Multinomial processing tree models: An implementation

XIANGEN HU

University of Memphis, Memphis, Tennessee

Multinomial processing tree (MPT) models have been widely used by researchers in cognitive psychology. This paper introduces MBT.EXE, a computer program that makes MPT easy to use for researchers. MBT.EXE implements the statistical theory developed by Hu and Batchelder (1994). This user-friendly software can be used to construct MPT models and conduct statistical inferences, including point and interval estimation, hypothesis testing, and goodness of fit. Furthermore, this program can be used to examine the robustness of MPT models. Algorithms for parameter estimation, hypothesis testing, and Monte Carlo simulation are presented.

Multinomial processing tree (MPT) models (Riefer & Batchelder, 1988) are a family of substantive models for cognitive psychology. This family of models makes the basic assumption that certain cognitive processes may be serial in nature—for example, that storage takes place before retrieval (Riefer & Batchelder, 1988), and these models represent these processes in terms of branching trees, with parameters being the conditional link probabilities from one stage to another stage. To understand MPT models intuitively, one may imagine that items are processed in a tree structure. There are several choice points. At each choice point, there are several possible choices, each associated with a probability. At the bottom of the tree are observable categories. MPT models have been used to analyze data from a variety of paradigms, including ones in the areas of memory, reasoning, perception, and social psychology (see Batchelder & Riefer, 1999, for a review).

Hu and Batchelder (1994) have formalized this family of models mathematically. This family of MPT models is a subfamily of multinomial models with special parameterization; specifically, the categorical probabilities of the multinomial takes the form of Equation 2 below. Multinomial models with such categorical probabilities provide a unified mathematical representation for the family of MPT models. Hu and Batchelder called the family of such models *general processing tree (GPT) models*. They studied the statistical inferences for GPT models, using the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977), and demonstrated that

the collection of GPT models has many statistically tractable properties. For the MPT models discussed in Riefer and Batchelder (1988) and Batchelder and Riefer (1999), the mathematical form of the categorical probability is in the form of Equation 2. The distinctions between GPT and multinomial binary tree are minor and are not important for the present paper; hence, the two names will be used interchangeably.

I will first introduce the formal definition of GPT models and then briefly review the existing programs for GPT models. The primary purpose of the present paper is to introduce the computer program. For this purpose, the algorithms will be briefly explained in the paper; detailed equations for the algorithm will be presented in the appendix.

DEFINITION OF GENERAL PROCESSING TREE MODELS

DEFINITION 1 (Hu & Batchelder, 1994). *Let*

$$\mathcal{M}(\Theta; \langle c_{ij} \rangle; \langle a_{ijs} \rangle; \langle b_{ijs} \rangle) \quad (1)$$

be a parametric multinomial model defined over J observable categories with S functionally independent parameters, $\Theta = (\theta_1, \dots, \theta_S)$. Then $\mathcal{M}$ is a general processing tree (GPT model) in case there are positive integers I_j ; nonnegative integers a_{ijs} and b_{ijs} ; and nonnegative reals c_{ij} so that the category probabilities $p_j(\Theta)$ can be written in the form of

$$p_j(\Theta) = \Pr(C_j; \Theta) = \sum_{i=1}^{I_j} p_{ij}(\Theta), \quad (2)$$

where

$$p_{ij}(\Theta) = \Pr(B_{ij}; \Theta) = c_{ij} \prod_{s=1}^S \theta_s^{a_{ijs}} (1 - \theta_s)^{b_{ijs}}, \quad (3)$$

is called branch probability, and B_{ij} is the i th branch leading to the j th category.

The author thanks William Batchelder, David Riefer, Edgar Erdfelder, and Richard Schweikert for their helpful comments and feedback during the development of the software MBT.EXE. The author thanks William Dwyer, William Marks, William Shadish, and Glenn Phillips for their helpful comments about the manuscript. The program can be obtained via anonymous ftp (<http://xhuoffice.psyc.memphis.edu/gpt/index.htm>). Correspondence concerning this article should be addressed to X. Hu, Department of Psychology, University of Memphis, Memphis, TN 38152 (e-mail: x.hu@mail.psyc.memphis.edu).

$$\sum p_j(\Theta) = 1,$$

where $c_{ij} > 0$, and the a_{ijs} , and b_{ijs} are nonnegative integer structural constants, $j = 1, \dots, J$, $i = 1, I_j$, $s = 1, \dots, S$.

