Approximate dynamic programming for rail operations

Warren B. Powell and Belgacem Bouzaiene-Ayari

Princeton University, Princeton NJ 08544, USA

Abstract. Approximate dynamic programming offers a new modeling and algorithmic strategy for complex problems such as rail operations. Problems in rail operations are often modeled using classical math programming models defined over space-time networks. Even simplified models can be hard to solve, requiring the use of various heuristics. We show how to combine math programming and simulation in an ADP-framework, producing a strategy that looks like simulation using iterative learning. Instead of solving a single, large optimization problem, we solve sequences of smaller ones that can be solved optimally using commercial solvers. We step forward in time using the same flexible logic used in simulation models. We show that we can still obtain near optimal solutions, while modeling operations at a very high level of detail. We describe how to adapt the strategy to the modeling of freight cars and locomotives.

For over 10 years we have been developing a series of models for optimizing locomotives and freight cars for a major freight railroad in the U.S. using the principles of approximate dynamic programming. The projects span operational planning to strategic planning which generally impose very different expectations in terms of the level of realism. In this paper, we review how these projects unfolded and the surprising level of detail that was required to produce implementable results, even for a strategic system.

The foundation of our solution strategy is approximate dynamic programming, which combines the flexibility of simulation with the intelligence of optimization. ADP offers three distinct features that help with the development of realistic optimization models in rail operations: a) It offers a natural way of decomposing problems over time, while still offering near-optimal solutions over the entire horizon. b) ADP allows us to model complex dynamics using the same flexibility as a simulation model. c) ADP uses the same theoretical framework as dynamic programming to solve multistage problems under uncertainty.

ADP is often presented as a method for solving multistage stochastic, dynamic problems. However, ADP can be thought of as a tool from three different perspectives: 1) as a decomposition method for large-scale, deterministic problems, 2) as a method for making simulations intelligent, and 3) as a set of techniques for solving large-scale (possibly stochastic) dynamic programs. Our original motivation for this work was as a decomposition technique for solving a very large-scale driver management problem ([1]). The work in locomotives described in this paper, while involving sources of uncertainty, has primarily focused on solving deterministic formulations. These problems produce very

ATMOS 2007 (p.191-208)
7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2007/1180

large-scale integer programming problems which have been widely approached using various heuristics (see [2] and [3]).

ADP offers two unexpected features for solving these large-scale problems. The first is that by breaking large problems into smaller ones, we can solve these subproblems optimally using commercial solvers such as Cplex. Thus, the problem of assigning locomotives to trains at a single yard (or in a region) at a point in time is solved optimally. We depend on approximations to capture the impact of decisions now on the future, so our overall solution is not guaranteed to be optimal, but comparisons against optimal solutions have been extremely encouraging.

The second feature is that ADP allows us to model problems at a much higher level of detail. It is typically the case that large deterministic models typically introduce operational simplifications that impact the accuracy of the model itself. ADP integrates simulation and optimization, allowing us to capture the characteristics of the resources being used, as well as various operational rules, at a very high level of detail. Thus, we are able to model each locomotive individually, capturing detailed features such as its precise horsepower and adhesion rating, its maintenance status, orientation on the track (is it pointing forward or backwards), special equipment and ownership. This high level of detail does not prevent us from solving subproblems to optimality.

Our work in freight transportation has spanned three classes of models: 1) strategic planning models, which address questions such as fleet size and scheduling design, along with more complex studies of transit time reliability and order acceptance policies, 2) short-term tactical planning, where we look several days into the future to anticipate shortages of equipment and to manage demands, and 3) real-time planning, where we wish to provide fast response to user inputs and overrides.

The use of approximate dynamic programming to solve large, time-staged optimization problems (which may or may not be stochastic) requires the use of special modeling tools that are less familiar to a math programming-based community (but common in simulation and control-theory communities). This paper provides a general introduction to this modeling and algorithmic framework, and then describes how it can be applied to both locomotive optimization and the optimization of freight cars. We discuss the limitations of classical optimization models of fleet management, focusing not as much on the issue of uncertainty but rather on the importance of capturing realistic operational details. We describe how the ADP paradigm makes it much easier to capture these details, without losing the important features of optimization.

1 Literature review

There is an extensive literature on optimization models for rail operations. These range from single commodity models for managing generic fleets of containers (e.g., [4] and [5], to multicommodity models for handling multiple equipment types with substitution ([6], [7], [8], [9], [10], [11], [12], [13], [14] and [15]). A

separate line of research has focused on handling the high level of uncertainty in the demand for freight cars ([16], [17]); this research has continued under the general heading of "stochastic fleet management" or "dynamic vehicle allocation" (see the reviews in [18] and [19], as well as [20]).

Many of these models are particularly well suited for managing fleets of containers (box cars, trailers, intermodal containers). A separate literature has evolved around the more complex problem of managing locomotives. This problem has been modeled almost exclusively as a large-scale integer programming problem (see [11] for a review of the literature as of 1998). There are a host of complicating issues with locomotives, including the cost of coupling and uncoupling groups of locomotives used to pull a single train, the handling of leader locomotives, shop routing and a heterogeneous fleet of locomotives with different levels of power (common in freight operations in the United States).

