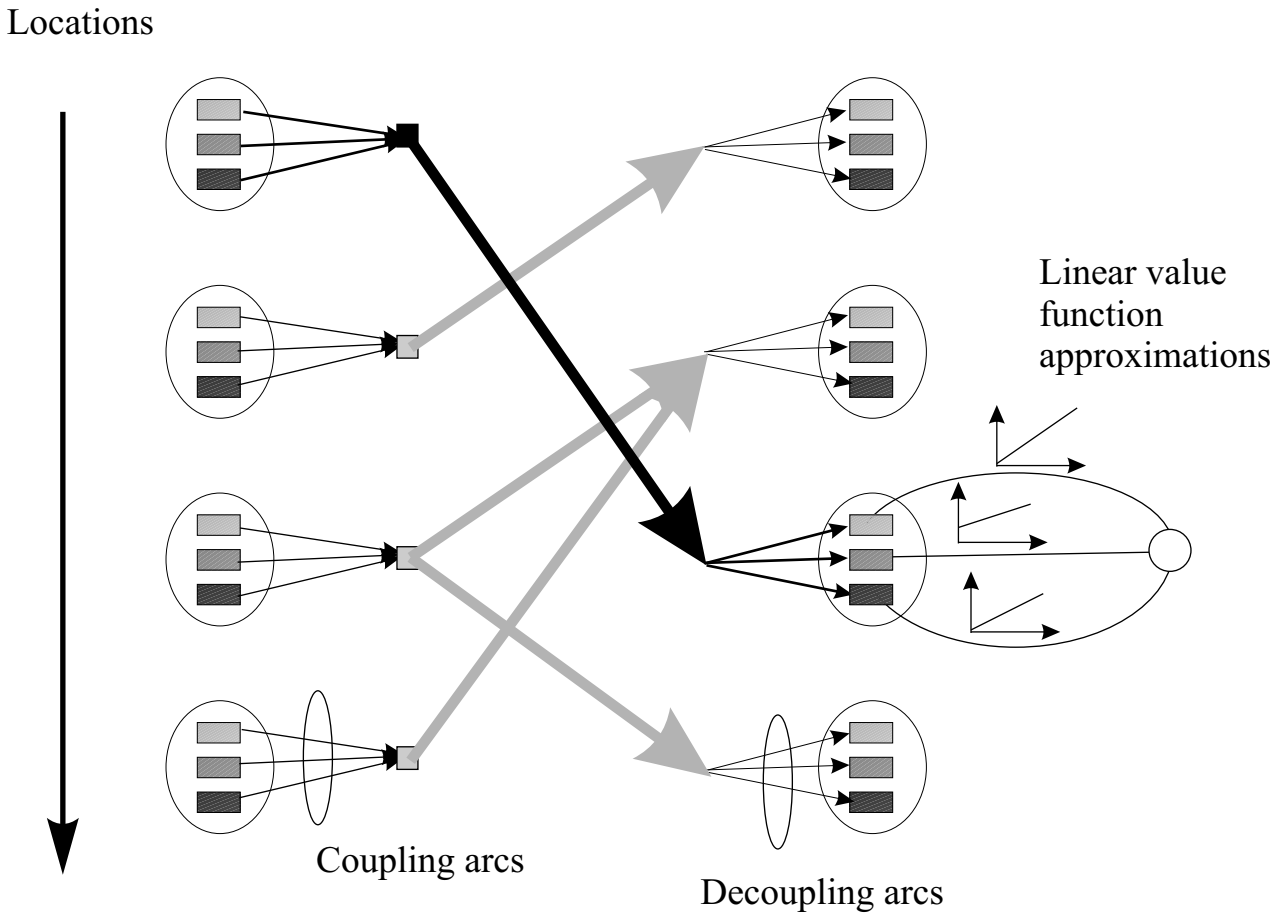


FIGURE 5. Network problem produced by multicommodity flow problems with linear value function approximations

This remains quite easy to solve, but suffers from all the problems we described earlier with linear approximations. Furthermore, the use of upper bounds to control the flows (especially repositioning decisions) becomes much trickier. It is important to keep in mind that the artificial upper bounds y_t are deterministic, and must work reasonably well under different sample realizations. The problem with these variables for multicommodity problems is that they do not handle very well the opportunities for substitution across resources. It might be preferable, for example, to have an upper bound that cuts across commodities, but then we destroy our nice network structure.

We can, instead, use separable nonlinear approximations just as we did with single commodity problems. This would involve solving subproblems of the form:

$$\widetilde{V}_t\left(R_t,\omega_t\right) = \max_{x\in\mathcal{X}} \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{d\in\mathcal{D}} c_{tid}^k x_{tid}^k + \sum_{t'>t}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{I}} \hat{V}_{t+1,jt'}^k(R_{t+1,jt'}^k + \bar{R}_{t+1,jt'}^k(x_t,\omega_t))$$

(61)

This problem is illustrated using figure 3. Unlike single commodity problems, however, this subproblem is a bit more complicated. Whereas nonlinear value functions produce nice network subproblems in the single commodity case, the use of nonlinear value functions gives us (possibly integer) multicommodity network flow problems. To see why we get multicommodity flow problems, we do not have to look any further than the constraints on the decisions in equation (58). These constraints bundle flows of different types of commodities. So why didn't this cause a problem when we used linear approximations? The reason was that the linear function approximation allowed us to write $R_{t+1,jt'}^k(x_t,\omega_t)$ in terms of x_{tid} directly and then use the separability of the linear approximation. Nonlinear approximations mean that the function is no longer separable in x_{tid}, which destroys our structure.

The good news is that the multicommodity flow problems we have to solve are not very large (that is, a single time period), and if we are interested in integer solutions, the LP relaxation almost always gives us integer solutions anyway. This is where our dynamic formulation is much easier than solving the rolling horizon formulation in equations (55)-(58). One-time period problems are much easier to solve than time-staged problems over even modest planning horizons.

These techniques work quite well on both deterministic and stochastic multicommodity flow problems. As with single commodity problems, we can obtain integer solutions as long as we use piecewise linear value function approximations. Table 4 demonstrates the effectiveness of the techniques on both deterministic problems (compared against the results of an LP solver) and stochastic problems (compared against deterministic rolling horizon approximations). Again, we see that the techniques provide near optimal solutions on deterministic problems, and results that significantly outperform rolling horizon models.

Percent of posterior optimal solution			
Problem	Linear	Nonlinear	Rolling horizon
Results of stochastic runs with varying number of locations			
10 locations	86.14	96.96	93.17
20 locations	78.65	93.28	86.84
40 locations	74.13	92.21	86.89
Results of stochastic runs with varying compatibility patterns			
Sub. matrix I	78.65	93.28	86.84
Sub. matrix II	80.59	95.40	90.87
Sub. matrix III	74.83	91.51	82.66
Sub. matrix IV	84.23	97.12	93.74
Results of stochastic runs with varying numbers of resources			
100 res.	74.19	84.87	76.81
200 res.	78.65	93.28	86.84
400 res.	84.41	96.51	91.67

TABLE 4. Performance of linear and nonlinear value function approximations against a deterministic rolling horizon procedure, from Topaloglu and Powell, 2000

There are other tricks and techniques associated with the use of value function approximations for multicommodity flow problems. The interested reader is referred to Topaloglu & Powell (2000).

