Optimal Dynamics

Sequential Decision Analytics for the Truckload Industry

December 14, 2020

Warren B. Powell
Chief Analytics Officer, Optimal Dynamics

Overview

“Artificial intelligence” has attracted considerable attention over the past decade, but the focus has been primarily on advances in **machine learning**, a field with a history that dates back to the early 1900s. Machine learning develops tools for estimating or predicting values we do not know, using information we do know.

Optimal Dynamics is moving artificial intelligence to the next level: making decisions. Decisions represent quantities or parameters that we control, such as the number of dedicated drivers, the price to bid in a lane, whether to accept an offered load at a specified price, or which driver to use now to move a load. The science of making decisions has typically fallen under the umbrella of **optimization**, which dates its roots to the late 1940s when linear programming first emerged. However, the major successes of this field have been limited to ***deterministic problems*****,** which describes problems where all information is known perfectly.

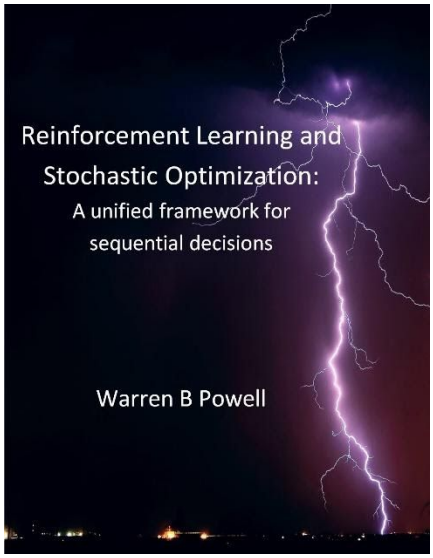
Deterministic optimization simply does not describe either the trucking industry, or any problem in the entire field of logistics. Uncertainty is as omnipresent in freight transportation and logistics as it is on Wall St., with the exception that transportation and logistics combines uncertainty with complex, high-dimensional physical processes. However, while the fields of machine learning and deterministic optimization are mature, with established textbooks, widely available (and often free) software libraries, and large communities of people trained in these tools, the same is not true when we combine decisions and uncertainty.

Our goal is to choose decisions that optimize a combination of metrics that might include operating costs, operating profit, customer service, and time-at-home for drivers, all in the presence of different forms of uncertainty such as customer requests, travel delays, market responses, driver behavior and equipment failures. Machine learning might use a historical dataset to estimate a ***statistical model*** to predict the decision a human would make. Optimal Dynamics uses advanced tools to design a ***policy*** for making decisions that outperform what a human can achieve. Below we bring out the similarities and differences between statistical models and policies.

The fields of statistical machine learning and deterministic optimization are well established, with a number of popular textbooks that present roughly the same material, supported by software that is often free, or available from commercial vendors. The study of making decisions under uncertainty, broadly known as “stochastic optimization,” also dates its modern roots to the 1950’s, but instead of evolving as a single, well-defined field, stochastic optimization is an umbrella for a wide range of fields. The graphic below shows the front covers of books spanning over 15 different research communities, using eight distinct modeling frameworks (with their own notation) and an array of

approaches for solving these problems. For years I have been referring to these communities as the “jungle of stochastic optimization.”

Building on a career of research working on a wide range of problems in freight transportation and logistics, as well as a variety of other applications (energy, finance, e-commerce, health, and the experimental sciences). This work has produced a universal framework that is summarized at jungle.princeton.edu which includes a video tutorial, several tutorial papers, links to undergraduate and graduate courses taught at Princeton University, an online book (using a teach-by-example style), and an advanced book (in progress) that can be downloaded here (the front cover is shown to the left).



The universal framework brings *all* the tools of stochastic optimization together, including one powerful idea that is widely used in practice but which has attracted almost no attention from the academic research community. My experience is that freight transportation and logistics is so rich, and so complex, that we need the complete toolbox.

