Approximate Dynamic Programming Captures Fleet Operations for Schneider National

Hugo P. Simao*    Jeff Day**    Abraham P. George*
Ted Gifford**    John Nienow**    Warren B. Powell*

*Department of Operations Research and Financial Engineering,
Princeton University
**Schneider National

January 2, 2010
Abstract

Schneider National needed a model that would capture the dynamics of their fleet of over 6,000 long-haul drivers so that they could determine where they should hire new drivers, estimate the impact of changes in work rules, find the best way to manage Canadian drivers, and experiment with new ways to get drivers home. To have confidence in the numbers, they needed a model that was as smart as their experienced team of dispatchers and fleet managers. For this purpose, we had to model drivers and loads at a high level of detail, capturing both complex dynamics and multiple forms of uncertainty. We used approximate dynamic programming to produce realistic, high quality decisions that capture the ability of dispatchers to anticipate the impact of decisions on the future. The resulting model produces results that closely match historical performance, while also yielding the marginal value of drivers and loads.
Schneider National is one of the three largest truckload motor carriers in the United States, with over 15,000 drivers. Over 6,000 of these drivers participate in the movement of one-way truckload movements, typically over distances ranging from several hundred to several thousand miles. These drivers often spend two weeks or more away from home, a job characteristic that contributes to driver turnover of 100 percent or more at most long-haul carriers. Schneider was interested in a model that would allow them design business policies that would, among other things, help them get their drivers home on time on a regular basis.

Schneider needed answers to a host of other questions. What would be the impact of changes in federal regulations governing drivers? What was the best way to manage drivers based in Canada? Where should new drivers be hired? How many teams (drivers that work in pairs which can operate 20 hours each day) should the company maintain? Is it possible to make commitments of when a driver will be given time at home?

To produce believable results, the model had to closely fit to actual fleet performance, matching the decisions of a skilled group of dispatchers supported by state of the art planning systems. To capture the behavior of drivers in a realistic way, it was necessary to model drivers using 15 separate attributes. All work rules had to be represented to capture driver productivity. We also had to model customer service requirements, and other operational details such as driver relays and the proper handling of geography-constrained drivers (such as the Canadian and regional drivers).

Perhaps the biggest challenge was the need to make decisions now that anticipate their impact on the future. Should we send a Texas-based driver into Massachusetts? Should we send a team, which is best used on long loads since they move more miles per day, into Washington State which primarily generates short loads into California? Dispatchers clearly think about the future, and it became clear that it was not enough to optimize decisions at a point in time; we had to optimize decisions over time. If we formulated the problem as a mathematical optimization problem, we would generate a linear program with literally millions of rows (constraints), and hundreds of millions of columns (decisions to assign drivers to loads over several weeks).

Even if we could solve such a model, we would be ignoring the inherent uncertainty of
truckload operations. For our purposes, the most important source of uncertainty was the customer demands which would arise randomly over time. This complicated the problem of getting drivers home. It would be nice to send a Virginia-based driver into Chicago knowing that a load that would take him home would be waiting for him there, but this is simply not how truckload operations work. Further complicating the problem is uncertainty in travel times.

We formulated the problem as a stochastic optimization model, and used the modeling and algorithmic framework of approximate dynamic programming (ADP). ADP is a simulation-based type of optimization algorithm that uses iterative learning to optimize complex, stochastic problems. The extensive literature on approximate dynamic programming (including its sister communities which go by names such as reinforcement learning and neuro-dynamic programming) has largely ignored the challenges of high-dimensional decision variables. For this project, we had to develop methods to handle driver assignment problems with up to 60,000 variables in each time period, millions of random variables (the demands), and a state variable with a virtually infinite number of dimensions.

To use a time-worn line, necessity is the mother of invention, and this project required the development of a novel strategy for this problem. The project combined results from three Ph.D. dissertations (Spivey (2001), Marar (2002), and George (2005)) to create a model that could handle the high level of detail, while also producing decisions that balanced current and future rewards. Equally important was the need to calibrate the model against historical performance, a dimension that has been largely ignored by the academic optimization community. A technical description of the model and algorithm is given in detail in Simao et al. (2009).

The operational problem

Truckload operations sound deceptively simple. You have a set of drivers who have to be assigned to a set of loads. When a driver is assigned to a load, the driver moves to the shipper, picks up a full truckload of freight which he then delivers. Once the truck is empty
again, the trucking company has to find a new load for the driver.

In reality, truckload operations are far more complex. Drivers are described by a multidimensional attribute vector which can include elements such as current location, driver type (teams, singles, etc), equipment type (tractor and/or trailer), current destination, estimated time of arrival, home domicile, and detailed driving and working hours. Loads can be similarly described by pickup and delivery time windows, customer type, priority, and type of appointments.

