An Adaptive Dynamic Programming Algorithm for the Heterogeneous Resource Allocation Problem

Warren B. Powell
Joel A. Shapiro
Hugo P. Simão

Department of Operations Research and Financial Engineering,
Princeton University, Princeton, NJ 08544
Technical Report CL-00-06

November, 2000

Abstract

We consider an aggregated version of a large-scale driver scheduling problem, derived from an application in less-than-truckload trucking, as a dynamic resource allocation problem. Drivers are aggregated into groups characterized by an attribute vector which capture the important attributes required to incorporate the work rules. The problem is very large: over 5,000 drivers and 30,000 loads in a four day planning horizon. We formulate a problem that we call the *heterogeneous resource allocation problem*, which is more general than a classical multicommodity flow problem. Since the tasks have one-sided time windows, the problem is too large to even solve an LP relaxation. We formulate the problem as a multistage dynamic program and solve it using adaptive dynamic programming techniques. Since our problem is too large to solve using commercial solvers, we propose three independent benchmarks, and demonstrate that our technique appears to be providing high quality solutions in a reasonable amount of time.

We consider the problem of optimizing, in a dynamic setting, the flows of 5,000 drivers who have to move 30,000 loads over a four day planning horizon. The problem arises in the context of planning the movements of drivers and loads for a major trucking company. Given the large numbers, it did not seem practical to plan the movements of each individual driver, but it was necessary to describe drivers using a six dimensional vector of attributes: current location, home domicile, driving hours, sleeper/nonsleeper, bid driver/not-bid, and days away from home. Loads are also described using a vector of attributes that include the origin and destination of the load, how long it has been delayed, the weight of the trailer, and the service requirements of the freight on the trailer. A load that is not moved at time t remains in the system at time $t+1$. Loads may take between two and 10 hours to complete, and a single driver may cover between four and eight loads over a four day horizon. We limit each driver to 10 hours of driving time after starting work, but do not capture details such as rests.

Another dimension of the problem is the evolution of information, which normally includes the loads to be moved, but which may also include information on costs and new drivers entering the system. We assume that decisions at time t must be made using the information available at time t. We may use forecasts of activities or functions that apply to the future, as long as they only use the information available at time t. This assumption makes it easy for us to consider stochastic versions of the problem, meaning that there are multiple possible outcomes of the future. In this paper, however, we wish to focus our attention on the issues that arise with the modeling and solution of heterogeneous resources. Although our techniques are easily adapted to stochastic versions (and we show how to do this), our presentation is made in the context of a problem where there is only one sample outcome of the future. Normally this would be equivalent to a deterministic formulation, but we still enforce the constraints on information availability.

The goal of the project was to develop a system that would run in real time, but which would provide tactical information to a group of centralized planners. As a result, we were not interested in assigning a particular driver to a particular load "here and now" but rather to provide forecasts of available drivers and loads waiting to be moved over a four day horizon. For this reason, we did not need to model each driver individually, but we did need to capture the attributes of a driver with sufficient accuracy to model basic flows. An optimization model was required since we needed to project decisions (assigning drivers to loads, repositioning drivers, holding freight) in the future.

The unique challenge that we faced involved both the large size of our problem, measured in

terms of the number of drivers and loads to be moved, and the relatively complex set of attributes that describe a driver. In principle, we could solve this problem using the column generation techniques developed for airline crew scheduling (Desrosiers, Solomon & Soumis (1995)), but two issues discouraged this approach: the sheer size of the problem (5,000 drivers, 30,000 loads) and the lack of fixed service times for the loads (loads can be served starting at an initial point in time or, at a penalty, any time after that).

Problems with many resources, such as fleet management problems, are typically modeled as multicommodity network flow problems. These models are relatively insensitive to the number of vehicles being managed, but are very sensitive to the number of different states that a resource can occupy. For example, a fairly complex instance of a fleet management problem might involve the flows of, say, 20 types of box cars between 100 locations, producing a state space of 2000 possible car-type/location combinations. By contrast, our problem produces a potential state space of over one million possible resource types. Thus, our problem shares the large state space of a crew scheduling problem and the large number of resources of a fleet management problem.