To understand this definition, let us consider a simple example of tossing three coins. Assuming that the observed outcome is the number of heads, the probabilities for the observed categories $P_j(\Theta)$, $j = 1, 2, 3$, can be obtained (see Equation 4, bottom of page). The structure constants are obtained in Table 1.

A multinomial model with categorical probability of the form in Equation 2 is called a GPT model, simply because there are possible tree representations for the model. For example, the coin-tossing experiment can be graphically represented (see Figure 1). To understand the meaning of the structure constants c_{ij} , a_{ijs} , and b_{ijs} , assume that the first coin is fair and that the other two coins are identical. With such an assumption, the number of parameters in the model is reduced to one ($\theta_1 = 0.5$ and $\theta_1 = \theta_2 = \theta$) and model Equations 4 are changed into

$$\begin{cases} P_1(\Theta) = \Pr\{H, H, H\} = .5\theta^2 \\ P_2(\Theta) = \Pr\{H, H, T\} = .5\theta(1-\theta) + .5(1-\theta)\theta + .5\theta^2 \\ P_3(\Theta) = \Pr\{H, T, T\} = .5(1-\theta)^2 + .5(1-\theta)\theta + .5\theta(1-\theta) \\ P_4(\Theta) = \Pr\{T, T, T\} = .5(1-\theta)^2 \end{cases} \quad (5)$$

and the structure constants are changed from those in Table 1 to those in Table 2.

PROGRAMS FOR GENERAL PROCESSING TREE MODELS

Some approaches to the implementation of MBT models already exist. The first approach is to use generic programming language and to write code for a specific model. For example, Riefer and Batchelder (1991) used a program written in Fortran for a simulation study of the storage-retrieval model (Batchelder & Riefer, 1980). The second approach is to write a user-friendly program for a specific model. For example, Erdfelder (1992) has written a program for the storage-retrieval model (Batchelder & Riefer, 1986), and in the same spirit, Hu (1990) has developed a software package for the source-monitoring model (Batchelder & Riefer, 1990). The third approach, and the most extensively used, is to develop a way to handle any tree model within a common framework—namely, one not limited by any specific model. For example, Dodson, Prinzmetal, and Shimamura (1998) have used Microsoft Excel solver to estimate parameters for

Table 1
Structure Constants for the Coin-Tossing Example
(With Categorical Probabilities in Equation 4)

i	j	p_{ij}	c_{ij}	a_{ij1}	b_{ij1}	a_{ij2}	b_{ij2}	a_{ij3}	b_{ij3}
1	1	$\theta_1\theta_2\theta_3$	1	1	0	1	0	1	0
1	2	$\theta_1\theta_2(1-\theta_3)$	1	1	0	1	0	0	1
2	2	$\theta_1(1-\theta_2)\theta_3$	1	1	0	0	1	1	0
3	2	$(1-\theta_1)\theta_2\theta_3$	1	0	1	1	0	1	0
1	3	$\theta_1(1-\theta_2)(1-\theta_3)$	1	1	0	0	1	0	1
2	3	$(1-\theta_1)(1-\theta_2)\theta_3$	1	0	1	0	1	1	0
3	3	$(1-\theta_1)\theta_2(1-\theta_3)$	1	0	1	1	0	0	1
1	4	$(1-\theta_1)(1-\theta_2)(1-\theta_3)$	1	0	1	0	1	0	1

Note— θ is the probability of observing a head in any trial for the unfair coin.

any MPT model. The program introduced in this paper belongs to the third approach.

There are obvious limitations to the first two approaches. The first approach can not be widely used because it is not user friendly and can be used only by those who know how to use the specific compilers. Furthermore, one has to write different codes for different models. The advantage of such an approach is the flexibility of the program. For example, a piece of code written in Fortran can be used in different platforms, so one can use it on whichever computer he or she is using. The second approach is very useful for a given model with a specific experimental design; however, it cannot be used for other designs or other models. The advantages for the second approach are its user-friendly nature (it does not require the user to write any code) and its speed, owing to the algorithms being optimized for the specific purpose.

The third approach is the most preferable. With this approach, one can analyze data by specifying a model and applying the general purpose program (or approach) to carry out the statistical inferences. Dodson et al.'s (1998) approach is one that is very useful. The advantages of Dodson et al.'s method are that most personal computers are equipped with Microsoft products and that it is easy to use. However, it can be used only to estimate parameters. In fact, there is other model information that is important to researchers, such as the asymptotic covariance matrix of the estimates (which can be used to obtain confidence intervals). Also, there are other aspects of the model that many researchers would like to explore, such as hypotheses about model parameters or the robustness of the model under violations of the assumptions.