There has been significant recent interest in models for locomotive optimization. [21] describes the use of modern branch and cut integer programming algorithms for the locomotive problem, which was applied to Canadian National Railway ([22]). [23] and [24] apply Benders decomposition to handle the simultaneous optimization of locomotives and cars. [2] presents a deterministic optimization model of locomotive operations that takes into account the issue of breaking up sets of locomotives that were joined to pull a previous train ("consist busting"). The model is designed for strategic planning purposes; it does not use a snapshot of the location of each locomotive, but instead works to identify repeatable cycles. The paper shows that the problem is NP-complete and presents a neighborhood search heuristic.

2 Modeling rail operations

The management of freight cars and locomotives are both instances of resource allocation problems. We begin by providing a general model, and then describe how this was adapted to handle freight cars and locomotives.

2.1 A general resource allocation model

Rail operations can be modeled as "resources" (locomotives, freight cars) that are serving "demands" (trains, customer orders). We model these using

$$
\begin{aligned}
a &= \text{the vector of attributes describing a resource,}\\
R_{ta} &= \text{the number of resources with attribute } a \in \mathcal{A} \text{ in the system at time } t,\\
R_t &= (R_{ta})_{a\in\mathcal{A}},\\
b &= \text{the vector of attributes describing a demand,}\\
D_{tb} &= \text{the number of demands of type } b \in \mathcal{B} \text{ in the system at time } t,\\
D_t &= (D_{tb})_{b\in\mathcal{B}}.
\end{aligned}
$$

We think of a (or a_t) as the state of a single resource, and R_t is the state of all the resources (the resource state vector). The state of our system is given by $S_t = (R_t, D_t)$, where t represents the time at which a decision is made, and S_t is the information available at time t. New information is represented as exogenous changes to the resource and demand vectors, as well as to other parameters that govern the problem. These are modeled using

$\hat{R}_{ta}$ = exogenous changes to R_{ta} from information that arrives during time interval t (between $t-1$ and t),

$\hat{D}_{tb}$ = exogenous changes to D_{tb} from information that arrives during time interval t (between $t-1$ and t).

$\hat{R}_{ta}$ would be used to describe exogenous changes to resources such as equipment failures and transit time delays. $\hat{D}_{tb}$ would normally be used to describe new customer requests, but could also be used to model changes in a customer request (something that will be useful in the freight car problem). We describe the exogenous information process generically using $W_t = (\hat{R}_t, \hat{D}_t)$. Throughout, we model information as if it were arriving in continuous time, where W_t is the information that arrived between decision epochs $t-1$ and t. We always let t index a decision epoch, not the time at which events actually happen (we can decide at noon that a locomotive arriving at 3pm should be assigned to a train leaving at 8pm).

Decisions are modeled using

$\mathcal{D}^D$ = decision to satisfy a demand with attribute b (each decision $d \in \mathcal{D}^D$ corresponds to a demand attribute $b_d \in \mathcal{B}$),

$\mathcal{D}^M$ = decision to modify a resource (each decision $d \in \mathcal{D}^M$ has the effect of modifying the attributes of the resource). $\mathcal{D}^M$ includes the decision to "do nothing,"

$\mathcal{D} = \mathcal{D}^D \cup \mathcal{D}^M$,

x_{tad} = the number of resources that initially have attribute a that we act on with decision d,

$x_t = (x_{tad})_{a \in \mathcal{A}, d \in \mathcal{D}}$.

For resource allocation problems, decisions always have to satisfy the constraints

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{ta}, \tag{1}$$

$$\sum_{a \in \mathcal{A}} x_{tad} \leq D_{tb_d}, \quad d \in \mathcal{D}^D, \tag{2}$$

$$x_{tad} \geq 0. \tag{3}$$

For specific applications (this is especially true with locomotives), there will be additional constraints. We let $\mathcal{X}_t$ be the feasible region, which would include (1)-(3) as well as any other constraints that may be necessary.

Our problem is determining how to make a decision. For now, we represent this step by assuming that we have a decision function, given by

$$X_t^\pi(S_t) = \text{a function that returns a decision vector } x_t \in \mathcal{X}_t, \text{ where } \pi \in \Pi \text{ is an element of the set of functions (policies) } \Pi.$$

The state of the system evolves over time in a way that is described using a transition function, represented using

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

The state transition function (known as the "system model" in some communities) can be broken down into components that act on specific parts of the state. State transition functions are very familiar to specialists in simulation and control, but not to the math programming community. It is important to realize that this single equation hides a tremendous range of rules and calculations that capture how the system evolves in time.

We are going to find it useful to divide the state transition into two steps: the pure effect of the decision, and the pure effect of information. We write this using

$$\begin{aligned} S_t^x &= \text{the post-decision state variable} \\ &= S^{M,x}(S_t, x_t), \\ S_{t+1} &= S^{M,W}(S_t^x, W_{t+1}). \end{aligned}$$

The post-decision state variable is going to play a particularly important role in our algorithmic strategy.

