Behavior Research Methods, Instruments, & Computers
2004, 36 (1), 29-40

A Matlab function to estimate choice model parameters from paired-comparison data

FLORIAN WICKELMAIER and CHRISTIAN SCHMID
Aalborg University, Aalborg East, Denmark

Tversky (1972) has proposed a family of models for paired-comparison data that generalize the Bradley–Terry–Luce (BTL) model and can, therefore, apply to a diversity of situations in which the BTL model is doomed to fail. In this article, we present a Matlab function that makes it easy to specify any of these general models (EBA, Pretree, or BTL) and to estimate their parameters. The program eliminates the time-consuming task of constructing the likelihood function by hand for every single model. The usage of the program is illustrated by several examples. Features of the algorithm are outlined. The purpose of this article is to facilitate the use of probabilistic choice models in the analysis of data resulting from paired comparisons.

In many empirical and experimental psychological studies, researchers rely on paired-comparison data in order to measure perceived magnitudes. Often, subjects are not capable of expressing their perceptions, impressions, or attitudes by means of an exact numerical value. On the other hand, a paired comparison between alternatives is generally feasible even if the stimuli involved are hard to distinguish or the dimension measured is not easy to comprehend. Making the task easier for the subject need not result in a measurement that lacks statistical power.

The first section of this article gives an outline of a family of probabilistic choice models as formulated by Bradley and Terry (1952), Luce (1959), Tversky (1972), and Tversky and Sattath (1979). These models are capable of deriving ratio scale measures of the stimuli from only binary judgments resulting from paired comparisons. Readers who are already familiar with these models are referred to the remainder of the article, where the Matlab function *OptiPt.m* is introduced, which allows for the straightforward fitting of the models.

Probabilistic Choice Models

A widely applied method of analyzing paired-comparison data is the BTL model as formulated by Bradley and Terry (1952) and Luce (1959). These authors showed that measurement on a ratio scale level can be established if the data satisfy certain structural assumptions. The probability of choosing x from a set of alternatives A can then be represented as

$$P(x, A) = \frac{u(x)}{\sum_{y \in A} u(y)}, \tag{1}$$

where u is a ratio scale. The BTL model leads to strong testable consequences. One of them is the independence of irrelevant alternatives. This implies that the probability ratio of choosing x from the set $\{x,y\}$ versus choosing y is not affected by any other alternative—for example, $\{x,y,z\}$. Formally,

$$\frac{P(x; y)}{P(y; x)} = \frac{P(x; y, z)}{P(y; x, z)} \tag{2}$$

is called the *constant ratio rule*. The constant ratio rule, however, is not likely to hold if the set of stimuli has some natural structure, as has been suggested by several counterexamples—for example, those from Debreu (1960) or Savage (see Luce & Suppes, 1965).

Consider the situation in which a subject is asked to choose between a trip to Florida (f) and two trips to California (c and c^+). (Rumelhart & Greeno, 1971, used a similar example to illustrate the shortcomings of the BTL model.) The two trips to California are identical except for a \$10 bonus for c^+. Suppose the subject is indifferent as to whether to travel to Florida or to California. Therefore, $P(f;c) = P(c;f) = .5$. Surely, he or she would prefer the bonus trip to California to the regular one; hence, $P(c^+;c)$ is close to one. Nevertheless, choosing between the three alternatives would presumably not result in a probability close to zero or one, which violates the constant ratio rule as formulated in Equation 2:

$$\frac{\overbrace{P\left(c; c^+\right)}^{\to 0}}{\underbrace{P\left(c^+; c\right)}_{\to 1}} \neq \frac{\overbrace{P\left(c; c^+, f\right)}^{>0}}{\underbrace{P\left(c^+; c, f\right)}_{<1}}. \tag{3}$$

This article was completed while the authors were employed at the Sound Quality Research Unit at Aalborg University. This unit receives financial support from Delta Acoustics & Vibration, Brüel & Kjær, and Bang & Olufsen, as well as from the Danish National Agency for Industry and Trade (EFS) and the Danish Technical Research Council (STVF). C.S. is presently at the Department of Experimental Psychology, University of Regensburg, Germany. The authors thank Sylvain Choisel for his contribution to the code of the Matlab function *OptiPt.m*. We also thank Karin Zimmer and Wolfgang Ellermeier for numerous helpful comments on earlier drafts of the article. We are grateful for the remarks of two anonymous reviewers and of the editor, which further improved the article. Correspondence concerning this article should be addressed to F. Wickelmaier, Department of Acoustics, Aalborg University, Fredrik Bajers Vej 7 B5, 9220 Aalborg East, Denmark (e-mail: fw@acoustics.dk).

Thus, stimulus similarity seems to be a major reason to abandon the BTL model.

Therefore, Tversky (1972) introduced a family of models that can cope with subgroups consisting of similar stimuli. He called the most general strategy, when choosing between alternatives, *elimination by aspects* (EBA). According to EBA, a subject prefers one stimulus over another because of a certain attribute this stimulus has that the other one does not have. Stimuli without this attribute are eliminated from the set of possible alternatives. If all the stimuli under consideration share the preferred attribute, it will be disregarded for the current decision. Thus, another discriminating attribute has to be found, and the elimination process restarts. In the example given above, EBA would predict that the choice between a trip to California and a trip to California plus a $10 bonus would, in essence, be a choice between obtaining an extra $10 or not.

EBA is a rather general approach for modeling paired-comparison data. It can be shown that the BTL model is a special case of the EBA model. If only one unique attribute characterizes each stimulus, EBA reduces to BTL.

Another special case of EBA describes a hierarchical decision strategy leading to so-called *preference tree*, or *Pretree*, models, as proposed by Tversky and Sattath (1979). According to Pretree, the attributes of the stimuli investigated are ordered in a hierarchical manner. Hence, the elimination process arrives at the final outcome much faster than it does with EBA. Consider the decision between several dishes at a restaurant (see Tversky & Sattath, 1979, for a similar example). These dishes may have hierarchically ordered attributes. They might, for example, fall into the two main categories of meat and fish. The meat category might again be divided into subcategories of, say, beef and other meats. Thus, a subject choosing the meat attribute could eliminate all alternatives of the fish category. If he or she chose the beef attribute, he or she could eliminate all nonbeef alternatives, and so on. The final outcome exhibits all of the desired attributes.