4.1.3. *Heterogeneous resources.* Heterogeneous resource allocation problems arise when the resources are relatively complex. These almost always arise when the resources are people, and they often arise when the resources are relatively complex pieces of equipment such as locomotives or airplanes. For example, in a driver management problem, the attribute of a resource might be:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} \text{Driver's home domicile} \\ \text{1 if driver represents a sleeper team. 0 otherwise.} \\ \text{The current/next terminal of driver } r \\ \text{The arrival time of driver } r \text{ at his current/next terminal.} \\ \text{The cumulative driving time of the driver.} \\ \text{The number of days away from home.} \end{pmatrix}$$

When routing and scheduling individual drivers, the attribute vector can become much more complex than this. These problems, however, are typically solved under assumptions of complete information (deterministic models), and are required to produce full schedules for individual drivers.

The management of locomotives might require the following vector of attributes:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = \begin{pmatrix} \text{Number of axles} \\ \text{H if it is a "high adhesion" locomotive, L otherwise.} \\ \text{The horsepower class of the locomotive.} \\ \text{The tractive effort rating of the locomotive.} \\ \text{Days remaining until the next required maintenance check.} \\ \text{The location where the locomotive should be maintained.} \\ \text{The identity of the train the locomotive came in on} \end{pmatrix}$$

These attribute vectors give a hint of the complexity that can arise when solving real resource allocation problems.

When the attribute vector is more complicated than a simple class and state, we refer to the problem as the *heterogeneous resource allocation problem*. These problems can be placed in the context of multicommodity flow problems by using the following observation. Let $a = (a^s, a^d)$ where a^s represent static elements of the attribute vector (elements which do not change when a decision is made) and where a^d captures the dynamic elements. In our driver example, $a^s = (a_1, a_2)$ while $a^d = (a_3, a_4, a_5, a_6)$. The static elements can be concatenated and viewed as a single resource class (or commodity) while the last four can be concatenated and viewed as a state variable. However, these problems do not satisfy the structure of multicommodity flow problems where the upper bound u_{tid} is keyed to the state of the resource.

Aside from this structural difference, the real difference between multicommodity flow problems and heterogeneous resource allocation problems is the size of the attribute space. In multicommodity problems, $a = (k, i)$, so the number of possible attributes is probably close to $|\mathcal{K}| \times |\mathcal{I}|$. If we are managing intermodal containers, we might find $|\mathcal{K}|$ is between 10 and 50, whereas the number of locations, given by $|\mathcal{I}|$, might be between 100 and 1,000. This means that the total size of the attribute space might be as large as 50,000, but is typically about 5,000. By contrast, a multidimensional attribute vector can easily have millions of possible combinations. When this is the case, the number of attribute vectors that actually occur are typically much smaller, but we do not know in advance which ones will be used.

A deterministic formulation of the heterogeneous resource allocation problem is given by:

$$\max \sum_{t' \in \mathcal{T}_t^{ph}} \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{t'ad} x_{t'ad} \tag{62}$$

subject to, for $t' \in \mathcal{T}_t^{ph}$:

$$\sum_{d \in \mathcal{D}_a} x_{t'ad} = R_{t'a} + \hat{R}_{t'a} \; \forall a \in \mathcal{A} \tag{63}$$

$$\sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}_a} x_{t'-\tau_{tad},ad} \delta_{t'a'}(t' - \tau_{tad}, a, d) = R_{t'a'} \; \forall a' \in \mathcal{A} \tag{64}$$

$$\sum_{a \in \mathcal{A}} x_{t'ad} \leq u_{t'd} \tag{65}$$

$$x_{t'ad} \geq 0 \tag{66}$$

where we adopt the convention that $x_{t''ad} = 0$ if $t'' < 0$. This is a hard problem as a result of its sheer size. For practical problems, it is virtually impossible to generate the complete attribute space even for a single time period, not to mention over all the time periods in a reasonable planning horizon.

Interestingly, this appears to be one of those problems which seems to be easier if we use stochastic techniques. So far, we have seen that stochastic techniques can work quite well on deterministic problems. Applying the same techniques we used previously, we find that our one-period problem becomes:

$$\widetilde{V}_t(R_t, \omega_t) = \max_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}_a} c_{tad} x_{tad} + \sum_{t' > t} \sum_{a' \in \mathcal{A}} \hat{V}_{t+1,a't'}(\bar{R}_{t+1,a't'} + R_{t+1,a't'}(x_t, \omega_t)) \tag{67}$$

subject to equations (63), (66), adapted to a single time period.

We can use either linear or nonlinear value function approximations, and we end up with the same basic subproblem structures as we did with multicommodity problems. For example, linear approximations reduce to networks such as that illustrated in figure 5, whereas nonlinear approximations produce subproblems that look like figure 3. The big difference arises because of the size of the attribute space. When we are solving multicommodity problems, it is normally the case that we would enumerate all possible values of $\mathcal{K} \times \mathcal{I}$ in advance. This means that we would have a resource constraint for every combination of k and i. This means that we will get a dual variable for every possible combination, and we will create a value function approximation for every possible combination, which gets updated at each iteration.

With the heterogeneous case, we cannot generate every element in $\mathcal{A}$. Instead, we have to generate attributes dynamically. Let:

$\mathcal{A}^n$ = The *active attribute space* that has been generated at iteration n.

We propose to use an increasing sequence $\mathcal{A}^n \subseteq \mathcal{A}^{n+1}$. This implies, however, that for a given attribute a and decision d the attribute $a^M_{t,a,d}$ may not have been generated yet. We need an approximation of $\hat{V}_{t'a'}$ for attributes $a' \notin \mathcal{A}^n$. For this purpose, we define:

$\mathcal{A}_i$ = The set of attribute vectors that have a common geographical location $i \in \mathcal{I}$.

$\bar{a}_i$ = The attributes of an artificial resource in location $i \in \mathcal{I}$ that will have the best possible behavior in that location.

We want $\bar{a}_i$ to have the behavior of a resource that is at least as good as any real resource. So, we assume that:

$$c(\bar{a}_i, d) \geq \max_{a \in \mathcal{A}_i, d \in \mathcal{D}_a} c(a, d) \tag{68}$$

We refer to the attribute vector $\bar{a}_i$ as the "best attribute" for location i. We could, of course, simply define a single "best attribute" that would apply system wide, but it seemed clear to us that we could get much better results if we tightened our bound by making it location specific. Note that it is not necessary that $\bar{a}_i \in \mathcal{A}$; if $\bar{a}_i \notin \mathcal{A}$, then $\bar{a}_i$ would simply be an empty resource bucket. What is important is that we have one attribute which is always present, which allows us to create a value function approximation which is an upper bound. This ensures that we will not artificially avoid a decision just because we underestimate the downstream value of the resource created by the decision.

