Optimization Models and Algorithms: An Emerging Technology for the Motor Carrier Industry

Warren B. Powell

Abstract—The emergence of new information technologies, tremendous competition, and the growth of very large carriers in truckload tracking is placing increasing emphasis on powerful new tools for planning operations. Increasingly, management in the motor carrier industry is being asked to review and critically evaluate these new technologies for their own operations. We review optimization models and algorithms for transportation, with the intent of providing lay management with at least a feeling of the range of technologies that are available, and a sense of the strengths and weaknesses of each. The presentation describes the different types of applications of optimization to transportation that have evolved, and outlines how different methods are suited to different types of problems. Special attention is given to real-time optimization for truckload motor carriers. A new interactive graphic simulation system, called MIDAS, (micro dispatch and simulation), has been developed for demonstrating optimization methods, allowing the user to compete against the optimization model in a real-time framework.

I. INTRODUCTION

Recent years have seen significant advances in information technology in transportation, the most dramatic being the use of radio tracking and satellite communications. Reduced communication costs, better fleet control, and improved order tracking are combining to make mobile communication and tracking systems part of the cost of doing business for large carriers. A significant, but far less visible, development in information technology has been the emergence of powerful optimization models and algorithms that can assist carriers with the actual process of making decisions. These methods, which often are described under banners like vehicle routing, work assignment or fleet management, allow the computer to make recommendations regarding work assignment and truck routing and scheduling. At the same time, advances have been made in the use of what is popularly referred to as artificial intelligence (AI), which approaches the problems from an entirely different perspective.

Several forces have encouraged the adoption of advanced planning systems. These include the following:

- Pressure to cut costs—This is nothing new, although deregulation has been a major culprit. However, the perennial intense competition in the truckload market has put particular emphasis on the need to be as efficient as possible.
- Increasing industry concentration—Shippers want to work with fewer, larger carriers. This simplifies billing, tracing, contract negotiations and increases leverage. Larger carriers are better able to provide the additional services that are becoming an accepted part of the industry. However, larger carriers are considerably harder to manage. Smaller carriers can sometimes be run more efficiently simply because a few good people are able to effectively manage the entire company.
- Declining cost and increasing power of computers—Requiring the computer to think, rather than just report, does put additional demands on the machine. The speed of new machines and the relatively low cost of this equipment make meeting this requirement much more cost effective. Particularly important has been the advent of workstations, which creates the possibility of highly economical turn-key systems.
- Advances in optimization technology—Advances in our ability to solve large problems over the last decade creates a tool that tends to look for a market. Universities and consulting firms involved in the development of these new tools are working to raise the level of awareness in industry of this new capability.

The problem that is most commonly approached using computer-based methods is that of routing a set of vehicles out of a depot among a group of customers. For truckload motor carriers, the problem does not require running a vehicle through multiple customers, performing pickups and deliveries, but rather requires the continuous assignment of drivers to loads and the empty repositioning of idle drivers. An important characteristic of the driver assignment problem is that it must be solved continuously, since information is constantly changing.

The purpose of this paper is to review the existing technologies for solving the driver assignment problem for truckload motor carriers. We begin with a short discussion of the different problems in transportation that are amenable to optimization methodologies, followed by a review of different optimization technologies that have proved effective in commercial applications. These include:

- local improvement heuristics;
- network flow algorithms;
- Lagrangian relaxation methods;
- set partitioning and linear programming formulations;
- artificial intelligence.

Next, truckload operations are presented to expose the dimensions of this problem, where we point out that running a truckload carrier is much more than determining which driver to assign each load. This discussion provides a foundation for presenting a relatively new type of model which combines driver assignment with other fleet management decisions. This model

Manuscript received June 1990. This work was supported by the National Science Foundation under a Presidential Young Investigator Award, Grant ECE-8451466.

The author is with the School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544.

IEEE Log Number 900760.
has been implemented within a package called MIDAS (micro dispatch and simulation), an interactive graphic simulation system that provides an effective environment for testing and developing optimization models for truckload carriers.

II. AN OVERVIEW OF MODELS IN TRANSPORTATION

A variety of problems has proved amenable to optimization methods. They vary in the type of problem being solved and the solution method. All, however, are actively being used in a variety of industrial applications. The purpose of this review is to place our discussion of fleet management models for truckload motor carriers within the broader context of transportation models.

A. Vehicle Routing

The classical vehicle routing problem involves running a small fleet of vehicles (perhaps between 10 and 50) out of one or more warehouses picking up or delivering shipments to a set of customers. Fig. 1 illustrates a set of vehicle tours covering a set of customers. The standard vehicle routing problem strives to minimize costs subject to vehicle capacity constraints, but in practice a variety of other considerations may need to be considered, including:

- maximum tour length restrictions;
- time window constraints on pickups or deliveries;
- handling multiple pickups and deliveries;
- multiple vehicle types;
- taking advantage of backhaul opportunities;
- maintaining customer inventories

The many variations of this standard vehicle routing problem are well documented in [4] or more recently in [12]. It has been the experience of practitioners in the field that almost every vehicle routing application is a little different from the next (see, for example, [30]). The basic vehicle routing problem is known to be mathematically difficult (it belongs to a class of hard problems that are referred to as NP complete), and as a result a broad range of heuristic procedures have been devised. These include:

- Tour construction procedures [27];
- Tour improvement procedures (this covers an entire family of procedures; [29], [21], and [24]).
- Cluster-first, route second procedures [28], [8], [15].

There are continuing efforts to solve vehicle routing problems exactly. Interestingly, the best successes have been in the simpler traveling salesman problem (see [22] and [1]) and the harder vehicle routing with time windows problem [7]. For the most part, however, commercial implementations continue to use heuristics.