This white paper is intended for people in industry with an analytical background, either in data sciences/machine learning, classical optimization, or any related fields typically taught in statistics, computer science, and operations research. It is not a complete tutorial, but it starts with familiar fields (machine learning, deterministic optimization) and builds a bridge to the rich problem domain of sequential decisions. Everything in this paper has been implemented at Optimal Dynamics to solve a range of decision problems in freight transportation and logistics, and has been implemented at a number of companies.

Sequential decision problems in freight transportation and logistics

In freight transportation and logistics, decisions span three time frames:

- Strategic planning - These are decisions that impact the company far enough in the future that the status of drivers and loads right now do not matter. Examples include how many dedicated drivers to have, where they should be domiciled, and the bids made to shippers for the upcoming quarter or year. Decisions in supply chain management might include the choice of suppliers and retailers, capacity planning, and raw material contracts. Strategic planning is always made in the presence of different forms of uncertainty.

- Tactical planning - These are decisions that impact the near future (perhaps 1-14 days out for a trucking problem, but 6-12 months for supply chain problems) that do depend on the state of drivers and loads now. The most

important is accepting offers for loads now to be moved in the future, but another might be developing plans to get drivers home. For supply chain management, tactical decisions include advanced inventory planning, signing contracts for production capacity in the future, pricing, and marketing.

- Real-time operations – These are the decisions that determine whether to use a company (dedicated) driver or not, which driver to assign to which load (or loads), where to relay loads, where to move empty, when to send a driver home, how to manage trailer pools, and spot pricing.

Both the strategic and tactical time frames require simulating real-time dispatching. So, at the heart of all three time frames is a single model that assigns drivers to loads (or tours), or manages inventories.

Note that these decisions come in several distinct classes.:

- Binary (accept the load or not).
- Discrete set (which carrier to call to move a load, which load to pull from a load board, which of a set of discrete prices to choose).
- Continuous vectors (setting prices across lanes or product lines).
- Discrete vectors (assigning a fleet of drivers to the set of available loads, allocating inventory to distribution centers).
- Multi-attribute (what *type* of driver do we want to assign to a load in the future; what *type* of product do we want to stock in inventory, where "type" consists of a vector of attributes).

I highlight these differences because there are entire research communities dedicated to each of these classes of decisions. My framework will cover all of them.

We represent decisions at time t by x_t, where x_t can be 0/1, an element of a set $\{x_1, \ldots, x_M\}$, or it can be a vector $x_t = \left(x_{tij}\right)_{i,j}$ where x_{tij} might be the flow from city i to city j at time t. When managing drivers, we often let

x_{tad} = The number of times we act on a resource (such as a driver) with attribute $a \in A$ with a decision $d \in D$ where d could be moving a driver empty, assigning a driver to a particular load, sending the driver home, or moving the driver to a trailer pool to change trailers. Note that the attribute a is typically a vector, that might include current/next location, domicile, equipment type, skills (e.g. hazmat), hours-of-service history, and nationality.

For all three of our time frames (strategic, tactical, real-time), we have to distinguish between decisions that are made before we start (this is represented by x_0) and the decisions that are made over time $(x_1, x_2, \ldots, x_t, \ldots)$. For strategic planning x_0 can represent the fleet size, driver domiciles or set of bids to customers. The decisions x_t for $t > 0$, would represent the simulation version of tactical and real-time decisions that are being simulated.

From deterministic optimization to sequential decisions under uncertainty

Our goal is to come up with a good set of decisions $x_0, x_1, \ldots, x_t, \ldots$ over time, but we have to do this as new information keeps arriving. If we are solving a deterministic linear program (I am keeping it simple to illustrate a point), we might write the problem as

$$\sum_{t=0}^{T} c_t x_t \tag{1}$$

subject to constraints on x_t at time t (for example, a driver cannot be in two places at the same time) and constraints that link x_t with $x_{t+1}, x_{t+2}, \ldots$ (for example, trailer or product inventories at time t+1 depend on your decisions at time t. There is a vast literature for dealing with variations of this problem.

