Journal of Machine Learning Research 9 (2008) 2079-2111 Submitted 8/08; Published 10/08

Value Function Approximation using Multiple Aggregation for Multiattribute Resource Management

Abraham George AGEORGE@PRINCETON.EDU
Warren B. Powell POWELL@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

Sanjeev R. Kulkarni KULKARNI@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA

Editor: Sridhar Mahadevan

Abstract

We consider the problem of estimating the value of a multiattribute resource, where the attributes are categorical or discrete in nature and the number of potential attribute vectors is very large. The problem arises in approximate dynamic programming when we need to estimate the value of a multiattribute resource from estimates based on Monte-Carlo simulation. These problems have been traditionally solved using aggregation, but choosing the right level of aggregation requires resolving the classic tradeoff between aggregation error and sampling error. We propose a method that estimates the value of a resource at different levels of aggregation simultaneously, and then uses a weighted combination of the estimates. Using the optimal weights, which minimizes the variance of the estimate while accounting for correlations between the estimates, is computationally too expensive for practical applications. We have found that a simple inverse variance formula (adjusted for bias), which effectively assumes the estimates are independent, produces near-optimal estimates. We use the setting of two levels of aggregation to explain why this approximation works so well.

Keywords: hierarchical statistics, approximate dynamic programming, mixture models, adaptive learning, multiattribute resources

1. Introduction

We consider the problem of managing resources (people, equipment) that can be described using a vector of attributes $a = (a_1, a_2, \ldots, a_M)$. Our work has grown out of a series of projects with industry and the military that involve managing resources over time under uncertainty. In all of these projects, we use algorithms that require estimating the marginal value of a resource with attribute vector a. As these projects have made the transition from laboratory experiments to industrial implementations, we have found that one and two dimensional attributes (for example, location and possibly equipment type) quickly grow to five or ten dimensions, with an exponential growth in the number of potential attributes. Examples of actual projects we have worked on which exhibit this behavior include:

- Managing pilots for business jets - The attributes of a pilot include elements such as home city, number of days away from home and the equipment that he is trained to fly. Decisions about pilots can include assigning a pilot to a particular flight, or a decision to send a pilot for training on a new type of aircraft.

- Managing locomotives - The decision to assign a particular locomotive to a particular train has to consider attributes such as the type of locomotive, the number of days until it has to be maintained, its current location and its home maintenance shop.

- Managing a fleet of freight cars - Freight cars have attributes such as location, time until arrival to a destination, loaded or empty status, ownership, and maintenance status.

- Managing a fleet of trucks to move loads - The truck can be described using attributes such as its current location, the home domicile of the driver, the maintenance level and whether it is being driven by a solo driver or a team of two drivers. Decisions include where to move to and whether to move loaded or empty.

- Managing cargo aircraft for the military - We have to decide which aircraft should be assigned to satisfy a particular requirement (a movement of freight or passengers). Choosing the best aircraft requires knowing the value of an aircraft at the destination which depends on the type of aircraft, cargo configuration, whether it is loaded or empty (and if loaded, the load characteristics), and its maintenance status.

- Managing blood inventories - Blood is characterized by blood type, age, location, and whether it has been frozen. New supplies of, and the demand for, blood is random.

All of these are examples of resource allocation problems where a decision has to be made now to act on resources (trucks, jets, locomotives) which will bring about a change in their attributes. Let $a \in \mathcal{A}$ be the attribute vector describing a resource now. If we act on the resource, we may produce a resource with attribute a' with value $v_{a'}$. In a dynamic programming setting, the value $v_{a'}$ refers to the solution of a finite horizon discounted reward dynamic program. In practical applications, we cannot compute $v_{a'}$ exactly, so we resort to Monte Carlo methods where we might observe random observations $\hat{v}_{a'}$ and use these to produce a statistical estimate $\bar{v}_{a'}$ (see Bertsekas and Tsitsiklis 1996 and Sutton and Barto 1998 for an introduction to the techniques of approximate dynamic programming). The problem is that in realistic problems, the attribute space $\mathcal{A}$ can be extremely large, and we may obtain only a few observations of $\hat{v}_{a'}$ for a particular a'. As a result, the statistical error in $\bar{v}_{a'}$ can be quite large.

One of the standard strategies in approximate dynamic programming is to aggregate the state (attribute) space. Instead of estimating $\bar{v}_a$, we might define an aggregation function $G(a)$ which produces an aggregated attribute $\bar{a}$ which has fewer outcomes. For example, a five-digit zip code can be aggregated up to a three-digit zip; a numerical attribute can be divided into fewer ranges; or an attribute can be completely ignored. The resulting smaller attribute space produces more observations of each attribute, but at a cost of aggregation error.

There are a variety of statistical strategies for estimating value functions which take advantage of the structure of a specific attribute vector a. In a trucking problem, we might design a statistical function that depends on the location of a driver, his days away from home, the fuel level of his tank and his home domicile. However, after designing a statistical model that works for this application, we would have to start from scratch if we wished to switch to another application. In fact, simply adding an attribute would require redesigning and refitting the statistical equation. This can be particularly hard when several of the attributes are categorical, and which interact to determine the effect of the attributes on the system. A truck driver might be characterized by his location and his home domicile; the value of a driver at a location depends very much on where he lives.

We are interested in developing a method for estimating the value $\bar{v}_a$ of a resource with attribute a, making minimal assumptions about the structure of the attribute space. We take advantage of the fact that for every application with which we are familiar, it is quite easy to design a family of aggregation functions $\mathcal{G}$ where $G^{(g)} : \mathcal{A} \rightarrow \mathcal{A}^{(g)}$ is an aggregation of the attribute space $\mathcal{A}$. For example, we can create an aggregation function simply by ignoring an attribute. Aside from assuming the existence of this family of functions, we make no further assumptions about the nature of the attribute space. For example, we do not even require the existence of a metric that would provide a measure of the distance between two attribute vectors, which prevents the use of standard methods such as non-parametric statistics or regression trees.

Aggregation has traditionally been a powerful technique in dynamic programming. A good general review of aggregation techniques is given by Rogers et al. (1991). Aggregation strategies in a dynamic programming setting may be governed by the desire to solve exactly a smaller dynamic program, or by the iterative nature of the algorithms. Techniques range from picking a fixed level of aggregation (Whitt, 1978; Bean et al., 1987; Athans et al., 1995; Zhang and Sethi, 1998; Wang and Dietterich, 2000), or using adaptive techniques that change the level of aggregation as the sampling process progresses (Mendelssohn, 1982; Bertsekas and Tsitsiklis, 1996; Luus, 2000; Kim and Dean, 2003), but which still use a single level of aggregation at any given time (many authors used a fixed level of aggregation to produce a smaller Markov Decision Process (MDP) that can be solved optimally). Tsitsiklis and Van Roy (1996) (see also Bertsekas and Tsitsiklis, 1996) show how value functions can be approximated using a fixed set of *features*; this strategy encompasses both static and hierarchical aggregation as special cases, but the use of these techniques in our setting is prohibitive because of the extremely large number of values that need to be estimated. Feng et al. (2003) presents a work that identifies state aggregations based on "structural similarity" where states are considered similar if they have similar value estimates or similar sets of successor states, rather than "input similarity" which is typically measured by some distance metric defined over the state space. Bertsekas and Castanon (1989) introduces a creative approach which adaptively clusters states with similar values of residual errors at each iteration, requiring no structure among the states of the system. While we also do not have any structure, we do take advantage of the presence of a family of aggregation functions, and our technique does not require the overhead of solving clustering problems. A nice discussion of aggregation and abstraction techniques in an approximate dynamic programming setting is given in Boutilier et al. (1999).

Instead of using a single level of aggregation, researchers have considered combining estimates from all the levels of aggregation at once. In the literature, there exist several techniques for combining estimates to improve accuracy (see Wolpert, 1992; LeBlanc and Tibshirani, 1996; Yang, 2001). It is well-known that if the estimates being combined are independent and unbiased, then it is optimal to combine them in inverse proportion to their variances (Guttman et al., 1965). When different estimates are based on different levels of aggregation, they are neither independent nor unbiased. It is also possible to use a weighted combination of estimates where the weights are estimated using regression techniques. For our application, there can be hundreds of thousands of such models, making the updating of regression models computationally expensive.

In this paper, we solve the problem of optimally combining (correlated) value estimates at different levels of aggregation in an approximate dynamic programming setting and derive expressions for optimal weights. The result generalizes a well-known result for optimally combining independent estimates. We point out that the independence assumptions used for deriving the results are true only in idealized regression settings, and not in an approximate dynamic programming setting.

The major contribution of this paper lies in finding that an inverse-variance weighting formula (adjusted for bias), which is optimal only when the estimates are independent, proves to be near-optimal even though estimates at different levels of aggregation are not independent. We explain this behavior analytically for the case with two levels of aggregation. We show that if we compute optimal weights (without assuming independence) and compare the results if we do assume independence, the results are the same for two extremes: when the difference between the aggregate and disaggregate value estimates is very large or very small. We show experimentally that the error for intermediate values is extremely small.

We also show, in the context of a single vehicle routing problem, that our weighting method produces value function estimates that are within five to ten percent of the optimal value functions, outperforming other estimates. The method of weighting a family of aggregate estimates is shown to naturally shift the weight from aggregate to disaggregate estimates as the algorithm progresses. We also demonstrate that this method is easy to implement in large-scale, on-line learning applications that arise in approximate dynamic programming, where it produces much faster convergence (which implies approaching a consistently better solution quality in a fewer number of iterations) than would be produced using a single, static level of aggregation. Further work on this application is explained in detail in Simao et al. (2008).

The paper is organized as follows. In Section 2, we describe a generic approximate dynamic programming technique, which estimates the value functions associated with various states. This section provides an introduction to the context in which our statistical estimation problem arises. The next three sections, however, focus purely on the statistics of aggregation outside of a dynamic programming setting. Section 3 provides a theoretical model of the sampling process and defines bias and variance for aggregated statistics. Then, Section 4 poses the problem of computing optimal weights for combining estimates of values at different levels of aggregation. The problem with this formula is that it is too expensive to use for our problem class. For this reason, we propose a simpler formula that assumes that statistics from different levels of aggregation are independent. In Section 5, we compare the two weighting formulas (with and without the independence assumption) for the special case where there are only two levels of aggregation which allows the optimal weights to be computed analytically. We show theoretically that assuming independent estimates introduces zero expected error at two extremes of the problem. We then show experimentally that ignoring the dependence between the estimates gives results that are very similar. In Section 6, we demonstrate our approximation method in the context of an approximate dynamic programming algorithm for solving a multiattribute resource allocation problem. We use both a single truck problem, which can be solved exactly, as well as a problem of managing a large fleet of trucks. We provide our concluding remarks in Section 7.

2. Approximate Dynamic Programming

This section is designed as a brief introduction to approximate dynamic programming, and introduces the context in which our problem arises. Our interest lies in the context of dynamic resource allocation problems. Dynamic programming techniques can be applied to solve these problems which are typically modeled as MDPs. Using the notation of Powell (2007), we let S_t be the state of our system. We also let $d \in \mathcal{D}$ be a type of a decision, and we let $x_d = 1$ if we choose decision d, and $x_d = 0$ otherwise. $x_t = (x_d)_{d \in \mathcal{D}}$ is the vector of decisions that we make at time t. Bellman's equation allows us to express the value of being in state S_t as

$$V(S_t) \quad = \quad \max_{x_t} \left[C(S_t, x_t) + \mathbb{E}\left\{ V\left(S_{t+1}(S_t, x_t, W_{t+1})\right) | S_t \right\} \right],$$

where W_{t+1} is a random variable representing new information that arrives between t and $t+1$. The exact values can be determined using traditional backward dynamic programming techniques such as value iteration and policy iteration. In these methods, the values are computed recursively starting from the final state, making use of the state transition probabilities.

