Published in final edited form as:

J Exp Anal Behav. 2014 January ; 101(1): 171–178. doi:10.1002/jeab.58.

RAPID GENERATION OF BALANCED TRIAL DISTRIBUTIONS FOR DISCRIMINATION LEARNING PROCEDURES: A TECHNICAL NOTE

Christophe J. Gerard, Harry A. Mackay, Brooks Thompson, and William J. McIlvane

Shriver Center, University of Massachusetts Medical School

Abstract

In simple and conditional discrimination procedures, a series of stimulus displays are presented, each of which is intended to occasion a response of some type. Regular readers of this journal are likely to be familiar with simultaneous and successive simple discrimination, matching-to-sample, and other conditional discrimination procedures used in the study of basic discriminative and relational learning processes (cf. Mackay, 1991; McIlvane, 2012). However, behavior analysis is not alone in employing such procedures as key elements of methodology. Other major users of multitrial procedures are neuroscientists, some with cognitive perspectives (e.g., in bioimaging applications), others with a behavioral orientation (e.g., in behavioral teratology), psychologists conducting research on cognitive functions (e.g., in memory and attention) and on behavioral processes (e.g., in discrimination learning), special educators, autism and early intervention specialists (e.g., in individualized and classroom procedures for minimally verbal students; applied behavior analysis), and speech/language pathologists (e.g., language intervention procedures; augmentative/alternative communication training).

Keywords

multitrial procedures; rapid session construction tool; discrimination

The present technical report is aimed at a number of considerations that apply to use of most matching-to-sample procedures and by extension to most other multitrial procedure applications: In what order are the stimuli to be presented? If choice (comparison) stimuli are presented on each trial, how are they chosen and/or combined—a consideration that is especially important when stimulus sets are larger than the number of display positions. If stimuli are to appear in different positions, how are those positions assigned? Might interaction of stimulus type with stimulus positions matter?

The history of experimental psychology has shown that stimulus order and position variables may matter a great deal—perhaps especially when procedures involve only a few

Corresponding Author: Harry A. Mackay, Shriver Center, University of Massachusetts Medical School, 200 Trapelo Road, Waltham, MA 02452, 781-642-0022, FAX: 781-642-0238, Harry.Mackay@umassmed.edu.

Readers interested in obtaining first-hand experience with the algorithms may contact Dr. Gerard to obtain instructions for accessing a demonstration website.

stimuli and few positions. For example, in conditional discrimination procedures, the occurrence of runs of trials in which the same comparison stimulus is scheduled as correct may induce temporary stimulus preferences that interfere with the reinforcement contingencies intended by the experimenter. Similarly, disruptive position preferences may result from repeated presentations of correct stimuli at the same position.

One early effort to control such variables was the Gellerman (1933) series—a template for presenting stimuli in two-choice, two-position simple discrimination procedures. The template was designed to render the positive and negative stimuli on trial N unpredictable from the outcomes of trials N-1, N-2, N-3, and so on. In particular, the series was constructed such that the same stimulus and/or position would not be correct many times in a row and obvious artifacts such as alternating S+/S– stimuli or correct/incorrect positions would not be introduced. Further early recognition of the relevance of these procedural matters is clear in Hilgard's (1951) discussion of them in the *Handbook of Experimental Psychology* (Stevens, 1951). The approach suggested for addressing these considerations, however, was to assess stimulus and position preferences of groups of participants before the start of an experiment and then, in training, use trial sequences that went against the preferences of half the participants and with the preferences of the other half. Notably, this approach did not involve the more desirable aim of eliminating potential sources of unwanted, interfering stimulus control from the procedural arrangements of assessment or of training sessions.