To my knowledge, there are only two general purpose software packages that are designed for the family of models described in Hu and Batchelder. The first one is MBT.EXE, which will be introduced in this paper. The other one is AppleTree (Rothkegel, 1997), which is a

$$\begin{cases} P_1(\Theta) = \Pr\{H, H, H\} = \theta_1\theta_2\theta_3 \\ P_2(\Theta) = \Pr\{H, H, T\} = \theta_1\theta_2(1-\theta_3) + \theta_1(1-\theta_2)\theta_3 + (1-\theta_1)\theta_2\theta_3 \\ P_3(\Theta) = \Pr\{H, T, T\} = \theta_1(1-\theta_2)(1-\theta_3) + (1-\theta_1)(1-\theta_2)\theta_3 + (1-\theta_1)\theta_2(1-\theta_3) \\ P_4(\Theta) = \Pr\{T, T, T\} = (1-\theta_1)(1-\theta_2)(1-\theta_3) \end{cases} \quad (4)$$

Figure 1. A multinomial processing tree model representation of an example of coin tossing.

version of MBT.EXE for the Mac operating system. MBT.EXE implements the statistical theory of MPT models (Hu & Batchelder, 1994), such as parameter estimation and hypothesis testing. Furthermore, MBT.EXE can be used to explore the robustness of a given model with simulations.

ALGORITHMS

Parameter Estimation

The algorithm for the parameter estimation is provided in the Appendix. The basic idea for parameter estimation is to obtain a set of parameters $\hat{\Theta}$ so that the distance between the observed and the expected categorical frequencies is minimized. A distance measure commonly used in statistics is the χ^2 distance measure

$$\chi^2[\mathbf{R}, \mathbf{P}(\Theta)] = \sum_{j=1}^J \frac{[n_j - NP_j(\Theta)]^2}{NP_j(\Theta)}, \quad (6)$$

where $\mathbf{R} = (n_1, \dots, n_J)$ is a vector of the response frequencies and $\mathbf{P}(\Theta) = [NP_1(\Theta), \dots, NP_J(\Theta)]$ is a vector of expected categorical frequencies, given parameter Θ . Equation 6 is used very often, simply because it is a distance measure, in which, at the same time, the minimized function is served as a test statistic for the goodness of fit, asymptotically distributed as chi-square distribution. For the same reason,

$$G^2[\mathbf{R}, \mathbf{P}(\Theta)] = 2 \sum_{j=1}^J n_j \ln \left(\frac{n_j}{NP_j(\Theta)} \right) \quad (7)$$

is also used as an alternative. It turns out that both Equation 6 and Equation 7 are just special cases of a family of distance measures that can be characterized as a power divergence family (Read & Cressie, 1988):

$$2nI^\lambda[\mathbf{R}, \mathbf{P}(\Theta)] = \frac{2}{\lambda(\lambda+1)} \sum_{j=1}^J n_j \left[\left(\frac{n_j}{NP_j(\Theta)} \right)^\lambda - 1 \right], \quad (8)$$

where λ can be any real number. For example, when $\lambda = 0$, Equation 8 is reduced to Equation 7, and when $\lambda = 1$, Equation 8 is reduced to Equation 6. A unified approach of goodness-of-fit measures in the form of Equation 8 has been systematically studied by Read and Cressie (1988). Differences between the parameter estimates for different λ values have been examined by García-Pérez (1994). Furthermore, for GPT models, parameter estimation on the basis of Equation 8 needs to be studied (Hu & Batchelder, 1994).

The estimation procedure that is used in MBT.EXE is based on the distance measure of Equation 8. The basic idea is to implement the EM algorithm (Dempster et al., 1977). An EM algorithm has been used to obtain maximum likelihood estimates (MLEs) on the basis of Equation 7. The EM algorithm is also called the *missing information principle*, and it is commonly used in statistics for parameter estimations for models with incomplete information. The example of the coin toss in the previous section is a model with incomplete information. In this model, some of the observable categories are combinations of the unobserved conditions (or branches). For instance, the observed category $\{H, H, T\}$ is a combination of three different branches: (HHT) , first two tosses are heads and last toss is tail; (HTH) , first and last tosses are heads and the second toss is a tail; and (THH) , first toss is a tail and the last two tosses are heads. Because the data for the model only consist of the number of heads (or tails) without considering the order, so the order of the outcome is not directly available in the data; hence, the observed data do not provide complete information about the details of the tosses. A complete information GPT model is a model in which all the branches are observed—namely, each of the observed categories contains only one branch. The EM algorithm is an iteration of trials; each contains an E-step and an M-step. In the E-step, complete information (all the branch frequencies) is obtained as an expectation of the branch frequencies, given the parameter values from the previous trial; in the M-step, MLEs of the parameters are obtained on the basis of the complete information model. MBT.EXE extends the algorithm to the general distance measure of Equation 8, so the M-step is not necessarily obtaining MLEs for the parameters. The reason MBT uses the EM algorithm is that both the E-step and the M-step in each trial