When making an assignment of a driver to a load, the dispatcher has to consider factors such as: the number of miles the truck must move empty to pick up the load, the ability of the driver to deliver the load on time, the nationality of the driver, the appropriateness of the length of the load for this driver type (teams are better used on long loads), the ability of the driver to get home on time after delivering the load, and the productivity of this type of driver. Dispatchers sometimes use complex strategies such as swapping drivers between loads enroute, and relaying loads (dropping the loads at a location that is not their destination) so that a better driver can complete the move.

Dispatchers want to minimize empty miles and move loads on time, but they have other goals they also have to manage. Driver turnover is a major issue. Two areas important to drivers are (a) getting them home on time (especially over a weekend), and (b) giving each a certain number of miles to move each week so he can generate income. Schneider can impact the ability of dispatchers to get drivers home both through the choice of loads they assign drivers to (which requires thinking about the future), and by choosing where to hire drivers (if they hire too many drivers in Texas, there may not be enough loads to return drivers home to Texas without incurring excessive empty miles). Schneider was also interested in making advance commitments to drivers, and wanted to be sure that the commitments could be met at a reasonable cost.

The issue of maintaining driver productivity (measured by the average length of the loads a driver was assigned to) was more complex. Teams need more miles because the work has to pay for two people. Drivers who own their own tractors need more miles (but not as much as teams) because they have to cover lease payments on their tractors. Single drivers
with company-owned equipment need the fewest miles. If a driver in a particular fleet (team, owner-operator, company driver) gets fewer miles than he expects, he may quit (and usually does), forcing the company to hire and train a new driver. However, there are no strict rules on the length of a particular load given to any driver. A team may be given a short load which repositions the team to a location where there are other long loads. Also, moving a short load is better than nothing. What matters is the average number of miles each week. If the model deviates significantly from historical averages, then driver turnover may be higher than expected, and studies performed using the model would not be viewed as being realistic.

Getting drivers home on time and maintaining a targeted length of haul were possibly the two most important issues in the development of the model. But we also had to consider on-time service, both when picking up and delivering a load. Movements of drivers had to carefully observe regulations on the number of hours a driver could move each day, and, most notoriously, the infamous “70 hours on duty in eight days” rule, which often limited a driver to substantially less than his allowed 14 hours on duty on a day.

To produce realistic results, we had to incorporate some of the more complex operating strategies used by the dispatchers, who struggled to meet numerous goals while keeping costs low. For example, it was not possible to have a driver move 300 miles empty just to pick up a load that would take him near his home. Instead, a dispatcher might assign driver A in Baltimore to load 1 going to Dallas and driver B in Atlanta moving load 2 going to Chicago, recognizing that driver A really needs to get to his home in Chicago. It might be possible to have these drivers meet where their paths cross, swap loads, and allow driver 1 to finish moving the load to Chicago, getting home on time with little or no additional empty movement costs.

**The optimization challenge**

There are two optimization models that are widely used in freight transportation. The first is a simple assignment problem where “resources” (in our case, this would be truck drivers, but in other applications might be freight cars, locomotives or planes) are assigned to “tasks” (loads of freight), as depicted in figure 1. This model represents every driver and load individually,
making it possible to capture the attributes of each driver and load at a high level of detail. The cost on a single arc assigning a driver to a load can capture the precise number of miles that the driver has to move empty to pick up the load, along with other issues such as valuing the ability of the driver to pick up and deliver the load on time, managing a load that has to move to or from Canada (and the driver’s ability to handle such a load), and recognizing whether the load helps the driver return home. While this model makes it possible to capture a high level of detail at a point in time, it is unable to capture the impact of decisions now on the future.

The second modeling strategy, and the most common way of modeling activities into the future is to use a time-space network, as depicted in figure 2. In this model, resources are represented by their location at a point in time. This particular model assumes that all resources at the same location are identical. This model proved useful when modeling fleets of identical trailers, but in practice the equipment usually comes in different sizes (older 45 foot trailers, the more common 48 foot trailers, and 53 foot trailers) and capabilities (such as refrigeration or special shock absorbers for more delicate freight).
We can handle different types of equipment if we use a multicommodity network flow model, such as that depicted in figure 3. Here, you have to imagine that we are flowing each equipment type over its own time-space network, but where loaded movements can be shared, such as when a load may be moved by more than one equipment type (which is common). These problems become hard to solve, because we are only interested in integer solutions, which is to say that we cannot assign half of a driver to a load. Modern solvers such as CPLEX have improved dramatically in their ability to handle large integer programming problems, but problems such as these can have hundreds of thousands of integer variables. And yet, we are still not even close to modeling the real problem.

At Schneider, one of the major challenges was that we had to model drivers, not trailers. While a trailer might be characterized by its equipment type, location and time, a driver has to be represented by fifteen distinct attributes. These include: current location (if the driver is enroute, it is the location he is headed to), his estimated time of arrival (if currently moving), domicile (his home location), driver type (teams or single drivers, company-owned equipment or owner operator, and other fleets that describe drivers who move primarily within a single region), days since last visit to home, next scheduled or desired time at home, road hours (how many hours he has been driving today), duty hours (how many hours he has been on
Figure 3: A multicommodity flow problem defined over a time space network, where we have represented three different types of equipment.