The Gellerman (1933) series was a standard for decades in research on discrimination learning. However, even that series was criticized for failure to control adequately possible stimulus position artifacts (Fellows, 1967). Moreover, as the complexities of procedures increase (e.g., more than a few stimuli, stimulus types, and/or positions), the challenges in making appropriate combinatorial decisions concerning stimulus order and position sequences become progressively greater. When one introduces a sample stimulus to establish four-term contingencies (Sidman, 1986), for example, one must not only vary the S +/S– comparison stimuli and positions unsystematically but also the sequence of samples (and sometimes their positions; cf. Galvão et al., 2005; Iversen, 1997). Five-term contingencies (i.e., contextual control of four-term contingencies, cf. Serna & Perez-Gonzales, 2003) present still greater combinatorial decision challenges.

How are Combinatorial Decisions Made Currently?

Perhaps the most common approach is to rely on “chance.” For example, automated random-number generators may be used to select the order of trial types for some types of experimental work. Another approach is to specify a few basic decision rules (e.g., the same stimulus should not be the correct choice for more than three consecutive trials), and to generate a series of trials by hand, making local alterations in the series until one does not detect obvious sources of artifact. Each of these approaches then allows an investigator to (1) construct a collection of templates for application to procedures with different numbers of stimuli, positions, and trial types, and (2) accumulate templates over time for re-use via point-by-point stimulus substitution (i.e., an extension of the approach represented in the Gellerman series).

Limitations of Current Approaches

By definition, unconstrained random trial sequence generation implements no decision rules. Thus, it cannot eliminate undesirable local trial-distribution sequences (e.g., the same stimulus or position *may* be correct many times in a row). Abelson (1995) used the term “lumpy” to describe undesirable repetitions that may occur in randomly generated sequences. To justify unconstrained trial-sequence generation in experimental work, investigators must assume that (1) participants will not be much influenced by possible lumpiness and/or (2) effects of lumpiness will wash out over time and/or (3) statistical methods can control for lumpiness effects. Assumptions 1 and 2 may or may not be true, depending on the population and task (e.g., college students doing simple matching-to-sample tasks are unlikely to abandon attending to the sample merely because trial sequences have some degree of lumpiness). Assumption 3 may be true when one’s interest is to compare groups of individuals on a statistical basis and very high accuracy is not deemed essential. When the interest is optimized individual performance, however, mere statistical control is not an appropriate strategy (cf. Sidman, 1960).

Similar potentially inappropriate assumptions underlie the use and reuse of templates collected over time. That practice suffers from the same difficulties as reliance on “chance”. Assumptions 1 and 2 may or may not hold, just as for local, within-session effects of lumpiness. However, the consequences of repeated use across sessions (e.g., learning sequences of correct keys regardless of stimuli presented) simply may take longer to appear. With respect to Assumption 3, statistical control again is not appropriate when optimal individual performance is desired.

As members of a research group established by Sidman, we are highly attuned to procedural variables that may influence behavior of individual participants in research and teaching environments. Since the earliest days of our operations, group members have sought a software tool that would allow us to generate optimally balanced trial sequences that could be applied to the general class of simple and conditional discrimination procedures. Unfortunately, as we implied earlier, the computational burden and associated processing time increases as the complexity of procedures increase (e.g., in terms of stimulus number, type, position, level of conditionality, etc.). At high levels of complexity, one may not be able to produce a solution within a practical time frame (cf. Krippendorff, 1986).

Over the years, members of our research group have tried several times to develop a balanced-trial generation (BTG) tool that was useful across the range of procedures employed. Some efforts were partially successful, but continued efforts toward ultimate resolution of this class of challenges were always deferred because the algorithms always had clear flaws (e.g., often very lengthy compute times that only sporadically yielded the desired balanced trial sequences). Thus, they were not practical for routine use, and we have continued to rely mostly on past templates and/or hand-generation methods.

With the ever-increasing speed of computers and ancillary technology, however, we decided recently to try once again. The software prototype described in this technical report represents the best solution to the problem that we have developed thus far. Our note will summarize development of algorithms implemented in prototype software. We report also

on internal efforts to test the algorithms and show that they operate according to specifications reliably and rapidly. Mindful that potential users might want to evaluate the program first hand, we offer a link to a website (see *Author Note*) to give interested colleagues access to our software.