Of particular importance is the evolution of the attributes of a specific resource. For this, we define the *attribute transition function* which describes the effect of a decision d on a resource with attribute a, after which we observe information W_{t+1} (information that arrives after time t). This is described using

$$a_{t+1} = a^M(a_t, d_t, W_{t+1}).$$

For notational convenience, we introduce the *resource transition function* that describes the collective effect of a set of decisions (described by the vector x_t) on the resource vector R_t using

$$R_{t+1} = R^M(R_t, x_t, W_{t+1}).$$

To write this out algebraically, we first give the post-decision version of the attribute transition function $a_t^x = a^{M,x}(a_t, d_t)$. It is useful to think of a_t^x as the

attribute of the resource which we *expect* to happen as a result of a decision. We then define the indicator function

$$\delta_{a'}(a,d) = \begin{cases} 1 \text{ if } a' = a_t^x = a^{M,x}(a_t, d_t), \\ 0 \qquad\quad \text{otherwise.} \end{cases}$$

This allows us to write the post-decision resource vector as

$$R_{ta'}^x = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} \delta_{a'}(a,d) x_{tad}.$$

We then let $\hat{R}_{t+1,a}$ be the exogenous change to the resource vector R_t^x as a result of exogenous information such as a transit time delay. This allows us to write

$$R_{t+1,a} = R_{ta}^x + \hat{R}_{t+1,a}.$$

For the moment, we model demands in a simple way. If a resource is assigned to a demand, then it is "served" and vanishes from the system. Otherwise, it is held to the next time period. Let

$$\begin{aligned} \delta D_{tb_d} &= \text{the number of demands of type } b_d \text{ that are served at time } t \\ &= \sum_{a \in \mathcal{A}} x_{tad} \quad d \in \mathcal{D}^D, \\ \delta D_t &= (\delta D_{tb})_{b \in \mathcal{B}}. \end{aligned}$$

The demand transition function can be written

$$\begin{aligned} D_t^x &= D_t - \delta D_t, \\ D_{t+1} &= D_t^x + \hat{D}_{t+1}. \end{aligned}$$

The last dimension of our model is the objective function. For our resource allocation problem, we define a contribution for each decision given by

$$c_{tad} = \text{contribution earned (negative if it is a cost) from using decision } d \text{ acting on resources with attribute } a.$$

The contribution function for time period t is assumed to be linear, given by

$$C_t(S_t, x_t) = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{tad} x_{tad}.$$

The objective function is now given by

$$\max_{\pi \in \Pi} E \left\{ \sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t)) \right\}.$$

One policy for solving this problem is a myopic policy, which involves making decisions using

$$x_t = \arg\max_{x_t \in \mathcal{X}_t} C(S_t, x_t). \tag{4}$$

Here, we simply ignore the impact of decisions now on the future.

Most railroads in North America use a simple myopic model for assigning freight cars to orders, although some use point estimates of supplies of and demands for cars. There are several potential problems with a myopic model. 1) We might assign a car available now (on Monday) to an order that does not have to be moved until Friday, that requires only a one-day transit time. This ties up the car for four additional days, when a different car (not yet known) might have covered the order. 2) It may be necessary to start moving cars now to orders that have not yet been called in (and which may be highly uncertain). 3) Often, multiple car-types can be used to cover a particular order. It is helpful to think about the value of different car-types at the destination of the order to determine the best car to assign right now. 4) A railroad might want to make decisions about whether to commit to a customer order for freight to be picked up a week or two in the future. Myopic models cannot help with these decisions. 5) There are numerous planning problems, relating to issues such as the value of freight, the value of cars of a particular type, the effect of transit time reliability and the value of advance notice from shippers that require the ability to model these effects.

This generic model for resource allocation problems allows us to describe both freight cars and locomotives quite easily.

2.2 An adaptation for freight car management

The generic model given in section (2.1) can be applied directly to freight car management. In the literature, the car distribution problem is almost always modeled as a multicommodity flow problem using decision variables given by

$$x_{tij}^k = \text{the flow of resources of type } k \text{ leaving node } i \text{ at time } t \text{ going to node } j.$$

We started a project with a major railroad using this same notation (see [25]), but quickly found that it simply did not capture important characteristics of the problem. By the completion of our project, we were using the following

attributes:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} = \begin{pmatrix} \text{Location (current or origin)} \\ \text{Destination} \\ \text{Departure time} \\ \text{Estimated time of arrival} \\ \text{Car type} \\ \text{Equipment status} \\ \text{Cleanliness} \\ \text{Shipper pool} \end{pmatrix}.$$

A major point of departure with classical deterministic models is that we model the time at which an event happens as an attribute, which can be modeled in continuous time, even if we make decisions in discrete time. Thus, a car can arrive at 7:33 am and depart at 11.52am. The importance of doing this took us by surprise, but laboratory experiments confirmed the feeling at the railroad that this was important.

The attributes of an order were given by

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{pmatrix} = \begin{pmatrix} \text{Number of orders} \\ \text{Pickup location} \\ \text{Delivery location} \\ \text{Call-in time} \\ \text{Pickup window} \\ \text{Delivery window} \\ \text{Loading time} \\ \text{Unloading time} \\ \text{Shipper/industry/commodity type} \\ \text{Car types allowed} \end{pmatrix}.$$

A significant issue with the modeling of car distribution was the complexity of the information process. Most models assume that everything is known in advance. The extensive literature on stochastic models assumes that demands are stochastic, but once they become known, everything becomes known. In practice, information evolves over time. For example, after the initial order is made (at the call-in time), we will know the origin of the order, but not the destination. The shipper does not let us know if the car is clean enough until the car is delivered to the shipper. Loading and unloading times are not known until the car is loaded or unloaded. The estimated time of arrival (for the car) evolves continuously over a trip.

