Hierarchical Aggregation Techniques for Estimating Value Functions for Dynamic Management of Multiattribute Resources

Warren B. Powell[*] Abraham George[†]

[*]Department of Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544
powell@princeton.edu

[†]Department of Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544
ageorge@princeton.edu

1 Introduction

There are a number of problems in transportation that involve the dynamic assignment of resources such as pilots, drivers, business jets, locomotives and tractors to tasks such as flights, loads, passengers, trains and trailers. Although academic models of these problems will typically simplify the characteristics of the resources, production systems have to incorporate a far higher level of detail, producing a rich set of attributes that capture the important characteristics that are needed to model the problem accurately. Methods exist to handle this detail in a deterministic world, but special challenges arise in the presence of uncertainty.

These problems are often solved myopically by optimally assigning the resources to tasks as they become known. A resource may be assigned to one task at a time (producing a series of dynamic assignment problems) or to multiple tasks (producing dynamic set partitioning problems). The focus of this research is on methods for approximating the value of resources in the future, so we are going to use the simplest possible dynamic model: we assume that resources are assigned to at most one task at a time, and that unserved tasks are lost to the system if they are not served in the time period in which they first become known.

2 A dynamic resource management model

To present a model of this simple problem:

$R_{ta} =$ The number of resources with attribute a available to be assigned at time t.
$\mathcal{A} =$ Space of attributes of a resource, with element $a \in \mathcal{A}$.
$L_{tb} =$ The set of tasks to be served at time t.
$\mathcal{B} =$ The space of attributes of a task, with element $b \in \mathcal{B}$.
$\hat{R}_{ta} =$ The number of resources that first become known at time t with attribute a.
$\hat{L}_{tb} =$ The number of tasks that first become known at time t with attribute b.

For our applications, the attribute vector a might have half a dozen to a dozen elements. The attribute space $\mathcal{A}$, discretized, might have several million elements. At any point in time, we may act on our resources with one of two types of decisions:

$\mathcal{D}^l =$ The decision to assign a resource to a task. If $d \in \mathcal{D}^l$, then b_d is the attributes of the task that we are assigning the resource to.
$\mathcal{D}^m =$ The decision to modify a resource (by moving it geographically, or in other ways such as repairing, cleaning, refueling, resting).
$d^\phi =$ The "null decision" which is the decision to do nothing.
$\mathcal{D} = \mathcal{D}^l \bigcup \mathcal{D}^m \bigcup d^\phi$.

A decision is represented using:

$$x_{tad} = \begin{cases} 1 & \text{If we act on a resource with attribute } a \in \mathcal{A} \text{ with decision } d \in \mathcal{D} \text{ at time } t. \\ 0 & \text{Otherwise} \end{cases}$$

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

If we act on a resource with decision d, we assume that we produce a modified resource with attribute a', which we represent using:

$$\delta_{a'}(t,a,d) = \begin{cases} 1 & \text{If decision } d\text{, applied to a resource with attribute } a\text{, produces a resource with attribute } a'. \\ 0 & \text{Otherwise} \end{cases} .$$

$c_{tad} =$ The contribution if we assign a resource with attribute a to a task with attribute b at time t.

The decisions have to be made subject to the *static constraints*:

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

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

In addition, we have to capture the dynamic constraints:

$$R_{t+1,a'} = \delta_{a'}(t,a,d)x_{tad} + \hat{R}_{t+1,a'} \tag{3}$$

We assume that tasks are not held from one period to the next. As a result, our task state vector is simply $L_{t+1,b} = \hat{L}_{t+1,b}$. Our state variable (the information we need to make a decision) is then given by $S_t = (R_t, L_t)$. If we wish to solve the problem using a myopic policy, we could form a contribution function:

$$C_t^\pi(x_t|S_t) = \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} c_{tad} x_{tad} \tag{4}$$

and then solve:

$$X_t^\pi(S_t) \quad = \quad \arg\max_x C_t^\pi(x|S_t) \tag{5}$$

subject to (1)-(2). Here, $\pi \in \Pi^M$ is a particular *policy* where Π^M is our class of myopic policies (there can be more than one due to other choices that might be made in the choice of contribution function or the resources that are considered in the optimization problem).

3 An approximate dynamic programming policy

Our goal is to produce a *dynamic policy* that considers the impact of decisions now on the future. One class of dynamic policies add to the myopic contribution function an estimate of the value of a resource in the future. Such a function would look like:

$$C_t^\pi(x_t|S_t) = \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} c_{tad} x_{tad} + \sum_{a'\in\mathcal{A}} \bar{v}_{ta'} \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} \delta_{a'} x_{tad} \tag{6}$$

where:

$\bar{v}_{ta'} =$ The approximate value of a resource with attribute a' *using the information available at time t.*

Thus, if we act on a resource with attribute a with a decision d, we produce a resource with attribute a', and we value this resource using a linear approximation with slope $\bar{v}_{ta'}$. Our challenge is estimating $\bar{v}_{ta'}$, which we do in a very simple way. We are going to step forward through time, using a Monte Carlo sample of new information $(\hat{R}_t(\omega), \hat{L}_t(\omega))$. In this case, solving the decision problem (5), even if we use (6) for the contribution function, is still a simple assignment problem. Let:

$\hat{v}_{ta'}(\omega) =$ The dual variable for equation (1).