When the state and action spaces become large, as in most real-life stochastic planning problems, it is not practical to enumerate the states to determine their values. In such problems, compact feature-based representations of the MDP, also called factored MDPs (see Boutilier et al., 2000) can be used to make the problem computationally tractable. Factored MDPs can be represented using a factored state transition model and a reward function that is additive. In these representations, a smaller set of variables (also called features or attributes) are used to describe the state of the system.

Dynamic resource allocation problems span dynamic vehicle routing (Gendreau and Potvin, 1998; Ichoua et al., 2005), where there has been recent interest in the application of approximate dynamic programming for the single vehicle routing problem (Secomandi, 2000, 2001). Powell and Carvalho (1998) uses an approximate dynamic programming algorithm for a fleet management problem, but the attributes of the vehicles were very simple. Powell et al. (2002) uses an approximate dynamic programming algorithm for multiattribute resources, but does not address statistical sampling issues. Spivey and Powell (2004) applies approximate dynamic programming for optimizing a fleet of vehicles, using a linear value function approximation that also requires estimating the value of a resource characterized by a vector of attributes. This research estimated the value of a resource at different levels of aggregation, but kept track of the variance of these estimates at each level of aggregation and always used the estimate that provided the smallest variance.

Resource allocation problems can be modeled by letting $a \in \mathcal{A}$ be an attribute vector (a may consist of categorical and numerical attributes), and by letting R_{ta} be the number of resources with attribute a. We then let $R_t = R_{ta})_{a \in \mathcal{A}}$ be the resource state vector. This research addresses problems where the vector a is large enough that the attribute space $\mathcal{A}$ becomes too large to enumerate. We develop these ideas in the context of a single entity. If a_t is the attribute of the entity at time t, then a_t is effectively our state variable.

In this section, we describe the basic approximate dynamic programming (ADP) strategy to solve the problem of managing a single resource with multiple attributes, the *nomadic trucker*. This is a single resource version of the dynamic fleet management problem, where there is a single trucker who needs to move between various locations to cover loads that arise and gains rewards in the process.

The state of the resource is defined by an attribute vector, a, composed of multiple attributes, which may be numerical or categorical. For the nomadic trucker problem, examples of attributes include the location of the truck, the home domicile of the driver and the number of hours driven. We could represent the attribute vector as $a = (a_{location}, a_{time}, a_{domicile}, \ldots)$. The state space, $\mathcal{A}$, for this problem would consist of all possible combinations of the attributes of the trucker. We can let the decision be represented by the vector $(x_d)_{d\in\mathcal{D}}$, but for a single entity problem, $\sum_{d\in\mathcal{D}} x_d = 1$, which means we can also write the problem as choosing a decision $d \in \mathcal{D}$. Typically, the set of potential decisions depends on the current state (attributes) of our resource, so we let $\mathcal{D}_a$ be the decisions available to a resource with attribute a. We assume that the impact of a decision d on a resource with attribute a is deterministic, and is given by the function $a' = a^M(a,d)$.

In approximate dynamic programming, we sample the various states by choosing decisions that are locally optimal based on current estimates of the value functions. For example, we could follow a procedure where we choose a decision that maximizes the sum of the one-period rewards and the future value (discounted by factor γ) as follows:

$$d(a,\omega) \quad = \quad \arg\max_{d\in\mathcal{D}_a(\omega)} \left\{ c(a,d,\omega) + \gamma v_{a^M(a,d)} \right\}.$$

Here, ω represents a sample realization of random information (for example, $\mathcal{D}_a(\omega)$ is a sample realization of the decision set), $a^M(a,d)$ is the state at the destination and $v_{a^M(a,d)}$ the value associated with $a^M(a,d)$. This model is easily generalized to handle stochastic transitions, but this is not relevant to the focus of this paper.

We outline the steps of a typical approximate dynamic programming algorithm for the nomadic trucker problem in Figure 1. This algorithm has two stages. In the forward pass, we use the current estimates of the optimal value functions to simulate a sample trajectory of the truck. The next state that is visited is determined using a transition function $a^M(a_m,d_m)$, as depicted in Equation 2, where the resource in state a_m undergoes a transformation to state $a_{m+1} = a^M(a_m,d_m)$ when acted upon by decision d_m. Once the end of the time horizon is reached, we perform a backward pass, where we first compute the observations of values of the various states in the current sample path using Equation 3. We point out that the estimates of the future values are discounted by a factor γ. We then use these to update the value estimates, as in Equation 4, and the associated statistics (number of observations and sample variance) of the states that are visited.

There are a number of variations of approximate dynamic programming. One family is known as $TD(\lambda)$-learning (see Sutton, 1988; Sutton and Barto, 1998), typically parameterized by an artificial discount factor λ. Using a pure forward pass algorithm is equivalent to $TD(0)$, while another variation follows a policy (determined by the current set of approximations), and then does a backward traversal to obtain updates of the estimate of the value of being in each state (this is equivalent to $TD(1)$). Another popular strategy is Q-learning (see Watkins, 1989), where we estimate the quantities $Q(a,d)$ which is the value of being in a state a and making decision d. Since the statistical problem of estimating the value of a state-action pair is, of course, even harder than the problem of estimating the value of being in a state, we have not used this approach. Since Q-learning allows you to determine a decision directly from the Q-factors (rather than solving an optimization problem), it is typically presented as a "model-free" algorithm (that is, one that does not require an explicit model of the transition function), although estimating the Q-factors does require some source that determines the next state given a state and action. All of these methods can be used without an explicit model of the exogenous information process (for example, we do not use a one-step transition function) as long as we have some mechanism for creating the sample realizations.

As with most ADP algorithms, the only way to obtain an estimate of the value of being in a state is to actually visit the state. In real applications, there may be millions of states but we may be limited to only thousands of observations. In practice, most states are never visited, and many are

```
Step 0. Initialize an approximation for the value function $\bar{v}_a^0$ for all attribute vector states $a \in \mathcal{A}$
    and set $n = 1$ .
Step 1. Iteration $n$:
    Step 2. Forward pass: Set $m = 0$ and randomly sample attribute vector $a_m$, but fixing the
        start time at the beginning of the time horizon.
        Step 3. Obtain the set of possible decisions, $\mathcal{D}_{a_m}(\omega)$.
        Step 4. Solve for the optimal decision, given the current value function estimates.
            $d_m(\omega) = \arg\max_{d \in \mathcal{D}_{a_m}(\omega)} \left[ c(a_m, d) + \gamma \bar{v}^{n-1}_{a^M(a_m,d)} \right]$   (1)
        Step 5. Evaluate the next state to visit:
            $a_{m+1} = a^M(a_m, d_m)$   (2)
        Step 6. If the end of the time horizon ($T$) is reached, then set $m = m+1$ and go to step
            3, else go to step 7.
    Step 7. Backward pass: For $j = m-1, m-2, \cdots, 0$, update the value function estimates as
        follows:
            $\hat{v}^n_{a_j} = c(a_j, d_j) + \gamma \hat{v}^n_{a_{j+1}}$   (3)
            $\bar{v}^n_{a_j} = (1-\alpha)\bar{v}^{n-1}_{a_j} + \alpha \hat{v}^n_{a_j}$   (4)
Step 8. Let $n = n+1$. If $n < N$ go to step 1, else for each state $a$, return the value function $\bar{v}^n_a$.
```

Figure 1: An approximate dynamic programming algorithm using a backward pass for the nomadic trucker problem

visited only a few times. As a result, there can be a high level of statistical noise in our estimates of the value of being in a state.

This section provides the context in which our adaptive learning problem arises. The next three sections consider the general problem of estimating a quantity (the value of a resource with attribute a) outside of the context of approximate dynamic programming. We assume we have a source of (unbiased) observations of the value associated with attribute a, from which we have to develop statistically robust (i.e., low-variance) estimates of the value associated with attribute a. We then use the method in the context of approximate dynamic programming to demonstrate that it produces better results than other methods, even though we no longer have unbiased observations.

3. The Statistics of Aggregation

In this section, we investigate the statistics of aggregation by studying a sampling process where at iteration n we first sample the attribute vector $a = \hat{a}^n$. We then use a sample realization of the random information which provides us with an unbiased observation of the value of the resource $\hat{v}^n$, producing a sequence of observations of (attribute vector, value) pairs. We wish to use this information to produce a statistically reliable estimate of the true value associated with a. The analysis in this section is not done in the context of dynamic programming (which allows us to assume that our observations of values are unbiased). Rather, it is intended as a pure study of the statistics of aggregation.

Our assumption that the observations of values, $\hat{v}^n$, are unbiased will not be true in a dynamic programming setting, but allows us to focus on the tradeoff between bias and variance.

We begin by defining the following:

$\mathcal{N} =$	The set of indices corresponding to the observations of the attribute vectors and values.
$\mathcal{S} =$	A sample of observations $(\hat{a}^n, \hat{v}^n)_{n\in\mathcal{N}}$.
$\nu_a =$	The true value associated with attribute vector a.
$N_a =$	The number of observations of attribute vector a given our sample $\mathcal{S}$.
$\hat{a}^n =$	The attribute vector at observation n.
$\hat{v}^n =$	The observation of the value corresponding to index n.
$1_{\{\hat{a}^n=a\}} =$	1, if the nth observation is of attribute vector a.

An estimate of ν_a can be obtained as an average across all the observations of values corresponding to a:

$$\bar{v}_a = \frac{1}{N_a}\sum_{n\in\mathcal{N}} \hat{v}^n 1_{\{\hat{a}^n=a\}}.$$

Throughout our presentation, we use the hat notation (as in $\hat{v}$) to represent exogenous information, and bars (as in $\bar{v}$) to represent statistics derived from exogenous information.

Consider a case where the attribute vector has more than one dimension, with A_i denoting the number of distinct states that attribute a_i can assume. The number of values that need to be estimated is $\prod_i A_i$. Needless to say, as the attribute vector grows, the state space grows exponentially, making it impossible to obtain statistically reliable estimates. One strategy is to resort to aggregation (such as dropping one or more dimensions of a) which can quickly reduce the number of values but introduces structural error. An alternative is to assume a structural property such as separability, which reduces the number of values to be estimated to $\sum_i A_i$. This has fewer values, but requires that we introduce separability as an approximation. In one of our trucking applications, one attribute is the location of the truck, while a second attribute is the driver's home domicile. The value of a driver in a location depends very much on his home domicile. Assuming these are independent would introduce significant errors.

In general, aggregation of attribute vectors is performed using a collection of aggregation functions, $G^g : \mathcal{A} \rightarrow \mathcal{A}^{(g)}$, where $\mathcal{A}^{(g)}$ represents the g^{th} level of aggregation of the attribute space $\mathcal{A}$. We define the following:

$a^{(g)} =$ $G^g(a)$, the g^{th} level aggregation of the attribute vector a.