When we have a sequential decision problem, we have to model the evolution of three variables over time:

$S_t =$ The state of the system at time t. $S_t = (R_t, I_t, B_t)$ captures the physical state R_t of the system (inventories, location of drivers, available loads); dynamic information I_t such as prices, weather; and probabilistic beliefs B_t that capture estimates of uncertain quantities that might be how the market responds to prices, the time it will take to complete a trip, or the reliability of a supplier. We can capture any form of uncertainty through the belief state, which also encourages active learning (such as trying a price just to see how the market responds).

$x_t =$ The decisions we make at time t. We assume x_t represents *any* decision, which covers the management of physical resources, pricing, or running experiments in a lab or the field, or even sharing information.

$W_{t+1} =$ Any new information that is revealed to us after we make the decision x_t (which means this information is not known when we determine x_t) and before time $t+1$ when we have to make the decision x_{t+1}. This information can be new customer demands, changes in prices or weather, unexpected delays unloading a truck, and updates to estimates of travel times. In a strategic planning model, we need to generate observations of W_{t+1} from a mathematical model, or by drawing from past history.

We can write the sequencing of state variables, decision variables, and new information as

$$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, S_t, x_t, W_{t+1}, \ldots, S_T, x_T).$$

When we are modeling a system in the computer, we require a set of equations known as the *transition function* that describes how the system evolves over time. We write this as

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

Note that this single, seemingly simple relationship can hide thousands of lines of code that describes how a complex system evolves over time.

We evaluate how well we are doing with a metric we call the "contribution function" $C(S_t, x_t)$ which can (and typically does) depend on dynamic information (such as prices) in the state variable S_t. We make decisions using a function called a *policy* which takes information in S_t and outputs a decision. We write this using $x_t = X^\pi(S_t)$, where the superscript π captures all the information about the type of function. Policies often depend on tunable parameters θ, so we might write the policy as $X^\pi(S_t|\theta)$.

Finally, we need to create sampled observations of the sequence $W_1, W_2, \ldots, W_T$. These are uncertain so we have to model that this is not just one set of possible values of this sequence, but many. We let ω be a sample realization of $W_1, W_2, \ldots, W_T$, where we let $W_t(\omega)$ be a sample realization of the information arriving at time t. We then let $\Omega = \{\omega^1, \omega^2, \ldots, \omega^N\}$ be a set of N sample realizations. We use this basic idea in strategic planning so that we can evaluate decisions under a variety of realizations of market demands and any other forms of uncertainty we wish to capture. When we are doing strategic planning, the initial state S_0 includes a lot of information such as the number of drivers in each domicile, but we do not need to know their initial positions. When we are doing tactical and real-time planning at time *t*, the initial state is S_t and includes the current status of each driver, as well as the set of loads that are known and committed.

In deterministic optimization, we want to find the best decisions x_t for $0 \le t \le T$. When we are solving sequential decision problems under uncertainty, we are trying to find the best *policy* $X^\pi(S_t)$, which is a method for making a decision regardless of the value of the state S_t. We then maximize (by convention) the *expected* performance of the policy over time. We write this objective as

$$\max_\pi F^\pi = E\left\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t)) | S_0\right\} \qquad (2a)$$

where $S_{t+1}(\omega) = S^M(S_t(\omega), x_t(\omega), W_{t+1}(\omega))$ for a sample path $W_1(\omega), \ W_2(\omega), \ \ldots, W_t(\omega), \ldots W_T(\omega)$. The expectation operator E can be viewed as an instruction to "take an average," which we can also write as

$$\max_\pi \bar{F}^\pi = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C(S_t(\omega^n), X^\pi(S_t(\omega^n))). \qquad (2b)$$

where $\bar{F}^\pi$ is an estimate of the value of following policy $X^\pi(S_t)$.