To create an attribute vector where we ensure that the contribution out of that attribute represents an upper bound over other attributes does not seem to necessarily ensure that the value of the attribute (which includes not only the immediate cost but also the downstream costs) is also an upper bound. The following proposition establishes this result:

Proposition 4.1. *(Powell et al. (2000b)) Assume that equation (68) holds and that* $\bar{v}^0_{\bar{a}_i t} \geq \bar{v}^0_{ta}$ *for* $a \in \mathcal{A}_i$. *Then:*

$$\bar{v}^n_{\bar{a}_i t} \geq \bar{v}^n_{ta} \ \forall a \in \mathcal{A}_i$$

In other words, we can ensure that our estimate of the value of our "best attribute" is going to be better than the value of resources with other attributes (at the same location).

This does not mean that any of these estimates are actually upper bounds over what the values should be. But, decisions are relative, so this is an important property.

The active attribute space, then, grows as the algorithm visits new states. We can describe the process using:

$$\begin{aligned}
\mathcal{A}_t^n &= \text{Set of attribute vectors that have been generated for time period } t \text{ in iteration } n.\\
\tilde{\mathcal{A}}_{tt'}^n &= \text{Set of attribute vectors that are generated for time period } t' \text{ when solving the subproblem for time period } t.\\
&= \{a' | x_{tad} = 1, \delta_{a't'}(t, a, d) = 1\}
\end{aligned}$$

Of course, the set $\tilde{\mathcal{A}}_{tt'}^n$ may include elements that are already in $\mathcal{A}_{t'}^n$. Our active attribute space is updated using:

$$\mathcal{A}_{t'}^{n+1} = \mathcal{A}_{t'}^n \bigcup_{t<t'} \tilde{\mathcal{A}}_{tt'}^n \tag{69}$$

This algorithm has been applied to the management of drivers for a major LTL trucking company. It scales easily to handle problems involving the management of thousands of drivers moving tens of thousands of loads between hundreds of different locations.

4.2. **Two layer resource allocation.** In the previous section, market demands, tasks, requirements, or other expressions of serving an exogenous customer were all modeled as upper bounds which limited our ability to make money or otherwise generate positive contributions. These upper bounds were expressed in the form u_{tid} for $d \in \mathcal{D}^s$, representing a limit on our ability to execute a decision to serve a task at time t. Generally, for a decision $d \in \mathcal{D}^s$ to serve a demand, we normally assume that $c_{tid} > 0$, whereas decisions to reposition a resource to another state, given by $d \in \mathcal{D}^r$, would incur a negative contribution. Implicit in this model is the assumption that if we do not serve a task at time t, then the positive contribution is lost. At no time do we ever make any decisions about the task itself.

Two-layer problems arise frequently because as a rule, we often have to make decisions about how a demand is satisfied. In the simplest case, we may have to decide whether to serve a task now or later. This is the basic case of demand backlogging. In truckload trucking, it is often the case that once we decide to serve a customer, we simply move the load from origin

to destination. We only have to decide when to serve the load. In more complex settings, we may have to decide how to serve the demand, which may have to progress through a series of steps before being completed.

When we make the transition to problems with two or more layers, we need to start distinguishing between important classes of resource layers. The first is whether they are *persistent* or *transient*. A persistent resource stays in the system when the decision is made to hold the resource. A transient resource vanishes. A *reusable* resource stays in the system after it is acted on; if it vanishes, it is nonreusable or *perishable* (the term "perishable" is awkward in the context of transportation and logistics, and appears to be better suited for consumer products).

A second critical dimension is whether the resource is *active* or *passive*. An active layer can be modified using a set of decisions. A passive layer can only work by coupling with other resource classes. A persistent, passive layer at a minimum has the property of (possibly) staying in the system when the action is to "do nothing" but a more interesting class is one that stays in the system even after it has been coupled and modified.

An example of these concepts arises in the case of a driver and a load. A driver can reposition from one location to another without a load, or it can move a load. Moving a load allows the driver to make money, but you cannot act on the load by itself. But if you do not move the load, it just sits there (although it may leave the system). If you do move the load, it vanishes from the system. The load is a persistent, but nonreusable class. The load becomes a reusable class if the driver moves the load to a relay point, drops off the load, and waits for another driver to pick it up.

We can turn this same example into a problem with two active layers. Assume that when we move a driver we mean that we are moving a driver that is an employee of our company. If we run short on drivers, we can contract out to another company to move the load, after which the driver becomes the responsibility of the outside company. From the perspective of the resources that we manage (our own drivers) it is as if we can move the load without a driver. This would be a problem with two active layers.

We start in section 4.2.1 with the simplest version of a two-layer problem where only one layer is reusable (we can actually make decisions that change its state) while the other is passive (demands that just sit there until they are served). Section 4.2.2 then describes problems where the second layer is reusable as well.

4.2.1. *One reusable layer.* Earlier, we introduced the notation that $\mathcal{R}^c$ represented the set of (discrete) resources in class c (or, R_c is the vector of resources in class c). This notation is especially useful when there are three or more classes of resources (some complex problems might have four or five classes of resources), since it saves us from creating an alphabet soup of variables to describe the different resource classes. But when there are only two classes, it is more convenient to use different variables for each layer. For this purpose, we let:

L_{tb} = The number of tasks with attribute vector b available at time t before any new arrivals have been added.
$\mathcal{B}$ = The space of possible task attributes, with element $b \in \mathcal{B}$.
L_t = The vector of tasks that we know about at time t before any new arrivals have been added.

This representation provides a certain symmetry with the representation of resources. However, it is also useful to define:

$\mathcal{L}$ = The set of task types (for example, each task type might represent an origin/destination combination.
L_{tl} = The number of tasks of type $l \in \mathcal{L}$.

$\mathcal{L}$ can be viewed as an indexing of the task attribute space $\mathcal{B}$. For our purposes, the latter representation is more convenient.

If we want to make a decision to serve a task, we let $\mathcal{D}^s = \mathcal{L}$ represent our set of possible types of tasks. L_{tl} is the number of tasks of type l at time t (some readers will prefer to use the variable u_{tl} to be the number of tasks of type l, since this variable later serves as an upper bound). For each task type in $\mathcal{L}$, there is a corresponding decision in $\mathcal{D}^s$ to serve a task of that type. For a decision $d \in \mathcal{D}^s$ there is a task type $l_d \in \mathcal{L}$, which means we can write $L_{l_dt} = L_{dt}$ for $d \in \mathcal{D}^s$. The number of resources we can assign to a task, then, is

limited by:

$$x_{tad} \leq L_{dt} \tag{70}$$

while resources limit us through the flow conservation constraint:

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{ta} \tag{71}$$

Equations (70) and (71) express the impact of resource and tasks on decisions. When we implement decision d, the impact on the resource is expressed through the modify function, while the impact on the task is that it leaves the system. The evolution of the resource state variable is given by equation (3). We can in principle use the same equation for the tasks, but the simplicity of our tasks encourages us to use simpler notation. We assume that if we act on a demand that it leaves the system. A demand that is not acted on may also leave the system (a customer refusal). For this reason, we define:

$L^h_{tt'}$ = The number of tasks that we knew about at time t which were actionable at time t' which were held at time t. Tasks which are not held include those that were served or which independently left the system.