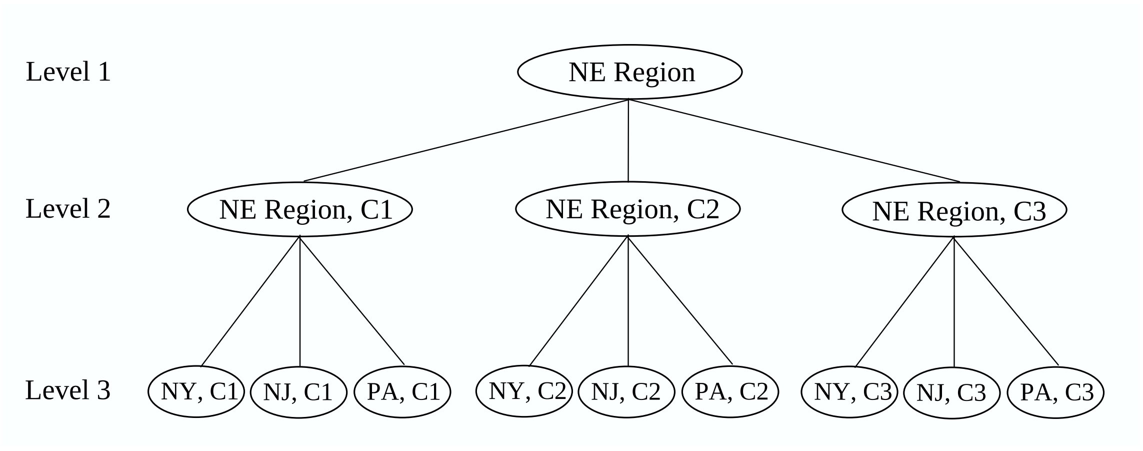

Figure 2: Aggregation of the state space for a multiattribute problem.

$\mathcal{G} =$ The set of indices corresponding to the levels of aggregation.

Aggregation can thus be used to create a sequence of state spaces, $\left\{\mathcal{A}^{(g)}, g = 1, 2, \ldots, |\mathcal{G}|\right\}$, with fewer elements than the original state space. This can be better illustrated using the example in Figure 2, where we consider the nomadic trucker problem with the state of the truck defined by two attributes - current location and capacity type. The number of possible states with three locations (NY, NJ and PA), and three capacity types (C1, C2 and C3), is nine at the most disaggregate level. The first-level aggregation function, $G^{(1)}$, involves aggregating the location to the regional level which reduces the number of states to three. The second-level aggregation function, $G^{(2)}$, would be defined as aggregating out the capacity type attribute completely, which leaves us with a single state. As in this example and in the experimental work to follow, it is usually the case that the gth level of aggregation acts on the $(g-1)$st level.

We let ε^n denote the error in the nth observation with respect to the true value associated with $\hat{a}^n$ (which, using the notation defined earlier in this section, would be represented using $\nu_{\hat{a}^n}$). For analysis purposes, we assume that the elements of the sequence $\{\varepsilon^n\}_{n \in \mathcal{N}}$ are independent and identically distributed, with a mean value of zero. This is, of course, an idealization, but it will help us understand the tradeoffs between structural errors (due to aggregation) and statistical errors. We can express the observed value as follows:

$$\hat{v}^n = \nu_{\hat{a}^n} + \varepsilon^n.$$

We define the following probability spaces,

$\Omega^a =$ The set of outcomes of observations of attribute vectors.

$\Omega^\varepsilon =$ The set of outcomes of observations of the errors in the values.

$\Omega =$ The overall set of outcomes

$= \Omega^a \times \Omega^\varepsilon.$

$\omega = (\omega^a, \omega^\varepsilon)$

$=$ An element of the outcome space.

We now define the following terms which will be useful in obtaining an estimate of the value associated with the attribute vector a at any level of aggregation:

$\mathcal{N}_a^{(g)}$ = The set of indices that correspond to observations of the attribute vector a at the gth level of aggregation

$= \{n \mid G^g(\hat{a}^n) = G^g(a)\}$.

$N_a^{(g)} = \left|\mathcal{N}_a^{(g)}\right|$.

$\bar{v}_a^{(g)}$ = The estimate of the value associated with the attribute vector a at the gth level of aggregation, given the sample, $\mathcal{N}$.

We can compute the estimate, $\bar{v}_a^{(g)}$, as

$$\bar{v}_a^{(g)} = \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \hat{v}^n.$$

We provide a numerical example to illustrate the idea of forming estimates at different levels of aggregation. Consider the state of a resource to be composed of two attributes, namely, location of the resource and resource type. There are four locations, namely, New York, Philadelphia, Boston and Washington. The type can be Single or Team. Thus, there are eight possible states. We use aggregation functions that aggregate out the type attribute and then the location attribute to obtain three different levels of aggregation. Suppose we have the following observations of state-value

a	Location	Type	N_a	$\bar{v}_a$	$N_a^{(1)}$	$\bar{v}_a^{(1)}$	$N_a^{(2)}$	$\bar{v}_a^{(2)}$
a_1	New York	Single	2	4.5	3	5.3	12	4.8
a_2	New York	Team	1	7.0				
a_3	Philadelphia	Single	3	3.7	4	3.3		
a_4	Philadelphia	Team	1	2.0				
a_5	Boston	Single	2	8.5	2	8.5		
a_6	Boston	Team	0	-				
a_7	Washington	Single	1	1.0	3	4.0		
a_8	Washington	Team	2	5.5				

Table 1: Numerical example illustrating the computation of value estimates using aggregation. For example, $\bar{v}_{a_1}^{(0)} = (7+2)/2 = 4.5$, and $\bar{v}_{a_7}^{(1)} = (5+1+6)/3 = 4.0$.

pairs - $\{(a_3,4),(a_4,2),(a_1,7),(a_5,8),(a_3,2),(a_8,5),(a_7,1),(a_5,9),(a_8,6),(a_2,7),(a_3,5),(a_1,2)\}$. We can form estimates of the values of the various states at the different levels of aggregation as illustrated in Table 1.

Now that we have an estimate of the value, ν_a, for each level of aggregation, the question arises as to what is the best level of aggregation. A traditional strategy is to choose the right level of aggregation by trading off statistical and structural errors to find a model with the least overall error.

In order to better understand these two kinds of errors in an aggregation setting, we first let $\overline{\delta}_a^{(g)}$ denote the total error in the estimate, $\overline{v}_a^{(g)}$, from the true value associated with attribute vector a:

$$\overline{\delta}_a^{(g)} = \overline{v}_a^{(g)} - \nu_a.$$

An important component of our prediction error will be aggregation bias. Consider our most recently observed attribute vector $\hat{a}^n$ and some other attribute a, where $\hat{a}^n$ and a may aggregate up to the same aggregated attribute at some level $g \in \mathcal{G}$, that is, $G^g(a) = G^g(\hat{a}^n)$ (for the moment, these are simply two attribute vectors). In our derivations below, it is useful to define a bias term,

$$\mu_a^n = \nu_{\hat{a}^n} - \nu_a.$$

We can use this notation to rewrite $\hat{v}^n$ as follows:

$$\begin{aligned}\hat{v}^n &= \nu_a + (\nu_{\hat{a}^n} - \nu_a) + \varepsilon^n \\ &= \nu_a + \mu_a^n + \varepsilon^n \quad \forall\, a, n.\end{aligned}$$

We can express $\overline{v}_a^{(g)}$ in terms of its bias and noise components as follows:

$$\begin{aligned}\overline{v}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} (\nu_a + \mu_a^n + \varepsilon^n) \\ &= \nu_a + \left(\frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \mu_a^n \right) + \left(\frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \varepsilon^n \right).\end{aligned}$$

We let,

$$\begin{aligned}\overline{\mu}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \mu_a^n, \\ \overline{\varepsilon}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} .\varepsilon^n.\end{aligned}$$

This enables us to express the total error as follows:

$$\overline{\delta}_a^{(g)} = \overline{\mu}_a^{(g)} + \overline{\varepsilon}_a^{(g)} \tag{5}$$

where $\overline{\mu}_a^{(g)}$ gives an estimate of the bias between the values of a at the gth level of aggregation and at the disaggregate level. $\overline{\mu}_a^{(g)}$ is a random variable that is a function of the set of points sampled. $\overline{\varepsilon}_a^{(g)}$ is an estimate of the random error that has zero expected value. By assumption, the variability in $\overline{\varepsilon}_a^{(g)}$ occurs because of the statistical noise in the observation of the values. We point out that the terms, $\overline{\delta}_a^{(g)}$, $\overline{\mu}_a^{(g)}$ and $\overline{\varepsilon}_a^{(g)}$, are not statistical estimators, because a knowledge of the true values is required for computing these. $\overline{\mu}_a^{(g)}$ is representative of the structural error that is introduced due to aggregation, while $\overline{\varepsilon}_a^{(g)}$ represents the statistical error due to noise in the observations. Moreover, these two error terms need not be uncorrelated in a general setting.

In an approximate dynamic programming setting, the right tradeoff between statistical and structural errors will change as we collect more observations. Furthermore, we generally do not control the sampling process of the attributes, and we will encounter instances where some regions of the attribute space $\mathcal{A}$ will be sampled more than others. Although it is common in practice to choose a single level of aggregation that produces the lower overall error, it can be useful to combine estimates from several levels of aggregation.

4. Combining Estimates

In this section, we propose methods to compute weights to combine value estimates that have been formed from a given set of observations. In the context of ADP, the weights are computed at a given iteration of the algorithm in Figure 1.

We consider a set of estimates, $\left\{\bar{v}_a^{(g)}, g \in \mathcal{G}\right\}$, of a value, ν_a, at different levels of aggregation. We let $\sigma_a^{(g)}$ denote the population standard deviation associated with the observations used to compute $\bar{v}_a^{(g)}$. Breiman (1996) proposes a method called stacked regression which in our setting would be equivalent to combining estimates at different levels of aggregation using

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

where $w^{(g)}$ is a set of weights for each level of aggregation. This method ignores the important feature that the best weighting depends on how many times we have observed a particular attribute. We prefer to use the strategy suggested by LeBlanc and Tibshirani (1996) (Section 8), where the weights depend on the attribute:

$$\bar{v}_a = \sum_{g \in \mathcal{G}} w_a^{(g)} \cdot \bar{v}_a^{(g)}.$$

The practical challenge here is that we have to estimate a set of weights $(w_a^{(g)})$ for *each* attribute a (that we observe). If we use classical regression methods for our applications, this can mean maintaining hundreds of thousands of regression models. Storing and updating these models is computationally demanding for large industrial applications. In this section, we develop both exact and approximate methods for estimating weights, where our approximation makes the assumption that the estimates $\bar{v}_a^{(g)}$ are independent. Section 5 presents theoretical and experimental arguments supporting the accuracy of the weights when we assume independence (even when the assumption is not even approximately true), which dramatically simplifies the procedure.

In Section 4.1, we formulate the problem of finding the optimal weights for the general case where the estimates may be biased and dependent on each other, and later derive these weights. However, the computation of these weights can prove cumbersome for large-scale problems. We provide, in Section 4.2, a simpler formula for the weights which assumes that the estimates are independent, but accounts for the possibility of biases in the estimates. In Section 4.3, we propose an approximation of the weights derived in Section 4.2, for the case where the bias and variance of the estimators are unknown.

4.1 Optimal Weights

We begin by finding the weighting scheme that will optimally combine the estimates at the different levels of aggregation, that is, the weights which give a combined estimate with the least squared deviation from the true value associated with attribute vector a. We can formulate the problem as follows:

$$\min_{w_a^{(g)}, g\in\mathcal{G}} \mathbb{E}\left[\frac{1}{2}\left(\sum_{g\in\mathcal{G}} w_a^{(g)} \cdot \bar{v}_a^{(g)} - \nu_a\right)^2\right], \tag{6}$$

subject to:

$$\sum_{g\in\mathcal{G}} w_a^{(g)} = 1. \tag{7}$$

In a setting where the estimates are unbiased, it is useful to have an affine combination of the estimates (LeBlanc and Tibshirani, 1996, Section 2) since the individual estimates and hence the affine combination are equal to the true value in expectation. Even though this is not necessarily true in a general setting, we choose to retain this constraint.