The vehicle routing problem, and its many cousins, is one of the richest problems in transportation both because of its wide applicability and its fundamental complexity from a mathematical point of view. The hundreds of articles written on the subject (the Bodin et al. [4] review contains more than 700 references) over the last four decades are a testament to the richness of the problem.

B. Driver Assignment

From the extreme complexity of the vehicle routing problem, there is a separate problem that arises in truckload trucking that we will call the driver assignment problem. Although this formulation will seem like a highly simplistic view of truckload operations, the basic problem of assigning drivers to loads falls within this category. One carrier (Schneider National, Inc.) has been solving an assignment problem in real-time as an aid to dispatchers.

The driver assignment problem can be described very simply. Consider the situation depicted in Fig. 2 where a set of idle drivers (depicted as triangles) must be assigned to a set of pending loads (depicted as squares). The problem is one of matching the right driver to the right load in such a way as to minimize total empty miles traveled (there are many other considerations, but this will serve our discussion now). The
resulting problem can be formulated as a network model, shown in Fig. 3. The value of this formulation is first that there are extremely fast and simple algorithms for solving this problem to optimality (discussed below). Second, it allows a fairly high level of detail to be incorporated into the cost of each assignment. For example, precise information about the location of a driver and a load can be used to develop an accurate estimate of the cost of assigning a driver to a load. It is also possible to add artificial costs and bonuses to reflect other factors, such as the desirability of using a particular driver for that shipper, or the degree to which a load helps get a driver home. If there are more drivers than loads, we introduce a dummy load to soak up the excess drivers. If there are 50 drivers and 40 loads, then we introduce a dummy load requiring 10 drivers. When a driver is assigned to a dummy load, this is equivalent to telling the driver to do nothing. Similarly, if there are more loads than drivers, then we create a dummy pool of drivers to handle the excess loads. This formulation of the driver assignment problem is myopic in the sense that it tries to do the best with what is known at the time. The model is unable to anticipate future loads. Thus, the best action at the moment might be to move a driver empty 80 miles to handle a waiting load, although a dispatcher might know that it is very likely that another load will be called in soon that is much closer. Despite these limitations, the assignment formulation does provide useful information to the dispatcher by virtue of the fact that it simultaneously considers all drivers and loads and makes recommendations based on the entire system.

C. Driver/Crew Scheduling

The driver or crew scheduling problem is the next step past the driver assignment problem. Here we typically have a set of tasks spread over time, and the problem is to develop an itinerary for each driver. This contrasts with the driver assignment problem where we have to assign each driver to a single task. For example, consider the set of tasks depicted in Fig. 4(a). The driver scheduling problem is to find a set of tours that describe what each driver will do over the planning interval, as depicted in Fig. 4(b). Thus, a solution to the driver scheduling problem is a sequence of tasks scheduled over time. Note that driver scheduling also contrasts with vehicle routing in that there is no need to cluster orders together.

Crew scheduling is widely used by the airline industry in order to work out itineraries for airline crews (see, for example, [16], [2], and [6]). Using a fixed set of flights, the problem is to determine an itinerary for each crew so that all the flights are covered at least cost. The biggest difficulty has been the presence of a variety of complex work rules which have the effect that the cost of a tour is not equal to the sum of the costs of the individual legs. For example, the cost of sending a crew over a particular leg may depend on how many hours the crew has worked that week, which would determine if the work falls outside of normal hours, thereby requiring overtime rates (or, preventing the assignment altogether because of legal limits on hours worked).

D. Dynamic Fleet Management

Dynamic fleet management problems arise in the context of large common carriers such as large truckload carriers, rail and container, as well as other applications such as one-way car and truck rentals and taxi fleet management. Given an initial distribution of a fleet of vehicles, as well as both current and forecasts of future demands, the problem is to determine what should be done now with the fleet in terms of holding vehicles in inventory, repositioning them empty to new locations, and moving them loaded to meet current demands. Dynamic networks have proved to be a powerful tool for modeling transportation problems in a variety of contexts, in-
including truck dispatching, railcar fleet management, airline scheduling and fleet management, and the management of international container fleets. An early report on the use of optimization methods in transportation, [23], is still an excellent reference. One of the first uses of dynamic network models can be found in [26] and [25]. The dynamic networks are formulated for, with particular emphasis on, incorporating forecasting uncertainties in [17], [18], and [20]. The problem is formulated in [11] as a large scale stochastic program, which provides a rigorous framework for handling the uncertainty in forecasting demand. A new method is presented that reduces the problem to a pure network, allowing it to be solved with highly efficient algorithms. A stochastic, dynamic network model for rail, which handles demand uncertainties as well as the tendency in rail to form inventories of cars at certain locations is presented in [14]. This model is currently being implemented at a major railroad. Shan [31] represents the problem of managing a fleet of railcars as a dynamic network.

Dynamic fleet management problems are classically approached using dynamic networks, as depicted in Fig. 5. We assume that the area over which the company is operating is divided into a set of regions or other discrete points (such as terminals, railyards, trailer pools or ports). Time is divided into discrete intervals (typically days or fractions of a day), and the model extends over a specific planning horizon which allows us to take into account forecasted events for a reasonable period into the future (the length of the planning horizon is often dictated by the nature of the application). We then form a dynamic network, where the nodes of the network each represent a particular region at a point in time. Network links carry flow from one region and time period to another region in time period. Different types of links in the network are used to model different activities. In network flow models, links are characterized by the cost of the link and its upper bound, which restricts the flow on that link. Table I provides a summary of different link types and their characteristics.