This problem does not fit cleanly in the framework of standard multicommodity flow formulations. As a result, we are introducing a new problem class, called the *heterogeneous resource allocation problem* (HRAP). Resources in multicommodity flow problems can be described by a commodity class k in a state i; decisions acting on a resource change its state, but not its class. In our problem, it is more convenient to describe a resource by a multidimensional attribute vector a; decisions acting on a resource are allowed to change the complete attribute vector in a general way. We use the term "heterogeneous" to capture the richness of the attribute vector a compared to the much simpler "commodity/state" description of a resource in the multicommodity literature. The HRAP is a special case of the dynamic resource transformation problem (dubbed "DRiP"), introduced by Powell & Shapiro (1999), and our notation is derived from this modeling framework. This paper introduces a mathematical formulation of the HRAP (which is much simpler than the very general formulation provided in Powell & Shapiro (1999)), and suggests a solution approach based on the principle of adaptive dynamic programming (see Powell, Godfrey, Papadaki, Spivey & Topaloglu (2000)). We suggest a method for handling the very large resource attribute space, and demonstrate the effectiveness of both the model and the algorithm on a very large scale driver scheduling problem.

Using the vocabulary of Powell & Shapiro (1999), our driver scheduling problem is a *two-*

layer resource management problem, in that we are managing both drivers and loads. A driver may be held at his current location, assigned to a load (which involves transporting the load to its destination), or repositioned (without a load) to another location. Drivers, then, satisfy the criteria of an *active resource layer*. A load available at time t that is not served is still available at time $t+1$. In our model, a load may be moved to its destination by a driver, or held in its location. Loads, then, satisfy the conditions of a *passive* resource layer. If we required that a load available at time t be moved only at time t, then this problem would reduce to a one-layer problem. In the actual problem, there are some loads which can be moved using "outside drivers" which are not explicitly modeled. If we had considered this dimension of the problem (which arises at only a few locations in the network) then loads would constitute an active resource layer.

The heterogeneous resource allocation problem arises in applications that involve relatively complex resources such as people and sophisticated equipment (such as locomotives and planes), with attributes that evolve over time. The most interesting problems involve drivers who have complex attribute spaces (such as domicile, skill levels, hours of service worked, time from home). The primary attributes that change are those related to time (hours of service worked, time from home). We have also seen instances of history-dependent routing, where a driver's options from terminal i depend on the path used to get to i. Chemical containers often possess the attribute of chemicals that have been carried since the last cleaning, which determine whether a trailer needs to be cleaned before carrying a certain type of product. Locomotives have the attribute of fuel level and maintenance status, as well as the attribute of which train they are attached to.

Prior research on dynamic resource allocation problems can be divided into two classes. The first is crew scheduling (CSP), which involves the routing and scheduling of airline crews, typically with complex attributes, over a set of flight legs. (Crew scheduling problems are deterministic, but the techniques can be applied to deterministic approximations of dynamic problems on a rolling basis.) While a variety of heuristics have been proposed, this class of problems can be solved effectively using column generation techniques. An excellent survey of this approach is presented by Desrosiers et al. (1995). In this context, the CSP is formulated as a set partitioning problem, with rows corresponding to flight legs, and columns corresponding to feasible pairings. In order to keep the number of columns manageable, the set partitioning problem is solved using Dantzig-Wolfe decomposition. Columns may be generated using heuristics (see Crainic & Rousseau (1987)) for maximum flexibility in handling work rules and non-additive path costs. They can, however,

be generated optimally by solving a resource-constrained shortest path problem in an expanded network (Desrosiers, Soumis & Desrochers (1984), Lavoie, Minoux & Odier (1988)).

