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

August 23, 2009

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 captured 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 capturing the marginal value of drivers and loads.

Schneider National is one of the three largest truckload motor carriers in the U.S., 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 help 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 match 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 our biggest challenge was the need to make decisions now that anticipated 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 thought 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 a driver to a load 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 50,000 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). This article provides a high level summary of the project, including only the modeling and algorithmic details that are required to appreciate the original methodological contributions that were needed for the project.

1 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 them a certain number of miles each week so they 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). 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 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.

2 The optimization model

By far the most common optimization model used in freight transportation, and one that is still widely used in industry even to model truckload operations, is a multicommodity flow formulation where x^k_{tij} might be the flow of trailers (drivers) of type k from city i to city j, departing at time t. We might let D^ℓ_{tij} be the number of loads of type ℓ moving from i to j, departing at time t. We could let r^ℓ_{ij} be the reward for moving loads of type ℓ from i to j, and let $c_{k\ell}$ be the "cost" of assigning drivers of type k to loads of type ℓ. If an assignment is considered infeasible (for example, assigning a 40' trailer to a load that requires a 53' trailer, or assigning a load that needs refrigeration to an unrefrigerated trailer) we will associate a very high cost to this assignment. A model based on this formulation produces a very large integer multicommodity flow problem, which still remains the subject of active research. Moreover, this model does not even come close to capturing the operational details needed to produce a realistic model of truckload operations.

Instead, we let a be a vector of 15 attributes describing a driver that includes: 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 away from home, scheduled time at home, road hours (how many hours he has been driving today), duty hours (how many hours he has been on duty today), and his duty hours over each of the previous seven days (to enforce the 70-hours-in-eight-days rule). Loads also have a complex vector of attributes, designated by b, that capture pickup and delivery windows, nature of the pickup and delivery appointments, and the priority of the shipper. We note that the estimated time of arrival of a driver, and the pickup and delivery windows of loads, could not be represented using a coarse discretization of time; these were all modeled down to the minute.

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 τ_{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

$$\begin{aligned} R_{ta} &= \text{the number of drivers with attribute } a \text{ at time } t, \\ R_t &= (R_{ta})_{a \in \mathcal{A}}. \end{aligned}$$

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

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

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^h_{ab}	=	the hard dollar contribution of assigning a driver with attribute a to a load with attribute b,
θ	=	a vector of bonuses and penalties that are used to produce specific model behaviors,
$c^s_{ab}(\theta)$	=	the soft dollar contribution of assigning a driver with attribute a to a load with attribute b.

$c^s_{ab}(\theta)$ is controlled by the vector of parameters θ 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

$$C_t(S_t, x_t|\theta) = \sum_a \sum_b (c^h_{ab} + c^s_{ab}(\theta))x_{tab}.$$

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 θ) which influences the behavior of the model. We let Π 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

$$\max_{\pi \in \Pi} F^\pi(\theta) = \mathbb{E} \sum_{t=0}^{T} C_t(S_t, X^\pi(S_t)|\theta). \tag{1}$$

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 $\bar{g}_i,\ i \in \mathcal{I}$ 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 $g_i^\pi(\theta),\ i \in \mathcal{I}$ be the same metric produced by our model using policy π. We measure how well we are matching historical performance using

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

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

3 Optimizing using approximate dynamic programming

It is tempting to solve the problem of assigning drivers to loads using a simple myopic policy which ignores the impact of decisions now on the future. This means that our dispatch policy $X^\pi(S_t)$ would be given by

$$X^\pi(S_t) = \arg\max_{x_t} C_t(S_t, x_t|\theta). \tag{2}$$

If we used this strategy, we would ignore the impact of putting a Texas-based team on a load that terminates in Massachusetts. Massachusetts might be a good source of long loads, or it may not have any loads that would get a driver back to Texas. We may not be concerned about getting a driver home if he is not scheduled to be home for a week or more, but if we need to get him home in three days, this is a problem. In fact, we tried this policy, and simply could not get the model to produce realistic results. Not only does a myopic policy do a poor job of solving the optimization problem in equation (1), it does a poor job of producing behaviors that match those of the company.