This is easily the part of the modeling framework that most people have trouble understanding, but it can be used to describe what a company (any company) is doing right now, with the sole exception that they do not have a formal process for

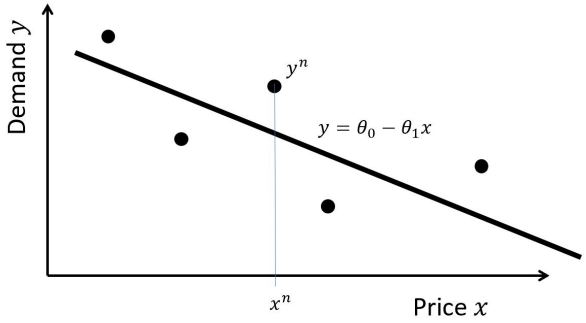
finding the best policy. Policies are often quite simple (buy low, sell high is an example of a simple policy). However, it is not uncommon that people use policies that they cannot describe. Not surprisingly, if we want computers to assist with making decisions, we have to be very clear and explicit not just about what decisions are being made, *but how*.

There is a mature, established community with tools for solving the deterministic optimization problem in equation (1). The same cannot be said for searching for the best policy, which partly explains the diversity of styles in the "jungle of stochastic optimization." At Optimal Dynamics, we are going to draw on a new universal framework that provides specific, practical paths to solving equation (2) (referring to both (2a) and (2b)).

From machine learning to sequential decisions

Given the high level of awareness of machine learning, it is helpful to contrast sequential decision problems with machine learning. Any machine learning problem can be represented as a function $f(x|\theta)$ which is used to estimate or predict an unknown quantity y (such as market demand in five days) given known inputs x (which can include both past history, and the prices we have chosen) and a set of tunable parameters θ.

We assume we are given a training dataset (x^n, y^n), $n = 1, \ldots, N$ of inputs x^n and responses y^n that we have observed in the past. A simple example relating demand ("y") to price ("x") is depicted in the figure to the right. We have to choose a type of function f from a family F, and then choose the tunable parameters $\theta \in \Theta^f$ where Θ^f is the set of tunable parameters associated with function $f \in F$. The type of function might be linear (as illustrated in the right to the right), or any of a number of nonlinear models, which would include neural networks.



The problem of finding the right function can be written as

$$min_{f \in F,\, \theta \in \Theta^f} \sum_{n=1}^{N} (y^n - f(\theta))^2 \qquad (3)$$

The art of machine learning is choosing the class of function $f \in F$. These functions fall in three overlapping classes:

- Lookup tables – Given a discrete $x \in X = \{x_1, \ldots, x_M\}$, produce an estimate θ_x. Here, X could be a set of shippers, or traffic lanes, or product types. If x_i is a

vector (e.g. origin, destination, day of week, product type, service requirements, refrigeration? hazmat?), then the number of elements in X can be huge, a problem known as the curse of dimensionality.

- Parametric models – The most famous is linear models of the form

$$f(\theta) = \sum_{i \in I} \theta_i \phi_i(x) \tag{4}$$

  where the functions $\phi_i(x)$ are known as features, and extract information from the input data x that is felt to be useful. This model is linear because it is linear in the parameters θ. A neural network is an example of a nonlinear model, which may exhibit a parameter vector θ with tens of thousands, to hundreds of thousands of parameters (some deep neural networks have hundreds of millions of parameters).

- Nonparametric models – The most common forms of nonparametric models use simple approximations (constants or linear models) for different regions of X. For example, we may feel that our example of demand as a function of price cannot be described using a single line, but we might fit different lines for low, moderate and high prices. This would be a nonparametric model.

Recently, a technology that has attracted considerable attention is neural networks which have excelled in recognizing images, sound, and text, as well as some other estimation problems. Statistical machine learning produces models that take information that we know (such as past market demands) to estimate or predict something we don't know (such as demands in the future). Models are estimated using training datasets, where the goal is always to predict the unknown quantity with the smallest possible error.

There is a misconception that neural networks can solve any problem. To be clear: neural networks are good at pattern recognition which are high-dimensional, unstructured, deterministic problems. Neural networks are extremely flexible, but this makes them vulnerable when applied to problems that exhibit noise, because they simply fit the noise.