The call-in process had to be modeled with some care. Initial orders (which include an estimate of the number of loads, pick-up location but not destination) are generally made the week before. But the railroad often has to move cars that are empty on Monday before orders arrive later in the week. If a shipper does

not place his order on, say, Wednesday, the order may arrive on Thursday or Friday, or not at all. Thus, the order process is not Poisson.

The contribution function depends on the shipper, the distance traveled (empty or loaded), and the degree to which the order is being picked up or delivered early or late.

2.3 An adaptation for locomotive operations

When assigning locomotives to trains, the first issue that has to be considered is how much power is needed to pull the train. A train might require 2.2 horsepower per trailing ton ("trailing tons" refers to the aggregate weight of all the cars being pulled). A train weighing 9,000 tons (gross weight, including the weight of the cars), requiring 2.2 horsepower per ton would require enough locomotives to provide 19,800 horsepower. This horsepower can be provided by a mixture of locomotives with anywhere between 1,700 to over 4,000 horsepower. Of course, we have to use an integer number of locomotives, and we can mix and match to produce the right amount of power. We could use seven 3,000 horsepower locomotives which produce 21,000 horsepower, or four 3,000 horsepower units with two 4,000 horsepower units for a total of 20,000 horsepower. As a result, this is a fairly challenging integer programming problem.

If we simply had to schedule a fleet of locomotives taking into consideration the mix of horsepower and integrality requirements, this by itself would be a fairly hard integer programming problem. We also have to consider the fact that if we group multiple locomotives to pull a single train (this group of locomotives is called a *consist*), there is a cost if we have to separate one or more locomotives from the consist. This introduces a significant complication, over and above the challenge of finding an integer number of heterogeneous locomotives to move a train. This complexity motivated the design of the neighborhood search heuristic reported in [2].

Our work has identified a number of other issues which have proven to be important not just for operational models (these tend to be more complex since the results have to capture enough realism for implementation), but also for strategic planning models. These details include the handling of leader-qualified locomotives, shop routing, late trains, equipment failures and foreign power.

Shop routing is particularly difficult. A locomotive can still pull a train while it is being routed to shop, but while we are routing a locomotive toward its shop location, we have to try to minimize how often consists are broken. Shop routing can not be solved independently of the original problem.

In strategic planning applications, it is also important to take into account the random additions and cancellations, as well as delays. If an extra train moves out of a yard 20 percent of the time (to various destinations), then we cannot pretend that we know exactly when, and to where, these additional trains will move.

3 Approximate dynamic programming

Approximate dynamic programming has been evolving as a powerful tool for solving more complex types of dynamic programs. In a series of papers motivated by problems in freight transportation, ADP has been adapted to solve multistage stochastic linear (and integer) programs. Classical dynamic programming starts with Bellman's equation, given by

$$V_t(S_t) = \max_{x \in \mathcal{X}_t} (C(S_t, x_t) + \gamma E \{V_{t+1}(S_{t+1})|S_t\}), \tag{5}$$

where $V_t(S_t)$ is the value of being in state S_t at time t, and γ is a discount factor. It is widely known that Bellman's equation is hard to use because of the "curse of dimensionality" which prevents us from solving (5) for each state S_t. If S_t is a vector (for our applications, S_t is a very high-dimensional vector), we cannot compute $V_t(s)$ for each state s.

In the remainder of this section, we describe a generic strategy for using approximate dynamic programming to solve resource allocation problems, and then describe how this was adapted for car distribution and locomotive optimization.

3.1 A generic ADP strategy

The approximate dynamic programming community replaces $V_t(S_t)$ with some sort of approximation which we denote $\bar{V}_t(S_t)$. For example, we might use

$$\bar{V}_t(S_t) = \theta_0 + \sum_i \theta_1 S_{ti} + \sum_i \theta_2 (S_{ti})^2.$$

Now, we just have to estimate the three parameters $(\theta_0, \theta_1, \theta_2)$. Aside from the issue of whether this is an accurate approximation, this strategy still assumes that we can compute the expectation in (5), and we need to find a high-dimensional vector x_t.

We avoid the expectation by formulating Bellman's equations around both the pre- and post-decision states S_t and S_t^x. This allows us to break equation (5) into two equations

$$\begin{aligned} V_t(S_t) &= \max_{x \in \mathcal{X}_t} (C(S_t, x_t) + \gamma V_t^x(S_t^x)), \\ V_t^x(S_t^x) &= E \{V_{t+1}(S_{t+1})|S_t^x\}. \end{aligned}$$

Here, $S_t^x = S^{M,x}(S_t, x_t)$ and $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$. We do not actually use $V_t(S_t)$. Instead, we replace $V_t^x(S_t^x)$ with an approximation $\bar{V}_t(S_t^x)$. We then make decisions using

$$x_t = \arg\max_{x \in \mathcal{X}_t} (C(S_t, x_t) + \gamma \bar{V}_t(S_t^x)). \tag{6}$$