We had to address the problem of not just optimizing the system at a point in time (this is what the policy in equation (2) does), we had to optimize over time. That is, we needed a policy that did a good job of solving the objective function in (1). This is exceptionally hard even if the problem were deterministic. The problem has to do with the complexity of drivers, since there will almost never be two drivers with the same attribute vector. To illustrate the challenge, assume that we only consider assigning a driver to at most 10 loads (out of the several thousand available at each point in time). To determine which of these 10 loads we should assign a driver to, we have to consider the value of this driver in the future after completing each load. To do this, we have to again consider assigning each of those future drivers to 10 more loads. If we look five loads into the future, we have 10^5 driver trajectories, and we have to do this for perhaps 2,000 drivers that are available to be assigned at each point in time (out of the 6,000 drivers that we are modeling). Finally, we have to consider all of these trajectories simultaneously, since each load can be moved at most once. So, even if we only look five loads into the future, we have 20 million trajectories that we have to optimize over. And this ignores the uncertainty in future demands.

There was another problem with using a myopic or rolling horizon policy. A major goal of the project was to determine the value of drivers based on domicile and driver type. For example, should they hire more teams based in Illinois? If R_{0a} is the number of drivers with a particular attribute vector a at time $t = 0$ (which captures the initial fleet), we would like to know the marginal value of adding a driver with attribute a. We were only interested in domicile (aggregated to the level of 100 regions) and driver type (for our initial study, there were five driver types), but this means we were interested in the marginal value of 500 types of drivers. We could run a simulation where we add, say, 10 teams in Texas and rerun the system, but this means we have to run 500 simulations to get these numbers.

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

$$V_t(S_t) = \max_{x_t} \big(C_t(S_t, x_t|\theta) + \gamma \mathbb{E}\left\{ V_{t+1}(S_{t+1})|S_t \right\} \big). \tag{3}$$

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 driver in the future. Let $\bar{v}_{ta'}$ be an approximation of the value of a driver with attribute a'. 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

$$X^\pi_t(S_t) = \arg\max_{x_t} \sum_a \sum_b (c^h_{ab} + c^s_{ab}(\theta) + \bar{v}_{t,a^M(a,b,\bar{\tau}_t)}) x_{tab} \tag{4}$$

So, we are simply adding $\bar{v}_{ta'}$ 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 $\bar{v}_{ta'}$. We do this by taking advantage of the fact that our dispatch problem in equation (4) is a linear program (actually, a simple network assignment model), which has to be solved subject to the flow conservation constraints

$$\sum_b x_{tab} = R_{ta}. \tag{5}$$

We note that equation (4) does not have an expectation as is the case in equation (3). 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 $a_t^x = a^M(a_t, b, \bar{\tau}_t)$ be the attributes we *expect* the driver to have after being assigned to a load with attribute b. We then let $\bar{v}_{ta}^{n-1}$ be our estimate of $\bar{v}_{ta}$ after $n-1$ iterations. Assume that a driver goes through attributes $a_{t-1}, a_{t-1}^x, a_t, a_t^x$ 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_{t-1}^x and a_t^x 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 $\hat{v}_{ta}^n$ be the dual variable for (5) in the n^{th} iteration of the algorithm, giving an estimate of the marginal value of a driver with attribute a. We then update $\bar{v}_{ta}^{n-1}$ using

$$\bar{v}_{t-1,a_{t-1}^x}^n = (1-\alpha_{n-1})\bar{v}_{t-1,a_{t-1}^x}^{n-1} + \alpha_{n-1}\hat{v}_{ta_t}^n \tag{6}$$

where α_{n-1} is a stepsize between 0 and 1. So, in equation (6), we are using the value of a driver, $\hat{v}_{ta_t}^n$, 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 $\bar{v}_{ta}$ 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). His 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 attributes ($100 \times 100 \times 5$), it meant that we still had to estimate three million values.