We state the following proposition for computing the optimal weights that solves the problem formulated in Equations 6-7:

Proposition 1 *For a given attribute vector, a, the optimal weights, $w_a^{(g)}$, $g \in \mathcal{G}$, to combine individual estimates that are correlated in a hierarchical fashion, are obtained by solving the following system of linear equations in (w,λ):*

$$\sum_{g\in\mathcal{G}} w_a^{(g)} \mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right] - \lambda = 0 \quad \forall\ g' \in \mathcal{G}, \tag{8}$$

$$\sum_{g\in\mathcal{G}} w_a^{(g)} = 1. \tag{9}$$

If the bias error, $\bar{\mu}_a^{(g)}$, is uncorrelated with the random error, $\bar{\varepsilon}_a^{(g)}$, then the coefficients of the weights in Equation 8 can be expressed as follows:

$$\mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right] = \mathbb{E}\left[\bar{\mu}_a^{(g)} \bar{\mu}_a^{(g')}\right] + \frac{\sigma_\varepsilon^2}{N_a^{(g')}} \qquad \forall g \le g' \text{ and } g, g' \in \mathcal{G} \tag{10}$$

where σ_ε^2 denotes the variance of the statistical noise in the observations.

Proof: The proof is given in appendix A. The derivation of Equation 8 involves using the Lagrangian for the problem stated in Equations 6-7 and performing some simple arithmetic on the corresponding first order optimality conditions. Equation 9 is identical to Equation 7 from the optimization formulation.

In the remainder of this analysis, our computations will be conditional on a given sequence of observed attribute vectors. In other words, all expectations and probabilities are computed with respect to the probability space, Ω^ε. We prove Equation 10 by simplifying the expression $\mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right]$ using some properties of hierarchical aggregation. $\square$

For the case where $g = 0$, we can use the result, $\mathbb{E}\left[\overline{\mu}_a^{(0)}\overline{\mu}_a^{(g')}\right] = 0$ (which follows from the property: $\mu_a^{(0)} = 0$), to further simplify (10) and obtain the following result:

$$\mathbb{E}\left[\overline{\delta}_a^{(0)}\overline{\delta}_a^{(g')}\right] = \frac{\sigma_\varepsilon^2}{N_a^{(g')}}. \tag{11}$$

We refer to the optimal weighting scheme as WOPT.

4.2 An Approximation Assuming Independence

It is a well-known result in statistics that if the estimates $\left\{\overline{v}_a^{(g)}, g \in \mathcal{G}\right\}$ were independent and unbiased, then the optimal weights would be given by

$$w_a^{(g)} = \frac{1}{{\sigma_a^{(g)}}^2/N_a^{(g)}}\left(\sum_{g' \in \mathcal{G}} \frac{1}{{\sigma_a^{(g')}}^2/N_a^{(g')}}\right)^{-1}. \tag{12}$$

We can obtain this result from proposition 1 as follows. If we assume that the estimates $\left\{\overline{v}_a^{(g)}, g \in \mathcal{G}\right\}$ are independent and unbiased, then the cross-terms in Equation 8 disappear, leaving behind the following modified relation:

$$w_a^{(g)}\mathbb{E}\left[\left(\overline{\delta}_a^{(g)}\right)^2\right] - \lambda = 0 \quad \forall\ g \in \mathcal{G}. \tag{13}$$

Solving Equations 13 and 9 gives us weights that are inversely proportional to the expected squared errors, $\mathbb{E}\left[\left(\overline{\delta}_a^{(g)}\right)^2\right]$. For the case of independent, unbiased estimates, $\mathbb{E}\left[\left(\overline{\delta}_a^{(g)}\right)^2\right]$ is identical to the variance, ${\sigma_a^{(g)}}^2/N_a^{(g)}$.

Solving the system of equations in Proposition 1 can be computationally expensive since in practice, there may be hundreds of thousands of models. For practical solutions, it will be useful to have an expression along the lines of Equation 12 for computing the weights, even though neither of the conditions (independence and absence of bias) holds true for estimates that arise from aggregation due to structural errors introduced in the process of aggregation. In order to adapt the simpler formula in (13) to the aggregation setting while acknowledging the bias, we first define:

$$\begin{aligned} \mu_a^{(g)} &= \text{Expected bias in the estimate, } \overline{v}_a^{(g)} \\ &= \mathbb{E}\left[\overline{v}_a^{(g)} - \nu_a\right]. \end{aligned}$$

For biased estimates, the total squared error can be expressed as the sum of bias and variance components, provided the bias and variance are independent of each other (Hastie et al., 2001, p. 24):

$$\mathbb{E}\left[\left(\overline{\delta}_a^{(g)}\right)^2\right] = \frac{{\sigma_a^{(g)}}^2}{N_a^{(g)}} + {\mu_a^{(g)}}^2. \tag{14}$$

We use this relation to modify the weights as follows:

$$w_a^{(g)} = \frac{1}{\frac{\sigma_a^{(g)^2}}{N_a^{(g)}} + \mu_a^{(g)^2}} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\frac{\sigma_a^{(g')^2}}{N_a^{(g')}} + \mu_a^{(g')^2}} \right)^{-1} \quad \forall\ g \in \mathcal{G}. \tag{15}$$

We call this weighting scheme, WIND.

4.3 Weighting by Inverse Mean Squared Errors

In the more realistic setting where the exact values of the parameters involved in the computation of weights as in Equation 15 are unknown, we propose using the plug-in principle (see, for example, Efron and Tibshirani 1993, chapter 4) where we use statistical estimates of the bias and variance to produce approximations of the weights. We first compute estimates of the bias and the variance using

$$\begin{aligned} s_a^{(g)^2} &= \text{The sample variance of the observations corresponding to the estimate } \bar{v}_a^{(g)} \\ &= \frac{1}{N_a^{(g)} - 1} \sum_{n \in \mathcal{N}_a^{(g)}} \left(\hat{v}^n - \bar{v}_a^{(g)} \right)^2 . \\ \tilde{\mu}_a^{(g)} &= \text{An estimate of the bias in the estimated value } (\bar{v}_a^{(g)}) \text{ from the true value} \\ &= \bar{v}_a^{(g)} - \bar{v}_a^{(0)} . \end{aligned}$$

The approximate weights on the estimates at different levels of aggregation are inversely proportional to the estimates of their mean squared deviations (obtained as the sum of the variances and the biases) from the true value:

$$w_a^{(g)} = \frac{1}{\frac{s_a^{(g)^2}}{N_a^{(g)}} + \tilde{\mu}_a^{(g)^2}} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\frac{s_a^{(g')^2}}{N_a^{(g')}} + \tilde{\mu}_a^{(g')^2}} \right)^{-1} \quad \forall\ g \in \mathcal{G}. \tag{16}$$

We refer to this formula as weighting by inverse mean squared errors (WIMSE). In the event that $N_a^{(g)}$ is too small or zero (which can happen in the early iterations and/or at the more disaggregate levels), it is difficult to form meaningful estimates of the variance and bias. In such a situation, we set the corresponding weight to zero.

Equation 16 is very easy to calculate even for large scale applications where we may observe hundreds of thousands of attributes. However, it produces the best results only when the estimates of values at different levels of aggregation are independent, an assumption that we cannot expect to hold true. In the next section, we present theoretical and experimental evidence supporting the claim that the error introduced from this assumption is negligible.

It is important to note that the use of the plug-in principle, which in this setting means using statistical estimates of parameters (the bias and variance), may result in some unexpected behavior when the number of observations is small. For example, the estimate of the total squared error in Equation 14 would be expected to decrease with each additional observation. When we use estimates of the bias and variance, this is no longer guaranteed, especially when $N_a^{(g)}$ is small. However, our empirical evidence is that it seems to behave as expected in an aggregate sense.

5. The Case for Assuming Independence

In this section, we justify our decision to ignore the dependence between the estimates from hierarchical aggregation, while combining them to form an improved estimate. We discuss the special case where we combine estimates from only two levels of aggregation, which enables us to obtain simple expressions for computing the various parameters. We assume that the statistical noise is independent of the attribute vector sampled and also that we know the probability distributions of the sampling of the attribute vectors and their values. These assumptions enable us to solve the optimality equations to obtain a solution explicitly. In Section 5.1, we analytically compare the two sets of equations (with and without assuming independence) for computing optimal weights. We provide an experimental comparison of the two methods, demonstrating the similarity in results, in section 5.2.

5.1 Analytical Comparison

For the two-level problem, we can obtain the optimal weights (WOPT) by solving the following system of equations:

$$\begin{aligned}
\mathbb{E}\left[\bar{\delta}_a^{(0)^2}\right] w_a^{(0)} + \mathbb{E}\left[\bar{\delta}_a^{(0)}\bar{\delta}_a^{(1)}\right] w_a^{(1)} - \lambda &= 0, \\
\mathbb{E}\left[\bar{\delta}_a^{(0)}\bar{\delta}_a^{(1)}\right] w_a^{(0)} + \mathbb{E}\left[\bar{\delta}_a^{(1)^2}\right] w_a^{(1)} - \lambda &= 0, \\
w_a^{(0)} + w_a^{(1)} &= 1, \\
w_a^{(0)}, w_a^{(1)} &\geq 0.
\end{aligned}$$

Since we are concerned with computing the weights for a particular attribute vector, we drop the index a in the following analysis. We obtain the value of $w^{(0)}$ as,

$$w^{(0)} = \frac{\mathbb{E}\left[\bar{\delta}^{(1)^2}\right] - \mathbb{E}\left[\bar{\delta}^{(0)}\bar{\delta}^{(1)}\right]}{\mathbb{E}\left[\bar{\delta}^{(0)^2}\right] + \mathbb{E}\left[\bar{\delta}^{(1)^2}\right] - 2\mathbb{E}\left[\bar{\delta}^{(0)}\bar{\delta}^{(1)}\right]}. \tag{17}$$

By assumption, the estimate at the disaggregate level is unbiased, that is, $\mu^{(0)} = 0$. We let $\mu^2 = \mathbb{E}\left[\bar{\mu}^{(1)^2}\right]$ denote the expected value of the square of the bias term at the aggregate level. Using Equations 10 and 11, we may write,

$$\begin{aligned}
\mathbb{E}\left[\bar{\delta}^{(0)}\bar{\delta}^{(1)}\right] &= \frac{\sigma_\varepsilon^2}{N^{(1)}}, \\
\mathbb{E}\left[\bar{\delta}^{(0)^2}\right] &= \frac{\sigma_\varepsilon^2}{N^{(0)}}, \\
\mathbb{E}\left[\bar{\delta}^{(1)^2}\right] &= \mathbb{E}\left[\bar{\mu}^{(1)^2}\right] + \mathbb{E}\left[\bar{\varepsilon}^{(1)^2}\right] \\
&= \mu^2 + \frac{\sigma_\varepsilon^2}{N^{(1)}}.
\end{aligned}$$

These results enable us to rewrite Equation 17 for computing the weights on the disaggregate estimate using the WOPT scheme (which we denote as w^{opt}) as follows:

$$w^{opt} = \frac{1}{1+\left(\frac{1}{N^{(0)}}-\frac{1}{N^{(1)}}\right)\frac{\sigma_\varepsilon^2}{\mu^2}}. \tag{18}$$

The competing scheme, WIND, assumes independence of the estimates. The weights at the disaggregate level are obtained using the formula:

$$w^{ind} = \frac{1+\frac{1}{N^{(1)}}\frac{\sigma_\varepsilon^2}{\mu^2}}{1+\left(\frac{1}{N^{(0)}}+\frac{1}{N^{(1)}}\right)\frac{\sigma_\varepsilon^2}{\mu^2}}. \tag{19}$$

We denote by $\tilde{v}^{opt}$ and $\tilde{v}^{ind}$ the estimates computed using the two weighting schemes.

$$\begin{aligned}\tilde{v}^{opt} &= w^{opt}\bar{v}^{(0)}+(1-w^{opt})\bar{v}^{(1)},\\ \tilde{v}^{ind} &= w^{ind}\bar{v}^{(0)}+(1-w^{ind})\bar{v}^{(1)}.\end{aligned}$$

We can write the difference between the estimates of the value obtained with and without the independence assumption as $\Delta = \tilde{v}^{opt}-\tilde{v}^{ind} = \Delta_w \cdot \Delta_v$, where $\Delta_w = w^{opt}-w^{ind}$ and $\Delta_v = \bar{v}^{(0)}-\bar{v}^{(1)}$. The following proposition establishes that Δ is small under certain conditions.