A second line of research has been in the dynamic fleet management literature, which easily handles fleets of thousands of vehicles, typically under the banner of the dynamic vehicle allocation problem. A review of this field can be found in Powell, Jaillet & Odoni (1995) and Powell (1988). This problem has been formulated deterministically by White (1972) and stochastically by Jordan & Turnquist (1983), Frantzeskakis & Powell (1990), and Cheung & Powell (1996). This work typically involves a single type of flow and tight time windows (the work of Jordan & Turnquist (1983) provides for backlogging).

Optimization models for multiple equipment types with substitution typically produce large (integer) multicommodity flow problems (see, for example, Hane, Barnhart, Johnson, Marsten, Nemhauser & Sigismondi (1995) and Rushmeier & Kontogiorgis (1997)). This problem has usually been solved in practice using network decomposition techniques, or by solving the LP relaxation, and then using rounding heuristics to obtain an integer solution. We could not solve even an LP relaxation of our problem using a commercial solver. We also experimented with network decomposition techniques, again without success. Our biggest difficulty with this class of methods was the one-sided time windows on the loads. Each load may be served at any point after its initial point of availability, although there is an increasing penalty as a load is delayed as the service requirement of each shipment on the trailer is violated.

One of the challenges that we had to address in this paper was the evaluation of the quality of our solution method, which depends on an approximation of the value function in a dynamic program. Prior research into this method, reported in Powell & Carvalho (1998*a*) and Powell & Carvalho (1997), addressed problems that were sufficiently small that continuous approximations could be solved to optimality using a commercial LP solver. By contrast, the problem we address in this paper is far too large to yield to such an approach, regardless of acceptable run times. For this reason, three measures of performance are provided. In the first, we test our adaptation of the method in Powell & Carvalho (1998*a*) on problems that can be solved as a linear program (since the LP produces non-integer solutions, this is a bound). Next, we take the full problem and solve a static approximation to optimality. Both of these benchmarks appear to be very tight. Finally, we compare our results to actual historical performance achieved by the motor carrier, and show that our method produces lower overall empty miles. While none of these three measures provides

a perfect yardstick, combined, they present a fairly compelling case supporting the quality of our solution.

This paper makes four principal contributions. 1) We formulate the aggregated driver management problem as a heterogeneous resource allocation problem. This appears to be a new problem class; it is not a multicommodity flow problem as it is traditionally defined, and the formulation is fundamentally different than that used in crew scheduling. 2) Although we focus our attention on a deterministic formulation (to simplify the notation), we explicitly model the evolution of information, and all of our techniques are easily adapted to stochastic data. Our equations for system dynamics also capture advance information, which is needed both to capture prebooking of loads, but also the difficult problem (in dynamic systems) of multiperiod travel times. 3) We formulate the problem as a multistage linear program, and solve it using adaptive dynamic programming. Although this work builds on the prior work of Powell & Carvalho (1998*a*), this paper appears to be the first to explicitly solve sequences of linear programs, using dual variables to update value function approximations (Powell & Carvalho (1998*a*) use the special structure of single commodity problems to avoid explicitly solving sequences of linear programs). 4) We demonstrate that the technique works well on a very large-scale, real-world dataset. Since no strict bounds were available for a problem of this size, we used three different types of numerical experiments: a) we applied the technique to a simpler problem for which an optimal solution exists, b) we compared our method on the full problem to the optimal solution of a static approximation of the full problem (using different levels of data aggregation), and c) we compared our solution to actual, historical performance. We offer that our method of evaluating our algorithm for such large problems is itself novel, and represents a new approach for evaluating heuristics when applied to problems of this scale.

In the next section we establish the notation which is used to describe the problem. Our notation captures a highly complex problem in a compact form that easily lends itself to introducing a wide range of operational issues in a simple way. We then present, in Section 2, the HRAP in a simultaneous formulation suitable for solution by integer programming. Unfortunately, the size of this formulation, and the presence of wide time windows, make it unsolvable by standard means. In response, we propose in Section 3 a new method for solving the heterogeneous resource allocation problem using adaptive dynamic programming. We show that a dynamic programming formulation of the HRAP is both natural and helpful in studying the problem, although the resulting optimality recursion is computationally intractable. To work around this difficulty, an approximation to the

value function is suggested that leads to the main discussion of this paper. In Section 4, we report on the results of three sets of experiments that provide a measure of the quality of the solution, and demonstrate the computational tractability on a large-scale application.