Even this reduced problem proved to be very difficult. We solved this problem by developing (specifically for this project) a hierarchical aggregation strategy which was later published in a leading machine learning journal (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-driver domicile (50,000 attributes 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

$$\bar{v}^n_a = \sum_{g \in \mathcal{G}} w^{(g,n)}_a \bar{v}^{(g,n)}_a .$$

where $w^{(g,n)}_a$ is the weight given to attribute a at the three levels of aggregation after n observations. The weights are computing using

$$w^{(g,n)}_a \propto (\bar{\sigma}^2_a)^{(g,n)} + \left(\bar{\mu}^{(g,n)}_a\right)^2$$

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 $\bar{v}^n_a$ which declines to zero as the number of observations grows. $\bar{\mu}^{(g,n)}_a = \bar{v}^{(g,n)}_a - \bar{v}^{(0,n)}_a$ 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 α_n. The best stepsize has to strike a balance between the rate at which $\bar{v}_a^n$ grows over the iterations (since it is estimating the value of a driver over his entire future), and the noise in the updates $\hat{v}_{ta}^n$. Not surprisingly, this varied significantly over the attributes. As we were developing this model, the design 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 and errors due to noise. This stepsize rule (also published in a leading machine learning journal, see George & Powell (2006)) is given by

$$\alpha_n = 1 - \frac{\sigma^2}{\left(1 + \lambda^{n-1}\right)\sigma^2 + (\beta^n)^2}, \tag{7}$$

where σ^2 is an estimate of the pure variance and β^n is an estimate of the bias between $\bar{v}_{ta}^{n-1}$ 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 $\bar{v}_{ta}^n$. We found that this stepsize rule produced more rapid convergence, and eliminated the need to tune parameters for more heuristic rules.

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 1. This improvement is significant since the first iteration, where we set $\bar{v}_{ta} = 0$, is equivalent to a myopic policy. The more meaningful validations are (a) that we produce results that closely match historical performance (shown in section 5), and (b) that the value functions $\bar{v}_{ta}$ accurately approximate the marginal value of increasing the number of drivers of a particular type (shown in section 6).

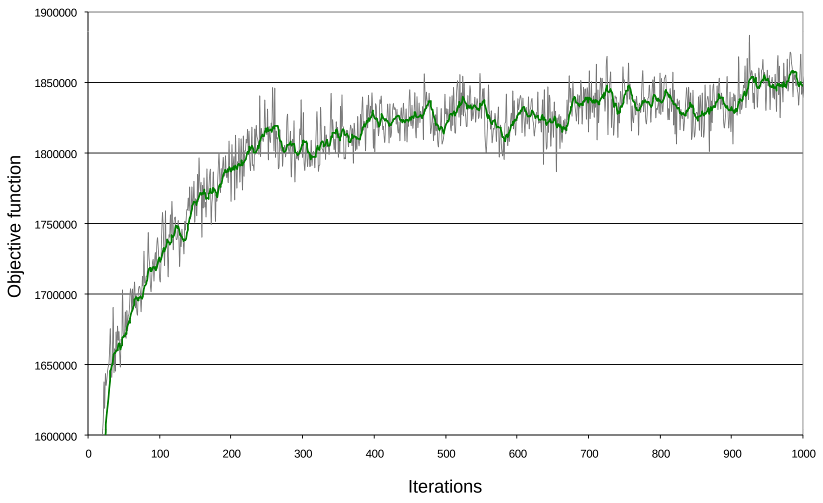


Figure 1: Improvement in objective function illustrating optimizing behavior.

4 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.

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 matched 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

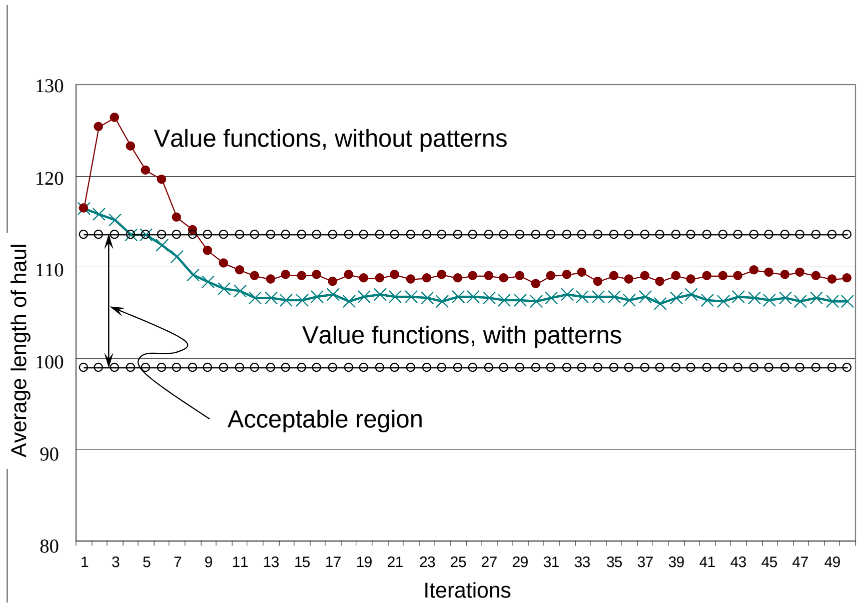


Figure 2: Effect of value functions and patterns on matching historical performance.