Development Processes

The computer used for algorithm development was a Dell Latitude (64 bits, 8GB Memory, Intel i5-2520M CPU @ 2.50 GHz). Algorithms were developed initially in JAVA (1.6.0_37 [32 bits] and subsequently in the Cygwin 1.7.17 C compiler (32 bits). Program outputs included (1) a table that provides a list of the sequence of sample:comparison mapping relations and positions of the stimuli for a simultaneous and/or delayed matching-to-sample session and (2) two matrices showing the distributions of each stimulus across the positions in which they serve as S+ and as S-. The solution output is provided as a fixed-width, space-delimited text that can be imported directly to Excel.

Phase 1 – Stimulus-Based Algorithm

The initial algorithm designed to construct sessions used a stimulus-based (SB) approach that assembles session content one stimulus at a time. The algorithm was developed in JAVA and used to generate trial sequences for a useful range of possible applications (2–8 comparisons per trial, 2–8 defined positions per trial, and 2–12 comparison stimuli with 1:1 mapping to corresponding sample stimuli). All comparison stimuli appeared equally often as S+ and S- in each position. The left portion of Table 1 lists further imposed constraints deriving from past practice in generating balanced trial sequences for our experimental work.

The SB algorithm begins by defining a selection pool consisting of the S+ and S- roles of all stimuli and all positions in which those stimuli can appear (i.e., A1+ [Stimulus A, in position 1, serving as S+], A1-, A2+, A2-, ... B1+, B1-, ... etc.). This pre-allocation process ensures a priori that quantitative constraints (i.e., the number of stimuli at a given location in a given role) will be met. The constraints shown in Table 1 can be adjusted merely by changing parameter values within the SB algorithm. For example, parameter modification could be used to modify the constraint that all S- stimuli appear in all locations equally often, thus permitting design of conditions that are appropriate when large numbers of stimuli are used. As the algorithm operates in practice, maximum values under the constraints are not reached frequently; their occasional occurrence adds to the unpredictability of which stimuli and positions will be S+ from trial to trial.

The trial sequence is assembled stimulus-by-stimulus through a recursive program. Currently, the stimulus selections are based on the probabilistic rules shown in Table 2. For example, as shown in the top row, after a stimulus has been selected for presentation in a trial, the probability that it will be selected also for the next trial is reduced by 10%. When a given trial is complete (all the stimuli have been chosen and intertrial and intratrial rules have been validated), the next trial is constructed and so on until the last trial for the session. When any constraint is not satisfied for the given stimulus selection, the recursive function “steps back” and tries an alternative to the stimulus previously validated.

To account for the nondeterministic nature of the algorithm (i.e., a possible solution may not be generated within a given search), a limiter parameter limits the search for a solution (heuristically defined as the following product; *number of trials* $\times$ *number of positions* $\times$ *number of stimuli* $\times 10$). For example, given a 72-trial target session with three locations and six stimuli (See Table 3, row 2), a search will be abandoned after selections of 12,960 stimuli without a solution; the program then automatically initiates new searches (up to 100 times).

Phase 2 – Trial-Based Algorithm

In Phase 2, the aim was to examine the construction of sessions by assembly of trials rather than one stimulus at a time. This trial-based (TB) algorithm was implemented in the C language for performance considerations (e.g., generation of optimized code compiled for a specific processor, efficient memory management, and easier portability to specialty applications such as hardware accelerators [field programmable gate arrays] and massively parallel processors [GPUs]). While the C implementation is not essential for our immediate purposes, its development expands capabilities that will support certain types of trial-generation applications at a later date (See Phase 3 for one example).