The links in the network describe the types of activities that occur (moving loaded or empty, holding in inventory). Separate from these are the flows of vehicles. Initial supplies of vehicles enter the network both in the first time period as well as later time periods (these are vehicles that are already moving and which do not become available until some point in the future).
Generally, all vehicles depart from the network over the salvage arcs and out the supersink. A supersink is used because we generally do not know a priori where the vehicles will end up. Salvage arcs serve the function of not only funneling all the flows out the supersink, but also provide an opportunity to place costs and bounds on the flows that terminate in each region. For example, it may be possible in the last time period to abandon a number of vehicles in a remote location (Alaska, North Dakota, or, depending on the application, Miami, FL). This occurs because the model does not capture the cost of having to return the vehicles to more productive locations. If this problem is ignored, the model may have end effects, which are distortions produced by truncating the planning horizon.

Issues that need to be addressed in dynamic networks include:

- uncertainty in the forecasts of demands and future vehicle inventories;
- setting the length of the planning horizons, and handling truncation problems;
- handling multiple equipment types (see [31]);
- representing driver movements and constraints;
- modeling the flexibility of demand to be satisfied in more than one time period;
- maintaining equipment pools in each region.

It is sometimes the case, particularly in trucking applications, that the dynamic network model can be formulated as a pure network. In this case, extremely efficient algorithms can solve problems involving over 50,000 links in a few minutes. However, real-world considerations can destroy the special structure required for pure networks. One of these is the presence of multiple equipment types (which is in particular a problem in rail and container applications), but other considerations such as modeling driver work rules or allowing flexibility in the time at which a load is picked up can also present complications. In most cases, specialized algorithms are developed that exploit the underlying network structure as much as possible.

III. Optimization Algorithms for Transportation Models

Perhaps more than any other problem area, transportation has motivated the development of a number of major results in optimization. The range of problems presented above draws on techniques extending from simple local search methods to advanced decomposition and primal partitioning techniques. In practice, however, a few methods have proved to be exceptionally powerful. Since these techniques are general and can be applied in different contexts, our presentation has separated the discussion of the models (Section I) from that of the methods, given in this section. Just the same, some techniques are useful only in certain problem contexts. The discussion here is not intended to serve as a comprehensive survey of these methods, but rather is intended to provide a working vocabulary of important methods. It will also provide a useful background when we present the methods used for solving real-time dispatching for truckload motor carriers. For example, the vehicle routing problem is probably the best known optimization problem in transportation, but the application is primarily in shipper logistics. Truckload motor carriers, which involve assigning drivers to loads and maintaining a fleet of vehicles, face a completely different problem. Just the same, the research behind these methods has played a major role in influencing the development of optimization methods in logistics.

The first step in understanding optimization methods for transportation is the role of models versus algorithms. The model is the translation of a real-world problem into a mathematical statement. An algorithm is a method for solving the model. It is often the case that there are several ways to model a particular problem. For more complex models, there is often more than one way to solve the model, leading to the use of different algorithms.

Four classes of solution are described. These include:

- local improvement heuristics;
- network flow algorithms;
- set partitioning methods;
- artificial intelligence.

The presentations are intended to communicate an intuitive sense of the nature of each approach with a minimum of mathematics.

A. Local Improvement Heuristics

Local improvement heuristics cover an entire family of procedures which involve finding an initial solution and then attempting to find better solutions. An improvement is typically constrained to involve only a few changes (hence it is a local improvement). The best illustration of a local improvement technique is in the context of vehicle routing, where local improvement methods are widely used. Consider the three vehicle tours in Fig. 3. There are two types of improvements for which we might look. The first is the sequencing of customers within a tour (this involves solving a traveling salesman problem). Remembering that computers are blind and cannot visualize the figure the way humans can, we might look for improvements by searching over all the customers looking for opportunities to make changes. Local improvements are often characterized by the complexity of the changes that are considered. A simple 2-opt change seeks to find improvements that require making only two changes at a time. For example, we might choose customer 4 in Fig. 3 and decide to drop the leg between 3 and 4. Now we have to look for a leg to add. If there are 10 customers in the tour, there are eight possible legs that could be added. A computer might have to search all eight, but we will cheat and choose to add leg 3 to 5. Now we have to drop one of the legs out of 5, and it is easy to see that we have to drop the leg 5 to 6. Finally, we have to close the loop by adding the leg from 4 to 6 producing a new tour, illustrated in Fig. 6. If this tour is less costly than the previous one, we keep it and continue to search starting with this new tour. Otherwise, we restore the original tour and try something else. This is considered a 2-opt change because we had to choose which leg to drop (we chose 3 to 4) and which leg to add (3 to 5). The other changes (dropping 4 to 5 and adding 4 to 6) were determined by the first two changes.

A 2-opt algorithm is generally very fast, but it is not very sophisticated because it cannot consider more sophisticated changes. For example, it may only be possible to find an improvement by changing three things at once. Algorithms that consider these more sophisticated changes are called 3-opt procedures. Of course, 3-opt procedures are much more expensive because they have to search over many more options. In addition, these changes have only considered improving the sequencing of customers within a tour. It may also be necessary to change what customers are served by each vehicle. A 2-opt customer interchange heuristic would consider interchanging two customers between two vehicles. For example, starting with the tour in Fig. 6, we might try to exchange customer 7 on one tour
with customer 9 on the other tour. Technically, in a 100 customer problem, there may be as many as 99 possible interchanges for every customer, creating 9900 possible customer interchanges. The real number is somewhat less because you would not interchange with customers already being served by the same vehicle. More significantly, commercial local improvement schemes use a range of pruning rules to eliminate interchanges that have little likelihood of success. Note: this pruning can restrict the quality of the final solution. Given that the basic idea behind a local improvement heuristic is relatively simple, it is the sophistication of the pruning rules that distinguishes a quality commercial code from a quick programming job. Restrictive pruning rules can make for a very fast code but a poor quality solution; loose pruning rules may give good solutions on small test problems, but may bog down in an operational environment on large problems.