Proposition 2
(i) $\lim_{\mu\to 0}\mathbb{E}[\Delta]=0$,
(ii) $\lim_{\mu\to\infty}\Delta=0$,
(iii) $\lim_{\sigma^2\to 0}\Delta=0$.

Proof:
(i) As $\mu\to 0$, $w^{opt}=0$ and $w^{ind}=N^{(0)}/\left(N^{(0)}+N^{(1)}\right)$. w^{ind} attains a maximum value of $1/2$ when $N^{(0)}=N^{(1)}$, but that would imply that $\bar{v}^{(0)}=\bar{v}^{(1)}\Rightarrow\Delta_v=0$. At the other extreme, if $N^{(0)}=0$, then $w^{ind}=0\Rightarrow\Delta_w=0$. For intermediate values of $N^{(0)}$, it is no longer true that the random variable Δ_v will always be zero (for statistical reasons), but we can show that its expectation will be zero using

$$\begin{aligned}\mathbb{E}[\Delta] &= \mathbb{E}\{\mathbb{E}[\Delta|\mathcal{N}]\},\\ \mathbb{E}[\Delta|\mathcal{N}] &= \mathbb{E}\left[\Delta_v\frac{N^{(0)}}{N^{(0)}+N^{(1)}}|\mathcal{N}\right]\\ &= \mathbb{E}[\Delta_v]\frac{N^{(0)}}{N^{(0)}+N^{(1)}}\\ &= 0.\end{aligned}$$

Since $\mu^2=0$, $\mathbb{E}[\Delta_v]=0$ and Equation 20 follows.
(ii) As $\mu\to\infty$, $w^{ind}\to w^{opt}\to 1$, which can be easily obtained by applying the appropriate limits in Equations 18 and 19. This is intuitive since with very high bias, the best strategy is to put all the weight on the most disaggregate level. As a result, $\Delta_w\to 0$.
(iii) As the variance goes to zero, $w^{(ind)}\to w^{(opt)}$ that again implies $\Delta_w\to 0$. □

Thus, the error from the independence assumption is small when the bias is high or low, or when the variance is low. The error will be highest for moderate values of the bias and higher values of the variance. Given that the errors vanish for the extreme cases, it is perhaps not surprising that the errors are never very large. We provide experimental evidence to support this conclusion in the next section.

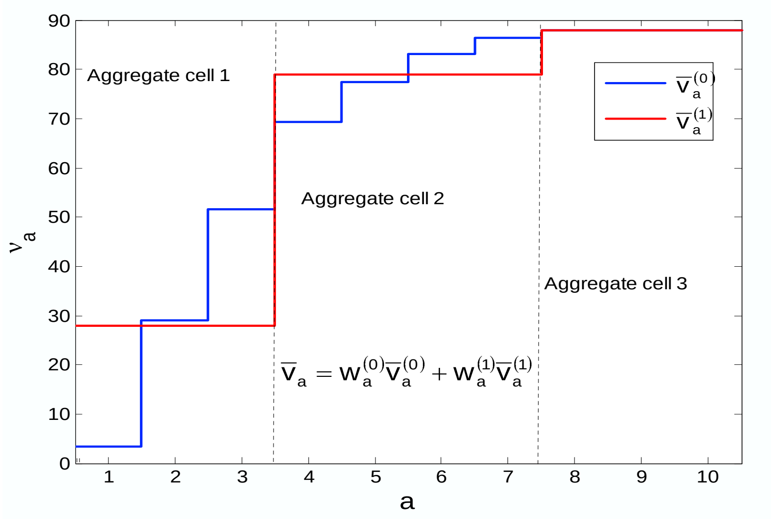

Figure 3: A piecewise constant function with its aggregate approximation. Estimates of values of each attribute vector are computed at both the aggregate and disaggregate levels. A weighted averaging is done to improve the estimates.

5.2 Experimental Results

In this section, we analyze the estimation of functions characterized by known parameters (which effectively requires that we know the actual function) in order to demonstrate the effectiveness of the optimal weighting strategy, as well as to serve as a benchmark for the strategy which assumes independent estimates at different levels of aggregation. We observe that the weights given by either method (Equations 18 and 19) are functions of the bias in the value at the aggregate level, the variance of the statistical noise in the observation of the values and the number of observations at either level. In order to compare the values of the weights from the competing strategies, we create scenarios with different combinations of the parameters that would produce significant changes in the weights. We then analyze how the variations in the weights given by WOPT and WIND affect the actual function estimates computed using the two schemes.

We consider a piecewise constant monotone function and its aggregate version as shown in Figure 3. We note that there are distinct regions in the domain where the bias is high, intermediate and zero - we expect the relative weights to be very different in these three regions. Figure 4 gives the weights (to be applied to the disaggregate level) produced by the optimal formula, WOPT, and the formula assuming independence, WIND, for each attribute a. The weights are obtained by sampling since the corresponding Equations (18 and 19) require the number of observations at the two levels of aggregation, $N^{(0)}$ and $N^{(1)}$. As we would expect, the optimal weights at the disaggregate level are zero when there is no structural error, in contrast to WIND. When the structural error is highest, the weights produced by the two methods are very similar. Note, however (consistent with our understanding from the previous section) that the weights are also quite different for the cells $a = 2$ and $a = 5$ where the aggregate and disaggregate functions are most similar (which means the bias is small). It is also the case that the weight to be given to the disaggregate level is also smallest when the bias is smallest.

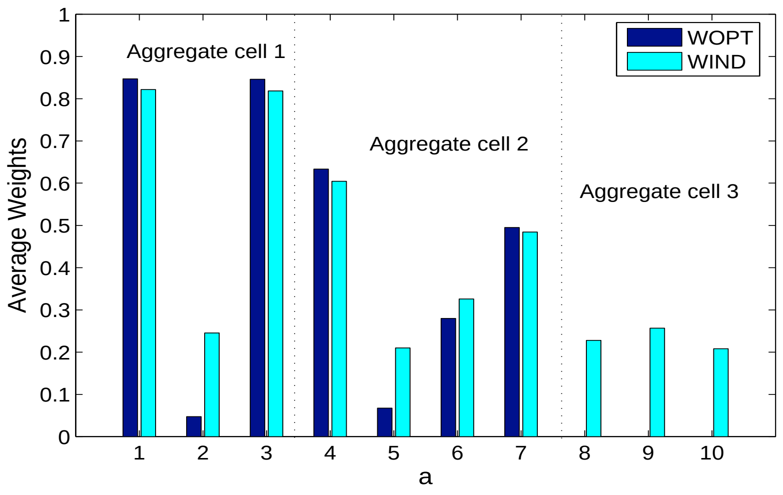

Figure 4: Comparison of the weights over the function domain

We have illustrated the difference in the weights produced by the two strategies, but less obvious is the difference in the estimates of the underlying function. In order to compare the two schemes, we developed a measure of the degree to which a weighting strategy reduced the variance of an estimate. We define the following:

$$\begin{aligned}
\tilde{v}^s_a &= \text{The value of the attribute vector } a \text{ as estimated by strategy } s. \\
\varepsilon^s &= \text{The sum of squared errors as estimated by strategy } s. \\
&= \sum_{a\in\mathcal{A}} (\tilde{v}^s_a - \nu_a)^2 \\
\varepsilon^G &= \text{The sum of squared errors using the static aggregation strategy which treats} \\
&\quad\ \text{the function as a constant over its domain.} \\
\theta^s &= \text{The performance measure for strategy } s. \\
&= 1 - \frac{\varepsilon^s}{\varepsilon^G}.
\end{aligned}$$

θ^s measures the degree of variability explained by a particular weighting strategy relative to using a single constant which can be thought of as a default strategy where all observations are aggregated together. θ^s is analogous to an R^2 measure commonly used in statistics.

A major factor in the performance of a weighting strategy is the relative size of the structural variation compared to the statistical noise. For this purpose, we define an index, ρ, that measures the ratio of the noise to the bias.

Figure 5 compares the performances of the two weighting strategies for three levels of noise. We observe that the performance of WOPT and WIND are almost identical even though there were situations where the weights given by the two schemes were significantly different. The similarity in the function estimates from the two strategies is explained by the analysis in Section 5.1.

We tested the relative performance of the two methods for other function classes. We summarize the results in figure 6 where we plot the performance measure as a function of the average number of observations per disaggregate cell. We observe that there is very little statistical difference between

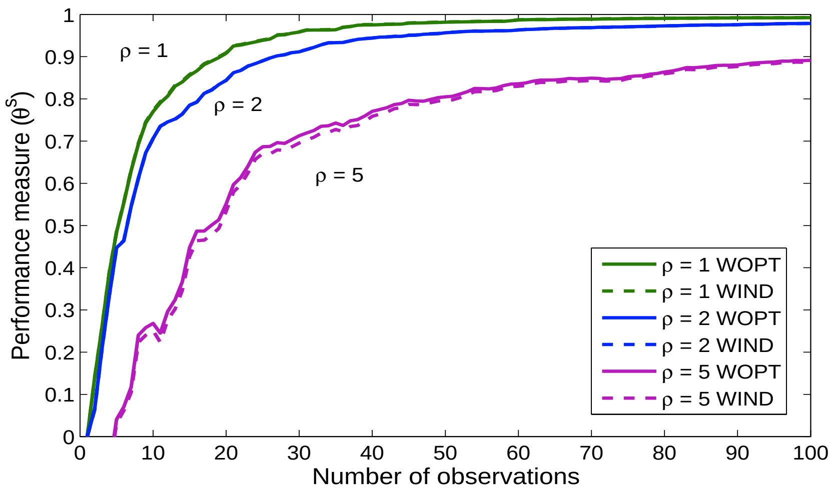

Figure 5: Comparison of the performance, as measured by θ^s, of WOPT and WIND in estimating the piecewise constant function

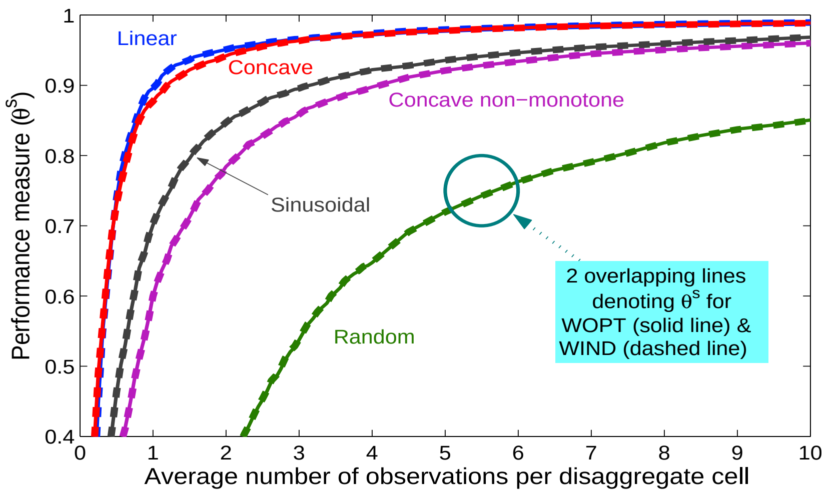

Figure 6: Comparison of WOPT and WIND for various function types using expected values of weights. The graph shows the average performance measure (θ^s) over 1000 samples for a moderate value of $\rho = 2$. WOPT is represented using solid lines and WIND, with dashed lines - the two are virtually indistinguishable.

the performance of the two methods. From this analysis, we conclude that WIND, which combines estimates assuming independence, will generally be a close approximation of WOPT. Of particular interest for our problem setting is that WIND is much easier to implement.