1 Problem Notation

In the presentation below, sets are always calligraphic (e.g. $\mathcal{D}$), the elements of vectors are always indexed by subscripts (e.g. x_{ijt} or x_t), and different types of vectors or sets are indicated using superscripts (x^h or $\mathcal{D}^l$). Just as we index vectors using subscripts, we indicate a subset of a set using a subscript (e.g. $\mathcal{D}_a$). Time is always indexed using t or t'.

We use the following notation:

Network

$\mathcal{C} =$ The set of cities between which movements are made. We always use i and j to index the set $\mathcal{C}$.

$\tau_{ij} =$ The travel time between cities i and j. We let τ^{max} be the largest travel time in the network.

$\mathcal{T} =$ The discrete set of time periods over which we are modeling our system.

Resources

We represent the resources using:

$\mathcal{A} =$ The attribute space, where $a \in \mathcal{A}$.

$R_{att'} =$ The number of resources that will be available with attribute vector a at time t' that we know about at time t.

$R_{tt'} = \{R_{att'}\}_{a \in \mathcal{A}}$

$R_t = \{R_{tt'}\}_{t' \geq t}$

$\hat{R}_{tt'} =$ The vector of resources that will first become available at time t' that we know about at time t.

The use of aggregation to create the vector $R_{att'}$ is an essential characteristic of our model. In fact, it is a fundamental dimension of the heterogeneous resource allocation problem.

To simplify our presentation, we assume that R_0 is given, and that there are no other changes in the set of resources over the course of our planning horizon. Example 1 provides an illustration of an attribute vector for a particular driver r at time t. Note that in this example there are two *static* attributes (the driver domicile and the sleeper flag) and four *dynamic* attributes.

Example 1(Resource Attribute Vector):

$\mathbf{a_1}$	=	The domicile terminal or 'home base' of driver r,
$\mathbf{a_2}$	=	1 If the driver represents a sleeper team, 0 otherwise.
$\mathbf{a_3}$	=	The current/next terminal of driver r,
$\mathbf{a_4}$	=	The ETA of driver r at his current/next terminal,
$\mathbf{a_5}$	=	The cumulative driving time of driver within a shift. r,
$\mathbf{a_6}$	=	The number of days that a driver has been away from home.

The double time indexing of the resource vector $R_{tt'}$ is relatively new (Powell & Shapiro (1999) introduce this notation for this problem class, but do not actually use it in any solution algorithms). For stochastic problems, it provides for a very general information arrival process where information about a resource can arrive before the resource does. To keep the notation simple, we are presenting all of our information as deterministic, but we are still capturing the evolution of this information over time. Later, this approach is going to prove critical to handling an important dimension of transportation problems, namely multiperiod travel times.

Tasks

Our tasks represent loads that need to be moved. Unlike the drivers, we do not attempt to aggregate the loads.

$\mathcal{L}_{tt'}$ = Set of tasks that we know about at time t that can be acted on at time t'.

$\mathcal{L}_t = \bigcup_{t' \geq t} \mathcal{L}_{tt'}$

$\mathcal{L}_{it}$ = Set of tasks whose origin location is location $i \in \mathcal{C}$ that we know about at time t.

b_l = The attribute vector of task $l \in \mathcal{L}_t$.

$\mathcal{B}$ = The set of possible outcomes of b_l.

$\hat{\mathcal{L}}_{tt'}$ = The set of new tasks that we first learn about at time t that can be acted on at time t'.

Example 2 provides an illustration of the set of task attributes that we considered.

Example 2(Task Attribute Vector):
$\mathbf{b_1}$ = The original terminal of load l.
$\mathbf{b_2}$ = The destination terminal of load l.
$\mathbf{b_3}$ = The time of availability of the load at its origin terminal.
$\mathbf{b_4}$ = The due time of the load at its destination terminal.
$\mathbf{b_5}$ = The priority class of the load.