Designing policies

We have just illustrated that the field of machine learning faces the challenge of choosing from three broad (and overlapping) classes of functions: lookup tables, parametric models, and nonparametric models. We now return to the problem of searching over policies π in equation (2). It turns out that there are four (meta)classes of policies, all of which will be used in complex problems such as freight transportation and logistics. The four classes consist of two broad classes, each of which can be further divided into two subclasses as follows:

- The policy search class – These are functions with tunable parameters that have to be tuned over time to find the values that work the best over time,

averaged over all the possible sample paths. This class consists of two sub-classes:
    - Policy function approximations (PFAs) – These can be simple rules (if it is Thursday we have to send this driver home), linear functions (we might set the discount as a declining linear function of inventory), nonlinear functions (such as neural networks) and nonparametric functions.
    - Cost function approximations (CFAs) – We can take a simplified optimization model (such as one that optimizes the assignment of drivers to loads) and parameterize it so that it works better over time, under uncertainty.

- The lookahead class – This policy makes better decisions now by estimating the costs or contributions in the future from a decision made now. This class also has two sub-classes:
    - Policies based on value function approximations (VFAs) – Imagine a Texas-based driver in St. Louis assigned to a load going to Boston. We do not know what will happen when he arrives in Boston, but we can approximate the value of the driver in Boston using a statistical model called a value function approximation.
    - Direct lookaheads (DLAs) – Google maps is the simplest example of a direct lookahead, where we plan a path all the way to the destination to determine if the driver should turn right or left. In truckload trucking, we might plan dispatches for a week into the future to determine if a load offered now, to be moved in the future, can be covered by one of our drivers. In a supply chain problem, we might have to plan ahead to anticipate bottlenecks to see if we should order more now.

Each of these four classes includes a range of variations, which is why they are called meta-classes. We describe each below in a little more detail.

PFA policies

The class PFA includes any function we might use in machine learning (lookup tables, parametric or nonparametric). The other three classes all require solving an embedded optimization problem. However, while a statistical model requires a training dataset to estimate the function, instead of fitting the function to data (as shown in equation (3)), we would use a performance metric $C(S_t,X^\pi(S_t))$ and the objective in equation (2) to determine a function we are using as a policy.

A simple example of a PFA arises in inventory planning where we order more inventory when the inventory falls below "s", at which time we order enough to bring the inventory up to "S." These are known as "(s,S)" policies, which have been proven to be optimal for very simple inventory problems (but not for the inventory problems we encounter in practice). Let $\theta=(s,S)$ represent these tunable parameters, and let $X^\pi(S_t|\theta)$ be the policy that depends on the state (which might just be the inventory, but can also include dynamic prices and forecasts), and the order-up-to policy π with parameters θ. We tune θ using equation (2), which we would write as

$$\max_{\theta} E\left\{\sum_{t=0}^{T} C\left(S_t, X^{\pi}\left(S_t|\theta\right)\right)|S_0\right\} \quad (5)$$

The problem with PFAs is that they cannot handle high-dimensional decisions such as dispatching drivers to loads. It is hard enough managing a single inventory where x_t might be the amount of new product to order at time t. Now imagine that $x_t = (x_{tad})_{a \in A, d \in D_t}$, where a is a vector of attributes describing a driver, and D_t is the set of possible decisions, which might include any location we may move a driver empty to, or any of a set of loads that we may ask the driver to pick up. The *dimensionality* of x_t for an actual trucking problem easily exceeds 10^{20}.

To understand the challenge of dimensionality, we may be able to limit the number of possible decisions for a single truck driver to 100 (sending it to nearby locations, or to a subset of nearby loads). If you have two drivers, you have roughly 100^2 possible decisions, since you have all the combinations of what both drivers can do at the same time. Now consider a carrier with 1,000 drivers, or 10,000 drivers. And this simple calculation does not come close to capturing all of the intricacies of modeling a modern truckload carrier.

CFA policies

An example of a simple CFA policy, used in the truckload industry since1980, would start with the problem of assigning drivers to loads. At time t, we might solve a single assignment problem that can be written as

$$\sum_{a \in A} \sum_{d \in D} c_{tad} x_{tad} \quad (6)$$

subject to constraints that reflect that we cannot assign a driver to two loads at the same time, and we cannot assign two drivers to the same load. This policy optimizes at time t, but ignores the impact of decisions now on the future, such as sending that Texas-based driver from St. Louis to Boston.