If we discretize the country into 100 regions (which is common for truckload motor carriers), and if we have only one equipment type, then our time-space diagram in figure 2 would have 100 nodes per time period. If we have five types of equipment, our multicommodity network would have 500 nodes per time period. If we model individual drivers, and capture only location (100), driver type (5), home domicile (100) and days since last visit to home (up to 30), we already have 1.5 million nodes per time period. When we capture all the attributes, the number of possibilities is effectively infinite.

A reasonable question is whether we are over-dramatizing the size of this problem. After all, if our fleet has only 6,000 drivers, why would we need so many nodes? For example, in the assignment model in figure 1, we can easily handle all the attributes of each driver and load. The problem arises when we want to look into the future. For example, Schneider has to consider whether to assign a driver domiciled in Texas to a load going to Massachusetts, who
has already been on the road for 15 days (and therefore would like to get home). A dispatcher needs to think about whether Massachusetts is a good location for such a driver. Dispatchers also have to think about whether a team should be sent to St. Louis, if most of the loads out of St. Louis are relatively short. In other words, it is very important to look into the future. Furthermore, one of the major goals of the model was to capture the marginal value of drivers based on their fleet (single, team, owner-operator) and domicile. This goal requires capturing the value of a driver over a period of time.

The complexity of the problem arises when we combine the realism with which we needed to model drivers and the need to make decisions that anticipate their impact in the future. Figure 4 illustrates an initial possible assignment of five drivers to five loads. Focus now on a single driver with attribute vector \( a_3 \). This driver might be assigned to each of the five loads, creating a new driver in the future with attributes that depend on the initial attribute vector \( a_3 \), along with the characteristics of each load. To know if we should assign our driver to each of these five loads, we have to think about what this particular driver would do after completing each of these five loads (and so on and so on).

Given the complexity of the attribute vector, each of the initial five drivers would create five unique potential drivers in the future after one assignment, and for each of these, five more if we look two assignments into the future, as illustrated in figure 4. Over the course of a month, where a driver might handle three loads per week (for a total of 12 assignments over the month), we might end up creating \( 5^{12} = 2.44 \times 10^8 \) (about 240 million) potential future drivers. Now multiply this one more time by the 6,000 drivers in our fleet.

This problem is hard enough as it is, but we now have to introduce the important dimension of uncertainty. The most significant source of uncertainty in truckload trucking is the demands of the customers, which come in on a rolling basis. In addition, we have to capture randomness in travel times. It is easy to claim that as a result of the uncertainty, we do not have to look into the future, but our work demonstrated that to accurately capture the behavior of Schneider’s experienced dispatchers, it was critical to think about the downstream impact of a decision. Furthermore, we have to keep reminding ourselves that we will need the marginal value of a driver over the entire simulation.
It turns out that the presence of uncertainty guided us to an elegant and practical solution. Even small versions of this problem cannot be solved exactly using any standard stochastic optimization framework. For this reason, we turned to the modeling and algorithmic framework of approximate dynamic programming, which offers not just a rigorous mathematical foundation, but it also has the important property that it is very intuitive. We start with the simple idea of solving a sequence of simple assignment problems such as what is depicted in figure 1. After assigning drivers to loads, we simulate random travel times and new loads, advance the clock, and solve the problem again. Of course, if we only did this, we would just be simulating a simple, myopic policy which would bring us none of the important qualities we need to solve this problem.

Instead, we solve a somewhat modified assignment problem. Rather than just assigning drivers to loads, we modify the assignment problem to capture an approximate value of drivers in the future. The idea is illustrated in figure 3. Here, we start with the same assignment problem we showed in figure 1, but now we have added an approximation of the value of a driver after completing each load. So, if we are considering assigning the driver with attribute vector $a_3$ to load 1, we would obtain a driver with attribute vector $a'_3$, with approximate value $\bar{v}(a'_3)$. The value $\bar{v}(a'_3)$ is estimated by using the marginal value of drivers in the future.
This simple idea introduced some technical complications (summarized in more detail in the appendix). First, it might be the case that we never actually assigned the driver with attribute $a_3$ to load 1, which means that we never were allowed to observe a driver with attribute $a'_{31}$ in the future. Instead, we viewed the problem of estimating the value $\bar{v}(a'_{31})$ as a statistical exercise where we need an estimate $\bar{v}(a)$ for any attribute $a$. While we can present this as a simple exercise in statistical estimation, estimating these values introduced enough issues to fill a doctoral dissertation and several research papers. For example, the value of a driver at one point in time depends on the value of drivers in the future (which are themselves approximations). Also, a characteristic of our problem is that there are many drivers in some parts of the country (such as Illinois and Georgia) and relatively few drivers in other locations (such as South Dakota and Nevada). We needed methods that allowed us to take advantage of the large number of observations in the more active parts of the country, while still handling the areas where there were relatively few observations.
Estimating the value function approximations required simulating the dispatch process (using a particular set of approximations) iteratively. After each iteration (where we would simulate a month of dispatching), we would use information from each assignment problem to update the value of drivers in the future. We would capture uncertainty by sampling any random variables (such as new loads or travel times) as we simulated decisions over the month. As a result of the need to sample from random quantities, we had to use smoothing techniques to balance out the noise. Again, this apparently simple step of smoothing proved to be another difficult research challenge.