Table 2
Structure Constants for the Coin-Tossing Example With
Parameter Constraints $\theta_1 = 0.5$ and $\theta_1 = \theta_3 = \theta$

<i>i</i>	<i>j</i>	p_{ij}	c_{ij}	a_{ij1}	b_{ij1}
1	1	$0.5\theta^2$	0.5	2	0
1	2	$0.5(1-\theta)\theta$	0.5	1	1
2	2	$0.5(1-\theta)\theta$	0.5	1	1
3	2	$0.5\theta^2$	0.5	2	0
1	3	$0.5(1-\theta)^2$	0.5	0	2
2	3	$0.5(1-\theta)\theta$	0.5	1	1
3	3	$0.5(1-\theta)\theta$	0.5	1	1
1	4	$0.5(1-\theta)^2$	0.5	0	2

Note.— θ is the probability of observing a head in any trial for the unfair coin.

of the iteration can be easily formulated in explicit forms. It is possible to use other maximization methods, such as Newton–Raphson, for the maximization of Equation 8. However, the EM algorithm used in MBT.EXE always keeps the parameters, which are probabilities, within the range of (0, 1), whereas other methods may try to maximize the distance measure (Equation 8) for parameter values that are outside of (0, 1).

MBT.EXE also makes it easy to test hypotheses among the parameters. As was discussed in Hu and Batchelder (1994), testing hypotheses in MPT models is straightforward. One can always represent the alternative hypothesis in terms of another MPT model and analyze the model. In MBT.EXE, two simple kinds of hypothesis testing are implemented: (1) setting parameters to constants, and (2) equating parameters to each other.

Fixing a parameter as a constant probability or constant proportion. There are three reasons that one wants to fix a parameter as constant. (1) The parameter in a model is no longer unknown. For example, given that one of the three coins is a fair coin, the model can be simplified by replacing the corresponding parameter with .5. (2) It is common that a given model may have more parameters than the degrees of freedom in the data, and the researcher wants to make the model identifiable (with fewer free parameters than the degrees of freedom in the data) by assuming some parameters as fixed constants. For example, one might want to set some guessing parameters as constant probabilities. (3) In other cases, some parameters in the model may correspond to fixed proportions. For example, in an experiment, the proportions of subjects in different experimental groups are controlled by the experimenter. This software takes these into account and provides some convenient features for the users.

To fix a parameter as a constant, there is no need to rewrite Equation 8—for example, to set $\theta_{s_0} = \alpha$ [constant $\alpha \in (0, 1)$]; the only modification of the algorithm is to add such constraints in Step 3 after Equation 11.

As I have mentioned, a parameter can be assumed to be a constant probability (such as in the case of constant guessing probabilities) or constant proportions (such as fixed proportions for experimental conditions). If the estimation is based on $\lambda = 0$, the computation of the covariance matrix of the parameter estimates differs for the two situations. For example, if θ_{s_0} is fixed as a constant probability, then θ_{s_0} will itself have variability and be treated as an estimated MLE that can contribute to the computing of the covariance matrix. If θ_{s_0} is fixed as a constant proportion, θ_{s_0} is a real constant and will not produce any variations in the model. In this case, if one wants to obtain confidence intervals for other estimates, it is important to remove the s_0 column and the s_0 row from the observed Fisher Information matrix.

Equating one parameter to other parameters. Setting parameters equal to each other is one of the most frequently used ways of formulating hypotheses, especially when dealing with observations from two or more groups of subjects (or two or more different items). Setting two or more parameters equal to each other means that what-

ever processes the parameters correspond to are not different. The source-monitoring example in Hu and Batchelder (1994, pp. 33–36) discusses the instance in which one wants to set $\theta_1 = \theta_2$, $\theta_3 = \theta_4$ (Model 5a) to represent the hypothesis that there are no differences for the two sources, both in stimulus detection and in source discrimination. Although equating one parameter to another parameter will produce another model, MBT.EXE makes it very easy. There is no need to retype model equations for the new model, if the original model already exists; one just changes the structural constants, as in the example from Hu and Batchelder (1994), and the algorithm is just slightly modified by adding Equation 10 in Step 2.