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 2 illustrates how the model learns to match historical patterns, 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 those in the next section.

5 Calibrating against history

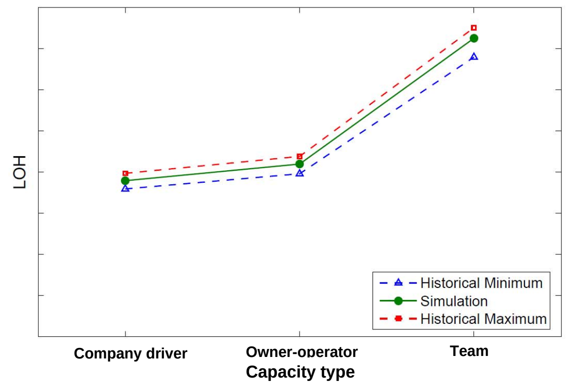
Despite an extensive academic literature on the optimization of truckload carriers, we cannot find a single instance of a model that has been shown to calibrate against actual historical performance (aside from Simao et al. (2009), on which this paper is based). The research community that works on the 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, and the fraction of times a driver was given time-at-home on schedule. These statistics were further divided by driver type.

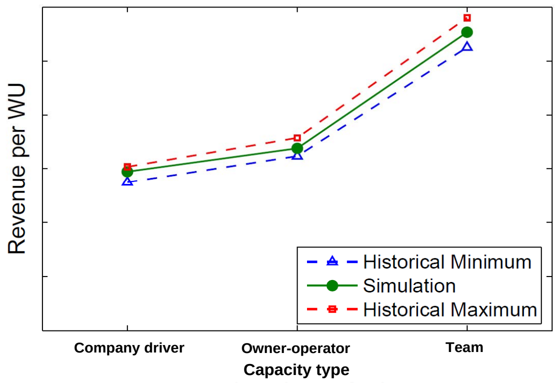
Comparisons were made between the results produced by the model, and those obtained in history, shown in figure 3. For each statistic, the company provided a range based on the variability they observed on a month to month basis. For each statistic, we were able to calibrate the model to match history. This required manually tuning the parameter vector θ, which is a fairly tedious process but nonetheless feasible.

6 Capturing the marginal value of drivers

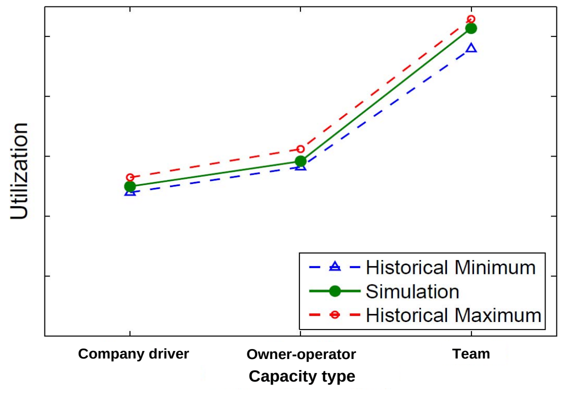
One of the applications of the model was guiding Schneider in the hiring of new drivers. For this application, we needed the marginal value of hiring, for example, teams who live in northern Illinois. This marginal value would reflect the cost of getting drivers home. It is important to emphasize that the marginal value of these drivers is very different than the



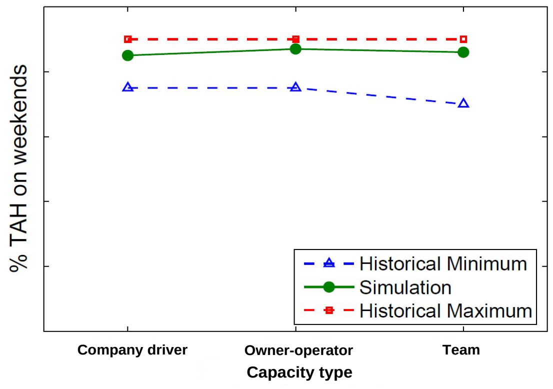
3a



3b



3c



3d

Figure 3: 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)).