We have undertaken extensive research in the use of approximate dynamic programming for fleet management problems, where we have shown that we can obtain solutions that are near optimal when compared against the optimal solution of deterministic, single and multicommodity flow problems (see, for example, Godfrey & Powell (2002), Topaloglu & Powell (2006)). But we are not able to obtain optimal solutions (or even tight bounds) for the problem class described in this paper. Instead, we first look for evidence that the algorithm is generally producing improvements in the overall objective function, as shown in figure 6. This improvement is significant since the first iteration, where we set $\bar{v}(a) = 0$, is equivalent to a myopic policy. The more meaningful validations are (a) that we produce results that closely match historical performance, and (b) that the value functions $\bar{v}(a)$ accurately approximate the marginal value of increasing the number of drivers of a particular type.

Matching patterns

As already mentioned in the previous section, simply optimizing the objective function was not enough to obtain realistic behaviors. A major issue faced by Schneider was the need to assign drivers to loads of an appropriate length. Teams expected the longest loads, while single drivers with company-owned equipment were given the shorter loads. However, it was possible to give a team a shorter load, and while single drivers with company-owned equipment were assigned shorter loads, they still needed to get enough miles per week to maintain their income.
Figure 6: Improvement in objective function illustrating optimizing behavior.

It is not possible to solve this problem by simply putting penalties when the length of a load is higher or lower than the average for a particular type of driver. Every type of driver needs to pull loads of different lengths - it is just the averages that count. It is not a problem to assign a team to a shorter load, as long as the average length of haul matches history. If we deviated significantly from historical averages, it is likely that we would start incurring higher driver turnover. The company was not willing to experiment with dispatch rules that deviated from history.

We solved this problem by introducing the idea of a pattern metric, proposed in Marar et al. (2006) and Marar & Powell (2009). Briefly, it involves adding a term that penalizes deviations from the historical percentage of times drivers of a particular type (e.g. team vs. single) take loads of a particular length (e.g. between 500 and 550 miles). As decisions are made over time, the model keeps track of an estimate of how often we are moving loads of a particular length over the entire horizon. We only introduce penalties when the average over the entire horizon starts deviating from historical performance. This logic also requires iterative learning, and as a result it fit naturally within the iterative learning of the value functions. See section 4.1 in Simao et al. (2009) for a description of how the pattern logic was applied to this problem.
Figure 7 illustrates how the model learns to match historical performance, using (a) just value functions (without a pattern) and (b) using value functions with a pattern. Our goal was to move a particular metric (in this case, average length of haul for a particular class of driver) within a range which the company determined would be acceptable. We note with interest that simply using value functions was able to move the model within the acceptable range. Introducing the patterns moved the metric within the allowable range more quickly, and moved it closer to the center of the range. There were other statistics, however, where the pattern logic played a more significant role. We highlight these in the next section.

**Calibrating against history**

Despite an extensive academic literature on the optimization of truckload carriers, we cannot find a single instance of a model that has been shown to calibrate against actual historical performance (aside from Simao et al. (2009), on which this paper is based). The research community that works on the development of models for freight transportation has primarily
focused on optimization models where the goal is to outperform the decisions made by a company. Since our model was to be used for strategic planning purposes, it was important that it produce realistic results. We needed the power of optimization both to produce decisions which closely matched historical performance, as well as to provide a method to simulate decisions as we changed the underlying business conditions.

An open question in the research was whether we could produce realistic behaviors. In fact, there was some risk that we would outperform the company. We found, however, that the model yielded results that closely matched historical performance. The company produced statistics describing the average length of haul, revenue per tractor per week, driver utilization (a measure of the total mileage traveled by a driver on a day), and the fraction of times a driver was given time-at-home on weekends. These statistics were further divided by driver type.

Comparisons were made between the results produced by the model, and those obtained in history, shown in figure 8. For each statistic, the company provided a range based on the variability they observed on a month to month basis. As depicted by the figure, for each statistic we were able to calibrate the model to match history.