Confidence Interval

In the case of $\lambda = 0$, MBT.EXE provides parameter estimates with estimated standard deviations. The standard deviations of the parameter estimates are used to construct confidence intervals for the estimates. Standard deviation of the parameter estimates are diagonal elements of the covariance matrix, which is the inverse of the observed Fisher information matrix $\mathbf{I}(\Theta)|_{\Theta=\hat{\Theta}}$, where $\mathbf{I}(\Theta)$ can be computed from the following formula:

$$\begin{aligned} (\mathbf{I}(\Theta))_{s \times r} &= \frac{\partial^2 \ln L(\Theta; \langle n_j \rangle)}{\partial \theta_s \partial \theta_r} \\ &= \begin{cases} \frac{\partial \alpha_s(\Theta)}{\partial \theta_r} - \frac{\partial \beta_s(\Theta)}{\partial \theta_r}, & \text{if } r \neq s \\ \frac{\partial \alpha_s(\Theta)}{\partial \theta_s} - \frac{\partial \beta_s(\Theta)}{\partial \theta_s} - \left[\frac{\alpha_s(\Theta)}{\theta_s^2} + \frac{\beta_s(\Theta)}{(1 - \theta_s)^2} \right], & \text{if } r = s. \end{cases} \end{aligned}$$

where

$$\begin{aligned} \alpha_s(\Theta) &= \sum_{j=1}^J n_j \sum_{i=1}^{I_j} \left[a_{ijs} \frac{p_{ij}(\Theta)}{p_j(\Theta)} \right], \\ \beta_s(\Theta) &= \sum_{j=1}^J n_j \sum_{i=1}^{I_j} \left[b_{ijs} \frac{p_{ij}(\Theta)}{p_j(\Theta)} \right]. \end{aligned}$$

Because the MLEs of the parameters are asymptotic normal, confidence intervals are obtained by $[\theta_s - c v_{ss}, \theta_s + c v_{ss}]$, where v_{ss} is the s th element on the diagonal of the covariance matrix and c is a constant that corresponds to the specific confidence interval. MBT.EXE provides a “covariance matrix” even when $\lambda \neq 0$. In such cases, the inverse of $\mathbf{I}(\Theta)$ is not identical to the covariance matrix; it nevertheless provides some information for the reliability of the parameter estimates.

Because the estimates are based on the observed categorical frequencies, the confidence intervals are influenced by sample size. Ideally, one will have enough observations that the estimated Fisher information matrix approximates the true Fisher information matrix. In reality, sample sizes are seldom large enough, so the stan-

dard deviation of the parameters cannot be taken seriously when sample size is small. Furthermore, relationships between parameters and data are different from parameter to parameter. For a given data set, some parameters can be estimated with small standard deviations, and some of the parameters are estimated with larger standard deviations. For example, in the source-monitoring example (Batchelder & Riefer, 1990; Hu & Batchelder, 1994), standard deviations for the item detection parameters are, in general, smaller than those for source discrimination parameters. For this reason, MBT.EXE has an option to obtain standard deviations of the parameters by Monte Carlo simulation.

Monte Carlo Simulations

The collection of MPT models is a subfamily of multinomial models. Hence, the statistical theory of MPT models is based on asymptotic properties of the multinomial distribution, and data are assumed to reflect multinomial trials with identically independent distribution (IID) requirements. However, in psychological experiments, not only are sample sizes limited by the nature of the experiments, but there may also be unavoidable individual differences for subject and item parameters. One concern for researchers is the robustness of a model if the IID assumptions are violated. MBT.EXE has implemented a Monte Carlo procedure that helps users to explore the robustness of their models. The basic Monte Carlo procedure is as follows:

1. Generate categorical probabilities:

(a) Generate link probabilities: With a given mean and standard deviation for each of the parameters, a random number is generated for each of the link probabilities on the basis of mean and standard deviation for that parameter. There are two options for the distribution of the parameters: beta distribution and uniform distribution:

- Given μ and σ for a parameter, if a parameter is assumed to be distributed as a beta distribution, $\beta(\alpha, \beta)$, given by probability density function

$$f(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}, x \in (0, 1),$$

where $\alpha, \beta > 0$, then a beta distribution, with integer parameters α and β ,¹ will be obtained that is closest to the specification. More specifically, α and β of the beta distribution are obtained by finding the best integers α, β , so that

$$\mu \approx \frac{\alpha}{\alpha + \beta},$$

$$\sigma^2 \approx \frac{\alpha\beta}{(\alpha + \beta + 1)(\alpha + \beta)^2};$$

then a random number from $\beta(\alpha, \beta)$ is drawn for the link probability of the parameter.