6. Experiments in an ADP Application

We implemented the hierarchical weighting strategy in the approximate dynamic procedure for solving the nomadic trucker problem described in Section 2. In Section 6.1, we describe the specifics of the problem instances that we consider. We also state the competing strategies that we compare in the experiments that follow. We then proceed to show the effectiveness of our hierarchical weighting scheme using two sets of experiments. In Section 6.2, we report on experiments where the discount factor is set to zero. In this case, the observations of values are unbiased, since they do not involve the estimates of values of future states. In Section 6.3, we present the results of experiments with positive discount factors. We have made available a collection of data sets used in these experiments on the following webpage - `http://castlelab.princeton.edu/`. Finally, in Section 6.4, we provide experimental results from applying our techniques on an industrial strength problem.

6.1 Experimental Design

We consider a problem where we specify the state of the truck using three attributes, namely, the current location, the day of week and the number of days away from home. The problem is rich enough to offer interesting opportunities for hierarchical aggregation, but small enough that we can solve the problem to obtain the exact solution.

The decisions are to be made over a finite time horizon of 21 time periods. The location attribute can be represented at two degrees of resolution - regions (eastern Pennsylvania, northern New Jersey) or geographical areas (Northeast, Midwest and so on). There are 50 locations at the region level which can be aggregated to 10 geographical areas.

The major contributor to the stochastic nature of the nomadic trucker problem is the uncertainty in the availability of loads in any particular location to be moved to other locations. The probability that a load is available to be moved from one location to another is dependent on the origin-destination pair. Another factor that influences the load availability is the day of week. Loads are more likely to appear during the beginning of the week (Mondays) and towards the end (Fridays). We use a probability distribution whereby the load availability dips during the middle of the week and is lower over the weekends. We introduce further uncertainty into the problem by allowing the one-period contributions to be moderately noisy.

The final attribute that we consider is the number of days that the driver is away from home. There is a penalty that we impose on moves that keep the driver away from his home domicile, which is a quadratic function of the number of days away from home. In order to keep the state space manageable (so we can obtain optimal solutions), we cap the number of days away from home at 12.

In Table 2, we list the aggregations that we use for the problem and the number of attribute states at each level of aggregation. For example, at aggregation level 1, the location attribute is aggregated from 50 regions to 10 geographical areas. We aggregate out the day-of-week attribute and retain the days-away-from-home attribute. Adding in the factor for 21 time periods, we have a total of 2541 possible states. The apparent discrepancy in the size of the state space at levels 0 and 1 arises because the days-away-from-home attribute is always set to 0 for the location corresponding to the home of the driver, while for all the other locations it can be any number from 1 to 12.

In order to compute the true values associated with each attribute vector, we use a standard backward dynamic programming algorithm. Our focus is on the problem of statistical estimation of the true values of the various states. In order to form estimates of these values we incorporate

g	Time	Location	Days-away-from-home	Day-of-week	$\lvert\mathcal{A}\rvert$
0	*	Region	*	*	86583
1	*	Area	*	-	2541
2	*	Area	-	-	210
3	*	-	-	-	21

Table 2: Aggregations for the multiattribute nomadic trucker problem. A '$*$' corresponding to a particular attribute indicates that the attribute is included in the attribute vector, and a '$-$' indicates that it is aggregated out.

our aggregation strategies in the approximate dynamic programming algorithm outlined in Figure 1. The value function estimate in Equation 1 is obtained as a weighted sum of the value estimates at various levels of aggregation:

$$\bar{v}_{a'}^{n-1} = \sum_{g\in\mathcal{G}} w_{a'}^{(g),n-1}\bar{v}_{a'}^{(g),n-1} \qquad \text{where } a' = a^M(a,d)\ \forall d\in\mathcal{D}_a$$

where $w_{a'}^{(g),n-1}$ denotes the weight, at iteration $(n-1)$, on the estimate of the value of attribute vector a' at the gth level of aggregation.

The methods that we compare use the following sets of weights:

1. Static Aggregation:

$$w_{a'}^{(g),n-1} = \begin{cases} 1 & \text{if } g \text{ is the fixed level of aggregation,} \\ 0 & \text{otherwise.} \end{cases}$$

2. Dynamic Aggregation (MINV):

$$w_{a'}^{(g),n-1} = \begin{cases} 1 & \text{if } g = \arg\min_{g'\in\mathcal{G}} \left(s_{a'}^{(g'),n-1}\right)^2 \\ 0 & \text{otherwise.} \end{cases}$$

3. WIMSE: $w_{a'}^{(g),n-1}$ is computed using Equation 16.

Equation 4 is replaced by a series of equations for updating the value function estimates at all the levels of aggregation corresponding to the currently visited attribute vector:

$$\bar{v}_a^{(g),n} = (1-\alpha)\bar{v}_a^{(g),n-1} + \alpha\hat{v}_a^n, \qquad a\in\mathcal{A}, g\in\mathcal{G}. \tag{20}$$

In the early iterations, especially for the more disaggregate levels, we often encounter the situation where the number of observations is too small (zero in certain cases) to form a meaningful estimate of a value. In such instances, we ignore the estimate at that level of aggregation. While forming a weighted sum of estimates, this usually results in all the weights being placed on the more aggregate levels.

For methods 2 and 3, we point out that the chosen level of aggregation and the weights on the estimates at different levels are dynamic in nature, in that they change with each iteration. This is brought out in the sections that follow.

The performance measure that we use for these experiments is based on the deviation of the value function approximations from the optimal value functions which can be obtained using a traditional backward dynamic programming algorithm such as value iteration. We first define

p_a	$=$	The steady state probability of being in state (equivalent to, in this situation, attribute vector) a.
N_a	$=$	The number of observations of or visits to state a.
ν_a	$=$	The true value associated with state a.
$\tilde{v}_a^s$	$=$	The value function approximation for state a estimate computed using strategy s.

Using these we may define the following performance measures:

$$E_1^s = \text{Error measure based on steady state probabilities, that is, a stationary distribution}$$

$$= \frac{\sum_{a\in\mathcal{A}} p_a\left(\tilde{v}_a^s - \nu_a\right)}{\sum_{a\in\mathcal{A}} p_a \nu_a} \times 100\%. \tag{21}$$

$$E_2^s = \text{Error measure based on the number of visits to each state}$$

$$= \frac{\sum_{a\in\mathcal{A}} N_a\left(\tilde{v}_a^s - \nu_a\right)}{\sum_{a\in\mathcal{A}} N_a \nu_a} \times 100\%. \tag{22}$$

6.2 Experiments on Myopic Data Sets

We first analyze the ability of our proposed method to make estimates from a purely statistical perspective. For this purpose, we maintain a zero discount factor, which means that the downstream values of the states are ignored while making the decisions at any time period. Instead the decisions are based on the one-period rewards alone. Using a discount factor of zero eliminates the bias that is introduced in the ADP procedure, which implies that the errors in the estimates are purely due to the noise in the observations. We compare the different estimation techniques in Table 3 where we tabulate the E_2^s values (as computed using Equation 22) for several problem instances. The static aggregation strategies for $g = 0, 1$ & 2 are denoted as *Disaggregate*, *Aggregate1* and *Aggregate2* respectively. We observe that WIMSE demonstrates lower errors than the other techniques.

6.3 Experiments on Non-myopic Data Sets

In this section, we discuss results obtained by using a positive discount factor. As such, the estimates of the future values have an impact on the current decisions. Figure 7 illustrates the variation in the weights on the estimates from different levels of aggregation, as a function of the number of observations, computed using the WIMSE weighting scheme. As expected, in the early iterations we place the highest weight on the most aggregate level since this offers the greatest statistical reliability. As the algorithm progresses, higher weights are put on the more disaggregate levels, with ultimately the highest weight on the most disaggregate level. It is interesting (and noteworthy) that the weights on the higher levels of aggregation drop quickly at first, but then stabilize, dropping

Problem #	Disaggregate		Aggregate1		MINV		WIMSE	
1	6.89	*0.07*	18.06	*0.06*	7.84	*0.13*	**5.54**	*0.08*
2	9.47	*0.07*	18.39	*0.12*	11.99	*0.28*	**7.88**	*0.09*
3	**2.92**	*0.04*	17.81	*0.07*	4.04	*0.07*	2.95	*0.04*
4	5.74	*0.08*	18.06	*0.09*	8.02	*0.16*	**5.56**	*0.07*
5	2.70	*0.05*	17.25	*0.07*	3.27	*0.11*	**2.44**	*0.05*
6	5.52	*0.09*	17.50	*0.09*	7.26	*0.10*	**5.10**	*0.10*

Table 3: Comparison of techniques for different problem instances (Disaggregate: $g = 0$, Aggregate1: $g = 1$). The performance measure used for each of the methods is the percentage deviation from optimality. Figures in italics denote the standard deviations of the terms to the left.

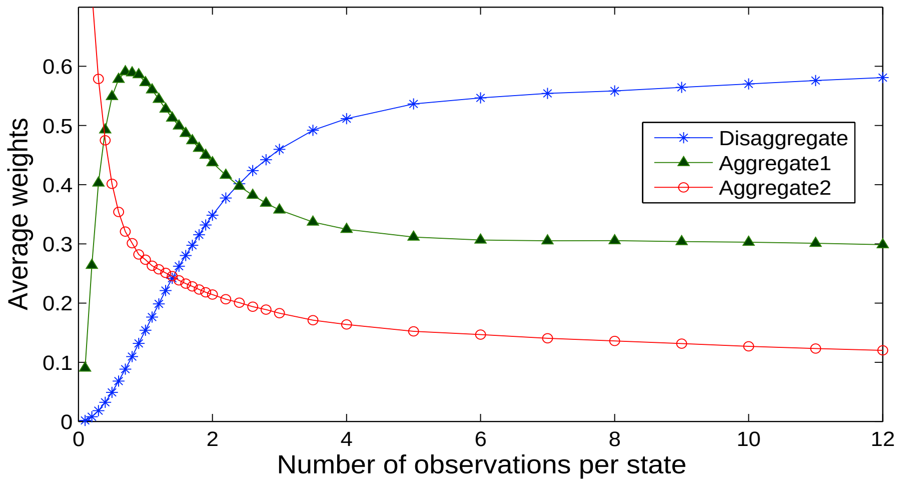

Figure 7: Average weights using hierarchical aggregation. (Disaggregate: $g = 0$, Aggregate1: $g = 1$, Aggregate2: $g = 2$).

very slowly. This behavior primarily reflects attributes with very low bias where the weight on the aggregate level will always remain fairly high.

In Figure 8, we compare the various techniques based on E_1^s values, that is, the percentage errors in the estimates from the optimal values weighted by the steady state probabilities under an optimal policy (see Equation 21). The steady state probabilities can be obtained in the process of computing the optimal values. This error measure enables us to determine how close the policy produced by the value function estimates is to an optimal policy. We see that the hierarchical weighting scheme outperforms all the static aggregation strategies as well as the dynamic aggregation strategy which picks the "best" level of aggregation, producing the least errors in the estimates.