More formally, let $T = \{x, y, z, \ldots\}$ be the total finite set of alternatives or stimuli under study, and let A denote any nonempty subset of T. Furthermore, let $x' = \{\alpha, \beta, \gamma, \ldots\}$ be the set of attributes that characterizes the alternative x. Then, according to EBA, the probability of choosing x from A is

$$P(x, A) = \frac{\sum_{\alpha \in x' \setminus A^0} u(\alpha) P(x, A_\alpha)}{\sum_{\beta \in A' \setminus A^0} u(\beta)} \tag{4}$$

(cf. Tversky, 1972, Equation 6), where A^0 is the set of attributes shared by *every* alternative in A (formally, $A^0 = \cap_{x \in A} x'$), A' is the set of attributes that belongs to *at least one* alternative in A ($A' = \cup_{x \in A} x'$), and A_α is the set of all alternatives in A sharing the attribute α ($A_\alpha = \{x \in A : \alpha \in x'\}$). If only binary choice probabilities are analyzed as they result from paired-comparison data, the general Equation 4 simplifies to

$$P(x; y) = \frac{\sum_{\alpha \in x' \setminus y'} u(\alpha)}{\sum_{\alpha \in x' \setminus y'} u(\alpha) + \sum_{\beta \in y' \setminus x'} u(\beta)} \tag{5}$$

(cf. Tversky, 1972, Equation 7), where $x' \setminus y'$ is the set of attributes characterizing alternative x, but not alternative y. Note that the EBA model distinguishes between the scale values of the stimuli and the values of their attributes (which are the model parameters): The scale values are defined as the sum of the respective parameters. In the BTL model, there is only one parameter per stimulus. It is not hard to see that Equation 1 is a special case of Equation 4 and, in the case of binary data, of Equation 5 as well. Also, the probabilities of any Pretree model for paired-comparison data can be expressed by Equation 5, but then the attributes have to be hierarchically structured.

Parameter Estimation

In order to obtain maximum likelihood estimates (MLEs) of the model parameters, the likelihood function of the model has to be specified. Since a paired-comparison matrix of n stimuli can be perfectly described by $\binom{n}{2}$ binomially distributed random variables, the likelihood function takes the shape

$$L = \prod_{i<j} \pi_{ij}^{Nij} \left(1 - \pi_{ij}\right)^{Nji}, \tag{6}$$

where i and j are the row and column indices, respectively, of the data matrix and N_{ij} is the ijth element. In the EBA model, the probabilities π_{ij} are computed by Equation 5. The MLEs, $^\circ(\alpha), ^\circ(\beta), \ldots$, are the values that maximize Equation 6. Most often, analytical solutions for maximizing the likelihood do not exist. Therefore, the MLEs have to be found by numerical optimization, using iterative methods.

The u parameters define a ratio scale on the set of attributes. Thus, one of them can be set to an arbitrary unit. Consequently the number of *free* parameters of an EBA model is always one less than the number of parameters in the likelihood function. This parameter surplus is not crucial when the MLEs are determined by numerical optimization. When statistical tests such as those presented below are employed, however, the number of free parameters has to be considered.

For the BTL model, Bradley (1955) has described how to estimate confidence intervals for the MLEs, but his approach can easily be generalized to apply to the whole EBA family. The Hessian matrix of the log-likelihood function is defined as the square matrix of second partial derivatives with respect to the model parameters:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \log L}{\partial u_1^2} & \cdots & \frac{\partial^2 \log L}{\partial u_1 \partial u_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \log L}{\partial u_k \partial u_1} & \cdots & \frac{\partial^2 \log L}{\partial u_k^2} \end{bmatrix}, \tag{7}$$

where k is the number of parameters. Plugging the vector of MLEs $\circ$ into Equation 7 allows one to construct the matrix $\mathbf{C}$, which is the inverse of the negative Hessian augmented by, respectively, a column and a row vector of ones and a zero in the bottom right corner:

$$\mathbf{C} = \begin{bmatrix} -\mathbf{H} & \mathbf{1} \\ \mathbf{1}' & 0 \end{bmatrix}^{-1}. \quad (8)$$

The first k rows and columns of $\mathbf{C}$ form the estimated covariance matrix of $\circ$. The variances, and thus the standard errors, of the MLEs can be estimated from the main diagonal of the covariance matrix. The 95% confidence intervals are obtained by

$$\circ \pm 1.96\sqrt{\mathbf{Diag}\left[\widehat{\mathbf{cov}}(\circ)\right]}. \quad (9)$$

Goodness of Fit

To check the goodness of fit of EBA models (including Pretree and BTL), it is convenient to compare the likelihood of the model with the saturated model that fits the data perfectly (cf. Wickens, 1982, chap. 6). In the saturated model, the probabilities π_{ij} are estimated by the relative frequencies, $\hat{\pi}_{ij} = N_{ij}/(N_{ij} + N_{ji})$. Therefore, the saturated model has $\binom{n}{2}$ free parameters. In the simplest case of the EBA model (i.e., BTL), the number of free parameters reduces to $n - 1$. Every additional parameter—for example, for a branch in a Pretree—has to be added, so in general the EBA model has $n - 1 + c$ free parameters. Note that the maximum number of Pretree parameters is $2n - 2$, whereas the maximum number of EBA parameters, $2^n - 2$, has to be reduced to a maximum of $\binom{n}{2}$ parameters if only binary data resulting from paired comparisons are available. For the statistical decision on whether or not the EBA model can account for the data, the likelihood ratio is computed. The expression

$$\chi^2 = -2\log\left[\frac{L_{\mathrm{EBA}}}{L_{\mathrm{SAT}}}\right] \quad (10)$$

is approximately χ^2-distributed, with $\binom{n}{2} - (n - 1 + c)$ degrees of freedom. The EBA model is rejected if the p value is less than 10% (rather than the conventional 5%, since one would like to increase the chance of detecting violations of the model).

A test statistic alternative to the one in Equation 10 is the common χ^2 test for goodness of fit:

$$\tilde{\chi}^2 = \sum_{i,j} \frac{\left(N_{ij} - \hat{\pi}_{ij}N\right)^2}{\hat{\pi}_{ij}N}, \quad (11)$$

where $N = N_{ij} + N_{ji}$. It accounts for the lack of fit between observed and predicted values. Equation 10 and Equation 11 are asymptotically equivalent—that is, $\chi^2 = \tilde{\chi}^2$ as $N \to \infty$.