- Given μ and σ for a parameter, if a parameter is assumed to be distributed as a uniform distribution with probability density function

$$f(x, a, b) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \subseteq (0, 1), \\ 0, & x \in (0, 1) - [a, b] \end{cases},$$

then a, b will be obtained on the basis of

$$\mu = \frac{a + b}{2},$$

$$\sigma^2 = \frac{(b - a)^2}{12}.$$

Then a random number from the distribution is drawn for the link probability of the parameter.

(b) Generate categorical probability: Having obtained the link probabilities for all the parameters, the categorical probabilities $p_j, j = 1, \dots, J$, can be determined with Equation 2.

2. Arrange the categorical probabilities: Arrange the categorical probabilities as $(q_0, q_1, \dots, q_{J-1}, q_J)$, where $q_n = \sum_{j=1}^n p_j, n = 0, \dots, J, q_0 = 0, q_J = 1$.

3. Simulate data: For each trial, a random number $r \in [0, 1]$ is generated from a uniform distribution. If $q_j \leq r < q_{j+1}$, then categorical frequency $C_{j+1}, j = 0, \dots, J - 1$.

4. Parameter estimation: Steps 1 to 3 will repeat until the number of trials equals the specified limit, at which point a multinomial frequency table is obtained. MBT.EXE then analyzes the frequency data, using the selected model.

MBT.EXE has implemented the above procedures and provides a Monte Carlo Simulation option. One can use this option to simulate N data sets based on the source-monitoring model with given parameter values (mean and standard deviation of the parameters), each with n trials. At the end, a table of N sets of parameter estimates will be obtained. The robustness of the model can be examined by comparing the differences of the means of the estimated parameters and the specified means of the parameters for the simulation. Currently, there is no mechanism in MBT.EXE that provides statistical tests for the robustness of MPT models, which is one of the limitations of MBT.EXE that I am currently working to improve on. Next, I will list a few limitations of MBT.EXE that will be improved on in the next version of the program.

LIMITATIONS

There are several limitations for the current version of MBT.EXE. The first limitation is that MBT.EXE can handle only MPT models in the form of Equation 2—namely, parameters are in the form of θ or $(1 - \theta)$, $\theta \in [0, 1]$. It is possible to have a model in which parameters are in other forms. Examples include the ABO blood group model in human genetics (Weir, 1990) or source-monitoring models with more than two sources (Batchelder, Hu, & Riefer, 1994; Riefer, Hu, & Batchelder, 1994), where parameters in the model have the form of $(\theta_1, \dots, \theta_K)$, $\sum_{k=1}^K \theta_k = 1$ (see Figure 2 as an illustration). MBT.EXE cannot analyze such models directly. In order to use MBT.EXE, one has

Figure 2. A portion of the tree where the link probability is not in the form of θ or $(1 - \theta)$. Instead, the link probabilities are in the form of $\theta_k, k = 1, \dots, K$, with domain restriction $\sum_{k=1}^K \theta_k = 1$.

to transform the models into the same functional format as Equation 2. In fact, trees such as those in the examples above can be easily transformed into the standard form with categorical probability in the form of Equation 2 (see Figure 3 as an illustration).

Another limitation of MBT.EXE is the memory limitation of MS-DOS. The compiler limits the size of the matrix declared to accommodate the 640K RAM limitation. So, MBT.EXE can handle models only within the following limitations: the number of trees altogether < 21 , the number of branches in all trees < 61 , the number of

observable categories in all trees < 41 , the number of parameters in all trees < 21 , and the number of branches that are combined into one category < 6 . The above limitations on the size of the model can be changed. For example, one can reduce the number of parameters and increase the number of observed categories.

The third limitation of MBT.EXE is that the Monte Carlo simulation can only simulate beta distributions with parameters α, β being small integers. It would be ideal to simulate models with parameters of any specified mean and standard deviations. The current version of MBT.EXE does not provide robustness tests for MPT models.

Finally, there are other statistical analysis options for MPT models that need to be implemented: power analysis, Bayesian estimation for the model parameters, and model selection and model misspecification tests, to list a few.