$\hat{v}_{ta'}(\omega)$ is a sample realization of a random variable $\hat{V}_{ta'}$ because it depends on the random sample that we took to obtain the new arrivals of resources and tasks. Since $\hat{v}_{ta'}(\omega)$ is random, we obtain an estimate of the value of a resource using a simple stochastic approximation procedure:

$$\bar{v}_{ta'}^n = (1-\alpha^n)\bar{v}_{ta'}^{n-1} + \alpha^n \hat{v}_{ta'}^n(\omega^n) \tag{7}$$

where α^n is a stepsize at iteration n. There is a sizable literature on stepsizes; these can be deterministic, such as $\alpha^n = a/(a+n)$, or stochastic, where the stepsize depends on the progress of the algorithm.

4 Using aggregation to improve accuracy

Regardless of how $\hat{v}_{ta}(\omega^n)$ is computed, we face the problem that if the attribute space $\mathcal{A}$ is large, we simply are not going to be able to compute very many observations $\hat{v}_{ta'}(\omega)$ for each element $a' \in \mathcal{A}$. Furthermore, experimental evidence indicates that there can be a high level of variability in these values. As a result, the approximate values $\bar{v}^n_{ta'}$ can exhibit a fairly high level of statistical error.

A natural way to overcome this statistical problem is to represent the approximate values at a more aggregate level. Let:

$$\begin{aligned} G^g : \mathcal{A} &\rightarrow \mathcal{A}^g & (8)\\ &= \text{A mapping from } \mathcal{A} \text{ to an aggregate attribute space } \mathcal{A}^g & (9)\\ \mathcal{G} &= (0, 1, \ldots, |\mathcal{G}|) & (10)\\ & \quad \text{A family of aggregation functions, where } g = 0 \text{ represents the most disaggregate level} & (11)\\ \bar{v}^g_{ta'} &= \text{The approximate value of a resource with attribute } a' \in \mathcal{A}^g. & (12) \end{aligned}$$

We compute $\bar{v}^g_{ta'}$ using:

$$\bar{v}^{g,n}_{ta^g} = (1 - \alpha^n)\bar{v}^{g,n-1}_{ta^g} + \alpha^n \hat{v}^n_{ta}(\omega^n) \quad a \in \mathcal{A}, a^g \in \mathcal{A}^g \tag{13}$$

where α^n is a stepsize between 0 and 1.

Aggregation has been a widely studied topic in dynamic programming where it has been viewed as a tool for reducing the size of the state space (see, for example, Mendelssohn, 1982, Bertsekas and Castanon, 1989). Our interest is using aggregation to produce more accurate estimates of a value function. One class of strategies involves using aggregation to reduce statistical noise at the cost of introducing structural error. In this paper, we investigate the idea of estimating the value of a resource at different levels of aggregation, and then using a weighted combination of these estimates, as in:

$$\bar{v}^n_{ta} = \sum_{g \in \mathcal{G}} w^g \bar{v}^{g,n}_{tG^g(a)} \tag{14}$$

Combining different estimates to produce a more robust statistic is a widely studied topic in statistics (see, for example, Hastie, Tibshirani and Friedman, 2001). This strategy can also be placed in the broader context of linear regression, where the weights are the coefficients of different explanatory variables (Tsitsiklis, and Van Roy, 1997, Bertsekas and Tsitsiklis, 1996). In this setting, it is useful to take advantage of recursive estimation techniques (see, for example, Ljung and Soderstrom, 1983) to estimate the parameters at each iteration.

George, Powell and Kulkarni (2003), have derived formulas for the weights that are shown to be optimal (in terms of minimizing the variance of the estimates). The idea is similar to the standard formula for combining the estimates of two parameters from Bayesian theory:

$$w^{(g)} = \frac{1}{\sigma^{(g)^2}} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\sigma^{(g')^2}} \right)^{-1} \quad \forall \ g \in \mathcal{G} \tag{15}$$

These weights use the inverse of the variance of the statistics. The weights are optimal if the variances are known, and if the statistics are independent. In practice, neither is true. George, Powell and Kulkarni (2003) found the optimal weights that account for the lack of independence of hierarchically estimated weights, but also found that while the weights can be significantly different from those given in equation (15), the quality of the estimates were virtually identical, especially for the case where the variances are not known *a priori.* For the case where the variances are not known, a variation of (15) proved to work quite well. The variation requires estimating the variances of the statistics from data. The standard formula for estimating variances is:

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

where $N_a^{(g)}$ is the number of observations of the g^{th} level of aggregation of attribute a. This formula, however, ignores the fact that the estimate of the value of a resource at an aggregated level will generally be biased (due to the structural error). We need to estimate the variance reflecting both the noise as well as the bias. For this purpose, we estimate:

$$
\begin{aligned}
\tilde{\mu}_a^{(g)} &= \text{An estimate of the bias in the estimate, } \bar{v}_a^{(g)}, \text{ with respect to the true value.} \\
&= \bar{v}_a^{(g)} - \bar{v}_a^{(0)}
\end{aligned}
$$

The weights, then, are given by:

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

5 Some experimental results

This formula was applied to a dynamic programming problem involving the management of drivers. For the purposes of estimating the value of drivers in the future, three types of attributes were considered:

- Location of the driver - This was represented at three levels of aggregation: "trade area" (the finest level, with about 400 points to cover the continental U.S.), "region" (100) and "area" (10).
- Driver domicile - This captures the home location of the driver, which were represented at the level of "regions" and "areas."
- Driver type - Drivers were organized into three major groups: teams (two drivers per tractor), solos (one company driver per tractor), and independent contractor (solo drivers who owned their own equipment).

Level of aggregation	Location	Driver domicile	Fleet type	Number of possible attributes
1	Trade area	Region	I	120,000
2	Region	Region	I	30,000
3	Region	Area	I	3,600
4	Region	X	I	300
5	Area	X	I	36
6	Area	X	X	12
7	X	X	X	1

Table 1: Levels of aggregation used to represent the driver attributes for the value function. “I” means the attribute was included without aggregation, “X” means the attribute was excluded.

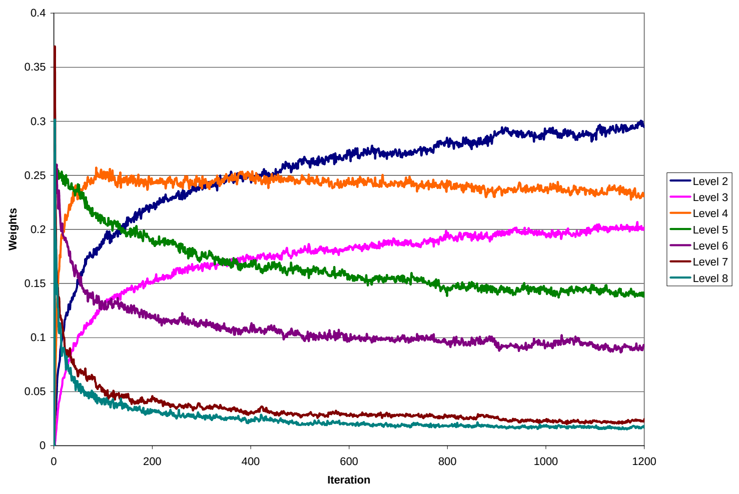

Figure 1: Weights at each level of aggregation at each iteration, showing the increasing weight given to the disaggregate levels as the algorithm progresses.

Each of these three dimensions can also be aggregated by completely ignoring them. Table 1 shows the seven levels of aggregation that we tested. Figure 1 shows the weights given to each level of aggregation as the algorithm progressed. The figure shows that the more aggregate levels were given higher weights in the beginning. After approximately 500 iterations, the weights decreased monotonically with the level of aggregation, with the most disaggregate level receiving the highest weight (on average).

This paper opens up a variety of issues that fall under the umbrella of “approximate dynamic programming.” This work is starting to emerge from the arena of small, specialized problems into problems that are of interest to the transportation and logistics community. This topic could be easily expanded into a tutorial that would be very relevant to anyone working on dynamic routing and scheduling problems.

References

D. Bertsekas. and D. Castanon "Adaptive aggregation methods for infinite horizon dynamic programming," *IEEE Transactions on Automatic Control* 34, 589–598 (1989).
D. Bertsekas. and J. Tsitsiklis *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA. 1996. A. George, W. B. Powell and S. Kulkarni "The statistics of hierarchical aggregation for multiattribute resource management," Technical report, Princeton University, Department of Operations Research and Financial Engineering, 2003. T. Hastie, R. Tibshirani and J. Friedman *The Elements of Statistical Learning*, Springer series in Statistics, New York, NY., 2001. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA., 1983. R. Mendelssohn, ''An iterative aggregation procedure for Markov decision processes', *Operations Research* 30, 62–73, 1982. J. Tsitsiklis, and B. Van Roy, ''An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control* 42, 674–690, 1997.