We need to create a value function approximation so that this problem can be solved using a commercial solver. For resource allocation problems, it is natural to create a value function approximation around the post-decision resource vector R_t^x (rather than the full state variable S_t^x). A simple value function approximation is linear in the resource state,

$$\bar{V}_t(R_t^x) = \sum_{a \in \mathcal{A}} \bar{v}_{ta} R_{ta}^x.$$

We have generally found that linear approximations are too unstable. A much better approximation uses separable, piecewise linear approximations which we write generically as

$$\bar{V}_t(R_t^x) = \sum_{a \in \mathcal{A}} \bar{V}_{ta}(R_{ta}^x),$$

where $\bar{V}_{ta}(R_{ta}^x)$ is a piecewise linear, scalar function. This approximation has proven to be very effective for fleet management problems (see [26], [27], and [25]). These functions can be estimated quite easily by using the dual variables for constraint (1). Thus, instead of using an estimate of the value of being in a state, we are using derivatives (or estimates of derivatives). [28] provides a simple description of an algorithm (the CAVE algorithm) for estimating these functions. [29] proves that these algorithms are convergent for special problem classes, and provides comparisons against optimal algorithms to support the claim that this approach offers very high quality solutions with fast convergence.

Figure 1 provides a detailed description of the steps of the algorithm. The algorithm is run iteratively, forward in time. At iteration n, we follow a particular sample path, indexed by ω^n, forward in time, making decisions using the value function approximation $\bar{V}_t^{n-1}(S_t^x)$ computed in the previous iteration. We represent the updating of the value function using

$$\bar{V}_{t-1}^n \leftarrow U^V(\bar{V}_{t-1}^{n-1}, S_{t-1}^{x,n}, \hat{v}_t^n),$$

where $U^V(\cdot)$ is a general updating strategy. There are numerous ways for performing this updating (in addition to the articles cited above, see the more complete treatment in [30]).

3.2 An adaption for freight car management

The algorithm described in the previous section can be applied almost directly to the freight car problem. The only adaptation involved the aggregation of the resource vector in the value function approximation. Section 2.2 describes an eight-dimensional attribute vector, which was needed to perform such calculations as computing the contribution function, and simulating the status of each car. For the value function, we used a three-dimensional attribute capturing location, estimated time of arrival and car type. This means that the dual variable for an eight-dimensional attribute vector, denoted $\hat{v}_{ta}$, was used to update a

Step 0. Initialization:
  Step 0a. Initialize $\bar{V}_t^0, \ t \in \mathcal{T}$.
  Step 0b. Set $n = 1$.
  Step 0c. Initialize S_0^1.
Step 1. Choose a sample path ω^n.
  Step 2. Do for $t = 0, 1, 2, \ldots, T$:
    Step 2a. Solve:

$$x_t^n = \arg\max_{x_t \in \mathcal{X}_t^n} \left(C_t(S_t^n, x_t) + \gamma \bar{V}_t^{n-1}(S^{M,x}(S_t^n, x_t)) \right) \tag{7}$$

and let $\hat{v}_t^n$ be the dual variables of the resource constraint (1).
    Step 2b. If $t > 0$, update the value function:

$$\bar{V}_{t-1}^n \leftarrow U^V(\bar{V}_{t-1}^{n-1}, S_{t-1}^{x,n}, \hat{v}_t^n).$$

    Step 2c. Update the states:

$$S_t^{x,n} = S^{M,x}(S_t^n, x_t^n),$$
$$S_{t+1}^n = S^{M,W}(S_t^{x,n}, W_{t+1}(\omega^n)).$$

Step 3. Increment n. If $n \leq N$ go to Step 1.
Step 4. Return the value functions $(\bar{V}_t^N)_{t=1}^T$.

Fig. 1. A generic ADP algorithm using dual variables to update the value function.

separable, piecewise linear value function approximation $\bar{V}_{ta}(R_{ta})$, where a is represented using a three-dimensional attribute vector.

Figure 2 illustrates what a subproblem looks like. Cars are assigned to known orders or to locations, where the value of a location is represented by a piecewise linear value function approximation. Note that a car may be available ("actionable") now or in the future, just as orders may be available to be moved now or at some point in the future. One problem that myopic models have is that a car available now may be assigned to an order that does not have to be moved for a week or more.

The car distribution problem required that we simulate randomness in customer demands (the number of orders from a location), transit times, load and unload times, the destination of an order (which became known only after the car was loaded) and the acceptability of a car to the shipper. These random variables were simulated as the system evolved through time.

The freight car management system can be run in three modes: a) as a real-time system for assigning cars to orders, b) as a short-term forecasting system, projecting activities over a two or three week period to help with demand management and fleet planning, and c) as a strategic planning system, which might be used to evaluate contracts, fleet mix, transit time reliability and customer behaviors.