Comparison of Models

In many applications, it is desirable to compare two EBA models directly. If the parameter space Ω' of one model is a proper subset of the other model's parameter space, Ω, the two models are nested. The restricted model EBA′ and the unrestricted model EBA can be tested against each other by using the likelihood ratio test of Equation 10:

$$\chi^2 = -2\log\left[\frac{L_{\mathrm{EBA}'}}{L_{\mathrm{EBA}}}\right] = 2\left[\log L_{\mathrm{EBA}} - \log L_{\mathrm{EBA}'}\right]. \quad (12)$$

The test statistic is approximately χ^2-distributed, having as many degrees of freedom as the difference between the numbers of parameters in EBA and in EBA′. The restricted model can be rejected if the likelihood ratio test is significant.

It is not in every case that two different EBA models are nested. The likelihood ratio test, however, is not appropriate if the two models are not nested. If this is the case, it is common practice to employ so-called information criteria as a tool for model selection. Akaike (1977) has provided a penalty function that takes into account both the likelihood of the model and the number of free parameters. For the EBA model, Akaike's information criterion (AIC) is defined as

$$\mathrm{AIC} = -2\log L_{\mathrm{EBA}} + 2(n - 1 + c). \quad (13)$$

When two models are compared using the AIC, the model with the smaller AIC should be selected.

The Function OptiPt.m

Pretree and EBA models can explain a great diversity of data sets for which the BTL model has to be rejected. Nevertheless, the BTL model is very popular among researchers, whereas the more general EBA and Pretree models are only rarely used. One of the main reasons appears to be that several computer programs are readily available for the estimation of BTL parameters, but not for EBA in general. Moreover, specifying the likelihood function of an EBA or Pretree model to a computer program is usually very cumbersome, and it takes considerable effort to write an estimation routine in a traditional programming language such as C, to handle more than one Pretree or EBA model.

The Matlab function OptiPt.m was written to overcome this unsatisfactory situation, since it draws heavily on the high-level computational commands lacking in many traditional programming languages. As we will show in the remainder of this article, it is capable of handling the whole EBA family (including the Pretree and BTL models) and is very easy to use and extremely flexible. Matlab is a popular mathematics and visualization software widely used by researchers in different fields to analyze data. OptiPt is designed for Matlab Version 6.0. It has been tested using Solaris and Windows versions of the Matlab software. No additional Matlab toolboxes are required to run the function. The source code of the function is given in the Appendix A. It may also be downloaded from www.acoustics.dk/~fw. Note: OptiPt is also available as an R function from www.acoustics.dk/~fw. R is a free statistical software package (see www.r-project.org).

Using OptiPt.m

To use the function OptiPt.m, copy it to your Matlab directory or to a directory in your Matlab path. For information on the usage of the function, type

>> help OptiPt,

and a short help message is displayed. (Note that >> is the Matlab prompt.) The complete syntax of the function is

>> [p,chistat,u,lL_eba,lL_sat,fit,cova] = OptiPt(M,A,s).

You call the function by its name, *OptiPt*. It expects two mandatory arguments *M* and *A*, and the optional argument *s*. *M* is a paired-comparison matrix, which is square, consisting of absolute frequencies and a main diagonal of zeros. The off-diagonal values should be greater than zero. *A* is a so-called cell array, an array that may consist of rows of different lengths. In *A*, the model will have to be specified, as we will show in the Examples section. Each row in *A* is a vector of all the parameters that belong to one alternative or stimulus. Thus, *A* has as many rows as there are alternatives or stimuli. The vector *s* carries the starting values for the estimation routine. If not specified, the search algorithm starts at $1/k$ for each parameter value, where k is the number of parameters.

The return values (*p*, *chistat*, *u*, *lL_eba*, *lL_sat*, *fit*, and *cova*) are optional. The parameter estimates are stored in the *p* vector. The *chistat* vector reports the χ^2 statistic according to Equation 10 as a measure of the goodness of fit of the model specified in *A*. In *chistat*, you also find the number of degrees of freedom needed for a statistical test. The *u* vector stores the scale values. Each scale value is the sum of the parameters that belong to one alternative or stimulus. Note that *p* and *u* are identical only for the BTL model, since there is only one unique parameter per stimulus. The log-likelihoods of the model specified in *A* and of the saturated model are stored in the *lL_eba* and *lL_sat* variables, respectively. The optimization algorithm searches for parameter values that maximize *lL_eba*. The saturated model, on the other hand, fits the data perfectly. Therefore, *lL_sat* is always greater than *lL_eba* for any model having fewer parameters. The *fit* matrix contains the predicted values of the fitted model. It can be used to calculate the goodness of fit according to Equation 11 and the residuals, in order to check for local misfits. The covariance matrix of the estimated parameters *p* is stored in the *cova* matrix. The diagonal of *cova* displays the variances of *p*. Take the square root of the variances in order to obtain the standard errors.

OptiPt automatically constructs the log-likelihood function of the model specified in *A* by first computing the set difference $x' \backslash y'$ for all pairs of stimuli. Then each factor of the likelihood function is calculated according to Equation 5. As a third step, rather than multiplying these factors as described in Equation 6, OptiPt takes the logarithm of each of them and sums them up, in order to compute the log-likelihood of the specified model. The optimization algorithm performs a direct search for the maximum likelihood estimates by calling the log-likelihood function with different parameter values. Thus, the maximum of the log-likelihood function is approached by numerical optimization without taking derivatives. OptiPt returns the parameter values that maximize the log-likelihood. In order to prevent the optimization algorithm from finding negative estimates, a constraint was built into the log-likelihood function: Whenever this function is called with at least one parameter less than or equal to zero, it returns minus infinity. The optimized parameters are then plugged into Equation 5 to yield the fitted paired-comparison matrix. Finally, a numerical Hessian matrix is computed and, according to Equation 8, augmented and inverted in order to estimate the covariance matrix of the parameters.