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 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).

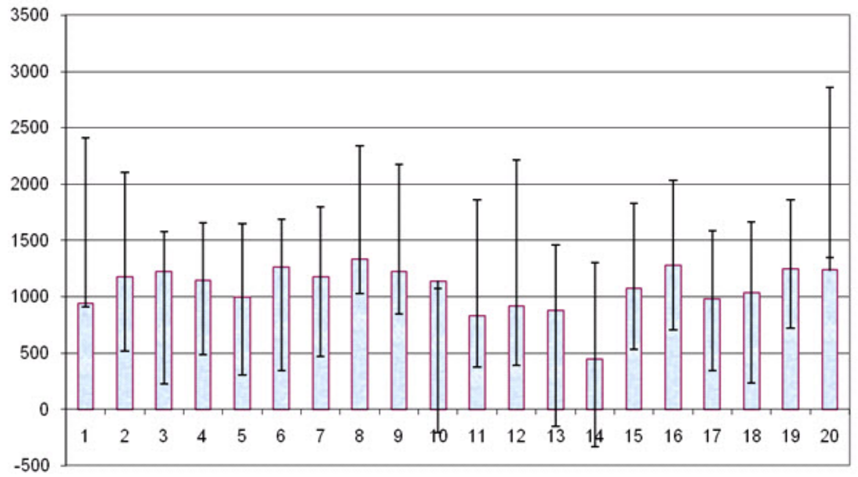


Figure 4: Simulated value of additional drivers compared to estimates based on value function approximations, for 20 different types of drivers (from Simao et al. (2009)).

The results are shown in figure 4. 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 estimate of the marginal values produced through simulation. However, from a single calibration run of the system, we obtain estimates $\bar{v}^n_{0a}$ for all combinations of the one hundred driver domiciles with the five driver types at the beginning of the simulation. That is, $\bar{v}^n_{0a}$ 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 $\bar{v}^n_{ta}$ for all attributes (location, driver type, domicile) and all time periods as part of the approximate dynamic programming algorithm. For driver valuations, we only use the estimates at the beginning of the simulation.

7 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.”

The following are brief summaries of several specific analyses with corresponding business benefits which have been carried out with the 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. Dept of Transportation has introduced several changes for driver work schedule constraints. Using TPS runs, Schneider was able to substantiate and quantify these impacts, allowing us 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 us 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 considerable 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 freight delivered by Schneider, 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.

References

George, A. (2005), Optimal Learning Strategies for Multi-Attribute Resource Allocation Problems, PhD thesis, Princeton University. 2

George, A. & Powell, W. B. (2006), 'Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming', *Machine Learning* **65**(1), 167–198. 13

George, A., Powell, W. B. & Kulkarni, S. (2008), 'Value function approximation using multiple aggregation for multiattribute resource management', *J. Machine Learning Research* pp. 2079–2111. 12

Godfrey, G. & Powell, W. B. (2002), 'An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times', *Transportation Science* **36**(1), 21–39. 13

Marar, A. (2002), Information Representation in Large-Scale Resource Allocation Problems: Theory, Algorithms and Applications, PhD thesis, Princeton University. 2

Marar, A. & Powell, W. B. (2009), 'Capturing incomplete information in resource allocation problems through numerical patterns', *European Journal of Operational Research* **197**(1), 55–58. 14

Marar, A., Powell, W. B. & Kulkarni, S. (2006), 'Capturing expert knowledge in resource allocation problems through low-dimensional patterns', *IIE Transactions* **38**(2), 159–172. 14

Powell, W. B. (2007), *Approximate Dynamic Programming: Solving the curses of dimensionality*, John Wiley and Sons, New York. 10, 11, 13

Simao, H. P., Day, J., George, A. P., Gifford, T., Nienow, J. & Powell, W. B. (2009), 'An approximate dynamic programming algorithm for large-scale fleet management: A case application', *Transportation Science* **43**(2), 178–197. 2, 10, 11, 15, 16, 17, 18

Spivey, M. J. (2001), The dynamic assignment problem, PhD thesis, Princeton University. 2

Topaloglu, H. & Powell, W. B. (2006), 'Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems', *Informs Journal on Computing* **18**(1), 31–42. 13