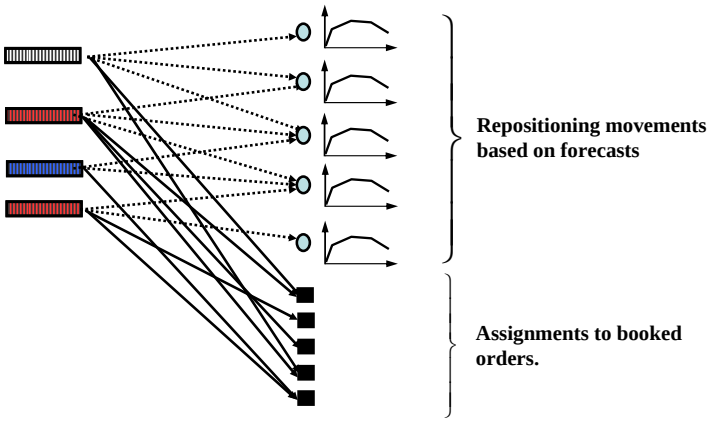

Fig. 2. The optimization model for cars at time t, showing assignment of cars to known orders and to value functions

3.3 An adaptation for locomotives

Modeling locomotives can be handled using the same framework, but locomotives are considerably more complicated. With freight cars, there is a constraint (equation (2)) that requires that we have one car per order. With locomotives, several locomotives may be used to move a single train. A train might require, for example, 13,000 horsepower. A single locomotive might have between 1,750 and 4,400 horsepower. The model has to mix and match locomotives to achieve at least 13,000 horsepower, but it is possible to assign more horsepower because the location to which the train is going needs additional locomotives. Locomotives may be "repositioned" either by putting more power than is needed on a train, or through the use of "light engine moves" which are locomotives moving without pulling any cars.

Locomotive assignment has to consider other issues. One attribute of a locomotive is the train-ID on which the locomotive arrived. If three locomotives share the same train-ID, then this means that they are coupled into a "consist" (locomotives have to be connected electrically and hydraulically to ensure that they move as a common unit). If there are three locomotives in a consist but we only want one or two of them, then we assess a consist-breakup cost (it

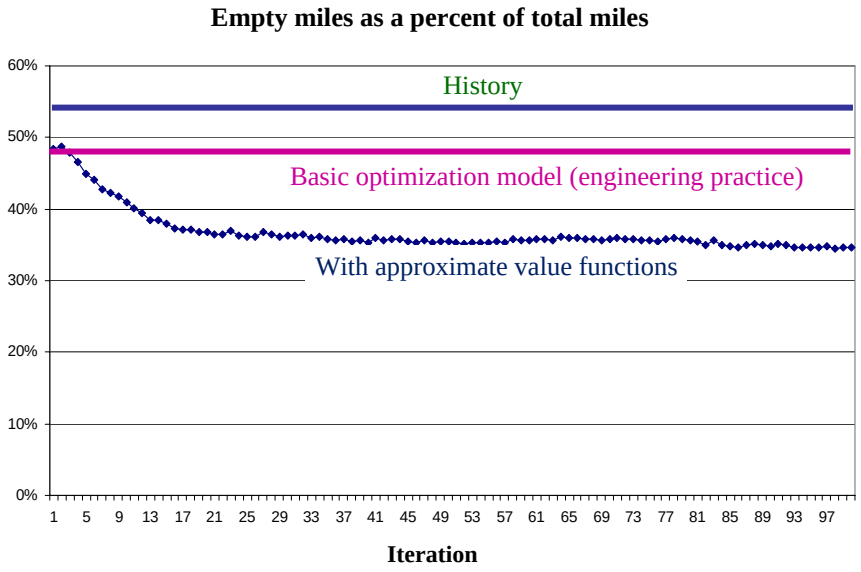

Fig. 3. Empty miles as a percent of total from history, with a myopic optimization model, and using approximate dynamic programming

also takes time). When we assign power to locomotives, we have to consider consist-breakup (some authors refer to this as "consist busting"). We also have to assign locomotives to trains that allow them to arrive at their shop location at the scheduled time.

As we determine a good set of locomotives to pull a train, we also have to take into account other requirements such as the need to have a leader-qualified locomotive, or other special equipment requirements. For example, sometimes a train moving up a steep grade requires the use of a radio-controlled locomotive positioned at the middle of the train. That means that one of the locomotives in the consist has to be equipped with radio power.

4 Experience with freight cars

An operational planning system based on approximate dynamic programming has been implemented at the Norfolk Southern Railroad, one of the two major freight railroads covering the eastern United States. We performed a set of experiments comparing a myopic model and the solution obtained using ADP to what was being achieved in history. The results are shown in figure 3. For this dataset, cars were running 54 percent empty in history. A myopic model reduced this to 48 percent, a result that is more than enough to justify the cost of the model. Approximate dynamic programming reduced this to almost 35 percent.