SUMMARY

This paper introduces a computer program, MBT.EXE, for analyzing MPT models. This program conducts standard statistical analyses, such as parameter estimation, hypothesis testing, and Monte Carlo simulation. The program is based on the mathematical form of MPT models (Hu & Batchelder, 1994) and the EM algorithm (Dempster et al., 1977). There are several friendly features in MBT.EXE that make using MPT models easy. For example, there is no need to retype model equations for certain types of hypothesis testing. Furthermore, the Monte Carlo simulation feature of MBT allows researchers to examine the robustness of their MPT models. There are a few limitations for using MBT.EXE, and a new version of MBT.EXE is under construction.

Figure 3. The parameters $\theta_k, k = 1, \dots, K$ can be partitioned (repeatedly). In this way, a new set of link parameters, in the form of θ and $(1 - \theta)$, can be obtained. Hence MBT.EXE can be used to analyze the model.

REFERENCES

- BATCHELDER, W. H., HU, X., & RIEFER, D. M. (1994). Analysis of a model for source monitoring. In G. H. Fischer & D. Laming (Eds.), *Contributions to mathematical psychology, psychometrics, and methodology* (pp. 51-65). New York: Springer-Verlag.
- BATCHELDER, W. H., & RIEFER, D. M. (1980). Separation of storage and retrieval factors in free recall of clusterable pairs. *Psychological Review*, **87**, 375-397.
- BATCHELDER, W. H., & RIEFER, D. M. (1986). The statistical analysis of a model for storage and retrieval processes in human memory. *British Journal of Mathematical & Statistical Psychology*, **39**, 129-149.
- BATCHELDER, W. H., & RIEFER, D. M. (1990). Multinomial processing models of source monitoring. *Psychological Review*, **97**, 548-564.
- BATCHELDER, W. H., & RIEFER, D. M. (1999). Theoretical and empirical review of multinomial processing tree modeling. *Psychonomic Bulletin & Review*, **6**, 57-86.
- DEMPSTER, A. P., LAIRD, N. M., & RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1-38.
- DODSON, C. S., PRINZMETAL, W., & SHIMAMURA, A. P. (1998). Using Excel to estimate parameters from observed data: An example from source memory data. *Behavior Research Methods, Instruments, & Computers*, **30**, 517-526.
- ERDFELDER, E. (1992). *Parameter estimation and goodness-of-fit tests for Batchelder and Riefer's storage-retrieval model: A manual for users of the SRM program* (Tech. Rep. 2). Bonn: Universität Bonn, Berichte aus dem Psychologischen Institut.
- GARCIA-PEREZ, M. A. (1994). Parameter estimation and goodness-of-fit testing in multinomial models. *British Journal of Mathematical & Statistical Psychology*, **47**, 247-282.
- HU, X. (1990). *Source: Computer program for source monitoring experiments*. Memphis: University of Memphis, Department of Psychology. (Available upon request)
- HU, X., & BATCHELDER, W. H. (1994). The statistical analysis of general processing tree models use the EM algorithm. *Psychometrika*, **59**, 21-47.
- READ, T. R. C., & CRESSIE, N. A. C. (1988). *Goodness-of-fit statistics for discrete multivariate data*. New York: Springer-Verlag.
- RIEFER, D. M., & BATCHELDER, W. H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, **95**, 318-339.
- RIEFER, D. M., & BATCHELDER, W. H. (1991). Statistical inference for multinomial processing tree models. In J.-P. Doignon & J.-C. Falmagne (Eds.), *Mathematical psychology: Current developments* (pp. 313-315). Berlin: Springer-Verlag.
- RIEFER, D. M., HU, X., & BATCHELDER, W. H. (1994). Response strategies in source monitoring. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **20**, 680-693.
- ROTHKEGEL, R. (1997). AppleTree [Computer Program]. Trier: University of Trier. Available: <http://www.psychologie.uni-trier.de:8000/projects/AppleTree.html>
- WEIR, B. S. (1990). *Genetic data analysis: Methods for discrete population genetic data*. Sunderland, MA: Sinauer Associates.

NOTE

1. Here α and β need to be integers, since the beta random numbers are generated by two gamma distributions where the parameters of the gamma distributions are required to be integers.