Capturing the marginal value of drivers

One of the applications of the model was guiding Schneider in the hiring of new drivers. We needed the marginal value of hiring, for example, teams who live in northern Illinois. This marginal value would reflect not only the revenues of the loads that these drivers can move, but also the cost of getting them home. It is important to emphasize that the marginal value of these drivers is very different from the average value of drivers with these attributes, a quantity that can be easily calculated just by tracking the paths of these drivers over the planning horizon. The marginal value requires that we understand how the solution would change if we added additional drivers of a particular type.

We compared the marginal value of a driver characterized by domicile and driver type as estimated by the value function approximations against the estimates produced by adding 10
Figure 8: System results compared against historical extremes for length of haul (LOH), revenue per working unit (WU), driver utilization, and percentage of driver time-at-home (TAH) spent on weekends (from Simao et al. [2009]).

drivers of a particular type and rerunning the system for several iterations. Given the noise in the estimates obtained by adding 10 drivers and rerunning the system, we repeated this exercise 10 times for each type of driver to obtain confidence intervals (in other words, we performed a simulation).

The results are shown in figure 8. We note that for the 20 different estimates (representing different combinations of driver domiciles and driver types), in 18 instances the value produced by the value function approximation fell within the 95 percent confidence interval from the simulation. We view this as a validation that the value function approximations are consistent with the estimates of the marginal values produced through simulation. However, from a single calibration run of the system, we obtain estimates $\bar{v}_0(a)$ for all possible combinations of driver
Figure 9: Simulated value of additional drivers compared to estimates based on value function approximations, for 20 different types of drivers (from Simao et al. (2009)).

domiciles with driver types at the beginning of the simulation. That is, $\bar{v}_0(a)$ is an estimate of the value of a driver with attribute $a$ at time 0, which provides an estimate of how the entire simulation should change if we added one more driver with attribute $a$ at the beginning of the simulation. We have to compute $\bar{v}_t(a)$ for all attributes (location, driver type, domicile) and all time periods as part of the approximate dynamic programming algorithm, though for driver valuations, we only use the estimates at the beginning of the simulation.

**Having an impact**

The tactical planning system (TPS, as it is known within the company) has been and continues to be used for a variety of analyses that lead to significant financial benefits through operational policy/procedure improvements, better informed negotiating positions, and cost reduction or avoidance. The principal benefit of this system over traditional aggregated-flow network models that we have used in the past has been the capability to capture driver and load attributes in great detail, producing a very realistic simulation of real-world operations. The particular strength of this modeling platform is that the system not only produces good (near-optimal) solutions to complex problem scenarios, but also provides comprehensive operating
characteristics and statistics which can be used to determine potential impacts and to uncover unintended consequences associated with proposed changes within a complex transportation network. In a statement issued by the company: “We firmly believe that the successful studies described below could not be achieved with any other modeling methodologies of which we are aware.” And this is coming from a company that has won the INFORMS prize for widespread use of operations research.

The following are brief summaries of several specific analyses with corresponding business benefits which have been carried out with TPS in the last several years.

- **Driver Time at Home (TAH)** - Over-the-road drivers are typically away from home for 2-3 weeks. To address driver retention, a business plan was approved to significantly increase the amount of time drivers spend at home, but through TPS runs the plan was found to have a $30M/yr potential negative impact on network operating costs, considerably outweighing the anticipated benefits. Using TPS, an alternative strategy was developed which provided 93 percent of the proposed self-scheduling flexibility while incurring an estimated cost impact of $6M/yr.

- **Driver Hours of Service (HOS) rules** - Over the last six years, the U.S. Department of Transportation has introduced several changes for driver work schedule constraints. Using TPS runs, Schneider was able to substantiate and quantify these impacts, allowing the company to effectively negotiate adjustments in customer billing rates and freight tendering/handling procedures, leading to margin improvements of 2-3 percent.

- **Setting appointments** - A key challenge in the order booking process is determining both the timing and flexibility of the pickup and delivery appointments. Using TPS, Schneider was able to quantify the impacts of different types of commitments, allowing it to identify the best choices. This produced margin impacts in the range of 4-10 percent with reduction of late deliveries exceeding 50 percent.

- **Cross Border Relay Network** - The Schneider freight network includes a large number of loads which move between the U.S. and Canada. Using TPS runs, Schneider was able to design a strategy that accomplished cross-border operations using only Canadian
drivers. This reduced the number of drivers engaged in border crossing by 91 percent, resulting in cost avoidance of $3.8M in training/identification/certification, and annual cost savings of $2.3M.

- Driver Domiciles - Schneider manages over 6,000 drivers who operate the long-haul network, which requires that drivers be away from home for weeks at a time. Getting these drivers home requires sending drivers to regions where there is a good likelihood of getting drivers home on time. As a byproduct of the approximate dynamic programming methodology, TPS provides an estimate of the marginal value of drivers for each home domicile. Schneider uses these estimates to guide its hiring strategy, leading to an estimated annual profit improvement of $5M.