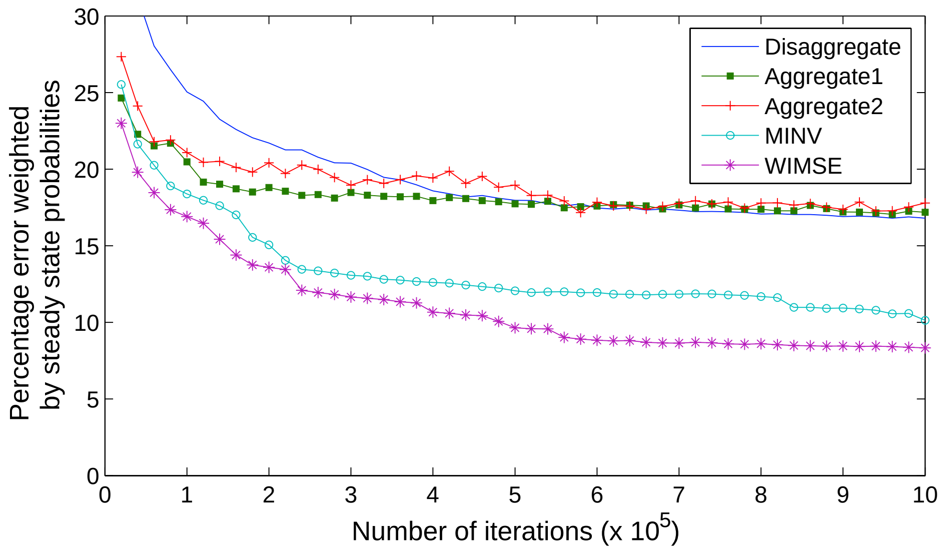

Figure 8: E_1^s values as a function of the number of observations.

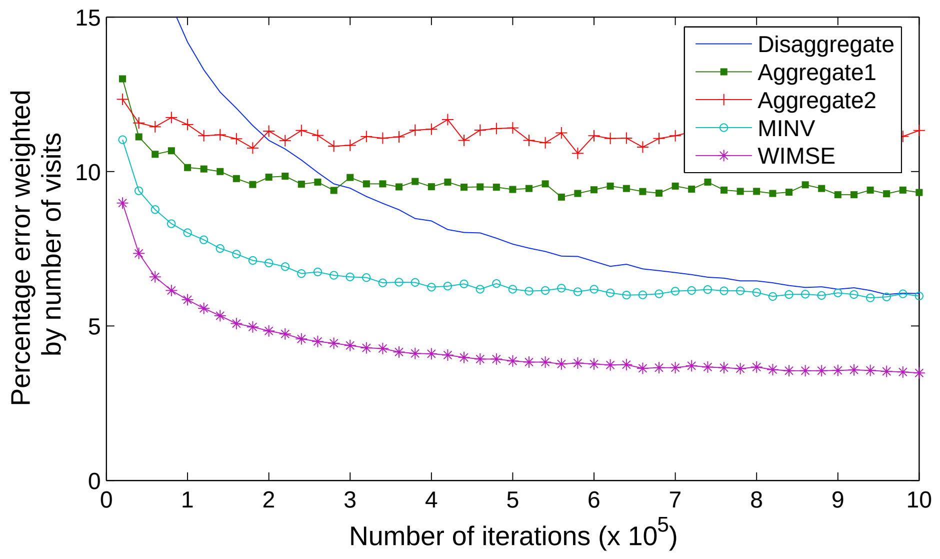

Figure 9: E_2^s values as a function of the number of observations.

Figure 9 shows the relative performance of the various schemes with respect to the errors weighted by the number of visits to the various states (E_2^s values). This error measure gives an idea of how well a particular strategy is able to estimate correctly the optimal value of the various states with higher weights on the errors in the estimates of states that are visited more frequently.

Problem #	Iterations	Disaggregate		Aggregate1		MINV		WIMSE	
1	20000	5.5	*0.1*	7.2	*0.1*	4.1	*0.0*	**2.9**	*0.0*
	50000	3.9	*0.0*	7.0	*0.1*	3.6	*0.0*	**2.2**	*0.0*
2	20000	5.3	*0.1*	6.8	*0.1*	4.8	*0.1*	**2.9**	*0.1*
	50000	3.7	*0.0*	6.6	*0.1*	4.4	*0.0*	**2.3**	*0.0*
3	20000	6.8	*0.1*	7.2	*0.1*	6.3	*0.1*	**4.3**	*0.0*
	50000	5.0	*0.0*	7.0	*0.1*	5.5	*0.1*	**3.6**	*0.0*
4	20000	11.0	*0.1*	9.8	*0.1*	7.0	*0.2*	**4.8**	*0.1*
	50000	7.7	*0.1*	9.4	*0.1*	6.2	*0.2*	**3.9**	*0.1*
5	20000	9.8	*0.1*	9.0	*0.2*	6.6	*0.1*	**5.0**	*0.1*
	50000	6.7	*0.1*	9.0	*0.1*	5.6	*0.1*	**3.8**	*0.1*
6	20000	9.9	*0.1*	8.3	*0.2*	7.1	*0.1*	**4.8**	*0.2*
	50000	6.7	*0.1*	8.2	*0.2*	6.4	*0.2*	**3.8**	*0.1*
7	20000	12.0	*0.1*	10.0	*0.2*	7.3	*0.1*	**5.7**	*0.2*
	50000	8.5	*0.1*	10.0	*0.2*	6.4	*0.1*	**4.7**	*0.1*
8	20000	11.0	*0.2*	8.6	*0.1*	7.9	*0.2*	**5.6**	*0.2*
	50000	7.8	*0.1*	8.3	*0.2*	7.2	*0.1*	**4.5**	*0.2*
9	20000	14.5	*0.4*	13.4	*0.3*	10.6	*0.5*	**8.4**	*0.3*
	50000	10.6	*0.4*	12.5	*0.3*	9.4	*0.4*	**7.3**	*0.3*
10	20000	13.8	*0.2*	13.1	*0.5*	10.6	*0.2*	**8.3**	*0.3*
	50000	9.7	*0.2*	12.7	*0.2*	9.0	*0.2*	**6.8**	*0.2*

Table 4: Comparison of techniques for different instances of the nomadic trucker problem. Two higher levels of aggregation are also used in the problem, but omitted from the tables, as they give inferior results. Figures in italics denote the standard deviations of the terms to the left.

Here again, WIMSE is found to consistently outperform the remaining techniques, producing error values that are much lower.

We now compare the various techniques for several problem instances. The major parameters that are varied to obtain the different problem sets are the discount factor, the probability distribution of the load availability at the various locations and the level of uncertainty in the contributions resulting from choosing a particular decision. We used discount factors of 0.80, 0.90 and 0.95, three different load probability distributions and two levels of uncertainty in the rewards. The E_2^s values for these problem instances are tabulated in Table 4. Here again, WIMSE outperforms all the other schemes consistently, irrespective of the number of iterations.

6.4 Experiments with a Fleet of Trucks

The techniques in this paper were designed to be applied to approximate dynamic programming algorithms for optimizing a fleet of trucks. To illustrate, we provide a very brief model of a multiattribute resource allocation problem that can be applied to the management of large fleets of trucks, freight cars, locomotives, military air cargo jets or business jets (we have industrial experience with all of these problems). Let

R_a	$=$	The number of resources with attribute a.
R	$=$	$(R_a)_{a\in\mathcal{A}}$
$\mathcal{D}$	$=$	The set of different decision types that can be used to act on a resource (e.g., moving from one location to another, repairing a vehicle).
x_{ad}	$=$	The number of times we act on a resource with attribute a using a decision of type $d \in \mathcal{D}$.
x	$=$	$(x_{ad})_{a\in\mathcal{A},d\in\mathcal{D}}$
c_{ad}	$=$	The contribution generated by x_{ad}.

For problems with moderately complex attribute vectors, we have found that we can obtain high quality solutions using dynamic programming approximations by solving (at iteration n) approximations of the form,

$$\tilde{V}^n(R^n) = \max_x \left(\sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} c_{ad}x_{ad} + \sum_{a'\in\mathcal{A}} \bar{v}^{n-1}_{a'} R^x_{a'}(x) \right) \tag{23}$$

subject to,

$$\sum_{d\in\mathcal{D}} x_{ad} = R^n_a \quad \forall\, a \in \mathcal{A}, \tag{24}$$

$$R^x_{a'} = \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} \delta_{a'} x_{ad}, \tag{25}$$

$$x_{ad} \geq 0. \tag{26}$$

This is a fairly basic model, but it is enough to illustrate the application (for a more complete description, see Spivey and Powell 2004). Problem 23 - 26 is a linear program, and returns a dual variable that we denote $\hat{v}^n_a$ for the resource constraint in Equation 24. In a real application, the attribute space $\mathcal{A}$ is large enough that we do not generate Equation 24 for each $a \in \mathcal{A}$. Instead, we might generate the equation only if $R^n_a > 0$ (for example). We can then update our value function approximation using

$$\bar{v}^n_a = (1-\alpha_n)\bar{v}^{n-1}_a + \alpha_n \hat{v}^n_a.$$

The problem we encounter, just as in the earlier sections, is that real problems might have attribute spaces with several million elements, and yet we can only run a few hundred iterations of the algorithm. Many of the attributes are never sampled, while others will have only a few observations. A small handful may receive dozens of observations. As a result, the statistical reliability of the approximations $\bar{v}^n_a$ can be quite low. The standard technique is to estimate the values at some aggregate level that trades off between the statistical reliability and cost of bias introduced by aggregation. The right level of aggregation depends not only on the specific attribute (some are sampled more often than others) but also on the number of iterations we have run the algorithm.

We tested our multilevel weighting strategy using this model to simulate the operations of a major truckload motor carrier. A detailed description of this problem can be found in Simao et al. (2008). The most disaggregate attribute vector used for the value function captured the location

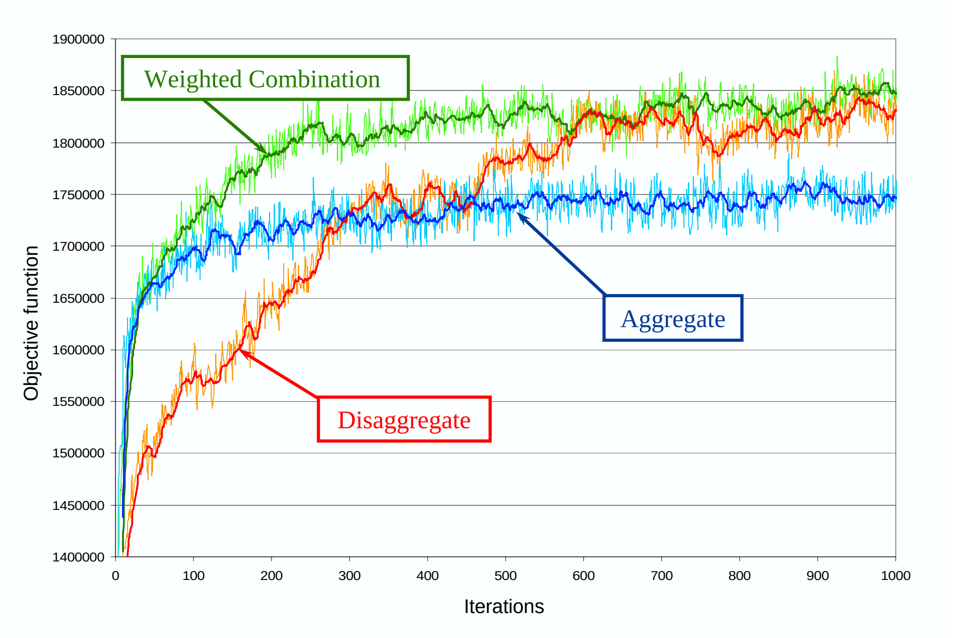

Figure 10: Comparison of using a single level of aggregation (aggregate and disaggregate) against a weighted aggregation strategy for optimizing a fleet of trucks.

of the driver (out of 100 regions), the driver's home domicile (out of 100 regions), and whether the driver was a single driver employed by the company, a contract driver, or a team (two drivers working together). At the most disaggregate level, the attribute space consisted of 30,000 elements. We considered an aggregate level which captured only the location of the driver (100 elements) and a hierarchical strategy that used a weighted estimate across four levels of aggregation.