The biggest limitation of local improvement heuristics is that they are unable to consider complex interchanges to find improvements. For example, it may be possible to find an improvement only if there is a three way interchange of customers among vehicles, and if simultaneously the system considers resequencing the order in which customers are handled within a tour. Even with effective pruning rules, there are simply too many such changes to be considered, and the algorithm would be far too slow. In practice, however, local improvement heuristics seem to give good quality solutions, and as a result have seen wide acceptance. They cannot handle extremely large problems, but are well suited to optimizing fleets of up to 50 trucks among several hundred customers. Problems of this size cover many situations in shipper applications.

Local improvement heuristics are by no means restricted to classical vehicle routing problems. Consider instead the driver assignment problem depicted in Fig. 3. Although this can be solved using efficient network algorithms, it is a good frame-

**Fig. 7.** (a) Driver assignments using nearest load logic. (b) Optimal driver assignments.

<table>
<thead>
<tr>
<th>TABLE II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distances from Drivers to Load</td>
</tr>
<tr>
<td>Driver \ Load</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

work to illustrate the use of local search techniques in a different context. Table II gives the distances from each driver to each load. When using a local improvement heuristic, it is necessary to first get an initial solution. One is to start with every driver doing nothing, but a more effective one (though not the best) would be to pass through all the drivers, assigning each driver to the closest available load. If we start with driver 1, we obtain the assignment given in Fig. 7(a), covering a total distance of 360 mi. Note that this results in the poor assignment of driver 5 to load 5 (this is obvious looking at the figure, but once again, computers cannot look at pictures).

A local improvement heuristic applied to this problem would work as follows. A 2-opt interchange procedure would try to
improve the solution by exchanging the assignments of two drivers. For example, we might interchange the assignments of drivers 1 and 2, letting driver 1 take load 2 and driver 2 take load 1. This interchange would actually raise costs by \((60 - 55) + (90 - 65) = 30\). A better interchange might be drivers 1 and 5, letting driver 1 take load 5 and driver 5 take load 1, which would change costs by \((135 - 55) + (50 - 120) = 10\), which also increases total costs. In fact, no change of the assignments of two drivers will improve this solution.

The optimal solution, shown in Fig. 7(b), requires switching driver 1 to handle load 2, driver 2 to handle load 3, driver 3 to handle load 4, driver 4 to handle load 5, and finally letting driver 5 handle load 1. The total distance covered is 345 mi, for a savings of 15 mi. In other words, we needed to make five changes simultaneously. On large problems, it would be impossible to design a local interchange heuristic that could have found this particular change.

The strength of local improvement methods is that they are simple to implement, and are very amenable to handling a variety of real-world constraints. For example, as part of a particular change, it is possible to check if the change would be feasible in terms of a variety of real-world constraints. For example, before accepting a particular change (such as that illustrated in Fig. 6) it is possible to check if the total length of the tour exceeds a limit on total driving time; we can check the time at which the vehicle arrives at different customers; or we can enforce a condition that the driver visit one customer before another. We can even evaluate the cost of the solution using complex strategies, such as accounting for driver overtime. It might be that the original tour in Fig. 6 required 30 min of driver overtime, while the improved solution resulted in a total tour length within the eight hour driving rule, producing cost savings that are greater than the simple savings in distance.

### B. Network Flow Algorithms

Certain problems can be formulated to take advantage of a special class of procedures known as network flow algorithms. The basic mathematics behind these methods have been known since the 1950's, but the major computational breakthroughs took place in the 1970's and the 1980's. In fact, most of the developmental work during the last decade has not been in the basic optimization algorithm, but rather in the translation of real-world problems into networks, an exercise referred to as modeling. Network algorithms can solve extremely large problems with exceptional speed. Also important is the fact that network models can be easily communicated via graphical means. The network in Fig. 3 represents a powerful visual image that is easily understood, and helps both the analyst and the manager in understanding the problem being solved.

An excellent example of a network flow problem is the driver assignment problem. Fig. 3 represents the problem as a network, which is characterized by the following components:

- **Nodes**—These are points where flow enters and leaves the network, or changes direction.

- **Links**—These represent ways in which flow can move from one node to the next. For example, assigning a driver to a load is represented as flow from a driver node to a load node.

Associated with nodes and links are various data that determine the demands placed on the network and the costs of different decisions. This includes:

- **Node data**—For each node there is either a surplus, representing flow entering the network, or a deficit, representing flow leaving the network. A driver is modeled as flow entering the network at a driver node (in other words, we have to do something with this driver), and a deficit might be associated with either a load node (meaning we have to get a driver to serve this load), or we might require all the drivers to leave through a supersink (see Fig. 5), which is another way of saying that all drivers must be moved forward in time (some way or another, to be determined by the optimization).

**Link data**—Each link is associated with a cost which a unit of flow incurs by moving over the link, and a bound, which restricts how much flow can be moved over the link.

In the driver assignment problem, both drivers and loads are represented as nodes. One unit of flow enters the network at each driver node, and a unit of flow leaves the network at each load node. The cost on each link might be the distance from each driver to each load. The problem is to determine how to assign flows over the network to minimize total costs. Modern network simplex codes on a mainframe or fast workstation can solve problems with hundreds of drivers and loads to optimality in less than one second! A modern workstation can outperform a mainframe, as well as provide a separate platform, which does not interfere with on-line operations. The methods behind network simplex and related algorithms are well understood (see, for example, [13]). A new algorithm, which uses a primal dual approach, has been shown to be even faster than network simplex [3].