- One of Schneider’s largest accounts asked for tighter time windows on delivered freight, covering 4,500 loads per month. Schneider used TPS to show that it would cost approximately $1.9M per year to meet this demand, and as a result the customer withdrew the request.

Appendix

The model

We model the attributes of a driver at time $t$ using the attribute vector $a_t$, and we similarly let $b$ be the attributes of a load. Also let $\tau$ be a random variable denoting the time to complete a task. We capture the complex dynamics of assigning a driver to a load using a function we refer to as the attribute transition function, represented by

$$a_{t+1} = a^M(a_t, b, \tau_{t+1}).$$

Here, $t$ represents a point in time where we make decisions. Although we model all arrivals and departures in continuous time, we approximate decisions as being made in discrete blocks of time (say, twelve-hour increments). $a_t$ is the attributes of a driver at time $t$, $b$ is the attributes of some load, and $\tau_{t+1}$ captures information about travel delays that is only learned between $t$
and \( t + 1 \). We note that we might decide at time \( t = 24 \) to assign a driver that is expected to arrive at time \( t = 25.3 \) to a load that needs to be picked up some time after \( t = 26 \) (implying the driver might have to wait), where, at time \( t = 24 \), we expect the driver to deliver the load at time \( t = 38.7 \) (this would be the expected time of arrival). By time \( t = 36 \), we might learn that the driver actually got delayed picking up the load. So, at time \( t = 24 \), \( a_{t+1} = a_{36} \) (we view \( t + 1 \) as being one time step into the future, translating to hour 36) is a random variable. Buried in the function \( a^M(a,b,\tau) \) is logic such as “if the driver runs out of hours at 11pm, insert 60 minutes of delay so that he continues driving after midnight when his clock resets.”

We capture all the drivers using the resource state vector defined by

\[
R_{ta} = \text{the number of drivers with attribute } a \text{ at time } t, \\
R_t = (R_{ta})_{a \in \mathcal{A}}.
\]

Here, \( \mathcal{A} \) is the set of all possible attribute vectors. For our problem, the set \( \mathcal{A} \) is huge (effectively infinite), so this is something we have to manage carefully. We model loads in a similar way by letting \( D_{tb} \) be the number of loads (demands) with attribute \( b \in \mathcal{B} \) at time \( t \), and let \( D_t = (D_{tb})_{b \in \mathcal{B}} \) be the vector of all the loads. The state of our system is given by \( S_t = (R_t, D_t) \).

We model our decisions using

\[
x_{tab} = \text{the number of drivers with attribute } a \text{ that we assign to loads with attribute } b \text{ at time } t, \\
x_t = (x_{tab})_{a \in \mathcal{A}, b \in \mathcal{B}}.
\]

We evaluate the contribution of an assignment using a mixture of hard dollars (the revenue generated by a load minus the cost of moving empty to pick up a load and the cost of actually moving the load) and soft bonuses and penalties. Let

\[
c_{ab}^h = \text{the hard dollar contribution of assigning a driver with attribute } a \text{ to a load with attribute } b, \\
\theta = \text{a vector of bonuses and penalties that are used to produce specific model behaviors,} \\
c_{ab}^s(\theta) = \text{the soft dollar contribution of assigning a driver with attribute } a \text{ to a load with attribute } b.
\]
\(c_{ab}(\theta)\) is controlled by the vector of parameters \(\theta\) that include bonuses for getting a driver home on time, and penalties for picking up or delivering a load in violation of the service commitment. The total contribution is given by

\[
C_t(S_t, x_t|\theta) = \sum_a \sum_b \left( c_{ab}^h + c_{ab}^s(\theta) \right) x_{tab}.
\]

In addition to the random travel times, we let \(\hat{R}_{t+1,a}\) be exogenous changes to the number of drivers with attribute \(a\) due to new information that arrived between \(t\) and \(t + 1\). \(\hat{R}_{t+1}\) can be used to model both transit delays, as well as equipment failures, drivers leaving the fleet and new drivers being hired. \(\hat{D}_{t+1,b}\) captures the number of new customer demands of type \(b\) that we learned about between \(t\) and \(t + 1\). We let \(W_{t+1} = (\hat{R}_{t+1}, \hat{D}_{t+1})\) be the vector of all the new information arriving between \(t\) and \(t + 1\). We represent the evolution of our system over time using a transition function that gives us

\[
S_{t+1} = S_M(S_t, x_t, W_{t+1}).
\]

\(S_M(\cdot)\) captures all the equations needed to update our system (this function uses the attribute transition function to determine the attributes of drivers in the future). This function handles loads being served, new loads arriving, and the change in the status of each driver after it is assigned to a load.