The results of the experiment are shown in Figure 10. For this problem, a single iteration of the ADP algorithm can take 20 minutes. Even if we had confidence that the solution quality would never be better, improving the rate of convergence is extremely important. Since it is required to run this model continually to perform policy analyses, it would be extremely advantageous if we could reduce the number of iterations needed for convergence from 1000 to 200. As expected, if we use only an aggregate attribute vector, we obtain fast convergence but it levels off quickly. Using the most disaggregate representation produces slow convergence but it eventually reaches a better solution. By contrast, using a weighted combination of estimates at different levels of aggregation produced the fastest convergence and the best results overall. We point out that faster convergence to a better solution does not necessarily imply gains in running time. Even though the weighted combination strategy has a greater number of computations per iteration compared to static aggregation schemes, we have observed that the increase in running time for additional computations is not significant.

7. Conclusion

There is a vast range of problems that can be described as multiattribute resource allocation problems that can be modeled as stochastic dynamic programs. All of these problems pose the statistical challenge of estimating the value of a resource as a function of a potentially complex vector of attributes. Requiring no more than a set of aggregation functions which are typically quite easy to design, we have proposed an adaptive weighting strategy that produces an estimate of the value of a resource with a particular attribute. The weighting system is quite easy to compute, and is implementable for very large scale problems with attribute spaces that are arbitrarily large, since the computational complexity is a function of how many attributes you actually visit. The weights adjust naturally as the number of observations change. Thus, it performs well in the early iterations when there are very few observations. As more observations are made of particular attributes, the weight given to disaggregate estimates increases, producing higher quality solutions.

Our most surprising result was the finding that a weighting system that assumed independence of the estimates at each level of aggregation worked so well. This assumption is clearly not accurate. However, our analysis showed that it had little or no effect on the accuracy of a prediction. If the difference between a disaggregate estimate and the corresponding aggregate estimate was large, the weights produced by our approximation closely matched the weights that would have been produced had we properly accounted for the correlation. If the difference between the disaggregate and aggregate estimates was small, the weights could be quite different, but in this case it did not matter. The result is a weighting system that is easy to compute, scales well to very large scale applications and provides very accurate estimates.

An interesting possibility for future work would be to analytically solve for the optimal weights to combine estimates when there are more than two levels of aggregation. Further, the heuristic techniques that we have proposed could be applied to a broader range of estimation problems, not restricted to applications in approximate dynamic programming.

Acknowledgments

The authors would like to recognize the helpful comments of several referees. This research was supported in part by grant AFOSR contract FA9550-08-1-0195, and by grant N00014-07-1-0150 from the Office of Naval Research through the Center for Dynamic Data Analysis.

Appendix A. Proof of Proposition 1:

We first prove (8). The Lagrangian for the problem formulated in (6)-(7) is

$$\begin{aligned} L(w,\lambda) &= \mathbb{E}\left[\frac{1}{2}\left(\sum_{g\in\mathcal{G}} w_a^{(g)}\cdot\bar{v}_a^{(g)} - \nu_a\right)^2\right] + \lambda\left(1 - \sum_{g\in\mathcal{G}} w_a^{(g)}\right) \\ &= \mathbb{E}\left[\frac{1}{2}\left(\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\bar{v}_a^{(g)} - \nu_a\right)\right)^2\right] + \lambda\left(1 - \sum_{g\in\mathcal{G}} w_a^{(g)}\right). \end{aligned}$$

The first order optimality conditions are easily shown to be

$$\begin{aligned}
\mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\bar{v}_a^{(g)}-\nu_a\right)\left(\bar{v}_a^{(g')}-\nu_a\right)\right]-\lambda &= 0 \quad \forall\ g'\in\mathcal{G}, \qquad (27)\\
\sum_{g\in\mathcal{G}} w_a^{(g)}-1 &= 0.
\end{aligned}$$

To simplify Equation 27, we note that,

$$\begin{aligned}
\mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\bar{v}_a^{(g)}-\nu_a\right)\left(\bar{v}_a^{(g')}-\nu_a\right)\right] &= \mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\bar{\delta}_a^{(g)}\bar{\delta}_a^{(g')}\right]\\
&= \sum_{g\in\mathcal{G}} w_a^{(g)}\mathbb{E}\left[\bar{\delta}_a^{(g)}\bar{\delta}_a^{(g')}\right]. \qquad (28)
\end{aligned}$$

Combining Equations 27 and 28 gives us Equation 8.

We now derive Equation 10. We assume that the bias error, $\bar{\mu}_a^{(g)}$, is uncorrelated with the random error, $\bar{\varepsilon}_a^{(g)}$. Using Equation 5, we obtain the relation,

$$\begin{aligned}
\mathbb{E}\left[\bar{\delta}_a^{(g)}\bar{\delta}_a^{(g')}\right] &= \mathbb{E}\left[(\bar{\mu}_a^{(g)}+\bar{\varepsilon}_a^{(g)})(\bar{\mu}_a^{(g')}+\bar{\varepsilon}_a^{(g')})\right]\\
&= \mathbb{E}\left[\bar{\mu}_a^{(g)}\bar{\mu}_a^{(g')}+\bar{\mu}_a^{(g')}\bar{\varepsilon}_a^{(g)}+\bar{\mu}_a^{(g)}\bar{\varepsilon}_a^{(g')}+\bar{\varepsilon}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right]\\
&= \mathbb{E}\left[\bar{\mu}_a^{(g)}\bar{\mu}_a^{(g')}\right]+\mathbb{E}\left[\bar{\mu}_a^{(g')}\bar{\varepsilon}_a^{(g)}\right]+\mathbb{E}\left[\bar{\mu}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right]+\mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right]. \qquad (29)
\end{aligned}$$

We notice that, $\mathbb{E}\left[\bar{\mu}_a^{(g')}\bar{\varepsilon}_a^{(g)}\right]=\mathbb{E}\left[\bar{\mu}_a^{(g')}\right]\mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\right]=0$. By a similar argument, $\mathbb{E}\left[\bar{\mu}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right]=0$. This enables us to rewrite Equation 29 as,

$$\mathbb{E}\left[\bar{\delta}_a^{(g)}\bar{\delta}_a^{(g')}\right] = \mathbb{E}\left[\bar{\mu}_a^{(g)}\bar{\mu}_a^{(g')}\right]+\mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right]. \qquad (30)$$

Since $g'>g$, the stochastic error term at level g', $\bar{\varepsilon}_a^{(g')}$, can be expressed as a combination of $\bar{\varepsilon}_a^{(g)}$ and some terms that are independent of it:

$$\begin{aligned}
\bar{\varepsilon}_a^{(g')} &= \frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}}\varepsilon^n\\
&= \frac{1}{N_a^{(g')}}\left(\sum_{n\in\mathcal{N}_a^{(g)}}\varepsilon^n+\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n\right)\\
&= \frac{N_a^{(g)}}{N_a^{(g')}}\bar{\varepsilon}_a^{(g)}+\frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n. \qquad (31)
\end{aligned}$$

Using Equation 31, we can rewrite the second term on the right hand side of Equation 30 term as follows:

$$\mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\bar{\varepsilon}_a^{(g')}\right] = \mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\cdot\frac{N_a^{(g)}}{N_a^{(g')}}\bar{\varepsilon}_a^{(g)}\right]+\mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\cdot\frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n\right]$$

$$= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \bar{\varepsilon}_a^{(g)}\right] + \frac{1}{N_a^{(g')}} \underbrace{\mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \cdot \sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right]}_{I}.$$

Since the individual observations are assumed to be independent, the term I can be further simplified as follows,

$$\begin{aligned} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \cdot \sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right] &= \mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\right] \mathbb{E}\left[\sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right] \\ &= 0. \end{aligned}$$

This implies that,

$$\begin{aligned} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \bar{\varepsilon}_a^{(g')}\right] &= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)^2}\right] \\ &= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbf{Var}\left[\bar{\varepsilon}_a^{(g)}\right] \\ &= \frac{N_a^{(g)}}{N_a^{(g')}} \frac{1}{N_a^{(g)}} \mathbf{Var}\left[\varepsilon^n\right] \\ &= \frac{1}{N_a^{(g')}} \sigma_\varepsilon^2. \end{aligned} \tag{32}$$

Combining Equations (30 and 32 gives us the result in Equation 10. This results generalizes for all $g \in \mathcal{G}$, since g and g' can be interchanged when $g > g'$.

References

M. Athans, D. Bertsekas, W. McDermott, J. Tsitsiklis, and B. Van Roy. Intelligent optimal control. Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 1995.

J.C. Bean, J.R. Birge, and R.L. Smith. Aggregation in dynamic programming. *Operations Research*, 35:215–220, 1987.

D. Bertsekas and D. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.

D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *J. of Artificial Intelligence*, 11:1–94, 1999.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000. URL `citeseer.ist.psu.edu/boutilier99stochastic.html`.

L. Breiman. Stacked regression. *Machine Learning*, 24:49–64, 1996.

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

Z. Feng, E. A. Hansen, and S. Zilberstein. Symbolic generalization for on-line planning. In Christopher Meek and Uffe Kjærulff, editors, *UAI*, pages 209–216. Morgan Kaufmann, 2003. ISBN 0-127-05664-5.

M. Gendreau and J. Y. Potvin. Dynamic vehicle routing and dispatching. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 115–126. Kluwer Academic Publishers, 1998.

I. Guttman, S.S. Wilks, and J.S. Hunter. *Introductory Engineering Statistics*. John Wiley and Sons, Inc., New York, NY, 1965.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in Statistics, New York, NY, 2001.

S. Ichoua, M. Gendreau, and J.-Y. Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225, 2005.

K.E. Kim and T. Dean. Solving factored MDP's using non-homogeneous partitions. *Artificial Intelligence*, 147(1-2):225–251, 2003.

M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91:1641–1650, 1996.

R. Luus. *Iterative Dynamic Programming*. Chapman & Hall/CRC, New York, 2000.

R. Mendelssohn. An iterative aggregation procedure for Markov decision processes. *Operations Research*, 30(1):62–73, 1982.

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

W. B. Powell and T. A. Carvalho. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Science*, 32(2):90–109, 1998.

W. B. Powell, J. A. Shapiro, and H. P. Simão. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science*, 36(2):231–249, 2002.

D. Rogers, R. Plante, R. Wong, and J. Evans. Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582, 1991.

N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers and Operations Research*, 27(11):1201–1225, 2000.

N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, (to appear), 2008.

M. Spivey and W. B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4): 399–419, 2004.

R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

R.S. Sutton and A.G. Barto. *Reinforcement Learning*. The MIT Press, Cambridge, Massachusetts, 1998.

J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

X. Wang and T. G. Dietterich. Efficient value function approximation using regression trees. In J. Boyan, W. Buntine, and A. Jagota, editors, *Statistical Machine Learning for Large Scale Optimization, Neural Computing Surveys*. 2000.

C.J.C.H. Watkins. Learning from delayed rewards. Ph.d. thesis, Cambridge University, Cambridge, UK, 1989.

W. Whitt. Approximations of dynamic programs I. *Mathematics of Operations Research*, 3:231–243, 1978.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Y. Yang. Adaptive regression by mixing. *Journal of the American Statistical Association*, 96, 2001.

Q. Zhang and S. P. Sethi. Near optimization of dynamic systems by decomposition and aggregation. *Journal of Optimization Theory and Applications*, 99(1):1–22, 1998.