Examples

The following gives an extensive example of the usage of the Matlab function OptiPt.m. It is based on the real-world data set reported by Rumelhart and Greeno (1971). Their stimuli consisted of nine celebrities, including three politicians (Lyndon Baines Johnson, Harold Wilson, and Charles de Gaulle), three athletes (Johnny Unitas, Carl Yastrzemski, and A. J. Foyt), and three movie stars (Brigitte Bardot, Elizabeth Taylor, and Sophia Loren). They presented all 36 pairs of stimuli to 234 subjects, asking them with whom they would rather spend an hour of conversation. The results are summarized in Table 1. Suppose you have saved these results in a tab-delimited text file named *matrix.txt*. You can easily read this file into the Matlab matrix *M* by typing

>> M = textread('matrix.txt','','delimiter','\t').

We will first try to fit a BTL model to the data. Therefore, we are going to specify the model by means of the cell array *A* as follows:

>> A = {[1];[2];[3];[4];[5];[6];[7];[8];[9]}.

Note that every row in *A* corresponds to one stimulus. Note further that there is only one entry in each row, since every stimulus has only one unique parameter. To estimate the BTL parameters and test the goodness of fit, enter

>> [p,chistat] = OptiPt(M,A).

This will start the estimation routine. During the estimation process, you will receive feedback about the number of iterations and function calls needed to maximize the log-likelihood function. The message "optimization terminated successfully" indicates that at least a local extremum has been reached before the algorithm stopped searching. The parameter estimates are now stored in *p*, but beware of interpreting them without looking at the fit of the model.

You are going to find a large value of the test statistic [$\chi^2(28) = 78.22$, $p < .001$], indicating that the BTL model cannot account for the data. This is not surprising, since the three politicians, for example, are more similar

Table 1
Aggregate Choice Frequencies Reported by Rumelhart and Greeno (1971)

	L.B.J.	H.W.	C.d.G.	J.U.	C.Y.	A.J.F.	B.B.	E.T.	S.L.
L.B.J.	0	159	163	175	183	179	173	160	142
H.W.	75	0	138	164	172	160	156	122	122
C.d.G.	71	96	0	145	157	138	140	122	120
J.U.	59	70	89	0	176	115	124	86	61
C.Y.	51	62	77	58	0	77	95	72	61
A.J.F.	55	74	96	119	157	0	134	92	71
B.B.	61	78	94	110	139	100	0	67	48
E.T.	74	112	112	148	162	142	167	0	87
S.L.	92	112	114	173	173	163	186	147	0

Note—Row stimuli are chosen over column stimuli.

to each other than to any other celebrity. Typically, the BTL model does not hold for data sets in which subgroups of stimuli are formed on the basis of similarity. Note, however, that sufficient statistical power (provided in the present example by the large sample size of $N = 234$) is required to be able to reject any given model. A small number of subjects of, say, 20 considerably reduces the chance for any specified model to be rejected. In such a case, however, OptiPt gives no warning messages; it assumes that the sample size is large enough.

As a second example, we are going to fit a Pretree to the data. The list of stimuli naturally suggests a tree structure with three branches corresponding to the three different occupations of the nine celebrities: (L.B.J., H.W., C.d.G.) (J.U., C.Y., A.J.F.) (B.B., E.T., S.L.). Tversky and Sattath (1979) investigated this kind of Pretree, depicted schematically in Figure 1.

To specify the Pretree model, we simply expand the cell array *A* by the three branch parameters 10, 11, and 12:

```
>> B = {[1 10];[2 10];[3 10];[4 11];[5 11];[6 11];
       [7 12];[8 12];[9 12]}.
```

Now, every stimulus is characterized by two parameters, its individual parameter and its branch parameter. Hence, every row of *B* consists of two elements. In order to estimate the model parameters and check the goodness of fit, type

```
>> [p,chistat,u,lL_eba,lL_sat,fit,cova] =
                                   OptiPt(M,B)
```

to start the estimation procedure. The complete output resulting from this command is listed in Appendix B. Again, the message "optimization terminated successfully" should be displayed. Looking at the model fit, you will find that the tree model can account for the data quite well [$\chi^2(25) = 30.17$, $p = .22$]; the alternative goodness-of-fit statistic according to Equation 11 amounts to $\tilde{\chi}^2(25) = 30.05$. The model parameters are stored in the vector *p*. They differ from the scale values in *u*, since each scale value is the sum of the individual and branch parameters characterizing a given stimulus. You can arbitrarily choose any stimulus to define the unit length, because *u* is a ratio scale. The standardized *u*-scale with the first stimulus defining the unit is obtained by

```
>> u/u(1),
```

but of course, you can choose any other stimulus by changing the index of the denominator. The standard errors of the parameter estimates *p* can be extracted from the covariance matrix by typing

```
>> se = sqrt(diag(cova)).
```

To evaluate the program, it is desirable to compare the output of OptiPt with the results obtained by Tversky and Sattath (1979). Unfortunately, these authors reported neither the parameter values nor the scale values estimated by their fitting a Pretree model. Rather, they gave a graphical representation of these values by depicting the estimated Pretree. In their Figure 7 (p. 555), the lengths of the branches are proportional to the parameter values. (Note that this does not hold for the merely schematical Pretree in our present Figure 1, in which the lengths of the branches reveal no information about the parameter values.) In order to extract the parameter values from Tversky and Sattath's Figure 7, we measured the lengths of the branches of the tree with a ruler. Table 2 shows the results of this "measurement" in its third column. The parameter values are standardized, using the first value as the unit length. Obviously, measuring parameters with a ruler might introduce some error. Nevertheless, the standardized parameter estimates of OptiPt are quite close to the ruler-measured values, as may be seen in the forth column of Table 2. Thus, it seems appropriate to conclude that the results of Tversky and Sattath were replicated by

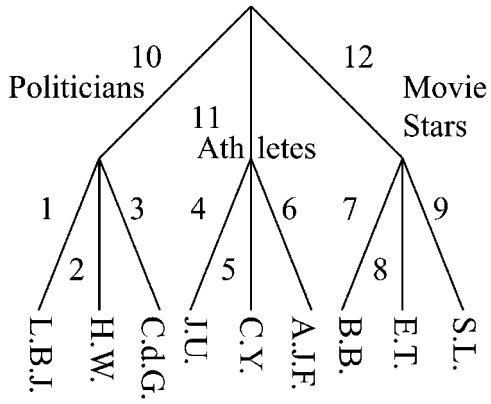

Figure 1. Schematical preference tree for choice among celebrities (Tversky & Sattath, 1979).

reanalyzing the data of Rumelhart and Greeno (1971) with the Matlab function OptiPt.m.