The final challenge involves actually making decisions. For the moment, we assume that we have some function \(X^\pi(S_t)\) that determines which driver should be assigned to each load, which drivers should sit idle, which should move empty, and which loads should be deferred or ignored. The function \(X^\pi(S_t)\) is often referred to as a policy. We view the function as being determined by a vector of parameters (such as \(\theta\)) which influences the behavior of the model. We let \(\Pi\) be the set of all possible values of these parameters, so that choosing \(\pi \in \Pi\) determines the policy. Our optimization challenge over a finite planning horizon \(T\), then, involves solving the problem

\[
\max_{\pi \in \Pi} F^\pi(\theta) = \mathbb{E} \sum_{t=0}^{T} C_t(S_t, X^\pi(S_t)|\theta).
\]  

(1)
This means that we are trying to find a policy (decision function) that maximizes the total contribution over our planning horizon (typically one month). Since there are uncertain elements, we want to make decisions that maximize some approximation of the expected value of this contribution. A central thesis of our approach is that solving this optimization problem would produce behaviors that closely match the historical performance of the company. In order to achieve that, the company designed a series of goals $\bar{g}_i, \ i \in I$ which capture metrics such as: average miles per load for different types of drivers, percent of time we get drivers home on time, on-time service and empty miles as a percent of total miles. We then let $g^\pi_i(\theta), \ i \in I$ be the same metric produced by our model using policy $\pi$. We measure how well we are matching historical performance using

$$H^\pi(\theta) = \sum_{i \in I} \beta_i (g^\pi_i(\theta) - \bar{g}_i)^2.$$ 

where $\beta_i$ determines the importance of the $i^{th}$ metric. Our strategy was to use optimization algorithms to find the best policy $\pi$, but we also have to separately tune the parameter vector $\theta$ to get the right performance.

The approximate dynamic programming algorithm

We can characterize an optimal policy for solving (1) using Bellman’s equation, which says that for each state $S_t$, we would choose an action by solving

$$V_t(S_t) = \max_{x_t} \left( C_t(S_t, x_t|\theta) + \gamma \mathbb{E} \left\{ V_{t+1}(S_{t+1}) | S_t \right\} \right).$$ (2)

The problem with Bellman’s equation is that it suffers from what has been called the three curses of dimensionality (Powell (2007)): the state variable $S_t$, the random variables in $W_t$ (which means that we cannot compute the expectation), and the decision $x_t$.

To obtain a good solution, we turn to the algorithmic framework of approximate dynamic programming. For a complete development of the ADP algorithm, we refer the reader to Simao et al. (2009). In a nutshell, this method requires estimating the marginal value of a driver after completing a load, based on what we know before we assign the driver to the load.
This value is then added to the contribution if we assign the driver to the load, so that we are now balancing the immediate contribution of an assignment with the downstream value of the resulting driver attributes in the future. Let $\bar{v}_t(a')$ be an approximation of the value of a driver with attribute $a'$, at time $t$. The attribute vector $a' = a^M(a, b, \bar{\tau}_t)$ is the future attribute of a driver with attribute $a$ if he is assigned to a load with attribute $b$, where $\bar{\tau}_t$ is the expected travel time based on what we know at time $t$.

Using approximate dynamic programming, we now have to solve a problem of the form

$$X^n_t(S_t) = \underset{x_t}{\arg \max} \sum_a \sum_b (c^h_{ab} + c^s_{ab}(\theta) + \bar{v}_t(a^M(a, b, \bar{\tau}_t)))x_{tab}$$

(3)

So, we are simply adding $\bar{v}_t(a')$ to the contribution of an assignment. This means that the problem of determining which driver to assign to each load at a point in time is no more difficult than it was with a myopic policy.

The challenge we face when using approximate dynamic programming is that we have to estimate the values $\bar{v}_t(a')$. We do this by taking advantage of the fact that our dispatch problem in equation (3) is a linear program (actually, a simple network assignment model), which has to be solved subject to the flow conservation constraints

$$\sum_b x_{tab} = R_{ta}.$$  

(4)

We note that equation (3) does not have an expectation as is the case in equation (2). We accomplish this by using the idea of the post-decision state (see Powell (2007), chapter 4, for a general discussion of this concept, and Simao et al. (2009) for the details of how we implement this idea for this project). In a nutshell, we let $a_t$ be the attribute of a driver at time $t$ just before he is dispatched, and let $a^x_t = a^M(a_t, b, \bar{\tau}_t)$ be the attributes we expect the driver to have after being assigned to a load with attribute $b$. We then let $\bar{v}_t^{n-1}(a)$ be our estimate of $\bar{v}_t(a)$ after $n - 1$ iterations. Assume that a driver goes through attributes $a_{t-1}, a^x_{t-1}, a_t, a^x_t$ where $a_{t-1}$ and $a_t$ are the attributes just before a driver is assigned to loads at time $t - 1$ and $t$, while $a^x_{t-1}$ and $a^x_t$ are the attributes that we expect the driver to have after completing these assignments, but before the driver has actually begun the assignment (which means that we
do not know about travel delays or the availability of new loads in the future). \(a_t^x\) is known as the *post-decision state variable* while \(a_t\) is the pre-decision state variable for a driver.