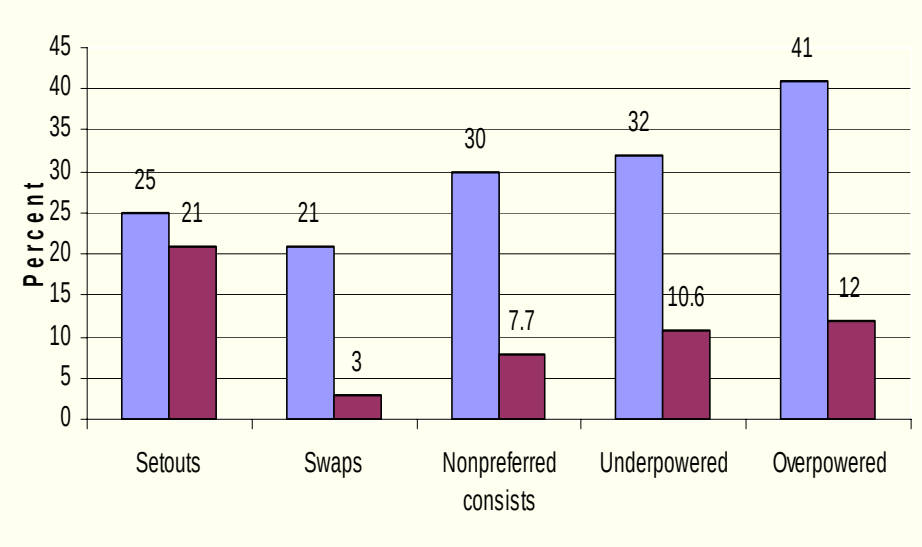

Fig. 4. Metrics from history and the model, where smaller is better

The freight car system can be run as a real-time assignment system (by solving the single subproblem at time 0), but its primary use has been to provide a forecast of activities over a three-week horizon. It can also be used to analyze history to suggest new routing patterns, or as a strategic planning model to help determine fleet size and mix, evaluate customers and analyze questions such as the effect of transit time and transit time reliability on fleet requirements.

5 Experience with locomotives

Figure 4 provides a measure of the performance of the model for one major railroad, where we compare the model to history using five different performance statistics such as setouts (breaking up a consist), swaps (exchanging locomotives between trains, often to get a particular locomotive to a shop) and using nonpreferred locomotives (the railroad preferred certain types of locomotives on certain types of trains). We were able to outperform the railroad on all major performance measures, including such detailed statistics as the productivity of locomotives while they are being routed to shop.

In 2006, we began development of a second generation locomotive model, drawing on a number of advances from our first generation model developed over the 1996-2002 period. Figure 5 illustrates train coverage as the algorithm adaptively learns the value function for the strategic planning model. The convergence is fast and extremely stable, representing a significant improvement over our first implementation (we attribute the stability to the use of nonlinear value function approximations).

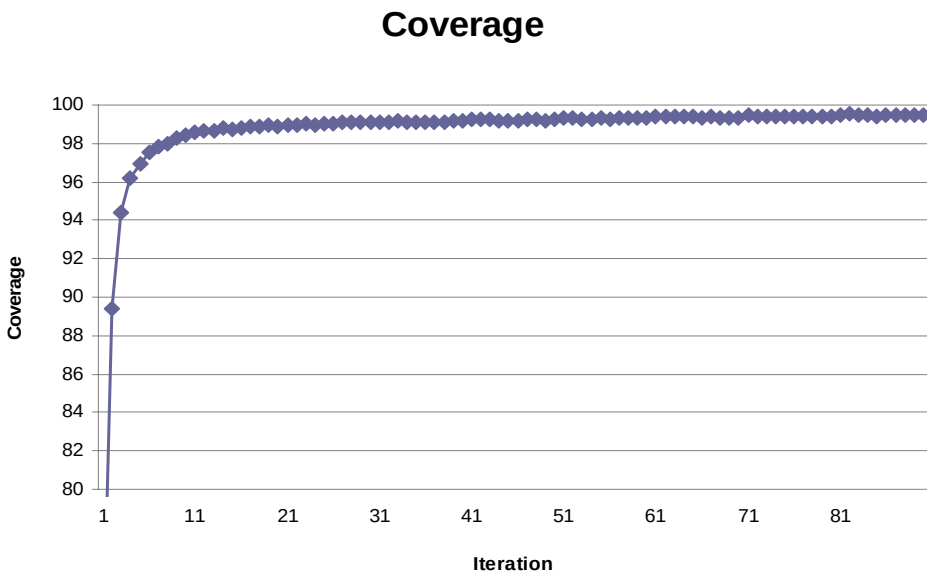

Fig. 5. Train coverage for strategic planning model during the learning process

One of our most difficult lessons has been the high level of detail required to perform accurate fleet sizing for strategic planning purposes. It is well known in the railroad modeling community that optimization models routinely recommend significant reductions in the number of locomotives. These "savings" arise not because of sophisticated algorithms finding optimal solutions, but rather in the many simplifications that are typically made in a mathematical model. We found that issues such as consist-breaking, leader locomotives and special equipment (ranging from radio controllers to coordinate different locomotives to the requirement for flush toilets in certain regions of the United States) can have a surprisingly significant impact on fleet sizing. Shop routing, and the proper handling of freight power, can also have significant impacts on fleet requirements.

6 Conclusions

Over 10 years of development with two separate railroads has shown us that we can handle the high level of complexity required to produce an accurate model of rail operations. For the car distribution problem, this means handling car attributes such as equipment type, maintenance status and ownership, but most importantly the complex information processes covering the number of cars being ordered, the destination of cars (known only after the car is loaded), load, unload and transit times, and the acceptability of a car. For locomotives, this has meant handling issues such as consists, horsepower and adhesion, maintenance status and ownership.