Finally, you may want to compare the BTL model and the Pretree model directly. Since the BTL model is nested into the Pretree, a likelihood ratio test can be employed. According to Equation 12, the test statistic is computed by

$$\chi^2 = -2\log\left[\frac{L_{\mathrm{BTL}}}{L_{\mathrm{PT}}}\right] = 2\left[\log L_{\mathrm{PT}} - \log L_{\mathrm{BTL}}\right], \quad (14)$$

where L_{BTL} and L_{PT} are the likelihoods of the BTL model and the Pretree, respectively. This is easily achieved by Matlab and OptiPt. Just enter

```
>> [p,chistat,u,btl] = OptiPt(M,A);
>> [p,chistat,u,pt] = OptiPt(M,B);
>> 2*(pt-btl)
```

to receive the test statistic. Subtract the number of free BTL parameters (9 − 1) from the Pretree parameters (9 − 1 + 3) to obtain the degrees of freedom for the test. When the likelihood ratio test is applied to the present data, the BTL model can be rejected in favor of the Pretree [$\chi^2(3) = 48.05$, $p < .001$]. The AIC, as formulated in Equation 13, may alternatively be used for the purpose of model comparison:

```
>> aic_btl = -2*btl + 2*(9-1)
>> aic_pt = -2*pt + 2*(9-1 + 3).
```

The two AICs amount to $\mathrm{AIC}_{\mathrm{BTL}} = 10{,}716.4$ and $\mathrm{AIC}_{\mathrm{PT}} = 10{,}673.5$, respectively, and therefore clearly argue for the Pretree.

Avoiding Local Extrema

Generally, optimization algorithms guarantee only finding local extrema (i.e., locally optimal minima or maxima). The optimization algorithm used by OptiPt is the Matlab function *fminsearch*, which is part of the Matlab standard distribution. It employs the Nelder–Mead simplex method (Nelder & Mead, 1965), which is known to have rather good convergence properties if the parameter space is of low dimensionality (Lagarias, Reeds, Wright, & Wright, 1998, investigated the convergence in one and two dimensions). With increasing number of parameters, however, fminsearch is likely to report just a local solution. The OptiPt.m function provides one with three kinds of diagnostic information that could help you to avoid getting stuck in a local extremum.

Table 2
Comparison of the Results of Tversky and Sattath (1979) and OptiPt

Parameter		Estimate		
Label	Number	Tversky and Sattath	OptiPt	*SE*
L.B.J.	1	1.00	1.0000	0.1116
H.W.	2	0.57	0.5416	0.0879
C.d.G.	3	0.40	0.3927	0.0735
J.U.	4	0.19	0.1803	0.0431
C.Y.	5	0.09	0.0729	0.0209
A.J.F.	6	0.19	0.1795	0.0454
B.B.	7	0.17	0.1641	0.0292
E.T.	8	0.43	0.4165	0.0538
S.L.	9	0.66	0.6401	0.0685
Politicians	10	0.31	0.3205	0.1300
Athletes	11	0.26	0.2450	0.0431
Movie stars	12	0.26	0.2549	0.0526

Note—The values in the third column were measured from Tversky and Sattath's Figure 7 (p. 555), using a ruler. The parameter estimates are standardized. The *SE* column shows the standard errors.

First, always make sure that the optimization algorithm stops with the message "optimization terminated successfully." Any other message, such as "maximum number of iterations has been exceeded," indicates that the algorithm stopped before reaching the supremum. In this case, the parameter values should not be interpreted. Second, EBA, Pretree, and BTL models are nested models if the parameter space of one model is a proper subset of the other model's parameter space. Thus, there is a natural order for the likelihood values of the models, determined by the number of dimensions of their parameter space. Note that the BTL model must have the lowest likelihood, since it has the lowest number of parameters. Every additional parameter must increase the likelihood. Furthermore, none of the models can have a likelihood that exceeds the likelihood of the saturated model. Any violation of this order indicates a local minimum. To check whether the likelihoods of your models are in the right order, make extensive use of the optional return values *lL_eba* and *lL_sat* of the OptiPt.m function. Since the log-function simply applies a monotonic transformation to the likelihoods, the log-likelihood of a Pretree model, for example, must always lie between the log-likelihoods of the BTL and the saturated model of the same data set. The third type of diagnostic information provided by OptiPt is the covariance matrix of the model parameters, contained in the optional return value *cova*. The main diagonal of the covariance matrix displays the variances of the MLEs. Enter

```
>> diag(cova)
```

to extract the variances from the *cova* matrix in Matlab. Any negative value in the main diagonal hints at an only locally optimal solution.

The general answer to the question of how to avoid local extrema is to optimize the starting values for the search algorithm. To achieve this with OptiPt, the optional function parameter *s* can be passed as an argument. Whenever one encounters a suspicious termination message or one finds the log-likelihoods not to be in the right order, the following method has proven successful: Run the program once, and take the results as starting values for the next run. This is easily done in Matlab by entering

```
>> [p,chistat] = OptiPt(M,A)
>> s = p;
>> [p,chistat] = OptiPt(M,A,s).
```

Note that the starting values have to be greater than zero. Hence, one may have to change elements of s if necessary. When a large number of stimuli and parameters are dealt with, it could also happen that this process of reestimating with optimized starting values has to be iterated several times until the log-likelihood of the specified model does not change any longer.

A second strategy to avoid local minima is to call OptiPt with randomly generated starting values. If this procedure is repeated, say, 10 times and the parameter estimates are the same, there is less doubt about a possible local extremum. The best approach to the problem is presumably to combine both methods: Run the function with different starting values, and plug in the estimates again.

Testing the Performance of OptiPt

The following section will report on a simulation study conducted to test the precision of the estimation algorithm. The general procedure thereby was as follows. First, the model structure needed to be specified. Second, the model parameters were chosen to have some arbitrary but fixed value. Third, a paired-comparison matrix was simulated by plugging the fixed parameters into the model's equations. Finally, the parameters were reestimated from the simulated paired-comparison matrix. The difference between the true parameter values and the estimates is an indicator of the precision of the estimation routine.