Decision variables

The types of decisions that we can make are represented using:

$\mathcal{D}_t$ = Set of decisions that we can apply to a resource.

$= \mathcal{D}_t^l \bigcup \mathcal{D}^m \bigcup \mathcal{D}^{hold}$, where:

$\mathcal{D}_t^l$ = The set of decisions to move a load (in this case, an element of $\mathcal{D}^l$ is a load available at that point in time, which means that $\mathcal{D}_t^l = \mathcal{L}_t$; we can think of $l \in \mathcal{L}_t$ as a *task*, while $d \in \mathcal{D}_t^l$ is a *decision to move a task*). For this reason, we define:

l_d = The task $l \in \mathcal{L}$ that corresponds to the decision $d \in \mathcal{D}^l$.

$\mathcal{D}^m$ = The set of decisions to move (reposition) a driver (without a load) to another city (in this case, $\mathcal{D}^m$ represents the set of cities a driver may be moved to).

$\mathcal{D}^{hold}$ = The decision to hold a resource (or do nothing).

$\mathcal{D}_{at}$ = The subset of decisions that can be applied to a resource with attribute a at time t, $\mathcal{D}_a \subseteq \mathcal{D}_t$.

We now define:

x_{adt} = The number of times we act on resources with attribute a with decision d at time t.

$x_t = \{x_{adt}\}_{a \in \mathcal{A}, d \in \mathcal{D}}$

This notation avoids the more common convention of using different variables for loaded and empty movements. It also makes it especially easy to include new classes of decisions (for example, hiring new drivers).

System dynamics

The evolution of our system over time is captured through the system dynamics. At its core, we define the *modify function* which is given by:

$$M(a,d,t) \quad = \quad (a',c,\tau) \tag{1}$$

where a is the vector of attributes that the decision is being applied to, d is the type of decision, and t is the time at which we are making the decision. The outcome of the decision is captured by the triplet (a',c,τ), where a' is the attributes of the modified resource, c is the cost of the action, and τ is the time required to complete the action (we assume throughout that $\tau \geq 1$). It is useful to define:

$c_{adt} =$ The cost of applying decision d to resource a at time t.

$c_t = \{c_{adt}\}_{a\in\mathcal{A},d\in\mathcal{D}}$

$\delta_{a't'}(a,d,t) = \begin{cases} 1 & \text{if } M(a,d,t) \text{ produces a resource with attribute } a' \text{ and if } \tau = t'-t. \\ 0 & \text{Otherwise} \end{cases}$

The dynamics for the resources are given by the equation:

$$R_{a'tt'} \quad = \quad R_{a',t-1,t'} + \hat{R}_{a'tt'} + \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}_{a,t-1}} \delta_{a't'}(a,d,t-1)x_{ad,t-1} \qquad \forall a' \in \mathcal{A}, t' \geq t \tag{2}$$

The variable $R_{att'}$, then, represents the vector of resources that we know about at the beginning of time t which we can act on at time t'.

To formulate the dynamics for the tasks, we first define:

$\mathcal{L}^h_{t-1,t'} =$ The set of tasks in $\mathcal{L}_{t-1,t'}$ that were held at time $t-1$ but which will next be available to be served at time t'. For our problem class, it is always the case that $t' = t$ (a task not served at time $t-1$ can always be served at time t).

When we modeled resources, the set $\mathcal{D}$ includes an explicit decision to hold a resource. With the tasks, we require a special set, $\mathcal{L}^h_{t-1,t'}$ to capture these "hold" decisions. The tasks that are not in $\mathcal{L}^h_{t-1,t'}$ that were in $\mathcal{L}_t$ include those which were served, or which were exogenously removed from the system (cancelled, expired, etc.)