A different example of a network problem is the dynamic fleet management problem illustrated in Fig. 5. Unlike the driver assignment problem, where links go directly from a driver node to a load node, this network involves routing flows of vehicles over space and time. Initial inventories of vehicles enter the network in the first few time periods and then must be pushed forward in time until they leave via the supersink. The intermediate nodes are called transshipment nodes since flows do not enter or leave the network at these points. For example, a vehicle might enter a node representing region 3 on day 4, after which the model will have to decide whether the vehicle should leave that node on an inventory link (meaning that the vehicle is being held overnight); on an empty link to another region; or outbound on another link representing a loaded move. The optimization model can decide to move vehicles empty based on the overall profitability of the move.

The incredible efficiency of network algorithms has helped spawn a cottage industry of specialists who try to model complex problems as networks. Networks represent a powerful tool, but they possess significant restrictions that can limit the range of problems that can be solved. Experts in the area are often very creative at getting around these limitations, but in some cases the technology just does not apply. For example, neither the vehicle routing problem depicted in Fig. 1 or the driver scheduling problem in Fig. 4(a) can be solved using network algorithms. The problem in Fig. 4(a) is particularly important since it illustrates the fact that assigning each driver to one load can be formulated as a network, while assigning a driver to a set of loads over time is not.

### C. Set Partitioning Methods

One of the most powerful concepts in optimization is the formulation and solution of what are known as set partitioning problems. Specifically, the problem of choosing the best tours shown in Fig. 8 to cover the loads that are shown is a classical
example of a set partitioning problem. We have a set of tours, and we wish to choose the best tours from this set to maximize profits or minimize costs, subject to the constraint that we cannot choose two tours to cover the same load. The set partitioning problem can be formulated as a classical linear programming problem and solved using commercial software packages. The real power of this approach is that very general rules can be used to determine which tours should be considered, and what is the cost of each tour. For example, if we want to get a driver home, then we can require that only tours that return a driver home be considered. We may also take into account the hours that a driver has worked and the desirability of allowing certain drivers to serve certain shippers. Normally, special routines are developed for generating tours that are tuned to a specific application, with almost no limit on their ability to incorporate a range of real-world considerations. For this reason, set partitioning techniques are expected to remain the most powerful method of formulating many real-world problems.

There are two general strategies for generating tours. One, referred to as tour enumeration, simply enumerates all reasonable tours, using a variety of pruning rules to determine what is reasonable. Note that these pruning rules may eliminate economical tours, bringing into question exactly what is meant by an optimal solution to this problem. If strong pruning rules are used, then the risk of eliminating some effective options is very real. On the other hand, if strong pruning rules are not used, then it is possible to generate thousands (or millions) of tours, producing an extremely large optimization problem that would challenge the most powerful computers and optimization packages.

Once a set of tours has been generated, the problem reduces to the difficult but straightforward task of choosing the best tours out of the set. Local improvement heuristics tend not to work well in these situations, and the problem cannot be solved using network algorithms. Langrangian relaxation methods can be used, but the most widely used approach (at this writing) is a standard linear programming code. Commercial packages can solve problems involving thousands of potential tours in a "reasonable" amount of time (a word of caution—the time required to solve these problems varies widely depending on the computer, the characteristics of the problem, and the specific software package being used). For problems with 50 to 100 drivers, this approach can be extremely effective. For larger problems, however, the number of potential tours can grow astronomically. Even with 50 drivers, it is possible to generate over 10000 tours (if we are trying to look several days into the future). Very large problems could produce several hundred thousand tours, and some problems in airlines have produced several million tours. For this reason, a popular approach is to use what is referred to as column generation techniques (for technical reasons, the research community refers to tours as columns). These methods do not enumerate all possible tours, but rather use optimization theory to generate the tours that are most likely to be used.

1) Lagrangian Relaxation Methods: There are many problems that would be relatively easy if we could simply ignore certain aspects of the problem. One of the best examples is the problem of scheduling a set of drivers over multiple loads as depicted in Fig. 4(b). Although the solution to this problem seems obvious, when there are many drivers and loads, and it is necessary to schedule each driver over potentially multiple loads, this problem can be exceptionally difficult. For example, Fig. 4(b) gives only the best tours. Fig. 8 shows that there are many more tours that might have to be considered (in a realistic, medium sized problems, we might have to consider tens of thousands of such tours). Note that there are two potential tours out of NYC, and that three separate tours (B, D, and E) cover the same load from PIT to NYC (this conflict is highlighted by the circle grouping the tours that cover this one load). Obviously, we cannot use all these tours, but rather have to choose the best ones. In doing so, we cannot choose two tours that cover the same load, and yet we would like to cover all the loads (although we may not be forced to do this).

What makes this problem hard is that we might like to use tour B in order to assign a driver out of NYC to the load going from NYC to PIT, and we would like to use tour D to assign a driver out of BAL to a load going to PIT. But we cannot use both of these tours, because they both cover the same load from PIT to NYC. Thus, we have to choose one tour or the other, and then generate even more tours to cover the remaining load. This problem has been complicated by the fact that the decisions are bundled, in the sense that the decision to use tours A, D or E are coupled by the fact that they all wish to cover the same load (as a result, these are not independent decisions). One way to unbundle them is to allow two or more of these tours to be used (which would seem to allow use to carry the same load from PIT to NYC twice), but to add a penalty onto the PIT to NYC load. When this approach is used, the resulting problem is generally very easy, or even trivial, to solve. However, it is possible that the optimal solution might use both tours A and D (meaning that there were two drivers willing to pay the penalty to use the same load), in which case we say that the solution is infeasible. This penalty, known as a Lagrange multiplier, is a pricing mechanism that allows us to sort out whether we should really use tour A, D or E. By adding a progressively higher penalty to the leg from PIT to NYC, eventually only one of these three tours will still want to cover this load.