It is well known that these problems cannot be solved optimally, producing an extensive literature on heuristics. However, these heuristics are typically used to find near-optimal solutions to simplified models, which invariably underestimate what is required to meet a set of demands (cars or locomotives). In many applications, the ability to handle uncertainty is important, although our models are frequently applied to history (which is deterministic). For example, it is not enough to plan the locomotive fleet size for a perfect schedule where there are no delays or failures. We have to anticipate that problems will arise, and plan for them. Approximate dynamic programming easily handles uncertainty, allowing us to produce robust solutions that will work in field implementations.

References

1. Powell, W.B., Shapiro, J.A., Simão, H.P.: An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. Transportation Science **36** (2002) 231–249
2. Ahuja, R.K., Liu, J., Orlin, J.B., Sharma, D., Shughart, L.A.: Solving real-life locomotive-scheduling problems. Transportation Science **39** (2005) 503–517
3. Glover, F., Kochenberger, G.: Handbook of Metaheuristics. Springer (2003)
4. White, W.: Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers. Networks **2** (1972) 211–236
5. Herren, H.: Computer controlled empty wagon distribution on the SSB. Rail International **8** (1977) 25–32
6. Glickman, T., Sherali, H.: Large-scale network distribution of pooled empty freight cars over time, with limited substitution and equitable benefits. Trans. Res. **19** (1985) 85–94
7. Dejax, P., Crainic, T.: A review of empty flows and fleet management models in freight transportation. Transportation Science **21** (1987) 227–247
8. Crainic, T., Rousseau, J.M.: Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. Transportation Research B **20B** (1988) 290–297
9. Haghani, A.: Formulation and solution of a combined train routing and makeup, and empty car distribution model. Transportation Research **23B** (1989) 433–452
10. Crainic, T.G., Laporte, G.: Planning models for freight transportation. European Journal of Operational Research **97** (1997) 409–439
11. Cordeau, J.F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science **32** (1998) 988–1005
12. Holmberg, K., Joborn, M., Lundgren, J.T.: Improved empty freight car distribution. Transportation Science **32** (1998) 163–173
13. Joborn, M.: Optimization of empty freight car distribution in scheduled railways. Ph.D. thesis, Department of Mathematics, Linkoping University, Sweden (2001)
14. Lingaya, N., Cordeau, J.F., Desaulniers, G., Desrosiers, J., Soumis, F.: Operational car assignment at via rail canada. Transportation Reesarch B **36** (2002) 755–778
15. Joborn, M., Crainic, T.G., Gendreau, M., Holmberg, K., Lundgren, J.T.: Economies of scale in empty freight car distribution in scheduled railways. Transportation Science **38** (2004) 121–134
16. Mendiratta, V., Turnquist, M.: A model for the management of empty freight cars. Trans. Res. Rec. **838** (1982) 50–55
17. Jordan, W., Turnquist, M.: A stochastic dynamic network model for railroad car distribution. Transportation Science **17** (1983) 123–145

18. Powell, W.B., Jaillet, P., Odoni, A.: Stochastic and dynamic networks and routing. In Monma, C., Magnanti, T., Ball, M., eds.: *Handbook in Operations Research and Management Science*, Volume on *Networks*, Amsterdam, North Holland (1995) 141–295
19. Powell, W.B., Bouzaiene-Ayari, B., Simao, H.: Dynamic models for freight transportation. In Laporte, G., Barnhart, C., eds.: Handbooks in Operation Research and Management Science: Transportation. (2006)
20. Crainic, T., Gendreau, M., Dejax, P.: Dynamic stochastic models for the allocation of empty containers. Operations Research **41** (1993) 102–126
21. Ziarati, K., Soumis, F., Desrosiers, J., Solomon, M.: A branch-first, cut-second approach for locomotive assignment. Management Science **45** (1999) 1156–1168
22. Ziarati, K., Soumis, F., Desrosiers, J., Gelinas, S., Saintonge, A.: Locomotive assignment with heterogeneous consists at CN North America. European journal of operational research **97** (1997) 281–292
23. Cordeau, J.F., Soumis, F., Desrosiers, J.: A Benders decomposition approach for the locomotive and car assignment problem. Transportation Science **34** (2000) 133–149
24. Cordeau, J.F., Soumis, F., Desrosiers, J.: Simultaneous assignment of locomotives and cars to passenger trains. Operations Research **49** (2001) 531–548
25. Topaloglu, H., Powell, W.B.: Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. Informs Journal on Computing **18** (2006) 31–42
26. Godfrey, G., Powell, W.B.: An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times. Transportation Science **36** (2002) 21–39
27. Godfrey, G., Powell, W.B.: An adaptive, dynamic programming algorithm for stochastic resource allocation problems II: Multi-period travel times. Transportation Science **36** (2002) 40–54
28. Godfrey, G.A., Powell, W.B.: An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems. Management Science **47** (2001) 1101–1112
29. Powell, W.B., Ruszczyński, A., Topaloglu, H.: Learning algorithms for separable approximations of stochastic optimization problems. Mathematics of Operations Research **29** (2004) 814–836
30. Powell, W.B.: Approximate Dynamic Programming: Solving the curses of dimensionality. John Wiley and Sons, New York (2007)