The system dynamics for tasks can now be written:

$$\mathcal{L}_{tt'} = \mathcal{L}_{t-1,t'} \bigcup \hat{\mathcal{L}}_{tt'} \bigcup \mathcal{L}^h_{t-1,t'} \tag{3}$$

There is an intentional symmetry between equations (2) for resources and (3) for tasks. We use vector notation for resources (since they aggregate into a nice state space) and set notation for tasks (because they do not aggregate well). Let:

$$S_{tt'} = \{R_{tt'}, \mathcal{L}_{tt'}\}$$

This allows us to write the state of our system as:

$$S_t = \{S_{tt'}\}_{t' \geq t}$$

It is useful to think of S_t as the *information state* of our system, because $S_{tt'}$ represents what we *know* about the system, even though we cannot act on the information until time t'. We may *plan* on acting on resources in $S_{tt'}$ for $t' > t$, but we may not *act* on them.

2 Optimization Formulation

In this section we present the optimization formulation of the HRAP. We begin by considering the objective function. For each $t \in \mathcal{T}$ we define

$$c_t(x_t) = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}_{at}} c_{adt} x_{adt} \tag{4}$$

Our optimization problem can now be expressed as:

$$\max_x \sum_{t \in \mathcal{T}} c_t(x_t) \tag{5}$$

subject to, for all $t \in \mathcal{T}$:

Physical constraints:

$$\sum_{d \in \mathcal{D}_{at}} x_{adt} = R_{att} \quad \forall a \in \mathcal{A} \tag{6}$$

$$\sum_{a\in\mathcal{A}} x_{adt} \leq 1 \qquad d \in \mathcal{D}_t^l \tag{7}$$

$$x_{adt} \geq 0 \quad \text{and integer} \tag{8}$$

System dynamics:

$$R_{a'tt'} = R_{a',t-1,t'} + \hat{R}_{a'tt'} + \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}_{a,t-1}} \delta_{a't'}(a,d,t-1)x_{ad,t-1} \qquad \forall a' \in \mathcal{A},\ t' \geq t \tag{9}$$

$$\mathcal{L}_{tt'} = \mathcal{L}_{t-1,t'} \bigcup \hat{\mathcal{L}}_{tt'} \bigcup \mathcal{L}^h_{t-1,t'} \qquad t' \geq t \tag{10}$$

The challenge of this formulation is its sheer size. The attribute space $\mathcal{A}$, for our application, has more than a million elements. The total number of tasks to be moved is over 30,000. The physical problem involves moving flows between 550 terminals over 30 (four hour) time periods.

It is useful to briefly compare our problem to the classical multicommodity network flow problem (see, for example, Frangioni & Gallo (1999), Jones, Lustig, Farvolden & Powell (1993), Assad (1987), Kennington & Helgason (1980), Ahuja, Magnanti & Orlin (1992)), which is typically stated in the following form:

$$\min \sum_{k\in\mathcal{K}} \sum_{i,j\in\mathcal{I}} x_{ij}^k \tag{11}$$

subject to:

$$\sum_{j\in\mathcal{I}} x_{ij}^j - \sum_{j\in\mathcal{I}} x_{ji}^k = b_i^k \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{12}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij} \quad \forall i,j \in \mathcal{I} \tag{13}$$

$$x_{ij}^k \geq 0 \quad \forall i,j \in \mathcal{I}, k \in \mathcal{K} \tag{14}$$

where x_{ij}^k is the flow of commodity $k \in \mathcal{K}$ between "states" i and j, $i,j \in \mathcal{I}$. In our version, we would add an additional time index t to account for transit times between states i and j.

Of course, the first big difference between the HRAP and a classical multicommodity flow problem is the number of different types of resources. In a multicommodity flow problem, the sets $\mathcal{K}$ and $\mathcal{I}$ are typically not "too large" making it possible to enumerate all possible combinations of $i \in \mathcal{I}$ and $k \in \mathcal{K}$. In our problem, the attribute vector a may have a number of elements, producing an attribute space $\mathcal{A}$ that is extremely large.