We would normally assume that $L^h_{t+1,t'} = L^h_{tt'}$ for $t' > t$, meaning that if a task is not actionable at time t, it should still be in the system at time t'. But our representation allows for order cancellations. This notation allows us to write the task dynamics as:

$$L_{t+1,t'} = L_{tt'} + \hat{L}_{tt'} + L^h_{tt'} \tag{72}$$

where $\hat{L}_{tt'}$, just as with $\hat{R}_{tt'}$, represents the tasks that first become known at time t and which are actionable at time t'.

Our resource state of the system is now given by the pair (R_t, L_t). We emphasize that this is our *incomplete* resource vector, since R_t and L_t do not include new resources and tasks that arrive in the system at time t. We can use the same dynamic programming recursion and approximations that we used with a single resource layer earlier by simply replacing R_t with (R_t, L_t). Using our dynamic programming approximations, we would have to solve:

$$\widetilde{V}_t(R_t, L_t, \omega_t) = max_{x \in \mathcal{X}} C_t(x, R_t, L_t) + \hat{V}_{t+1}(R_{t+1}(\omega_t), L_{t+1}(\omega_t)) \tag{73}$$

which we would solve subject to constraints (70) and (71), as well as the system dynamics (3) (the updating of the number of resources) and (72) (the updating of the number of tasks).

Number of Locations	With time windows			Without time windows		
	Horizon Length			Horizon Length		
	15	30	60	15	30	60
20	99.0%	99.2%	99.5%	100.00%	100.00%	100.00%
40	98.2%	98.4%	98.9%	100.00%	99.99%	100.00%
80	97.5%	97.0%	97.6%	99.99%	100.00%	99.99%

TABLE 5. Performance of nonlinear approximation on problems where tasks have non-zero time windows (two-layer problem) and tight time windows (true one-layer problem).

To solve (73) we can resort to the tricks we used for the one-layer problem. Assume, for example, that we want to work with a linear approximation. We would simply write:

$$\widetilde{V}_t\left(R_t, L_t, \omega_t\right) = max_{x\in\mathcal{X}} C_t(x, R_t, L_t) + \hat{v}^R_{t+1} R_{t+1}(\omega_t) + \hat{v}^L_{t+1} L_{t+1}(\omega_t) \tag{74}$$

We would estimate $\hat{v}^R$ and $\hat{v}^L$ by using the dual variables on the constraints (71) and (70), and applying our standard smoothing techniques.

When the tasks are a passive layer, it is not unreasonable to use the approximation $\hat{v}^L = 0$. This means that we try to cover a task at time t, but if we cannot, we simply hold it until time $t + 1$ and hope to cover it then. Table 5 shows the results of experiments using a nonlinear approximation on a resource allocation problem where tasks have time windows (but where we use $\hat{v}^L = 0$) and problems where tasks must be served at a point in time, where both experiments are run on datasets without any uncertainty (which allows us to get tight bounds using an LP solver). The results indicate (on these deterministic datasets) that we are obtaining virtually optimal solutions when the time windows are tight (a true one-layer problem) whereas we are one or two percent below optimal when we use $\hat{v}^L = 0$.

To test the value of using both resource and task gradients we need to work on a problem where both resource and tasks may be held before being assigned, where we can also readily obtain optimal solutions, at least in the form of posterior bounds. (Posterior bounds are computed by finding the optimal solution after all the information becomes known). A problem that readily lends itself to this test is the *dynamic assignment problem*. The dynamic assignment problem involves the assignment of resources and tasks over time, but where once a resource is assigned to a task, they both vanish from the system. But, if a resource or task is not assigned in time period t, they are available in time period $t+1$. The decision to

Type of experiment	Myopic	Resource Gradients	Resource and Task Gradients
Deterministic	88.4	93.4	97.5
Stochastic	86.6	89.2	92.8

TABLE 6. Results of value function approximations for deterministic and stochastic experiments expressed as a percent of the posterior optimal solution. Each statistic is an average over 20 datasets.

assign a resource or a task now has to take into account the value of the resource or task in the future.

The dynamic assignment problem is a special version of a two-layer problem, where we arbitrarily designate the "resources" as the active resource layer, while tasks are the passive layer. An important application is the load matching problem of truckload trucking, where we have to assign drivers to loads. Over time, drivers become available and loads are called in. After a driver is assigned to a load, both "vanish" from the system.

What makes the dynamic assignment problem special is that it is easy to solve the problem after all the resources and tasks become known to get a tight upper bound. In contrast with our earlier resource allocation problem, this is a problem where a myopic solution is not only interesting, it is what is normally done in practice. Experiments were run on 20 deterministic and 20 stochastic datasets, comparing a myopic solution ($\hat{v}^R = \hat{v}^L = 0$), against algorithms with just resource gradients ($\hat{v}^L = 0$) and algorithms using both resource and task gradients. The results are shown in table 6, which suggest about a three to four percent improvement by adding in task gradients. We would conclude that while the improvement is not dramatic, it is certainly significant.

4.2.2. *Two reusable layers.* Consider the problem of moving a boxcar loaded with freight from origin to destination to serve the customer. This process occurs in a series of steps. When the boxcar is pulled from the shipper's dock, it is pulled to a yard where it is added to a *block* which represents a set of cars that will move together over one or more trains. When a train moves, it pulls a set of blocks which share a common segment (a common intermediate destination). When the block reaches its destination, it is probably the case

that some of the cars have also reached their destination, but others may have to continue on. These cars are pulled out and added to a new block, which again will move over one or more trains before again reaching a new, intermediate destination.

Trains, of course, are pulled by locomotives. Thus, to move a set of cars from one location to another, it is necessary to couple the cars together, move them to an intermediate destination (the destination of the block) and then uncouple them. Both the locomotive and the boxcars stay in the system. The locomotives have to be allocated to new trains, and decisions have to be made about how to route the boxcars. Thus, both are active resources.

Two layer problems arise in other settings. Truckload motor carriers have to manage drivers and loaded trailers. Once a driver picks up a load, it may be necessary to move the load to a terminal where it is stored for a few days waiting for its final delivery appointment. The driver will be assigned to a new load, and at a later point a new driver will come in to pick up the original load. As with the boxcars, at the end of the first move, both the driver and the load remain in the system.

A third-party logistics provider also faces two-layer problems if they have the responsibility for moving and storing product, as well as managing the driver. It is necessary to load up the driver's vehicle, move the product, store the product, and continue to manage the driver.

The modeling of a two-layer problem is virtually equivalent to the modeling of a two-layer problem with a single reusable layer (but where both layers are persistent). All we need to do is change the system dynamics so that both layers are handled in the same way. So, instead of using the simple task dynamics of equation (72), we would model tasks using (3). Basically, the reader has to understand that when we face a true two-layer problem, whatever we do for the so-called "resource layer" is the same as what we have to do for the "task layer." We have to capture not only the value of the resource in the future, but also the value of the task.

4.3. **Multiple layers.** Real problems are invariably even more complicated than the problems that we have addressed. For example, a driver has to use a tractor to pull a trailer to pick up a load of freight. Furthermore, we may need pallets or special loading equipment to help handle the load. A locomotive needs both fuel and a crew to pull boxcars with freight.

A chemical products company has to specifically manage the driver, tractor, trailer, chemical product, and the customer tank (a five layer problem).