OptiPt calls the built-in Matlab fminsearch function to execute the search for the best parameters. Fminsearch evaluates the function to be minimized (in this case, the negative log-likelihood function) and tries different parameter configurations, starting from the initial values in the starting vector, in order to achieve a minimum function value. The search is successfully terminated if the return value of the minimized function and the optimized parameter values do not change by more than 0.0001 in two successive function calls. The stopping criteria are the default values in Matlab; they cannot be passed to OptiPt as optional arguments. Advanced users, however, can change the stopping rules, if they find it necessary, by editing the source code of OptiPt and by passing additional options to fminsearch by means of the *optimset* command. (Consult the Matlab help for detailed information on fminsearch and optimset.)

In the remainder of this section, we will report the method and the results of a simulation study, in which we tried to reestimate the parameters of three nested models in order to check the quality of the search algorithm. To illustrate the findings we encountered in our simulation studies, we will show a typical example of the results.

Consider a set of five stimuli $T = \{a,b,c,d,e\}$. Since we are dealing with paired-comparison data, we have $\binom{5}{2} = 10$ independent data points. Hence, the total amount of $2^5 - 2 = 30$ parameters of a saturated EBA model (the number of proper nonempty subsets of T) has to be reduced to 10 parameters, in order for the model to be identifiable. We specified the structure of a 10-parameter EBA model, as is depicted in Figure 2; its parameters were randomly chosen from a uniform distribution ranging from 0 to 10. As a result, the parameters were set to

$$\begin{aligned} p_1 &= 1.1228 & p_6 &= 3.1357 \\ p_2 &= 2.8673 & p_7 &= 3.5723 \\ p_3 &= 9.6698 & p_8 &= 3.1550 \\ p_4 &= 2.3594 & p_9 &= 6.2415 \\ p_5 &= 3.3741 & p_{10} &= 6.0702. \end{aligned} \tag{15}$$

We inserted these values into Equation 5 to compute the predicted paired-comparison matrix, multiplying the relative frequencies by $N = 1{,}000$. In Matlab, the 10-parametric EBA model is specified by

```
>> EBA = {[1 6 7 9];[2 6 7 10];[3 7 9 10];
          [4 8];[5 8]}.
```

When calling OptiPt with the predicted matrix and the cell array EBA as arguments, we obtained the parameter estimates. Since there is no error in the data, however, we iterated the estimation procedure as described in the previous section until the χ^2 value was close to zero be-

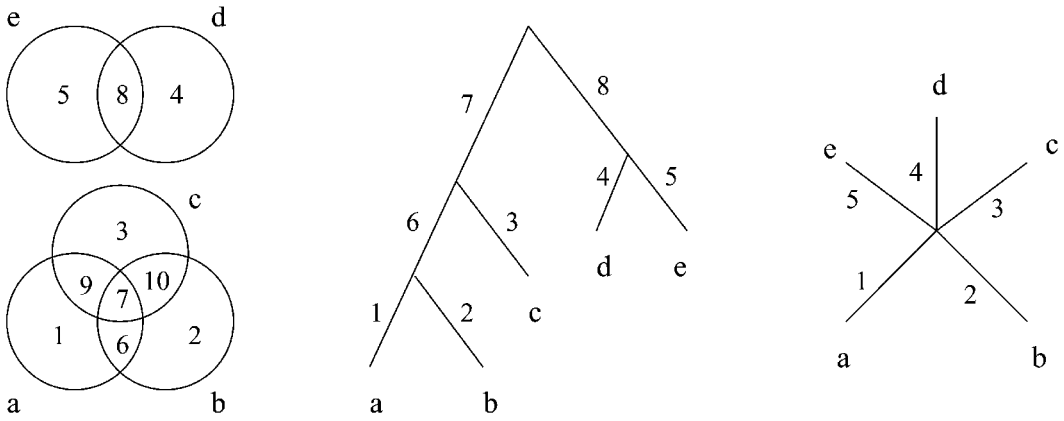

Figure 2. Three nested models: EBA, Pretree, and BTL models (from left to right).

fore we took the parameter estimates for granted. After a few iterations the estimates were

$$p = 0.0508\ 0.1492\ 0.5330\ 0.1331\ 0.1904\ 0.1895\ 0.1890\ 0.1780\ 0.3647\ 0.3551.$$

These estimates are unique up to multiplication by a positive constant, since the parameters are ratio scaled. To find the best constant c, we stored the true values of Equation 15 in the Matlab vector *true* and entered

```
>> c = mean(true./p).
```

Multiplying p by $c = 18.2298$ leads to

$$p = 0.9257\ 2.7201\ 9.7172\ 2.4269\ 3.4706\ 3.4545\ 3.4453\ 3.2451\ 6.6490\ 6.4727$$

as a good approximation of the true values in Equation 15, the maximal estimation error being 17.55%.

Setting p_9 and p_{10} to zero reduces the EBA model to the nested Pretree depicted in Figure 2. Again, we computed the predicted data matrix and passed it on to OptiPt, specifying the model by

```
>> PT = {[1 6 7];[2 6 7];[3 7];[4 8];[5 8]}.
```

Only one iteration was needed to reach a χ^2 value of 0.00. Multiplying the parameter estimates by $c = 15.7150$ resulted in

$$p = 1.1229\ 2.8674\ 9.6691\ 2.3598\ 3.3748\ 3.1353\ 3.5722\ 3.1542,$$

which is very close to the true values in Equation 15 (maximal estimation error: 0.25%). Finally, we turned the Pretree into a BTL model by setting the branch parameters p_6, p_7, and p_8 to zero. We computed the paired-comparison data and specified the model by

```
>> BTL = {[1];[2];[3];[4];[5]}.
```

Again, OptiPt provided the parameters after the first call. After multiplication by $c = 15.4007$, the parameter estimates were almost perfect copies of the true values (maximal estimation error: 0.07%):

$$p = 1.1228\ 2.8675\ 9.6692\ 2.3594\ 3.3743.$$

Two conclusions can be drawn from this simulation study. First, the precision of the estimation algorithm proved to be very satisfactory. Second, models with fewer than $\binom{n}{2}$ free parameters may be estimated from paired-comparison data, but the more parameters there are, the harder it becomes for the search algorithm to find the global extremum in a single run. As the number of the parameters increases, the precision of the search algorithm decreases slightly because of the enlarged search space. The results are summarized in Table 3.

