Online Appedix: Multiattribute Search: Empirical Evidence and Information Design

Pedro M. Gardete Megan Hunter

March 27, 2024

1 Data Cleaning

A few unusual patterns emerged while analyzing the clickstream data. First, a few sessions were very short, resembling 'bouncing' behaviors, in which a user visits the website and leaves without triggering additional action flags, often adding up to a very short overall session time. This type of activity is typically generated by users who visited the website by mistake, or by 'bots' scraping the Internet for content. These users are likely to have very different objectives than purchasing a car on the platform. We control for this type of behavior by eliminating sessions that triggered only a page visit or that took less than 5 seconds in terms of the overall measured activity (e.g., a session that triggers two events, with only 3 seconds apart, is eliminated). We also removed a few users with very unusual browsing behaviors. For example, users who exhibited activity across multiple IP addresses belonging to different countries within a very short period of time were eliminated from the sample.

2 Programming the Decision Tree

The decision tree has three fundamental objects. The first one is a node, which corresponds to each possible "action state" (set of search actions taken) that the consumer may face during her search. By action state, we mean a value that identifies which search actions the consumer has already taken.

The second fundamental object is a partition. This object identifies a set of eligible simulations at a given node. Each node contains a set of eligible partitions, each of which the consumer may encounter at that node. For example, when the consumer searches an action with a binary value, she partitions the set of eligible simulations into two subsets, and selects one depending on the observed value. The corresponding child node features both of those partitions.

Finally, the third object is an (search) action, which belongs to a specific partition. Each partition may contain multiple admissible search actions. The action object has two roles. First, it links a partition to its child partitions and their probabilities. For example, if a

consumer takes a given search action while in a given partition, she will transition to different partitions of a sub-node with different probabilities. Second, an action also contains its own value, which is obtained by integrating the value of the possible future paths it can lead to.

The connections among all of these objects describe the DM's dynamic problem. Below, we provide a stylized representation of each of the fundamental decision-tree objects. (The actual representation is more complex in that involves vectors of pointers and additional properties and methods.)

Node:

```
node{

    int id;
    vector of <partition> v_p;

}
```

Partition:

```
partition{

    vector of <action> v_a;
    vector of <int> simulations;
    vector of <terminal_utilities> v_tu;

}
```

Action:

```
action{

    vector of <partition> v_p;
    vector of <transition_probabilities> v_tp;
    double EV;

}
```

In order to understand the links among the objects above, consider Figure 4 in the main paper. When the consumer starts her search, she is in node 1, which contains a single partition. This single partition is saved as an element of the vector "v_p" of node A. The partition is associated with an object that has a vector of possible actions "v_a", a vector of identifiers of the simulations it contains (field "simulations") and a vector of terminal utilities. In the example, vector "v_tu" has three elements, corresponding to the inside and outside options as well as the action of searching x_1. Finally, each action element in vector "v_a" has three fields. First, an action keeps track of the partitions it can transition the decision-maker to (in node A, the search action can transition the decision-maker to nodes B and C). It also keeps track of the transition probabilities to each of those partitions (field "v_tp"), and

finally, it keeps track of its own expected value (field "EV"). In Figure 4, when the decision-maker searches x_1 she understands that the belief space in node A will be partitioned into the draws in node B or the draws in node C.

We create the decision tree bottom-up, starting at the last level of nodes corresponding to full information. We populate the last-level nodes with single belief draws. We then proceed to populate all direct parent nodes. For each one, we cycle through the elements in the partitions of the child nodes and merge according to the action taken. The procedure stops once the top of the tree is reached. The estimation code is written in C++ and is available from the authors.

3 Covariance With Unobservable Characteristic

In the estimated model we have assumed the observable characteristics are independent of the unobservable one. This assumption is valid to the extent that consumers react to vehicle photos in an idiosyncratic fashion and is commonly maintained in the literature. In this section we outline detailed steps in order to estimate the covariance vector between the observable characteristics and the unobservable one.