Briefly, the TB implementation performs an exhaustive enumeration of all the trials required to constitute a fully balanced session (where each stimulus appears equally often as S+ and S – in each location). The pre-generated trials are then arranged in an unsystematic order to form the session. The “search” algorithm consists merely of verifying that intertrial constraints are met. When a given trial is not acceptable (given the current trial sequence), all subsequent trials in the sequence are evaluated for substitution (in a lazy-Susan fashion). When no available trial fits the current trial sequence, the preceding trial is reevaluated. The algorithm continues this trial reordering until success is achieved (the last trial meets the constraint) or time runs out (the search limit as defined above has been reached). The program will automatically initiate new searches (up to 10,000) then generate the user output.

Because the algorithm computes all possible trials, the combinatorial process with large stimulus sets yields a very large number of trials. Thus, this method is best suited to the case where the number of stimulus positions, stimuli, and mapping relations is not as large as those handled by the SB program.

Phase 3 – Trial-Based Construction of Sessions for Learning by Exclusion

The TB implementation was refined to allow trials of different types to be intermixed in different proportions. A learning-by-exclusion procedure (McIlvane & Stoddard, 1981) was selected as a test case. Algorithms scheduled not only baseline arbitrary matching trials but also the exclusion and control trials that are used in that teaching method. In this extended algorithm, we retained the same characteristics (e.g., stimuli presented, order, spatial positions, constraints, etc.) that are important in generating trial sequences in general.

Algorithm Evaluations

Stimulus-Based Implementation

Qualitative evaluation—Initially, a variety of test sessions were prepared by entering values that specified their quantitative characteristics and constraints on the ordering of trials. Members of the development team studied the outputs manually in order to assess directly the effects of enabling, disabling, and changing the constraint values either singly or in combination (e.g., disabling the constraint limiting reuse of a particular stimulus as sample to three consecutive trials. Please note that these values cannot be changed in the demonstration program offered in the *Author Note*). In addition, the results obtained by using particular constraint values could be examined across repeated test runs, thus enabling detection of failure to reach a solution or the occurrence of trial sequences that violated the programmed constraints. Feedback from the development team led to refinements of the algorithm prior to formal testing for the purposes of this report.

Quantitative evaluation—We relied also on automatically collected quantitative data to measure the impact of different parameter settings on performance in order to tune the program. Data such as execution time, processor cycles, success probability of a given validation rule, number of incorrectly positioned trials, and search termination threshold were generated for large numbers of trial runs (at least 100).

Trial-Based Implementation

The TB implementation has been used thus far to explore how trial sequences might be generated virtually instantaneously when applications warrant that approach. We have conducted qualitative and quantitative testing of the type just described with a subset of balancing challenges that represents commonly used procedures. Because compiled C code runs much faster than JAVA, we increased the number of runs allowed to find a solution to 10,000 (i.e., 100 times more than with the SB program), a limit that could be set higher still to optimize program yield (i.e., percent successful completion; See Tables 4 and 5). After the process of session generations, evaluations, and algorithm refinements was concluded with satisfactory outcomes, the more complex exclusion training sessions were constructed and evaluated in the same manner.

Test Results

Table 3 presents representative quantitative data resulting from tests of the SB algorithm. The four columns on the left show the session characteristics. The most important data are shown in the rightmost two columns, which present, respectively, the success rate (percentage of successful search on the first attempt across 100 runs), and the mean search time across all attempts regardless of success.

Table 3 shows that the algorithm computed balanced-trial solutions for every configuration tested. For more than half of the tested configurations successful search occurred within 15 s on average for 100 test runs. For all but a few of those remaining, the SB algorithm performed very well (median first-attempt success rate 96%) with search times ranging from <1 s – 2.5 min. A major outlier (4% success rate) was for the configuration 8, 8, 8.

Increasing the search limiter value improved success rate (37%) although mean search time increased from just under 2 min to almost 12 min. A similar outcome occurred with configuration 6, 6, 6 when we reduced the search limiter value to 1,296. In this case, success rate declined (to 2%) but was offset by a decline in average search time. Because the algorithm is automatically rerun after an unsuccessful search, reducing the search limiter reduced the net time to a solution. Although a more detailed examination of the optimal stimulus, trial, and search limiter combinations is needed, the SB algorithm may be usefully employed at suboptimal combinations when search time is not a critical factor.