Let \(\hat{v}_n^{a_t}\) be the dual variable for (4) in the \(n^{th}\) iteration of the algorithm, giving an estimate of the marginal value of a driver with attribute \(a_t\). We then update \(\bar{v}_{t-1}^{n-1}(a)\) using

\[
\bar{v}_{t-1}^{n-1}(a_{t-1}) = (1 - \alpha_{n-1})\bar{v}_{t-1}^{n-1}(a_{t-1}) + \alpha_{n-1}\hat{v}_t^{a_t}
\]

where \(\alpha_{n-1}\) is a stepsize between 0 and 1. So, in equation (5), we are using the value of a driver, \(\hat{v}_t^{a_t}\), with attribute \(a_t\) (just before he is assigned to a load) to update his attributes evaluated at \(a_{t-1}^x\), which is what we thought \(a_t\) would be at the time that we assigned the driver at time \(t - 1\).

There were a number of technical hurdles that we had to overcome as we developed an ADP-based strategy for this problem. First, we took advantage of the fact that we did not literally have to estimate \(\bar{v}_t(a)\) for a fifteen-dimensional attribute vector. Instead, for the purpose of approximating the value of a driver in the future, we only required three attributes (in addition to time): the location of the driver, his domicile and his driver type. The other attributes (such as how many hours he was driving on each of the last eight days) were needed to compute what the driver could do moving forward, but were not felt to affect the value of a driver for the purpose of making dispatch decisions (more precisely, we simply did not have enough observations to estimate the contributions of these other attributes). Location and domicile were represented by dividing the U.S. into 100 regions. For a problem with 60 time periods (two dispatch decision periods per day over 30 days) and 50,000 attribute vectors \((100 \times 100 \times 5)\), it meant that we still had to estimate three million values.

We solved this estimation problem by developing (specifically for this project) a hierarchical aggregation strategy (George et al. (2008)). In this method, we estimated the value of a driver at a location (producing 100 values per time period), the value of a driver with a location-driver type pair (producing 500 values per time period), and the value of driver location, driver type, and driver domicile (50,000 values per time period). These estimates, which were
denoted \( \bar{v}^{(2)} \), \( \bar{v}^{(1)} \) and \( \bar{v}^{(0)} \), respectively, were combined into a single estimate using

\[
\bar{v}^n(a) = \sum_{g \in \mathcal{G}} w_a^{(g,n)} \bar{v}^{(g,n)}(a).
\]

where \( w_a^{(g,n)} \) is the weight given to attribute \( a \) at level of aggregation \( g \) after \( n \) observations. The weights are computed using

\[
w_a^{(g,n)} \propto (\bar{\sigma}^2_a)^{(g,n)} + (\bar{\mu}_a^{(g,n)})^2
\]

where the weights are normalized (for each attribute \( a \)) so they sum to 1. Here, \( (\bar{\sigma}^2_a)^{(g,n)} \) is the variance of the estimate \( \bar{v}^{(g,n)}(a) \), which declines to zero as the number of observations grows, and \( \bar{\mu}_a^{(g,n)} = \bar{v}^{(g,n)}(a) - \bar{v}^{(0,n)}(a) \) is an estimate of the bias relative to the most disaggregate estimate. The bias generally does not go to zero as the number of observations grows. There are many attributes which receive few observations, and for these we depend on estimates at more aggregate levels. A detailed description of the method is given in George et al. (2008).

The second problem was the design of a stepsize rule to determine \( \alpha_n \). The best stepsize has to strike a balance between the rate at which \( \bar{v}^{n-1}_t(a) \) grows over the iterations (since it is estimating the value of a driver over his entire future), and the noise in the updates \( \hat{v}_t^a \). Not surprisingly, this varied significantly over the attributes. As we were developing this model, the choice of an appropriate stepsize rule proved to be particularly frustrating. We undertook the development of a new stepsize rule which optimally balances errors due to bias with errors due to noise. This stepsize rule (also published in a leading machine learning journal, see George & Powell (2006)) is given by

\[
\alpha_n = 1 - \frac{\sigma^2}{(1 + \lambda^{n-1}) \sigma^2 + (\beta^n)^2},
\]

where \( \sigma^2 \) is an estimate of the pure variance and \( \beta^n \) is an estimate of the bias between \( \bar{v}^{n-1}_t(a) \) and what should be the true value. This stepsize rule, dubbed the bias adjusted Kalman filter (BAKF) rule in Powell (2007) (to reflect its relationship to the Kalman filter gain rate), produces a stepsize that ranges between \( 1/n \) when the bias is zero or the variance is very high, and 1 as the variance drops to zero (relative to the bias). We use a different stepsize for each
\( v_{t}^{n-1}(a) \). We found that this stepsize rule produced more rapid convergence, and eliminated the need to tune parameters for more heuristic rules.

References