Because the covariance between observed characteristics can be estimated in a first stage, it is kept constant through the estimation of the structural parameters. In contrast, the covariance parameters relating the unobserved characteristic with the observed ones are unknown and need to be estimated simultaneously with the preference parameters. As the guesses for the covariance parameters change, so will the simulations of the unobserved characteristic, as in usual simulation-based estimation models.

One complication that arises when incorporating an unobserved characteristic in the context of state-partitioning trees is that the probability of two draws of a continuous variable coinciding is equal to zero. This uniqueness means that searching the unobserved characteristic not only informs the decision maker (DM) about it, but it also informs the DM of all of the other search characteristics perfectly. Consider the example in Table **??**. Unless the cost of evaluating characteristic ν is prohibitively high, the DM should opt to immediately learn ν, because that decision will also provide full information about the remaining characteristics (e.g., learning that $\nu = 0.3$ informs the DM that she is in case 1) with certainty.

Table 1: Example of Simulations with Continuous Unknown Characteristic ν

Simulation #	x_1	x_2	ν
1	1	1	0.3
2	1	2	-1.3
3	2	2	0.6

An apparent solution would be to discretize the space of ν, just as one can do with continuous observable characteristics. However, that discretization procedure will fail to update the likelihood for small optimization steps. This is a common issue, usually present in the context of accept/reject estimators (see **?**).

Table 2: Example of Simulations with Discrete Signal $\widehat{\nu}$

Simulation #	x_1	x_2	$\widehat{\nu}$
1	1	1	1.5
2	1	2	0
3	2	2	1.5

Instead, assume that when consumers take a search action with respect to the unobserved characteristic, they learn a signal $\widehat{\nu} = \nu + \eta$, where η is normally distributed $N\left(0, \sigma_\eta^2\right)$. For estimation purposes, signal $\widehat{\nu}$ has a fixed support, for example based on the sparse grid proposed by **?**. In this way, the number of support points for ν grows with the number of vehicles in the consumer's consideration set. Given the covariance matrix Σ, which includes the observable characteristics plus the unobserved characteristic ν, it is possible to write the distribution of $\widehat{\nu}$ conditional on the observable characteristics. For a specific parameter guess, we take draws of $\widehat{\nu}$ conditional on observed characteristics, from the fixed support. The simulation probability of each support point $\widehat{\nu}_i$ is given by $\check{Pr}\left(\widehat{\nu}_i \middle| x_1, x_2\right) = \frac{f_{\widehat{\nu}}(\widehat{\nu}_i|x_1,x_2)}{\sum_{j=1}^{NS} f_{\widehat{\nu}}(\widehat{\nu}_j|x_1,x_2)}$ where $f_{\widehat{\nu}}\left(\widehat{\nu}_i \middle| x_1, x_2\right)$ is the density of the signal conditional on the observable characteristics, and NS is the number of sparse grid support points used. As NS grows, our simulation probability $\check{Pr}\left(\widehat{\nu}_i \middle| x_1, x_2\right)$ approaches the exact probability $Pr\left(\widehat{\nu}_i \middle| x_1, x_2\right)$ induced by the normal distribution.

Table **??** provides an example. Suppose that only three draws were taken, and the sparse grid of $\widehat{\nu}$ is $\{-1.5, 0, 1.5\}$. According to Table **??**, searching the unknown characteristic ν partitions the simulation set into subsets $\{1, 3\}$ and $\{2\}$. The expected value of ν can be calculated easily. Before any search activities take place, the DM is in an initial partition P_1, and the expected value of ν is calculated according to

$$\begin{aligned} E\left(\nu \middle| P_1\right) = \frac{1}{3}\big[&E\left(\nu \middle| x_1 = 1, x_2 = 1, \widehat{\nu} = 1.5\right) \\ &+ E\left(\nu \middle| x_1 = 1, x_2 = 2, \widehat{\nu} = 0\right) \\ &+ E\left(\nu \middle| x_1 = 2, x_2 = 2, \widehat{\nu} = 1.5\right)\big] \end{aligned}$$

However, if the DM searches the unobserved characteristic and finds $\widehat{\nu} = 1.5$ such that her partition is $P_2 = \{1, 3\}$, the expected value of the unobserved characteristic becomes

$$\begin{aligned} E\left(\nu \middle| P_2\right) = \frac{1}{2}\big[&E\left(\nu \middle| x_1 = 1, x_2 = 1, \widehat{\nu} = 1.5\right) \\ &+ E\left(\nu \middle| x_1 = 2, x_2 = 2, \widehat{\nu} = 1.5\right)\big] \end{aligned}$$

The utility from the observed characteristics is calculated similarly.