Table 4 shows results from the TB implementation of Phase 2. The algorithm yielded first-attempt success rates across 10,000 runs for every combination of configuration and limiter value tested (range: 4–73%, mean = 33%). For each configuration, the smallest limiter value tested was increased by a factor of 10 and, then, 100 for additional tests. The mean first-time success rate for each value (not shown) was 29%, 33% and 39%, respectively. The mean search times (not shown) were 0.12 s, 0.10 s and 0.95 s, respectively. Had the searches been permitted to go on longer, the success rates would have been higher.

The results of Phase 3 testing are listed in Table 5. It presents the performance of the TB algorithm for a 72-trial teaching session with various trial types (36 baseline, 18 teaching and 18 control trials) for three limiter configurations. The smallest limiter value yields a first attempt success rate of 34% with a search time of 6 ms.

Concluding Comments

The results show that the algorithmic approaches in development yield well-balanced sequences of matching-to-sample trials within a practical time frame when implemented on current desktop or laptop computing equipment. For the simple sample:comparison relations we have evaluated—and certainly for applications to simple discrimination procedures—our algorithm could be used routinely in research that uses multitrial methods within the scope evaluated and in applications of those methods in therapeutic or educational settings.

With many stimulus sets of the size and type used in past and current behavioral experiments, such sequences can be generated very rapidly. When the algorithm was used with larger stimulus sets, it often yielded solutions in only slightly more time. Even in those complex cases in which the algorithm fails to yield solutions instantaneously, it can do so when the number of searches scheduled is increased and/or when a search process requiring minutes rather than seconds is deemed acceptable (e.g., for generating pre-session templates).

One possible limitation on our work to date: We have tested the current algorithm only with three-term and four-term contingencies. We have not yet formally evaluated whether this algorithmic approach can be extended successfully to five-, six-, and higher-term contingencies (e.g., Gatch & Osborne, 1989). We decided to publicize the work at this point because the algorithm does appear to fulfill its intended function with stimulus sets like those used in many current applications. Because there are only a few extant examples of behavior-analytic research using five- or higher-term contingencies, we are not sure whether the further development necessary to address such contingencies is warranted by current

need. Moreover, it is not clear to us whether five-, six-, or n-term contingencies actually present any inherent combinatorial explosion challenges—despite the seeming complexity of those procedures. There are alternate conceptions of complex conditional discrimination procedures (e.g., the separable compound formulation of Stromer, McIlvane, & Serna, 1993) that suggest a straightforward approach to modifying or supplementing the present algorithm such that higher-order contingencies are handled with no more difficulty than seemingly simpler ones.

Regarding applications to larger sets of ordered pairs than we have tested thus far, there would almost certainly come a point at which the number of trials needed to yield a completely balanced session would become impossibly large from a practical perspective. That limitation does not seem to present problems for the approach. When the number of ordered pairs becomes very large, it seems to us that the need for an exhaustively calculated combinatorial solution is obviated. Even random generation of trial types would likely serve in that case, because the problem of generating lumpy trial sequences diminishes as the number of possible trial types increases. All that would be needed to manage large stimulus sets would be an algorithm with simple constraints to forestall the very rare occurrences of undesirable repetition of the same trial types and correct positions along with a procedure to roughly equalize the frequency of positions used.

Several future directions for our work seem obvious. First, we envision implementing the SB algorithm in C to leverage the intrinsic benefits of that language and allow comparison with the TB approach. Second, we want to explore further heuristics that might lead to reduced compute times and higher success rates with both algorithms. Finally, we hope to investigate how the algorithms perform when implemented on hardware with faster processing speed. Doing that might obviate the need to further tune the algorithms for practical implementation.