APPENDIX
Algorithms

Parameter Estimation

As in Hu and Batchelder (1994), categorical probabilities of MPT models are in the form of Equation 2. One component of Equation 2 is called a branch probability (Equation 3). The pa-

rameter estimation procedure obtains parameter values that minimize the distance measure, in the form of a power divergence family (Read & Cressie, 1988). If observed categorical frequencies are obtained in an experiment, the procedure to obtain the estimates of parameters (assuming that the model with the data can give unique MLEs of the parameters) is as follows, where symbol λ is the parameter in the power divergence family (Equation 8), $\Re$ stands for the set of real numbers, and $\Re^+$ for the set of positive real numbers.

Step 0: (1) Input branch probabilities in the form of Equation 3, p_{ij} , $i = 1, \dots, I_j$, $j = 1, \dots, J$. (2) Input $\lambda \in \Re$. (3) Input observed category frequencies n_j , $j = 1, \dots, J$. (4) Input a positive real number $\varepsilon \in \Re^+$.

Step 1: Input initial values of the parameters $\Theta^{(0)} = (\theta_1^{(0)}, \dots, \theta_S^{(0)}) \in (0, 1)^S$.

Step 2: Compute $M_\lambda(\Theta^{(n-1)})$, where $M_\lambda(\Theta) = [\phi_1^{(\lambda)}(\Theta), \dots, \phi_S^{(\lambda)}(\Theta)]$. If there is no parameter restriction for $\phi_s^{(\lambda)}(\Theta)$, then $\phi_s^{(\lambda)}(\Theta)$ is computed by

$$\phi_s^{(\lambda)}(\Theta) = \frac{\sum_{j=1}^J \left[\left(\frac{n_j}{np_j(\Theta)} \right)^\lambda \sum_{i=1}^{I_j} \left(\frac{n_j p_{ij}(\Theta)}{p_j(\Theta)} a_{ijs} \right) \right]}{\sum_{j=1}^J \left[\left(\frac{n_j}{np_j(\Theta)} \right)^\lambda \sum_{i=1}^{I_j} \left(\frac{n_j p_{ij}(\Theta)}{p_j(\Theta)} (a_{ijs} + b_{ijs}) \right) \right]}; \quad (9)$$

otherwise, if there is a parameter restriction, such as $\theta_{s_0} = \theta_{s_1}$, then $\phi_{s_0}^{(\lambda)}(\Theta)$ and $\phi_{s_1}^{(\lambda)}(\Theta)$ are computed by

$$\begin{aligned} \phi_{s_0}^{(\lambda)}(\Theta) &= \phi_{s_1}^{(\lambda)}(\Theta) \\ &= \frac{\sum_{j=1}^J \left[\left(\frac{n_j}{np_j(\Theta)} \right)^\lambda \sum_{i=1}^{I_j} \left(\frac{n_j p_{ij}(\Theta)}{p_j(\Theta)} (a_{ijs_0} + a_{ijs_1}) \right) \right]}{\sum_{j=1}^J \left[\left(\frac{n_j}{np_j(\Theta)} \right)^\lambda \sum_{i=1}^{I_j} \left(\frac{n_j p_{ij}(\Theta)}{p_j(\Theta)} [(a_{ijs_0} + b_{ijs_0}) + (a_{ijs_1} + b_{ijs_1})] \right) \right]}; \end{aligned} \quad (10)$$

$s = 1, \dots, S$.

Step 3: Compute

$$\Theta^{(n)} = \Theta^{(n-1)} - \varepsilon(\Theta^{(n-1)} - M_\lambda(\Theta^{(n-1)})). \quad (11)$$

If any of the parameters is fixed as a constant, such as $\theta_{s_0} = \alpha$, assign the constant to that parameter.

Step 4: Compute

$$\text{FIT}^{(n)} = \frac{2}{\lambda(\lambda+1)} \sum_{j=1}^J n_j \left[\left(\frac{n_j}{np_j(\Theta^{(n-1)})} \right)^\lambda - 1 \right].$$

Step 5: Compare $\text{FIT}^{(n)}$ with $\text{FIT}^{(n-1)}$.

— If $0 < \text{FIT}^{(n-1)} - \text{FIT}^{(n)} < 10^{-8}$, then go to **Step 6**.

— If $\text{FIT}^{(n-1)} - \text{FIT}^{(n)} < 0$, then adjust ε (to readjust the step size of the iteration. If $\lambda \geq 0$, then decrease ε ; if $\lambda < 0$, then increase ε), go to **Step 1**.

— If $10^{-8} < \text{FIT}^{(n-1)} - \text{FIT}^{(n)}$, then go to **Step 2**.

Step 6: Record $\Theta^{(n)}$ as the estimates of the parameters and $\text{FIT}^{(n)}$ as the goodness of fit of the model.