Multilayer problems are inherently complex, so it is especially important to adopt elegant, compact notation. Our earlier modeling framework introduced the concept of resource classes $\mathcal{C}^R$, where R_t^c is the vector of resources in class c, and $\mathcal{A}^c$ is the attribute space for class c. Resource layering helps us handle the problem when decisions have to be made for resources that are coupled together. For example, a locomotive attached to an inbound train is quite different than the same locomotive that is not attached to an inbound train. The deliveries that can be made by a particular driver, tractor and trailer depends on the characteristics of all three.

One challenge faced by multilayer problems is that of solving a single period subproblem. Two-layer problems have the fundamental structure of transportation problems and assignment problems. Three-layer problems are much harder to solve.

It is perhaps not surprising that multilayer problems are often (but not always) solved as sequences of two-layer problems. In an LTL carrier, one person will manage drivers, another will plan the loading of trailers, and a third makes sure that the tractor pools are adequate. In railroads, locomotives, boxcars, and crews are all managed by distinctly different groups. But, a truck dispatcher has to manage both drivers (with their tractors), trailers and loads.

4.4. **Bundling.** Up to now, we have assigned resources to tasks with the tacit assumption of one resource per task. These are called "one to one" problems, and arise when we have to assign a driver to pull a load, or the assignment of a boxcar to a customer order. But it is often necessary to consolidate freight into a single container. In this section, we consider two special cases. The first involves the batching of dozens, or even hundreds, of shipments on a single trailer going between a pair of points. In the second, we address the problem of clustering tasks with different characteristics. This might arise when putting orders together with different delivery dates, or with different final destinations (otherwise known as the vehicle routing problem).

4.4.1. *Batch dispatching.* The simplest batch dispatching problem arises in LTL trucking where shipments accumulate at a terminal until there is enough to satisfy the criteria for

sending the truck. In most problems, the arrival rate of shipments is not constant over time. For example, at an end of line terminal, arrivals occur primarily in the evening as shipments are unloaded from trucks that were in the city during the day. Most of the time, the dispatch rule is pretty simple. It is either "dispatch when full" or a variation such as "dispatch when full, but no later than a cutoff time," where the cutoff time ensures that the carrier can make service. The challenge always arises when there is no more freight and the truck is only partially full. While all carriers focus on service, any carrier will have difficulty sending a truck over a long distance when it is only 20 percent full.

For regional carriers, there are typically very few options for routing freight. Some trucks will go directly from one city to another, carrying only freight between those two cities. A few carriers work exclusively this way, but this operating concept is impossible to grow past one or two dozen terminals. As a rule, most freight has to be handled through a single distribution facility. If a truck is not full enough to send through the facility, either the carrier has to send the truck partially loaded, or hold the freight until the next cycle.

Long haul carriers have more options. A trailer at an origin end of line such as Boston may be loading shipments to carry to a distribution center (or breakbulk) at a destination region such as Texas. If there is not enough freight to fill the trailer, the carrier has the option of filling the trailer with freight and moving the trailer (either full or partial) but only to the nearest distribution center (sometimes called the "origin breakbulk") that would be in the northeast. There, the freight may be completely or partially sorted onto trailers that leave to many other terminals.

Efforts have been made to formulate the problem of determining where to send trucks as integer programming models. Even static models of regional carriers can be intractably large, and optimal algorithms have not proven effective. Local search heuristics for optimizing static networks have been effective, and in particular, local search heuristics that work interactively with a planner have been widely adopted. But, even heuristic optimization models for the dynamic case have not been effective. Simulation models which use simple policies to determine when a truck should be dispatched remain the only effective tool in engineering practice, and we are not aware of any serious progress toward optimizing dynamic problems.

In this section, we again focus on dynamic problems and illustrate how the techniques that we have presented earlier for resource allocation problems can again be effective in this setting. As before, our solution approach will be one which solves a sequence of relatively simple problems by stepping through time. We can use either a simple myopic rule, or apply our adaptive dynamic programming techniques (see Papadaki & Powell (2001)). We consider only the case of dispatching trucks over a single link, but we allow ourselves to consider the case where there are different types of customers. This is particularly important in LTL trucking, where there are high and low priority customers, as well as customers who have been waiting different lengths of time. Finally, we do not assume steady state behavior.

Model parameters

$\mathcal{K}$ = Set of customer classes.
c^d = Cost to dispatch a vehicle.
c_i^h = Holding cost of class i per time period per unit product.
$c^h = (c_1^h, c_2^h, ..., c_m^h)$
K = Service capacity of the vehicle, giving the total number of customers who can be served in a single dispatch.

Activity variables

R_{kt} = Number of customers in class k waiting at time t before new arrivals have been added.
$R_t = (R_{kt})_{k \in \mathcal{K}}$
$\hat{R}_t$ = Vector random variable giving the number of arrivals in time t of each type of customer.
$R_t^+ = R_t + \hat{R}_t$.

Decision variables

x_{tm} = The number of customers in class m who are served at time t.
$X_t^\pi(R_t^+)$ Decision function giving the vector x_t as a function of the complete resource vector R_t^+.

We define a family of decision functions $(X_t^\pi)_{\pi\in\Pi}$. It is useful for us to define an indicator variable $z_t = 1$ if a vehicle is dispatched and 0 otherwise. We let $Z_t(x_t)$ be a decision function where $Z_t = 1$ if $\sum_{k\in\mathcal{K}} x_{kt} > 0$, and 0 otherwise. Note that we are assuming that there is at most one dispatch per time period (since time periods can be made smaller, this does not pose a significant limitation).

Our one period cost function is given by:

$$C_t(R_t, \hat{R}_t, x_t) = c^d Z_t(x_t) + c^h(R_t^+ - x_t)$$

The objective function is now given by:

$$F(S_0) = \min_{\pi\in\Pi} E\left\{\sum_{t=0}^{T-1} C_t(R_t^+, X_t^\pi(R_t^+))\right\}$$

We follow our standard methodology and propose to solve the dynamic programming approximation:

$$\widetilde{V}_t(R_t, \omega_t) = \min_{x_t} C_t(R_t^+, x_t) + \hat{V}_{t+1}^n(R_t^+(\omega_t), x_t) \tag{75}$$

The simplest approximation, which is also surprisingly effective, is to use a linear approximation:

$$\hat{V}_t(R_t) = \hat{v}_t R_t \tag{76}$$

These batch processes are not linear programs, so we do not have access to dual variables. But, we can use finite differences. Let:

$$\tilde{v}_{kt} = \widetilde{V}_t(R_t + e_k, \omega) - \widetilde{V}_t(R_t, \omega)$$

where e_k is a $|\mathcal{K}|$-dimensional vector with a single 1 in the k^{th} element (when there are a lot of product classes, it is fairly easy to devise schemes to approximate $\tilde{v}_{kt}$ using derivatives for only a few product classes). As before $\tilde{v}_{kt}$ is a statistical estimate, and we perform smoothing to find the approximation $\hat{v}_{kt}$.