In summary, we believe (and hope) that we may be on the verge of resolving the technical challenge of producing a useful utility for developing complete balanced trial sequences for many applications of simple and conditional discrimination procedures. We invite readers of this note to evaluate whether our algorithmic approach does in fact yield the complete balanced-trial solutions by using the link provided in the *Author Note*. If the algorithm performs as expected in beta testing, then we hope to develop the utility into software that can be maintained and upgraded for future computing environments.

Acknowledgments

Development of the computer algorithms was supported in part by NIH grants HD04147, DC10365, and MH90272 and in part by funding from the Commonwealth Medicine Division of the University of Massachusetts Medical School. Address technical comments and/or questions to Christophe Gerard (christophe.gerard@umassmed.edu) and other types to Harry Mackay (harry.mackay@umassmed.edu). We thank William Dube and Ellen Isley for comments on an earlier version of the manuscript.

References

Abelson, Robert P. *Statistics as a Principled Argument*. Hillsdale, N.J.: L. Erlbaum Associates; 1995.

- Galvão OF, Barros RS, Brandão S, Lavratti CM, Santos JR, Brino ALF, McIlvane WJ. Extent and limits of the matching concept in *Cebus apella*: A matter of experimental control? The Psychological Record. 2005; 55:219–232.
- Gatch MB, Osborne JG. Transfer of contextual stimulus function via equivalence class development. Journal of the Experimental Analysis of Behavior. 1989; 51:369–378. [PubMed: 16812584]
- Gellerman LW. Chance orders of alternating stimuli in visual discrimination experiments. Journal of Genetic Psychology. 1933; 42:206–208.
- Hilgard, ER. Methods and procedures in the study of learning. In: Stevens, SS., editor. Handbook of Experimental Psychology. New York, NY: Wiley; 1951. p. 517-567.
- Fellows BJ. Change stimulus sequences for discrimination tasks. Psychological Bulletin. 1967; 67:87–92. [PubMed: 6045339]
- Iversen IH. Matching-to-sample performance in rats: A case of mistaken identity. Journal of the Experimental Analysis of Behavior. 1997; 68:27–45. [PubMed: 16812849]
- Krippendorff K. Combinatorial explosion. Web Dictionary of Cybernetics and Systems. 1986; 14 Principia Cybernetica Web.
- Mackay, HA. Conditional stimulus control. In: Iversen, IH.; Lattal, KA., editors. Experimental analysis of behavior, Part 1. Amsterdam: Elsevier; 1991. p. 301-350.
- McIlvane, WJ. Simple and complex discrimination learning. In: Madden, GJ., editor. APA handbook of behavior analysis. Washington, DC: American Psychological Association; 2012. p. 129-163.
- McIlvane WJ, Stoddard LT. Acquisition of matching-to-sample performances in severe retardation: Learning by exclusion. Journal of Mental Deficiency Research. 1981; 25:33–48. [PubMed: 6454002]
- Serna RW, Pérez-González LA. An analysis of generalized contextual control of conditional discriminations. Journal of the Experimental Analysis of Behavior. 2003; 79:383–394. [PubMed: 12908764]
- Sidman, M. Tactics of scientific research. New York, NY: Basic Books; 1960.
- Sidman, M. Functional analysis of emergent verbal classes. In: Thompson, T.; Zeiler, MD., editors. Analysis and integration of behavioral units. Hillsdale, NJ: Lawrence Erlbaum Associates; 1986. p. 213-245.
- Stromer R, McIlvane WJ, Serna RW. Complex stimulus control and equivalence. The Psychological Record. 1993; 43:585–598.

Table 1

Additional constraints on balanced trial generation

Constraint Imposed	Maximum value
Same stimulus in consecutive position	3
Same stimulus as consecutive S+	3
Same stimulus as consecutive S−	$(\text{No. of Positions} \times 2) - 1$
Same position as consecutive S+	3
Same position as consecutive S−	$(\text{No. of Positions} \times 2) - 1$

Table 2

Probability weights for stimulus selection in the SB algorithm.