There are a number of ways to influence the behavior of this policy. For example, let τ_{tl} be the amount of time load l has been waiting as of time t. Let c_{tdl} be the cost of assigning driver d to load l, and let $x_{tdl} = 1$ if we assign driver d to load l at time t. We can write our policy as

$$X^{\pi}(S_t|\theta) = \sum_{a \in A} \sum_{d \in D} (c_{tdl} - \theta \tau_{tl}) x_{tdl} \quad (7)$$

where θ is a tunable parameter that allows us to control the importance of delaying a load. Now we face the problem of tuning θ, which we do by rewriting equation (2a) as

$$E\left\{\sum_{t=0}^{T} C\left(S_t, X^{\pi}\left(S_t|\theta\right)\right)|S_0\right\} \tag{8}$$

Delaying a load is actually just one of a host of issues when assigning drivers to loads. We may also wish to consider early or late pickups or deliveries, encouraging sleeper teams on long loads, making sure the driver is hitting revenue targets, and estimates of when we might get the driver home. We can write this more general utility function using

$$\sum_{a\in A}\sum_{d\in D}\left(c_{tad}+\sum_{i\in I}\theta_i\phi_i(a,d)\right)x_{tad} \tag{9}$$

Designing the features $\phi_i(a,d)$, $i\in I$, and then tuning the parameters $\theta_i, i\in I$, requires considerable experience working both with real data, and real operations. The policy in (9) is a classical example of a cost function approximation. Of particular value is that it can handle very high-dimensional decisions x_t (for example, over 100,000 dimensions).

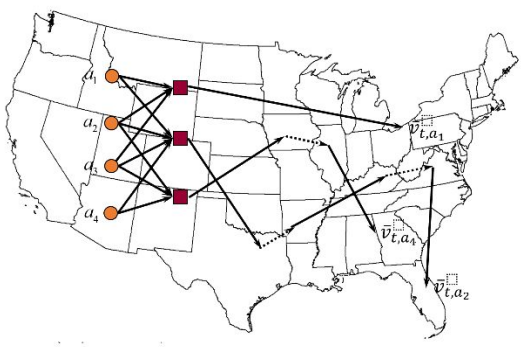
VFA policies using approximate dynamic programming

A limitation of the policy in equation (9) is that it does not capture the impact of a decision now on the future, such as what happens when our Texas-based driver is assigned to a load going to Boston. Looking into the future in freight transportation is complicated because it will depend on the loads that will be available when the truck arrives in Boston. Of course, we do not know which loads will be available at the time that we have to choose which driver to move the load, and it matters since we may be trying to get the driver back home to Texas.

Solving this problem required over 20 years of research, and resulted in a method that we have been calling *high-dimensional approximate dynamic programming*. The core idea is to draw on a nonstandard form of Bellman's equation that can be written

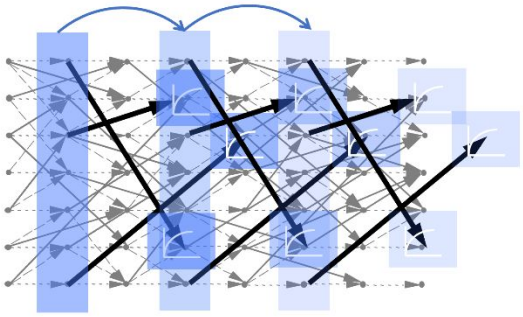
$$\sum_{a\in A}\sum_{d\in D}\bar{c}_{tad}x_{tad}+\sum_{a\in A}\bar{v}_{ta}R^x_{ta} \tag{10}$$

where $\bar{v}_{t,a}$ is the estimate, given what we know at time t, of a driver with attribute a. R^x_{ta} is known as the post-decision resource vector: it is the number of drivers with attribute a resulting from the decision x_t using the information we know now. R^x_{ta} is a linear function of x_t, which means we can solve (10) using modern solvers such as Gurobi or Cplex (to name two), which allows us to scale to thousands of drivers. Equation (10) is illustrated in the figure to the right, which illustrates that we can assign a driver to one load, or a sequence of loads, ending in an approximate value of the truck at the end of its tour through known loads.