Solving equation (75) using a linear value function approximation is pretty easy. If we assume that $z_t = 1$ (meaning that we are going to dispatch the vehicle), then finding the optimal x_t is usually a simple sort. In fact, it is possible to show that the simple rule of putting the most valuable products in the truck is the best. This means that we really only have to calculate (75) for $z_t = (0, 1)$ and find the best value.

Step 1 Given R_0 : Set $\hat{V}_t^0 = 0$ for all t. Set $R_0^n = R_0$ for all n. Set $n = 1$, $t = 0$.
Step 2 Choose random sample $\omega = (\omega_0, \omega_1, ..., \omega_{T-1})$.
Step 3 Calculate

$$z_t^n = \arg \min_{z_t \in \{0,1\}} \left\{ cz_t + c^h \cdot (R_t^n + \hat{R}_t^n - z_t X(R_t^n + \hat{R}_t^n)) + (\hat{v}_t^n) \cdot (R_t^n + \hat{R}_t^n - z_t X(R_t^n + \hat{R}_t^n)) \right\}$$

and

$$R_{t+1}^n = R_t^n + \hat{R}_t^n - z_t X(R_t^n + \hat{R}_t^n)$$

Then define:

$$\tilde{V}_t^n (R_t^n) = \min_{z_t \in \{0,1\}} \left\{ c^d z_t + c^h \cdot (R_t^n + \hat{R}_t^n - z_t X(R_t^n + \hat{R}_t^n)) + (\hat{v}_t^n) \cdot (R_t^n + \hat{R}_t^n - z_t X(R_t^n + \hat{R}_t^n)) \right\}$$

Step 4 Update the approximation as follows. For each $k = 1, ..., m$, let:

$$\tilde{v}_{kt}^n = \tilde{V}_t^n (R_t^n + e_k) - \tilde{V}_t^n (R_t^n)$$

where e_k is an $|\mathcal{K}|$-dimensional vector with 1 in the k^{th} entry and the rest zero.

FIGURE 6. Adaptive dynamic programming algorithm for the batch dispatch problem

The steps of the algorithm are given in figure 6.

Table 7 summarizes the results of a series of experiments where the linear approximation was tested on a problem with a single customer class. For this special case, it is possible to solve the optimality recursion using standard backward dynamic programming techniques. The table shows the relative error over the optimal using between 25 and 200 iterations. Also shown is the performance of a myopic, go-when-filled policy where the vehicle is not allowed to be held more than time τ (where τ was optimized for each dataset). Three classes of datasets were tested, reflecting the difference between the holding cost c^h and the per-customer dispatch cost c^d/K. The results suggest that the heuristic provides near-optimal performance. Most significantly, there is no difficulty extending it to problems with many customer classes.

It is important to emphasize that since most companies dispatch trucks using myopic rules, a heavily engineered myopic policy can do a superb job of mimicking the real world. These policies will be more sophisticated than even an optimized go-when-filled strategy such as that illustrated in table 7. From a modeling perspective, the issue is not so much whether a dynamic programming approximation will outperform a myopic heuristic. Of much greater significance is whether the model will yield good, realistic results without the heavy engineering.

Method:	linear	linear	linear	linear	DWF-
Hold/dispatch	Number of iterations				TC
cost	(25)	(50)	(100)	(200)	
$c^h > c^d/K$	0.077	0.060	0.052	0.050	0.774
$c^h \simeq c^d/K$	0.048	0.033	0.023	0.024	0.232
$c^h < c^d/K$	0.030	0.022	0.017	0.016	0.063
Average	0.052	0.038	0.031	0.030	0.356

TABLE 7. Fractional error of total cost with respect to the optimal cost (from Papadaki and Powell (2000))

4.4.2. *Clustering.* The second type of batching involves the clustering of resources together. We might have to group several locomotives to pull one train, two or three "pups" to be pulled by one tractor, or the clustering of several deliveries onto one delivery vehicle. All of these problems involve a function that is nonseparable in the set of resources that are being bundled together. Locomotives may be attached together, so if we assign one locomotive to a train, we generally need to assign the other locomotives that the first locomotive may already be attached to. Pups need to be matched based on weight and service requirements. Deliveries should be grouped that form an efficient vehicle tour.

The general clustering problem can be expressed using a contribution function $C_t(x_t)$ which is a nonlinear, nonseparable function of the decision vector x_t. Fortunately, most problems are not quite this general. Let R_t be our vector of active resources (our trucks), and let $\mathcal{L}$ be the passive resource layer that we are coupling to (our deliveries, or tasks). We let R_{ta} be the number of resources with attribute a, and we let u_l be the size of each task, expressed in the same units as the resources (we may let R_{ta} be the capacity of the vehicles, and u_l be the size of each task). In many routing and scheduling problems, each individual vehicle $r \in \mathcal{R}$ will have its own unique attribute vector a_r, in which case R_{ta} would always refer to a single vehicle (but, this is not always the case). We let x_{alt} be the number of resources of type a that are being coupled to task l. Thus, we would have both a flow conservation

constraint on the resources:

$$\sum_{l\in\mathcal{L}} x_{tal} = R_{ta} \ \forall a \in \mathcal{A}, \tag{77}$$

and a coupling constraint:

$$\sum_{a\in\mathcal{A}} x_{tal} \leq u_{tl} \ \forall l \in \mathcal{L} \tag{78}$$

Let $x_{tl} = (x_{tal})_{a\in\mathcal{A}}$ be the vector of decisions describing the assignment of resources with attribute a to task $l \in \mathcal{L}$. Now let $c_{tl}(x_{tl})$ be the cost of assigning a vector of resources x_{tl} to task l. This allows us to write:

$$C_t(x_t) = \sum_{l\in\mathcal{L}_t} c_{tl}(x_{tl}) \tag{79}$$

It is, of course, a nice simplification when we can write $c_{tl}(x_{tl}) = \sum_{l\in\mathcal{L}} c_{tal}(x_{tal})$ which is to say, a separable function across resources. This might be the case when assigning several locomotives to a single train, or two or three pups to the same tractor. But, it is not going to be true when assigning multiple deliveries (or pickups) to the same vehicle, since the total cost depends on the tour that can be formed to complete the pickups. (Just the same, a separable approximation is the basis of a popular vehicle routing algorithm, Fisher & Jaikumar (1981), as well as research in routing, Bramel & Simchi-Levi (1995)). For this reason, researchers find that they have to dynamically find the actual tour, even if it will change as new information arrives (Gendreau et al. (1999), Regan et al. (1998)). These problems can be solved with any of a host of vehicle routing algorithms (Laporte (1992), Fisher (1995)). As of this writing, it is not known whether precise routing and scheduling outperforms a good approximation when demands are highly dynamic. For example, Powell et al. (2000*c*) show that for the load matching problem of truckload trucking (a form of dynamic assignment problem), a solution that uses a discounted approximation of the future (which is neither an optimal myopic solution, nor an attempt to optimize over the entire horizon) outperforms optimal myopic solutions in a dynamic setting.