Table 3
Exemplary Results of the Simulation Study

Parameter	True Value	EBA	Pretree	BTL
p_1	1.1228	0.9257	1.1229	1.1228
p_2	2.8673	2.7201	2.8674	2.8675
p_3	9.6698	9.7172	9.6691	9.6692
p_4	2.3594	2.4269	2.3598	2.3594
p_5	3.3741	3.4706	3.3748	3.3743
p_6	3.1357	3.4545	3.1353	–
p_7	3.5723	3.4453	3.5722	–
p_8	3.1550	3.2451	3.1542	–
p_9	6.2415	6.6490	–	–
p_{10}	6.0702	6.4727	–	–

Note—The estimates are close to the true values. The accuracy increases as the number of parameters decreases.

Conclusions

In many empirical studies, paired-comparison data are collected in order to measure subjective magnitudes on scales resulting from the BTL model. Stimulus similarity, however, often causes the BTL model to be rejected. Therefore, Tversky (1972) proposed a family of models, the EBA models, that can handle stimulus subgrouping due to similarity. Use of these models in applied research, however, has been restricted by a lack of adequate software permitting flexible model specification and stringent testing.

This article introduced the new Matlab function OptiPt.m for the estimation of EBA, Pretree, and BTL model parameters. Its usage was illustrated by a classical example from the literature. Detailed instructions were given on how to apply the function effectively. The precision of the estimation algorithm has been shown to be very satisfactory. It is the authors' hope that this article will encourage other researchers to rediscover Tversky's EBA models and to use them as widely as the BTL model.

REFERENCES

AKAIKE, H. (1977). On entropy maximization principle. In P. R. Krishnaiah (Ed.), *Applications of statistics* (pp. 27-41). Amsterdam: North-Holland.

BRADLEY, R. A. (1955). Rank analysis of incomplete block designs: III. Some large-sample results on estimation and power for a method of paired comparisons. *Biometrika*, **42**, 450-470.

BRADLEY, R. A., & TERRY, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, **39**, 324-345.

DEBREU, G. (1960). Review of R. D. Luce's Individual choice behavior: A theoretical analysis. *American Economic Review*, **50**, 186-188.

LAGARIAS, J. C., REEDS, J. A., WRIGHT, M. H., & WRIGHT, P. E. (1998). Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on Optimization*, **9**, 112-147.

LUCE, R. D. (1959). *Individual choice behavior: A theoretical analysis.* New York: Wiley.

LUCE, R. D., & SUPPES, P. (1965). Preference, utility, and subjective probability. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology* (Vol. 3, pp. 249-410). New York: Wiley.

NELDER, J. A., & MEAD, R. (1965). A simplex method for function minimization. *Computer Journal*, **7**, 308-313.

RUMELHART, D. L., & GREENO, J. G. (1971). Similarity between stimuli: An experimental test of the Luce and Restle choice models. *Journal of Mathematical Psychology*, **8**, 370-381.

TVERSKY, A. (1972). Elimination by aspects: A theory of choice. *Psychological Review*, **79**, 281-299.

TVERSKY, A., & SATTATH, S. (1979). Preference trees. *Psychological Review*, **86**, 542-573.

WICKENS, T. D. (1982). *Models for behavior: Stochastic processes in psychology.* San Francisco: Freeman.

APPENDIX A
Source Code of OptiPt.m

```
function [p,chistat,u,lL_eba,lL_sat,fit,cova] = OptiPt(M,A,s)
% OptiPt parameter estimation for BTL/Pretree/EBA models
%   p = OptiPt(M,A) estimates the parameters of a model specified
%   in A for the paired-comparison matrix M. M is a matrix with
%   absolute frequencies. A is a cell array.
%
%   [p,chistat,u] = OptiPt(M,A) estimates parameters and reports
%   the chi2 statistic as a measure of goodness of fit. The vector
%   of scale values is stored in u.
%
%   [p,chistat,u,lL_eba,lL_sat,fit,cova] = OptiPt(M,A,s) estimates
%   parameters, checks the goodness of fit, computes the scale values,
%   reports the log-likelihoods of the model specified in A and of the
%   saturated model, returns the fitted values and the covariance
%   matrix of the parameter estimates. If defined, s is the starting
%   vector for the estimation procedure. Otherwise each starting value
%   is set to 1/length(p).
%   The minimization algorithm used is FMINSEARCH.
%
%   Examples
%     Given the matrix M =
%                          0    36    35    44    25
%                         19     0    31    37    20
%                         20    24     0    46    24
%                         11    18     9     0    13
%                         30    35    31    42     0
%
%     A BTL model is specified by A = {[1];[2];[3];[4];[5]}
%     Parameter estimates and the chi2 statistic are obtained by
%       [p,chistat] = OptiPt(M,A)
%
%     A Pretree model is specified by A = {[1 6];[2 6];[3 7];[4 7];[5]}
%     A starting vector is defined by s = [2 2 3 4 4 .5 .5]
%     Parameter estimates, the chi2 statistic, the scale values, the
%     log-likelihoods of the Pretree model and of the saturated model,
%     the fitted values, and the covariance matrix are obtained by
%       [p,chistat,u,lL_eba,lL_sat,fit,cova] = OptiPt(M,A,s)
%
% Authors: Florian Wickelmaier (wickelmaier@web.de) and Sylvain Choisel
% Last mod: 03/JUL/2003

I = length(M);  % number of stimuli
mmm = 0;
for i = 1:I
  mmm = [mmm max(A{i})];
end
J = max(mmm);  % number of pt parameters
if(nargin == 2)
  p = ones(1,J)*(1/J);  % starting values
elseif(nargin == 3)
  p = s;
end

for i = 1:I
  for j = 1:I
    diff{i,j} = setdiff(A{i},A{j});  % set difference
  end
end

p = fminsearch(@ebalik,p,optimset(`Display','iter','MaxFunEvals',10000,...
    `MaxIter',10000),M,diff,I);  % optimized parameters
lL_eba = -ebalik(p,M,diff,I);  % likelihood of the specified model

lL_sat = 0;  % likelihood of the saturated model
for i = 1:I-1
  for j = i+1:I
    lL_sat = lL_sat + M(i,j)*log(M(i,j)/(M(i,j)+M(j,i)))...
                    + M(j,i)*log(M(j,i)/(M(i,j)+M(j,i)));
  end
end
```