There is a significant amount of work to estimate $\bar{v}_{t,a}$ which is beyond the scope of this article, other than to say that it draws on the fields of approximate dynamic programming (sometimes referred to as reinforcement learning) and machine learning.

The real value of a VFA-based policy such as (10) is that it allows us to capture the impact of a decision now on the future (approximately), without solving a problem that is any larger than the CFA policy in equations (7) or (9). Equation (10) provides a policy for dispatching drivers that can be simulated over time, as depicted in the figure to the left. This is how we can simulate fleets at a high level of detail in the strategic planning model, and it also produces the policy that can be used to dispatch drivers now, while capturing the value of drivers in the future.



DLA policies

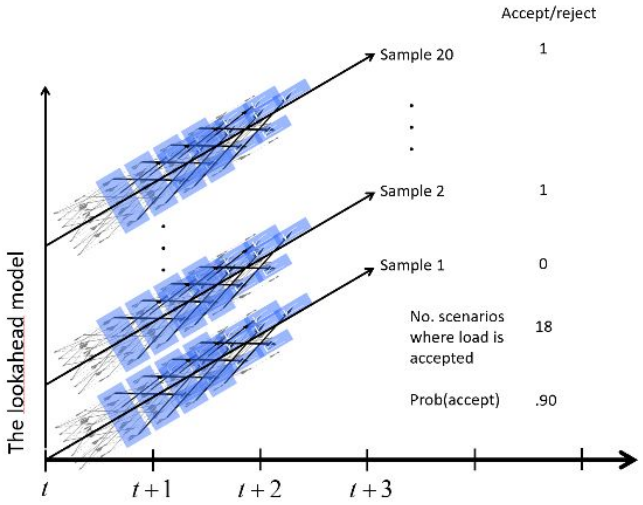
The simplest example of a direct lookahead policy is Google maps, which plans a path all the way to the destination to determine which turn to make right now. Google maps is using the simplest form of lookahead, which is to say that it assumes a deterministic estimate of the future. This approximation works when the uncertainty is in the travel times, since we just use a best estimate. We cannot do this in truckload trucking, since the uncertainty is in whether we have a load from region i to region j at some time t' in the future. Let's say that the probability there is a load is 0.10. If we use a point estimate, would we assume there is one tenth of a load?

We can use value function approximations to approximate the value of a driver in the future, but we cannot use this approach for load acceptance, where we need to make a decision to accept a load offered today (which might be Monday) to be picked up and moved in the future (say Friday). We make this decision based on the revenue generated by the load, whether we have a driver available to move the load (and drivers may move over 100 miles empty to pick up the load), and the opportunities for the driver after dropping off the load. On Monday, we will not know if we are going to have a driver available, or the competing opportunities for this driver.

This is a case of needing a *stochastic lookahead policy*, which involves solving

$$X^{\pi}(\theta) = argmin_{x\_t}\left(C(S_t, x_t) + \tilde{E}\max_{\tilde{\pi}}\left\{\tilde{E}\sum_{t'=t+1}^{t+H} C\left(\tilde{S}_{tt'}, \tilde{X}^{\tilde{\pi}}\left(\tilde{S}_{t,t'}|\theta\right)\right)|\tilde{S}_{t,t+1}\right\}|S_t, x_t\right) \quad (11)$$

Equation (11) is daunting, but this is the basis of the load acceptance model used by Optimal Dynamics. We simulate the VFA-based policy (equation (10)) to simulate dispatching in the future. This is how we create the inner policy-within-a-policy $\tilde{\pi}$ in equation (11), as depicted in the figure below. We then do this 20 times following different sample paths, and derive an estimate of the probability that an offered load now is covered in the future by counting how many times the load is covered. All of this is done continuously over time, responding to changing information on drivers, booked loads and rolling forecasts.