Most efforts in the literature have focused on solving the problem myopically, which means forming vehicle tours using the vehicles and customer demands that are known at time t (see, for example, Gendreau et al. (1999) and Regan et al. (1998)). This means solving sequences of problems of the form $\min_x C_t(x_t)$ using a vehicle routing algorithm. A significant challenge

in this setting is the computational problem of solving a VRP under the pressure of time-staged demands, which limit the amount of time we have to actually solve the problem. We are not aware of efforts to solve VRP's using deterministic forecasts of future demands, which not only makes the problem much larger, but also creates other practical challenges (if the forecast is an expectation, we face the problem of routing an integer vehicle to pick up an expectation of the customer demand, which will typically not be either feasible or realistic).

We can approach the dynamic vehicle routing problem using the same strategies as we have reviewed for other resource allocation problems. We can solve the problem myopically, or incorporate value function approximations which are estimated through adaptive learning. There is very recent research into using neuro-dynamic programming methods (Secomandi (2000) and Secomandi (2001)), but this work considers only a single vehicle. A challenge in designing value function approximations for dynamic VRP's is both the large size of the attribute space, and the complexity of the true value function. It is not clear that the simple linear or separable, nonlinear functional approximations that we introduced earlier will be successful. Also, many problems have some degree of advance information. We do not know at what point a myopic model, using advance information outperforms an adaptive, dynamic programming model. All of these are open research questions.

Dynamic vehicle routing problems exhibit other characteristics unique to the problem class. Many (but not all) vehicle routing problems require forming complete tours where the driver terminates at the depot at the end of the day. In a dynamic setting, it is possible to require that any tour always terminate at the depot, or to form tours that are not complete (at time t), responding to new demands as they arise. Furthermore, it may be necessary to form tours that do not serve all the customers at time t. For example, it may make sense to hold off on picking up a request from a customer in a part of town where there are no other requests to be served (right now), in the hope that other calls will come from that part of town, allowing the vehicle to serve several customers at once.

Thus, the decision to form a tour at time t may leave some customers unserved (consider the dynamic assignment problem above), and may produce a tour that leaves a vehicle at a location other than his home depot at some point during the day (requiring us to complete

his tour at a later time). We may not serve all the customers at time t, also requiring us to think about the impact of forcing some deliveries until later in the day. We could use a myopic model that tries to cover only the deliveries we know about, using only the vehicles we know about. We would require this model to cover all deliveries with tours that always finish at the depot, or cover only some of the deliveries, with a tour that does not finish at the tour. The myopic model could be expanded by the use of simple rules, such as "do not deliver to a part of town unless there are at least three orders, or unless it is after 3pm." We could use a rolling horizon model by optimizing the problem using a mixture of known and forecasted demands. Or we could resort to our adaptive dynamic programming techniques. The last approach would require that we devise an effective approximation strategy, and a method for estimating and updating the function.

5. Implementation issues for operational models

It is easy to draw the conclusion from our presentation that the important issue in modeling freight transportation is designing models and algorithms that account for information that is not yet known. In practical implementations, the real issue tends to be in the form of bad data, which could otherwise be described as data that is not yet known, but should be. The problem is that we do not know in advance what data is bad, but we do know when we do not like a solution.

A byproduct of capturing the organization and flow of information is that it produces models where the original problem is broken into a number of pieces. Modeling the evolution of information over time produces models that are solved sequentially over time (rather than one big model over time). Modeling the organization of information and decisions produces a multiagent structure that further breaks the problem into subproblems. Not only are these subproblems much easier to solve, they are a lot easier to diagnose. That is, if the model recommends a decision that is not what a dispatcher would do, it is a lot easier to determine if the problem is data, model, algorithm or software.

We are not aware of any formal research into the modeling and algorithmic issues of dealing with bad data, but this is one of the characteristics of operational models that is markedly different than planning models. In a planning setting, we generally assume the

data is fine, and we rarely have operations people looking over our shoulder criticizing the solution. Just as important, the precise solution (how the freight is moving) is less important than aggregate performance statistics.

In an operational setting, dispatchers typically already know what they would do, and if the model disagrees, then you face the challenge of trying to find out whether the discrepancy is because of a problem or if the model is simply displaying intelligence.

6. Summary remarks

Problems in freight transportation and logistics cover a range of problem classes, but most can be characterized by dynamic information processes. In this chapter, we provide an overview of the most important operational settings, and provide a notational framework that captures most of these problems. We then summarize four major classes of algorithms, each based on different classes of information, that can be used to solve these problems. Finally, we illustrate these algorithms in the context of some of the major problem classes. We made a point of avoiding detailed descriptions of models that are unique to specific modes (such as the blocking problem of railroads, or crane scheduling for intermodal ports), preferring instead to provide foundational models that could be adapted to different settings.

The design of models and algorithms for dynamic problems is relatively immature compared to the extensive body of research on deterministic problems. Not only are the models of physical operations quite young, there has been surprisingly little formal research on costing models, and virtually no research (that we are aware of) governing the design of information systems (which ultimately is what really controls operations).

References

Bertsekas, D. & Tsitsiklis, J. (1996), *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA. 4

Bodin, L., Golden, B., Assad, A. & Ball, M. (1983), 'Routing and scheduling of vehicles and crews', *Computers and Operations Research* **10**(2), 63–211. 1

Bramel, J. & Simchi-Levi, D. (1995), 'A location-based heuristic for general routing problems', *Operations Research* **43**, 649–660. 84

Brown, G., Graves, G. & Ronen, D. (1987), 'Scheduling ocean transportation of crude oil', *Mgmt. Sci.* **33**, 335–346. 1

Cheung, R. & Powell, W. B. (1996), 'An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management', *Operations Research* **44**(6), 951–963. 3

Cheung, R. K.-M. & Powell, W. B. (2000), 'SHAPE: A stochastic hybrid approximation procedure for two-stage stochastic programs', *Operations Research* **48**(1), 73–79. 41

Crainic, T. & Laporte, G. (1997), *Design and Operation of Civil and Environmental Engineering Systems*, Wiley-Interscience, New York, chapter Planning Models for Freight Transportation, pp. 343–394. 1

Crainic, T. & Rousseau, J.-M. (1988), 'Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem', *Transportation Research B* **20B**, 290–297. 1

Crainic, T. & Roy, J. (1988), 'OR tools for the tactical planning of freight transportation', *European Journal of Operations Research* **33**, 290–297. 1

Crainic, T. & Roy, J. (1992), 'Design of regular intercity driver routes for the LTL motor carrier industry', *Transportation Science* **26**, 280–295. 1

Crainic, T., Ferland, J. & Rousseau, J.-M. (1984), 'A tactical planning model for rail freight transportation', *Transportation Science* **18**(2), 165–184. 1

Crainic, T., Gendreau, M. & Dejax, P. (1993), 'Dynamic stochastic models for the allocation of empty containers', *Operations Research* **41**, 102–126. 3

Desrosiers, J., Solomon, M. & Soumis, F. (1995), Time constrained routing and scheduling, *in* C. Monma, T. Magnanti & M. Ball, eds, '*Handbook in Operations Research and Management Science*, Volume on *Networks*', North Holland, Amsterdam, pp. 35–139. 1

Dror, M. (1993), 'Modeling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution', *European Journal of Operations Research* **64**(3), 432–441. 4

Dror, M., Laporte, G. & Trudeau, P. (1989), 'Vehicle routing with stochastic demands: Properties and solution frameworks', *Transportation Science* **23**, 166–176. 4