Lagrangian relaxation is a scientific approach to relaxing constraints (in this case, allowing two or more drivers to carry the same load) and then adding penalties whenever we violate one of these constraints (see [8], [9]). The difficult step is in determining how large the penalties should be. This is complicated by the fact that we have to solve this problem for many drivers and loads simultaneously. As a rule, Lagrangian relaxation methods can often provide good solutions (often within 5% of the optimal solution) in a reasonable time. However, the success of the technique is highly dependent on the characteristics of the problem, and careful experimental work is needed to document its performance in different applications.
2) Linear Programming: The most "standard" approach to solving large set partitioning problems is the use of commercial linear programming packages. Commercial packages can solve problems involving thousands of potential tours in a "reasonable" amount of time (a word of caution—the time required to solve these problems varies widely depending on the computer, the characteristics of the problem, and the specific software package being used). For problems with 50 to 100 drivers, this approach can be extremely effective. For larger problems, however, the number of potential tours can grow astronomically. Even with 50 drivers, it is possible to generate over 10,000 tours (if we are trying to look several days into the future). Very large problems could produce several hundred thousand tours, and some problems in airlines have produced several million tours. For this reason, a popular approach is to use what is referred as column generation techniques. These methods do not enumerate all possible tours, but rather use optimization theory to generate the tours that are most likely to be used.

Set partitioning methods are widely used by the airlines for scheduling crews. Since airline schedules are fixed, it is possible to take some time to determine the best set of schedules (a given crew schedule might run over a period of a week). In trucking, the problem is typically much more dynamic, providing considerably less time to solve the problem but generally not requiring the planner to look as far into the future.

The solution of large set partitioning problems continues to challenge the optimization community, and remains the subject of very active research. With modern mainframes (and certain very fast workstations), and modern optimization methods, the research community has achieved some success solving these large problems. However, the several hours required to solve an airline crew scheduling problem may not be feasible for a trucking company trying to make a decision right now. A considerable amount of this work is being undertaken by private companies that, for obvious reasons, do not publish their methods or results.

D. Artificial Intelligence

A considerable amount of visibility has been given to artificial intelligence (AI), both in the academic and consulting communities. While no attempt is made here to provide a full introduction to artificial intelligence, a few observations can be made based on this author's experiences with artificial intelligence in consulting and in research.

First, it should be said that artificial intelligence is a powerful and flexible concept with wide applicability, in particular to problems that resist a mathematical formulation. Applications from diagnosing medical problems to choosing the right wine for dinner illustrate situations where AI is a useful concept. At the same time, there is no rigorous definition of exactly what is AI, and it has become a catchall umbrella that has been used to describe extremely simple heuristic rules that have been used for generations to solve problems in the real world. Most disturbing to this author, however, is that a consulting firm may give a presentation on the full power and flexibility of the AI framework, and then deliver a simple set of heuristics, perhaps using a specialized AI language such as LISP, which could have been written without any prior knowledge of AI. In short, let the buyer beware.

AI works well in instances where issues and trade-offs are difficult to model or quantify. On the other hand, it works very poorly when there are an extremely large number of options to consider. For example, determining the best load to assign a driver to might be an excellent application of AI since there may only be 10 or 20 loads to consider, and there may be a number of "soft" issues that need to be taken into consideration. On the other hand, the problem of determining the best assignment of 10 drivers to 10 loads simultaneously would be extremely difficult to handle in an AI context, since there are over three million possible ways of assigning 10 drivers to 10 loads.

A common misconception is to assume that if you are using optimization then you are not using AI. Optimization experts have for many years been using AI concepts within their optimization models under the label of simple rules and heuristics. Now, these same professionals will call these simple rules "AI." While experts in AI do not recognize the coding of these simple rules as true artificial intelligence, the logic is often as sophisticated as the "AI" packages delivered by some consulting firms.

The biggest drawback of a pure AI system in a transportation context is that specific mathematical properties of the optimization problems that arise in transportation are being ignored. An optimization package that uses these powerful results, and also uses the flexible rules in knowledge-based AI systems, will always outperform a system that does not use this information. In addition, it is very tempting to code the system using one of the very flexible AI language environments, which can greatly reduce system development time. This author has seen several instances of AI systems that were coded in an AI language but without actually using any AI concepts whatsoever. The result is often a package that is incredibly slow compared to a system coded in a native language such as C or Fortran.

IV. Optimization Models for Real-Time Dispatching for Truckload Carriers

Large truckload motor carriers pose a special challenge for optimization because the problems are large, dynamic and combine a variety of complex issues. The discussion above provides an important context for discussing this problem, in part because such a large part of the research and consulting community has focused on these problems and approach truckload trucking from this perspective. In the first section below, we briefly review the major elements of truckload operations, bringing out the different dimensions of the problem. Then, we present an optimization model that is able to use the highly efficient network algorithms discussed above.

A. An Overview of Truckload Operations

We turn now to the problem of managing a large fleet for a truckload motor carrier. This problem is different than classical vehicle routing or crew scheduling problems that have received so much attention primarily because the problem is so dynamic. In addition, there is more to the problem than just assigning drivers to loads.