This is a nice illustration of a direct lookahead policy using a VFA-based policy as the "policy within a policy." We think this may be the first time this has been done, since typical policies within a stochastic lookahead model are much simpler. However, this is an application where the realistic simulation of drivers in the future is needed to make credible decisions about whether a load will be accepted.

Digital twins: the power of simulators

A critical component of the modeling system is a well calibrated simulator, which is increasingly being referred to as a "digital twin." Simulators are used for strategic planning, where they can be used for a variety of questions. In truckload trucking, these might be

- How many company (dedicated) drivers should the company have, recognizing that during peak periods loads will have to be pushed off to brokerages which are quite expensive?
- Where should the drivers be located?
- What shippers (and which lanes) should the carrier be serving, and at what price?

While these are probably the most important strategic questions, there are others:

- What is the cost if a shipper asks for tighter time windows?
- What is the cost of trying to get drivers home more frequently?

Questions for managing supply chains might include:

- Which suppliers should be used?
- What mode should be used?
- How much production and inventory capacity should be provided?
- How should the product be priced and marketed?

In addition to these basic (but challenging) questions is the need to tune a wide range of parameters. For example, if a manufacturer is using an (s,S) inventory policy, these need to be tuned *for each part*. However, in complex systems, we often have to balance competing priorities. In trucking, we want to minimize empty miles, but maximize on-time pickups and deliveries and, particularly important, manage drivers so they get home on time. This requires the use of bonuses and penalties to balance the importance of each of these objectives.

Simulators are critical for performing this tuning. Finding the best bonuses and penalties requires the same tools as finding the best (s,S) parameters, or the best allocation of drivers across potential domiciles. Without a simulator, optimizing these parameters requires a lot of guesswork.

Simulators have to be carefully calibrated to be credible. While this requires a significant amount of engineering of the dynamics of the system, inverse modeling, which is exactly analogous to fitting a machine learning model to data (comparable to the problem in equation (3)). In machine learning, the function $f(x|\theta)$ is an analytical function which is trivial to calculate (even if it is a deep neural network). By contrast, if $f(x|\theta)$ is a simulator with inputs x with tunable parameters θ, then we have to deal with the fact that $f(x|\theta)$ is expensive, noisy and nondifferentiable. A powerful strategy for overcoming these problems is to fit a surrogate model using the tools of nondifferentiable stochastic optimization.

A unified modeling system

A key feature of this modeling framework is that the strategic, tactical and operational planning systems are all based on the same logic for optimizing the assignment of drivers to loads. The dispatch model in equation (10), which is used in real-time operational planning, is simulated into the future to perform load acceptance in equation (11), and is imbedded in the strategic planning system which simulates dispatch over a planning horizon (perhaps a month, but it could be a year).

This nesting of the dispatch system means that any improvements made to the logic in any of the systems improves the performance for all of the systems. The strategic planning system is a particularly effective environment for testing new ideas quickly, in a realistic setting. If issues arise when using the system for real-time operational dispatching, changes made to address these issues are immediately reflected in both the tactical and strategic systems.

Closing remarks

This paper summarizes the general methodology used by Optimal Dynamics to handle the complex sequential decision problems, in the presence of different forms of uncertainty, that arise in freight transportation and logistics. This framework was developed over the course of a career. All of the modeling strategies outlined in this

paper have been implemented at Optimal Dynamics and are being used by a number of carriers.

While we cannot come close to describing all the details (and the engineering of these models and algorithms is implemented in almont a million lines of code), we hope this summary gives a sense of the general framework, and how it relates to more established fields such as deterministic optimization and machine learning. This framework is completely general, and can be applied direction to complex supply chain problems, with the twist that we need to introduce the dimension of multiagent decision making.

Determined readers who wish to develop a command of these ideas can go to http://www.castlelab.princeton.edu/jungle (or just enter jungle.princeton.edu in your browser) for extensive tutorial material. An introduction to the unified framework can be accessed at http://tinyurl.com/sequentialdecisionanalytics; a more advanced treatment is found in http://castlelab.princeton.edu/RLSO/.