Fisher, M. (1995), Vehicle routing, *in* C. Monma, T. Magnanti & M. Ball, eds, '*Handbook in Operations Research and Management Science*, Volume on *Networks*', North Holland, Amsterdam, pp. 1–33. 1, 84

Fisher, M. L. & Jaikumar, R. (1981), 'A generalized assignment heuristic for vehicle routing', *Networks* **11**(2), 109–124. 84

Frantzeskakis, L. & Powell, W. B. (1990), 'A successive linear approximation procedure for stochastic dynamic vehicle allocation problems', *Transportation Science* **24**(1), 40–57. 3

Gendreau, M., Guertin, F., Potvin, J. & Taillard, E. (1999), 'Parallel tabu search for real-time vehicle routing and dispatching', *Transportation Science* **33**, 381–390. 4, 84

Glickman, T. & Sherali, H. (1985), 'Large-scale network distribution of pooled empty freight cars over time, with limited substitution and equitable benefits', *Trans. Res.* **19**, 85–94. 1

Godfrey, G. & Powell, W. B. (2000), An adaptive, dynamic programming algorithm for stochastic resource allocation problems II: Multi-period travel times, Technical report, Department of Operations Research and Financial Engineering, Princeton University. CL-00-05. 4

Godfrey, G. & Powell, W. B. (to appear), 'An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times', *Transportation Science*. 4

Haghani, A. (1989), 'Formulation and solution of a combined train routing and makeup, and empty car distribution model', *Transportation Research* **23B**(6), 433–452. 1

Herren, H. (1977), 'Computer controlled empty wagon distribution on the SSB', *Rail International* **8**(1), 25–32. 3

Jaw, J., Odoni, A., Psaraftis, H. & Wilson, N. (1986), 'A heuristic algorithm for the multi-vehicle many-to-many advanced request dial-a-ride problem with time windows', *Transportation Research* **20B**, 243–257. 4

Jordan, W. & Turnquist, M. (1983), 'A stochastic dynamic network model for railroad car distribution', *Transportation Science* **17**, 123–145. 3

Laporte, G. (1992), 'The vehicle routing problem: An overiew of exact and approximate algorithms', *European Journal of Operations Research* **59**, 345–358. 84

Laporte, G. & Louveaux, F. (1990), Formulations and bounds for the stochastic capacitated vehicle routing problem with uncertain supplies, *in* J. Gabzewicz, J. Richard & L. Wolsey, eds, 'Economic Decision-Making: Games, Econometrics and Optimization', North Holland, Amsterdam. 4

Leddon, C. & Wrathall, E. (1967), Scheduling empty freight car fleets on the louisville and nashville railroad, *in* 'Second International Symposium on the Use of Cybernetics on the Railways, October', Montreal, Canada, pp. 1–6. 3

Mendiratta, V. & Turnquist, M. (1982), 'A model for the management of empty freight cars', *Trans. Res. Rec.* **838**, 50–55. 3

Misra, S. (1972), 'Linear programming of empty wagon disposition', *Rail International* **3**, 151–158. 3

Muriel, A. & Simchi-Levi, D. (to appear), Supply chain design and planning - applications of optimization techniques for strategic and tactical models, *in* S. Graves, ed., '*Handbook in Operations Research and Management Science*, Volume on *Supply Chain Management*', North Holland, Amsterdam. 6

Papadaki, K. & Powell, W. B. (2001), A scalable adaptive dynamic programming problem for a batch production process, Report CL-01-01, Princeton University. 80

Powell, W. B. (1986*a*), 'A local improvement heuristic for the design of less-than-truckload motor carrier networks', *Transportation Science* **20**(4), 246–257. 1

Powell, W. B. (1986*b*), 'A stochastic model of the dynamic vehicle allocation problem', *Transportation Science* **20**, 117–129. 3

Powell, W. B. (1987), 'An operational planning model for the dynamic vehicle allocation problem with uncertain demands', *Transportation Research* **21B**, 217–232. 3

Powell, W. B. (1989), 'A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy', *Transportation Science* **23**(4), 231–243. 41

Powell, W. B. (1996), 'A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers', *Transportation Science* **30**(3), 195–219. 3, 4

Powell, W. B. & Carvalho, T. A. (1998), 'Dynamic control of logistics queueing network for large-scale fleet management', *Transportation Science* **32**(2), 90–109. 3, 60

Powell, W. B., Godfrey, G., Papadaki, K., Spivey, M. & Topaloglu, H. (2000*a*), Adaptive dynamic programming for multistage stochastic resource allocation, Technical Report CL-01-03, Department of Operations Research and Financial Engineering, Princeton University. 5

Powell, W. B., Jaillet, P. & Odoni, A. (1995), Stochastic and dynamic networks and routing, *in* C. Monma, T. Magnanti & M. Ball, eds, '*Handbook in Operations Research and Management Science*, Volume on *Networks*', North Holland, Amsterdam, pp. 141–295. 4

Powell, W. B., Shapiro, J. A. & Simão, H. P. (2000*b*), An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem, Technical Report CL-00-06, Department of Operations Research and Financial Engineering, Princeton University. 5, 70

Powell, W. B., Shapiro, J. A. & Simão, H. P. (2001), A representational paradigm for dynamic resource transformation problems, *in* R. F. C. Coullard & J. H. Owens, eds, 'Annals of Operations Research', J.C. Baltzer AG, pp. 231–279. 8, 25

Powell, W., Towns, M. T. & Marar, A. (2000*c*), 'On the value of globally optimal solutions for dynamic routing and scheduling problems', *Transportation Science* **34**(1), 50–66. 84

Psaraftis, H. (1988), Dynamic vehicle routing problems, *in* B. Golden & A. Assad, eds, 'Vehicle Routing: Methods and Studies', North Holland, Amsterdam, pp. 223–248. 4

Psaraftis, H. (1995), 'Dynamic vehicle routing: Status and prospects', *Annals of Operations Research* **61**, 143–164. 4

Regan, A., Mahmassani, H. S. & Jaillet, P. (1998), 'Evaluation of dynamic fleet management systems - simulation framework', *Transportation Research Record* **1648**, 176–184. 4, 84

Secomandi, N. (2000), 'Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands', *Computers and Operations Research* **27**(11), 1201–1225. 4, 85

Secomandi, N. (2001), 'A rollout policy for the vehicle routing problem with stochastic demands', *Operations Research* **49**(5), 796–802. 4, 85

Stein, D. (1978), 'Scheduling dial-a-ride transportation systems', *Transportation Science* **12**, 232–249. 4

Stewart, W. & Golden, B. (1983), 'Stochastic vehicle routing: A comprehensive approach', *Eur. J. Oper. Res.* **14**(3), 371–385. 4

Topaloglu, H. & Powell, W. B. (2000), Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems, Technical Report CL-00-02, Department of Operations Research and Financial Engineering, Princeton University. 67

Turnquist, M. (1986), Mov-em: A network optimization model for empty freight car distribution, School of Civil and Environmental Engineering, Cornell University. 3

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

Wilson, N. (1969), Dynamic routing: A study of assignment algorithms, Ph.d.thesis, Department of Civil Engineering, MIT, Cambridge, MA. 4