Truckload operations can be reduced to four major components:

- Driver assignment—Determining what driver to assign to each load.
- Empty repositioning—Determining how to manage the fleet when there are more drivers than loads.
- Load selection/evaluation—Determining which loads to accept when there are more loads than drivers.
- Load solicitation—When more drivers than loads are forecasted, load solicitation is the process of calling shippers and trying to attract freight. By calling the right shippers,
carriers can increase the likelihood of attracting loads moving to areas where they most need the equipment.

The driver assignment problem, although the easiest to state, is the most complicated in actual practice because of the range of issues that must be balanced. These include the following.

- Minimizing total empty miles.
- Satisfying driver requests (in particular, to return home) and, in general, balancing driver needs. For example, we may want to give priority to a driver that has been waiting a long time, as long as this does not mean driving too many additional miles.
- Satisfying shipper needs—The most important is covering the load, but next to this is assigning a reliable driver to handle the load. The importance of the shipper, and the carrier's past service performance with this shipper play key roles here.
- Handling maintenance requests—Although a smaller issue, occasionally it is necessary to take into account other issues such as equipment maintenance.

Classical applications of operations research methods focus on the driver assignment problem. Knowledgeable experts in the motor carrier industry will recognize that the other three areas, in particular load selection/evaluation and load solicitation, properly and aggressively managed, can have a much larger impact. However, the last three components of truckload operations all require a model which looks into the future and forecasts demands and truck movements. At the same time, improved methods for driver assignment and empty repositioning can often be much easier to implement, both because the instructions are relatively simple and they impact only the dispatchers and customer service people. Load solicitation, on the other hand, requires the cooperation of sales and marketing, as well as the shippers, and the impact of load solicitation can be more difficult to measure.

There are several characteristics of the driver assignment problem that make it much more difficult than classical vehicle routing problems or the crew scheduling problems faced by the airlines. Driver assignment must be performed on a rolling basis, with decisions being made under pressure over the course of the day, responding to shipper requests that are continually being called in. In addition, drivers are also constantly checking in, often changing their status or their estimate of when they will arrive. One important implication of this is that the classical use of optimization methods does not apply. Specifically, it is common to develop an optimization model which reads in data, generates a network, optimizes the network, and outputs the results. "Fast" models are those requiring only a few minutes on the computer. However, driver assignment requires making decisions while the driver is on the phone (with mobile communications, this is becoming less of an issue). It is not uncommon for a driver to call in, tell the dispatcher that he is empty and available (possibly before he was expected to call in) and ask what his next assignment is. The dispatcher cannot tell the driver to call back in ten minutes while the model is running.

A second difficulty with the driver assignment problem is that it may be very difficult to differentiate good answers from bad ones. In the case of the vehicle tours in Fig. 6, it is very easy to compare two solutions by just calculating the cost of each. In the driver assignment, the quality of a driver assignment may depend on what loads are called in over the next hour. For example, Fig. 9 illustrates what appears to be a good driver assignment, where two drivers cover two loads. But in fact, it might have been better to let driver 1 handle load 2, counting on another driver to arrive later to handle load 1, and anticipating that another load will materialize near driver 2. Instead of being an obscure example, evaluations such as this are made all the time. Of course, sometimes the dispatcher gets lucky, and sometimes he does not. But a few quick phone calls can sometimes change the picture, when a customer service representative "generates" a load near driver 2 (load solicitation) while a different sales rep calls the shipper for load 1 and asks if the load can be picked up tomorrow (when we know we have a driver coming in). In view of these complications, how should different strategies be compared?

B. An Optimization Model for Truckload Motor Carriers

In Fig. 3, we show that the problem of assigning drivers to loads can be formulated as a network problem. This formulation, however, cannot take into consideration any of the future impacts of decisions made now. A dynamic network model, such as that shown in Fig. 5, can handle the dynamic aspects, but loses the high level of detail afforded by the assignment model. Drivers and loads are aggregated by region and time period, with a significant loss in accuracy. A separate, but more subtle problem with the network shown in Fig. 5 is that it assumes that all future demands are known with certainty, a very strong assumption in the motor carrier industry.

We can overcome the assumption that all future demands are known with certainty by using the approach developed in [11]. Aside from providing a more realistic model of the future, this approach reduces what is often a large, dynamic network extending up to two weeks into the future to a much smaller network that more accurately depicts the quality of the information that is really known when a dispatch decision is made. This method also retains the pure network structure of the problem, allowing us to continue to use the powerful network algorithms referred to earlier. However, this model, like that in Fig. 5, also still uses the aggregation of the country into coarse regions.

An extremely powerful approach is to combine the stochastic network model developed in [11] with the assignment network shown in Fig. 3, producing the hybrid network given in Fig. 10. In this formulation, the assignment network is combined with the forecast network, allowing us both the high level of detail
required to assign individual drivers to individual loads, as well as the forecasting capabilities of the dynamic network model. In the assignment network, individual drivers and loads are represented explicitly. Loads are represented as links moving from an individual origin node for that load, but terminates at a node that represents only a region and time period.

From that node, the driver may flow onto other known loads that are to be picked up in the future (such as the load originating from region 3 on day 2, terminating in region 2 on day 4) or may flow into the stochastic links, which capture the value of uncertain forecasted loads.

The power of the hybrid formulation is that it can handle situations where there are too many drivers or too many loads. If there are too many drivers, then the system can forecast future opportunities and make recommendations to hold or reposition drivers based on anticipated future loads. If there are too many loads, then we can choose the best load based on the downstream economics of each load (what is the value of another truck at the destination of each load).