Condition	Probability weight
Stimulus selection	–10%
Stimulus at given position	–10%
Stimulus in S+ configuration	–10%
S+ at a given position	–10%
S+ position	–50%

Table 3

Illustrative results of Phase 1 tests using the SB algorithm for 1:1 sample:comparison matching relations (identity or arbitrary).

Defined Positions	Comparisons Presented Per Trial	Comparison Stimulus Set	Generated Trials	Search Limiter	Successful Search (%)	Mean	
						Search Time (s)	
3	2	2	48	2,880	91	1	1
3	3	6	72	12,960	100	3	3
3	3	7	84	17,640	100	5	5
3	3	8	96	23,040	100	8	8
3	3	9	81	21,870	100	4	4
3	3	10	90	27,000	100	6	6
3	3	11	99	32,670	100	9	9
3	3	12	108	3	87	2	2
3	3	12	108	38	100	0.9	0.9
3	3	12	108	194	100	0.7	0.7
3	3	12	108	388	100	0.9	0.9
3	3	12	108	3,888	100	2	2
3	3	12	108	38,880	100	14	14
4	2	2	48	3,840	89	2	2
4	2	3	48	5,760	97	4	4
4	3	3	48	5,760	96	3	3
4	4	5	60	12,000	94	10	10
5	5	5	75	18,750	70	24	24
6	6	6	72	1,296	2	6	6
6	6	6	72	2,592	26	13	13
6	6	6	72	25,920	49	60	60
6	6	7	42	17,640	97	24	24
6	6	8	48	23040	99	31	31
6	6	9	54	29160	99	41	41
6	6	10	60	36,000	89	78	78
6	6	11	66	43,560	74	104	104
6	6	12	72	51,840	54	154	154

Defined Positions	Comparisons Presented Per Trial	Comparison Stimulus Set	Generated Trials	Search Limiter	Successful Search (%)	Mean Search Time (s)
7	7	7	49	24,010	76	60
8	8	8	64	40,960	4	113
8	8	8	64	204,800	37	700

Illustrative results of Phase 2 tests using the TB algorithm to construct sessions of 1:1 sample:comparison matching relations (identity or arbitrary).

Table 4

Defined Positions	Comparisons Presented Per Trial	Comparison Stimulus Set	Generated Trials	Search Limiter	Successful Search (%)		Mean Search Time (s)	
					Search	Success	Search	Time
2	2	2	224	896	56	56	0.01	0.01
2	2	2	224	8,960	58	58	0.02	0.02
2	2	2	224	89,600	73	73	0.11	0.11
3	3	3	234	2,106	38	38	0.10	0.10
3	3	3	234	21,060	50	50	0.06	0.06
3	3	3	234	210,600	52	52	0.53	0.53
3	3	4	216	2,592	17	17	0.29	0.29
3	3	4	216	25,920	23	23	0.11	0.11
3	3	4	216	259,200	23	23	1.04	1.04
3	3	5	180	3,240	4	4	0.01	0.01
3	3	5	180	32,400	7	7	0.16	0.16
3	3	5	180	324,000	10	10	1.67	1.67
4	4	4	192	3,072	21	21	0.20	0.20
4	4	4	192	30,720	28	28	0.16	0.16
4	4	4	192	307,200	37	37	1.42	1.42

Illustrative results of Phase 3 tests using the TB algorithm to construct sessions of 1:1 sample:comparison matching relations (identity or arbitrary) for exclusion training.

Table 5

Defined Positions	Comparisons Presented Per Trial	Comparison Stimulus Set	Generated Trials	Search Limiter	Successful Search (%)	Mean Search Time (s)
3	3	4	72	864	33	0.006
3	3	4	72	8,640	50	0.037
3	3	4	72	86,400	55	0.282