APPENDIX A (Continued)

```
fit = zeros(I);  % fitted PCM
for i = 1:I-1
  for j = i+1:I
    fit(i,j) = (M(i,j)+M(j,i))/(1+sum(p(diff{j,i}))/sum(p(diff{i,j})));
    fit(j,i) = (M(i,j)+M(j,i))/(1+sum(p(diff{i,j}))/sum(p(diff{j,i})));
  end
end

chi = 2*(lL_sat-lL_eba);
df =  I*(I-1)/2 - (J-1);
chistat = [chi df];  % 1-chi2cdf(chi,df)];  % goodness-of-fit statistic

u = sum(p(A{1}));  % scale values
for i = 2:I
  u = [u sum(p(A{i}))];
end

H = hessian(`ebalik',p',M,diff,I);
C = inv([H ones(J,1); ones(1,J) 0]);
cova = C(1:J,1:J);

function lL_eba = ebalik(p,M,diff,I)  % computes the likelihood

if min(p)<=0  % bound search space
  lL_eba = inf;
  return
end

thesum = 0;
for i = 1:I-1
  for j = i+1:I
    thesum = thesum + M(i,j)*log(1+sum(p(diff{j,i}))/sum(p(diff{i,j})))...
                    + M(j,i)*log(1+sum(p(diff{i,j}))/sum(p(diff{j,i})));
  end
end
lL_eba = thesum;

function H = hessian(f,x,varargin)  % computes numerical Hessian

k = size(x,1);
fx = feval(f,x,varargin{:});
h = eps.^(1/3)*max(abs(x),1e-2);
xh = x+h;
h = xh-x;
ee = sparse(1:k,1:k,h,k,k);

g = zeros(k,1);
for i = 1:k
  g(i) = feval(f,x+ee(:,i),varargin{:});
end

H = h*h';
for i = 1:k
  for j = i:k
    H(i,j) = (feval(f,x+ee(:,i)+ee(:,j),varargin{:})-g(i)-g(j)+fx)...
                / H(i,j);
    H(j,i) = H(i,j);
  end
end
```

APPENDIX B
Sample Output of OptiPt.m (See the Examples Section for Details)

```
Optimization terminated successfully:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04


p =

  Columns 1 through 7

    0.2860    0.1549    0.1123    0.0516    0.0209    0.0513    0.0469

  Columns 8 through 12

    0.1191    0.1831    0.0917    0.0701    0.0729


chistat =

   30.1663   25.0000


u =

  Columns 1 through 7

    0.3777    0.2466    0.2040    0.1217    0.0909    0.1214    0.1198

  Columns 8 through 9

    0.1920    0.2560


lL_eba =

  -5.3257e+03


lL_sat =

  -5.3106e+03


fit =

  Columns 1 through 7

         0  151.7911  168.0190  176.9900  188.5925  177.0748  177.6394
   82.2089         0  135.6456  156.6913  170.9534  156.7932  157.4720
   65.9810   98.3544         0  146.5816  161.8499  146.6893  147.4079
   57.0100   77.3087   87.4184         0  166.6236  117.2719  117.8844
   45.4075   63.0466   72.1501   67.3764         0   67.5996  100.9599
   56.9252   77.2068   87.3107  116.7281  166.4004         0  117.7693
   56.3606   76.5280   86.5921  116.1156  133.0401  116.2307         0
   78.8693  102.4461  113.4627  143.2449  158.7971  143.3542  167.8648
   94.5307  119.1920  130.2268  158.6188  172.6645  158.7194  186.2584

  Columns 8 through 9

  155.1307  139.4693
  131.5539  114.8080
  120.5373  103.7732
   90.7551   75.3812
   75.2029   61.3355
   90.6458   75.2806
   66.1352   47.7416
         0   92.2326
  141.7674         0
```

APPENDIX B (Continued)

```
cova =

  Columns 1 through 7

    0.0010    0.0007    0.0006   -0.0001   -0.0001   -0.0001   -0.0001
    0.0007    0.0006    0.0005   -0.0001   -0.0000   -0.0001   -0.0001
    0.0006    0.0005    0.0004   -0.0001   -0.0000   -0.0001   -0.0001
   -0.0001   -0.0001   -0.0001    0.0002    0.0001    0.0001   -0.0000
   -0.0001   -0.0000   -0.0000    0.0001    0.0000    0.0001   -0.0000
   -0.0001   -0.0001   -0.0001    0.0001    0.0001    0.0002   -0.0000
   -0.0001   -0.0001   -0.0001   -0.0000   -0.0000   -0.0000    0.0001
   -0.0003   -0.0002   -0.0002   -0.0000   -0.0000   -0.0000    0.0001
   -0.0004   -0.0003   -0.0002   -0.0000   -0.0000   -0.0000    0.0001
   -0.0010   -0.0008   -0.0007    0.0001    0.0000    0.0001    0.0001
   -0.0001   -0.0001   -0.0001   -0.0001   -0.0001   -0.0001    0.0000
   -0.0000   -0.0000   -0.0000    0.0000    0.0000    0.0000   -0.0001

  Columns 8 through 12

   -0.0003   -0.0004   -0.0010   -0.0001   -0.0000
   -0.0002   -0.0003   -0.0008   -0.0001   -0.0000
   -0.0002   -0.0002   -0.0007   -0.0001   -0.0000
   -0.0000   -0.0000    0.0001   -0.0001    0.0000
   -0.0000   -0.0000    0.0000   -0.0001    0.0000
   -0.0000   -0.0000    0.0001   -0.0001    0.0000
    0.0001    0.0001    0.0001    0.0000   -0.0001
    0.0002    0.0002    0.0002    0.0000   -0.0001
    0.0002    0.0004    0.0003    0.0000   -0.0001
    0.0002    0.0003    0.0014    0.0002    0.0002
    0.0000    0.0000    0.0002    0.0002    0.0001
   -0.0001   -0.0001    0.0002    0.0001    0.0002
```

(Manuscript received September 19, 2002;
revision accepted for publication September 9, 2003.)