V. MIDAS—A SIMULATION AND GAMING SYSTEM

There can be a variety of different models suggested to solve a specific problem, each varying in terms of how they handle system effects, forecasting uncertainties along with a variety of soft issues. Short of actually implementing each one and undertaking real-world experiments (aside from the risks associated with experimenting with your company, this approach is not as rigorous as it may seem), a method is needed to test and evaluate different dispatch models. The problem of comparing different methods for assigning drivers, selecting loads and repositioning empty drivers is particularly difficult. One of the only truly reliable methods is to compare alternative strategies in the context of a simulation. Although they have the obvious drawbacks associated with any computer model, simulation models allow situations to be modeled in a controlled, experimental environment. Different methods for managing the fleet can be compared under the same demands, starting with the same initial conditions. If we wish to compare two procedures for managing a fleet, we simply run two simulations using the same demands and the same initial truck locations. The simulations must be run for a sufficiently long period of time to capture the dynamic effects of decisions.

MIDAS (micro dispatch and simulation) is an interactive graphic simulation system for testing alternative methods for managing the operations of a truckload motor carrier, covering driver assignment, empty repositioning and load selection (load acceptance and rejection). The system was designed with several objectives in mind.

Validation—Large systems can contain significant errors either in the input data, or in the logic of the software itself. The graphical interface of MIDAS has proved to be an invaluable tool in simply debugging the entire system and gaining confidence in the results.

Understanding—By watching the dispatch decisions as they are being made, it has been possible to develop a much greater degree of insight and understanding than was achievable using an earlier batch-mode simulator. We are able to question dispatching decisions as they are being made, developing a better understanding of the trade-offs that are being made.

Communication—Perhaps the single greatest value of the system is its ability to communicate what automated dispatching is all about. Optimization models have an inherent "black box" quality to them, and the ability to watch them perform in a clear way. People have more confidence in using optimization when they can see it in action first.

Evaluation—Our original and primary objective was to use MIDAS to compare alternative optimization models for managing a fleet. We could test and refine a given optimization model by changing parameters and assumptions and then running the simulation to evaluate its performance.

Once MIDAS was developed, we found we could quickly modify it for another purpose. It is common in the development of optimization models to compare one algorithm against another. In dynamic optimization, we have consistently found that our toughest competitor was not another algorithm, but a human making decisions manually. MIDAS includes an option whereby the optimization logic is disabled, and the user must specify all the driver assignments, including empty repositioning and load selection. This is done via a mouse where the user may select drivers and loads to indicate his choice for an assignment.

The MIDAS screen is illustrated in Fig. 11, where the system is zoomed in on New Jersey and eastern Pennsylvania. (The real system uses a range of colors, but for reproduction purposes a black and white edition has been produced.) In the lower left hand corner, a full map of the United States is shown, along with a rectangle that specifies the coverage of the main window. The user may move this rectangle around or change its size to control the viewing screen. Within the screen, the following items or activities are shown.

Idle drivers—These are shown as small, lightly shaded circles.

Pending loads—These are shown as solid dots surrounded by a larger dot. When a load is first called in, these dots are made relatively large. After about a minute, they shrink to a smaller size.

Region centroids—Although trucks and loads are located at the level of a five digit zip code, the country is assumed to be divided into a set of regions for forecasting purposes. The centroid of each region is given as a shaded square.

Loaded movements—These are shown as solid lines, depicting the entire length of the move. The locations of the truck is shown by a gray dot, which moves along the solid line depicting the process of a truck moving loaded.

Deadhead movements—The empty move from a driver’s current location to a load is shown as a dashed line. As with the
loaded movements, the location of the truck is depicted using a gray dot that moves with the truck.

Empty repositioning moves—These are empty movements where the driver is not moving to a specific load, but is rather being moved empty in the direction of where additional loads are expected to arise. Empty repositioning moves are depicted as dotted lines, always terminating at a region centroid.

MIDAS has been used to perform a variety of experiments evaluating not only variations of the network optimization model posed above, but also to pit human dispatchers (students) against the optimization model. These experiments have shown that as long as the problem is not too large, and as long as there is not too much time pressure, humans can compete effectively against the optimization model. The optimization model pulls ahead as the problem becomes larger and there is more time pressure.

Interestingly, the optimization model also excels in the presence of a number of "soft" issues (which affect the choice of the best driver for a load) since these factors are difficult to display graphically. The graphical display gives the human an excellent command of the relative distances between drivers and loads, but it is difficult to simultaneously display information about the quality of the load; whether a driver wants to go home (and where); the qualifications of a driver for a particular load, or whether the driver has the equipment to handle a load. All of this information can be displayed, but the screen can become very cluttered and the dispatch process can become very slow.

REFERENCES


Warren B. Powell received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge.

He joined the Department of Civil Engineering and Operations Research at Princeton University, Princeton, NJ, in 1981. His research since then has focused on developing and implementing operations research-based models for the motor carrier industry. This work has included the development and implementation of a large tactical and strategic planning system for LTL motor carriers and models for shipper logistics. Most of his work has revolved around dynamic fleet management models for truckload motor carriers, with particular emphasis on real-time dispatching and fleet management for the truckload industry. He is also working on dynamic models for routing drivers, tractors and trailers for LTL carriers.

Dr. Powell is a recipient of a Presidential Young Investigator Award from the National Science Foundation and second place in the 1987 Edelman Award for the practice of operations research. He is currently an Area Editor for the Journal of Operations Research, as well as serving on the boards of Transportation Science and Transportation Research, and is the chairman of the Transportation Science Section dissertation prize competition. He has co-authored over 40 refereed journal articles and is a co-chairman of TRISTAN, a conference for research in transportation and logistics.