As before, the simulation draws over $\widehat{\nu}$ may not change across small optimization steps. However, the conditional expectations above do change continuously with the covariance parameters, meaning that this approach does produce a likelihood function that is sensitive to small changes to parameter values.[1]

[1]Note also that the loss in efficiency from not updating the simulations of $\widehat{\nu}$ continuously with the covariance parameters decreases as the number of simulations increases.

It remains to discuss the identification of the covariance parameters between the observed characteristics and the unobserved utility component ν. Whereas search costs are identified by the *average* propensity of each search action being taken, the covariance parameters are identified by information-dependent search patterns concerning the unobserved characteristic. For example, suppose a consumer knows a given vehicle's price. One can imagine that if the price is moderately high, she may opt for the outside option immediately, conditional on the preference and search cost parameters. If the price were lower, it is possible that the consumer would have continued her search of the vehicle's attributes, including component ν. The price threshold at which the consumer is indifferent between continuing her search for ν identifies the covariance between price and the unobserved component: If the consumer selected the outside option without searching further, despite facing only a moderate price point, then it must have been that the residual variance of the unknown component was not high enough to justify further search. On the other hand, continuing search despite the current low prior on the vehicle's expected utility is a signal of high variance of component ν, conditional on the known characteristics. The parameters of the variance-covariance matrix thus identify the residual variance of component ν, conditional on knowledge of the vehicles' remaining characteristics.[2]

4 Scaling the Search Problem Up

While the method we propose helps tackle complex search problems significantly, it does not do away with the demanding requirements of search problems. The actual size of the problem depends on a number of factors, such as the number of alternatives considered, the number of search actions contemplated, the number of search characteristics, and the number of levels within each characteristic (1 previous owner, 2 previous owners, 3 previous owners, etc). It is challenging to precisely estimate computation requirements since decision trees associated with search are largest "in the middle", since in its beginning no information is yet known, and in the end all information is available. In contrast, the middle of the decision tree features the maximum number of possible search paths + possible feature combinations.

Still, in order to provide a sense of how fast requirements grow with the size of the problem, we outline the memory footprints we encountered in our context, in Table **??**.

[2]The question of separate identification of the preference and covariance parameter may linger in the reader's mind. To this end, notice that in our setting the preference parameters are identified from the subset of consumers who take all available search actions, leaving partial search decisions of other consumers to identify the covariance between the observable characteristics and the unobservable component.

Table 3: Memory Usage Associated with Consideration Set Size of each Customer

Consideration set size	Total Memory Use (Traditional Approach)	Peak Memory Usage (In Tandem Approach)
1 alternative	0.6 MB	0.5 MB
2 alternatives	2.6 MB	0.8 MB
3 alternatives	25 MB	1.5 MB
4 alternatives	301 MB	126 MB
5 alternatives	3 GB	1.1 GB
6 alternatives	28.9 GB	10.2 GB
Our Sample:	289 GB	30 GB

Note: The values above were measured by considering estimation with individual consumers at a time. While the actual memory per consumer is significantly lower when the full sample is considered, the relative magnitudes are informative. The "Our Sample" row depicts the highest memory use observed during estimation of the model using the sample of 2,659 consumers.

First, we note that the in-tandem approach significantly reduces the memory needed to contemplate the search activity of our sample. Second, note that memory necessary for estimation increases exponentially in the number of alternatives contemplated by consumers, keeping other factors constant. While our method makes problems of moderate size addressable, new approaches are still needed to tackle problems where consumers search attributes of many alternatives in piecemeal fashion.