There is a second more technical difference. The formulation captured by equations (11) - (14) assumes that the flow across all commodities k between i and j are restricted by an upper bound u_{ij} that is also indexed by i and j. Our problem lacks this property. Each task is characterized by a vector of attributes b, and if task l has attribute vector b_l, then the cost of assigning a resource with attribute a to task l (at time t) could be represented by c_{alt} (or c_{adt} where $d \in \mathcal{D}^l$ corresponds to task l). The implication is that we do not require that our resource be in any particular state before it serves a task. In principle, *any* resource can serve any task at some cost.

We can convert a multicommodity flow problem into our notation if we view $a = (i, k)$ as the attribute vector of a resource, but it is not possible to write our problem strictly in the form of an MCNF. We would not recommend that MCNF problems be reformulated in our notation, since we lose some of the structural properties of the problem class, but it is important to emphasize that our formulation does, in fact, represent a new problem class. MCNF formulations often arise in transportation when modeling the flows of vehicles, which are generally modeled as relatively simple resources. They are generally characterized by a vehicle type (the commodity) and a state (typically the location). Our resources, representing aggregations of drivers, are much more complex.

Classical crew scheduling models (for example, Desrosiers et al. (1995)), fall in our problem class, but the most common solution approach based on column generation converts the problem into a multicommodity network flow problem by finding itineraries for individual crews using a special subproblem solver (which captures all the physics of an individual crew) which then allows the problem to be reduced to an MCNF in the form of a set partitioning master problem. These techniques can be applied to our problem class, but they do not generally work well with very large problems, or with problems where the tasks have wide time windows (as ours do). Applied to our problem class, standard column generation methods would suffer from the high level of degeneracy that arise in home domiciles where there will be a large number of drivers with the same attribute vector.

3 Solution Algorithm

Our strategy for solving this problem is to use adaptive dynamic programming, outlined in section 3.1. Due to the sheer size of the attribute space, we use a dynamic state generation approach (comparable to row generation in linear programming), described in section 3.2. Throughout, our

presentation is done in the context of a deterministic problem. However, all of the steps of the algorithm can be easily modified to handle stochastic problems by using a simple resampling process at each iteration. The problem is converted to an optimal control problem which requires the use of upper bounds on decision variables to stabilize the solution. The process of updating the upper bounds is given in section 3.3. Finally, section 3.4 describes the complete algorithm.

3.1 Adaptive dynamic programming

We propose to solve our problem using adaptive dynamic programming. We start by writing the dynamic programming recursion:

$$V_t(S_t) = \max_{x_t} c_t(x_t) + V_{t+1}(S_{t+1}(x_t)) \tag{15}$$

subject to constraints (6) through (8), where $V_T(S_T) = 0$. Classical discrete dynamic programming techniques cannot be applied to problem (15). The standard problem of a large state space certainly applies here, but recent progress in forward dynamic programming partially mitigates the state space issue (see Bertsekas & Tsitsiklis (1996) and Sutton & Barto (1998)). The real problem in our setting is the size of the feasible region, which eliminates the use of any algorithms that require enumerating over all possible decisions. This means that even if we knew the value function for all possible (discrete) values of the state S_{t+1}, we still could not solve the problem.

Instead, we turn to adaptive dynamic programming using functional approximations for the value function. Our goal in the design of a functional approximation for $V_t(S_t)$ is to retain the basic structure of the one-period problem. If we drop the value function from equation (15) completely, then we have a simple transportation problem, illustrated in figure 1. Here, we have to assign supplies of drivers, given by the vector R_t, to individual loads, given by the set $\mathcal{L}_t$. The cost of assigning a driver in bucket a to a load l with attribute vector b_l at time t is given by c_{alt} (or equivalently, c_{adt} with $d \in \mathcal{D}^l$). The number of links in this network are dramatically reduced by carefully constructing the sets $\mathcal{D}_a$. For example, we only consider assigning drivers at a location i to loads that originate at location i. A driver in location i may move a load in location j by first repositioning from i to j and then taking a load.