Application of an Automated Item Selection Method to Real Data

Martha L. Stocking, Len Swanson, and Mari Pearlman

Educational Testing Service

A method of automatically selecting items for inclusion in a test that has constraints on item content and statistical properties was applied to real data. Two tests were assembled by test specialists who assemble such tests on a routine basis. Using the same pool of items and the same constraints, the two tests were reassembled automatically. Test specialists not involved in the original manual assembly compared the tests constructed manually to the tests constructed automatically. The results indicated that the progress of automated test assembly methods lies in improving item banking systems, classification schemes, and quality control measures, rather than in the development of different algorithms or in the improvement of computer time and cost. *Index terms: heuristic algorithms, mathematical programming, test assembly, test construction, test design.*

The process of test construction is time consuming and expensive. Every year public and private testing organizations spend millions of dollars writing and editing items and then assembling these items into test forms for some measurement purpose. Until recently, the process of test construction or test assembly has been virtually unassisted by modern psychometrics.

A practical application of item response theory (IRT) suggested by Lord (1980) and Birnbaum (1968) is to use IRT in the test assembly process. But the test assembly process they suggested has little or no practical (as opposed to measurement) advantage over test assembly methods using conventional statistics without additional assistance from computers. In the last 10 years, many test assembly paradigms employing a combination of IRT, computers, and mathematical programming methods or heuristic methods have been proposed (e.g., Ackerman, 1989; Adema, 1988; Baker, Cohen, & Barmish, 1988; De Gruijter, 1990; Theunissen, 1985, 1986; Van der Linden, 1987; Van der Linden & Boekkooi-Timminga, 1989). Generally, these formulations have been successful, but typically have not been applied to test construction problems involving very large item pools and/or very large numbers of constraints on item selection. For this purpose, Swanson & Stocking (1993) developed a model and a heuristic particularly well suited for solving very large item selection problems.

This paper presents an application of the Swanson & Stocking (1993) weighted deviations model and heuristic algorithm using real data and real constraints actually employed in the manual test assembly process. First, two tests were assembled manually. Then, the tests were reassembled from the same item pool using the automated item selection algorithm. The results of the manual and automated assemblies were compared in terms of item overlap, statistical properties, and content properties.

Theoretical Framework

The test construction problem can be formulated as an optimization problem in which some aspect of the items to be selected for a test is optimized, subject to constraints on other item properties.

APPLIED PSYCHOLOGICAL MEASUREMENT
Vol. 17, No. 2, June 1993, pp. 167-176

0146-6216/93/020167-10$1.75

Typically, the variables representing the exclusion or inclusion of an item in a test are allowed to take on the values of only 0 or 1, respectively; hence, it is frequently called a binary optimization problem.

The typical formulation of a binary optimization problem consists of an objective function to be minimized or maximized subject to a set of linear constraints. Many different objective functions are conceivable; Van der Linden & Boekkooi-Timminga (1989, Table 3) listed the objective functions in a number of useful models. They also provided an extensive discussion of possible practical constraints on item selection, such as item content, item type, and statistical properties.

The advantage of this formulation is that it provides an elegant way of expressing both statistical and nonstatistical specifications in mathematical terms as constraints on test optimization. This formulation also makes it possible to solve the test construction problem using classical linear or integer programming algorithms. In particular, if binary programming models such as these have integer solutions (i.e., if they are feasible), then solutions can be found using standard mixed integer linear programming algorithms. Considerable attention has been focused on these algorithms, as well as ways of improving the standard algorithms by increasing their execution speed (e.g., Adema, 1989; Boekkooi-Timminga, 1989).

However, models such as those discussed by Van der Linden and Boekkooi-Timminga (1989) do not always have feasible solutions. Or, there may be a feasible solution, but the problem is so large that it is uneconomical to find this solution in practice. This may occur for many different reasons—some constraints may be difficult or impossible to meet, or the item pool may not be rich enough to satisfy all constraints simultaneously. The larger the number of constraints, the more likely it is that the binary programming problem will have feasibility problems because of the complexity of the interaction of the constraints.

The Weighted Deviations Model

The Swanson & Stocking (1993) weighted deviations model explicitly recognizes that models such as these may not have feasible solutions, and seeks to produce the best integer solution possible under the existing conditions. The motivation for this approach is that although an item pool may not support the construction of an ideal test, it remains the task of the test specialist to produce the best possible test under the circumstances.

Therefore, a reasonable model to facilitate this approach is to consider test specifications as "target" properties desired in the resultant test rather than as true constraints in the mathematical sense of binary programming. Deviations from the desired test properties then are incorporated in the objective function that is to be minimized. In addition, the test specialist can weight each property to reflect the relative importance of failing to satisfy any particular property in the test assembly. Thus, this model attempts to satisfy the target test properties by minimizing the aggregate failures, and at the same time attempts to provide some control over which are (or are not) allowable failures by weighting each property. This mirrors the goals of test specialists who regularly must assemble perhaps less than perfect tests from less than optimal item pools.

The Heuristic

The Swanson and Stocking (1993) weighted deviations model is expressed as a binary programming problem and can be solved using standard mixed integer linear programming algorithms, which will produce the best possible solution by minimizing the deviations from the target test properties. However, very large problems may be difficult to solve using standard algorithms—both from the perspective of computer storage requirements and computer time. Swanson and Stocking proposed a heuristic solution based on the nature of the model: Select items in such a way that the weighted

sum of the positive deviations from the target properties is as small as possible. As a sequential process, the heuristic seeks to monotonically increase progress toward the optimal solution.

The algorithm selects each successive item in the test by first estimating the extent to which the test specifications would be expected to be satisfied if all remaining items in the test were selected at random. The reason for including this step is to focus attention on those specifications that are the most difficult to satisfy, because they represent item properties that are over- or under-represented in the pool. The algorithm then uses this information to compute, for each eligible item in the pool, the expected value of the objective function—that is, the expected weighted deviations from the desired test properties, if the item were included in the test. This computation accounts for all previous items selected, the characteristics of the particular item for which the expected value of the objective function is being computed, and the expected characteristics of the remaining items in the test. The algorithm then selects the item that results in the smallest expected weighted deviation from the target test properties.

This heuristic conforms to the stated objective of minimizing the deviations from the desired test properties. The algorithm has been implemented and used extensively to solve a variety of test construction problems, including those reported here.

Application

Manually constructed verbal and quantitative tests were used. The 70-item verbal test consisted of three sections: reading comprehension (RC, 23 items); sentence correction (SC, an indirect measure of writing that was 27 items long); and critical reasoning (CR, a measure of verbal reasoning that was 20 items long). The quantitative test consisted of two parallel problem-solving (PS) sections, each 20 items long, that contained a balance of arithmetic, algebra, and geometry items; and a third section, 25 items long, that contained data sufficiency (DS) questions.

The pool from which the verbal test was constructed consisted of 1,538 items, including 55 reading comprehension passages with 11 to 12 items associated with each passage. The pool from which the quantitative test was constructed consisted of 853 items, including 11 quantitative stimuli with 4 to 5 items associated with each stimulus. Both pools were (independently) calibrated using LOGIST (Wingersky, 1983) with the three-parameter logistic model (see Lord, 1980). Information about the items, including their estimated item parameters and content classifications, was stored in a PC-based item-banking system.

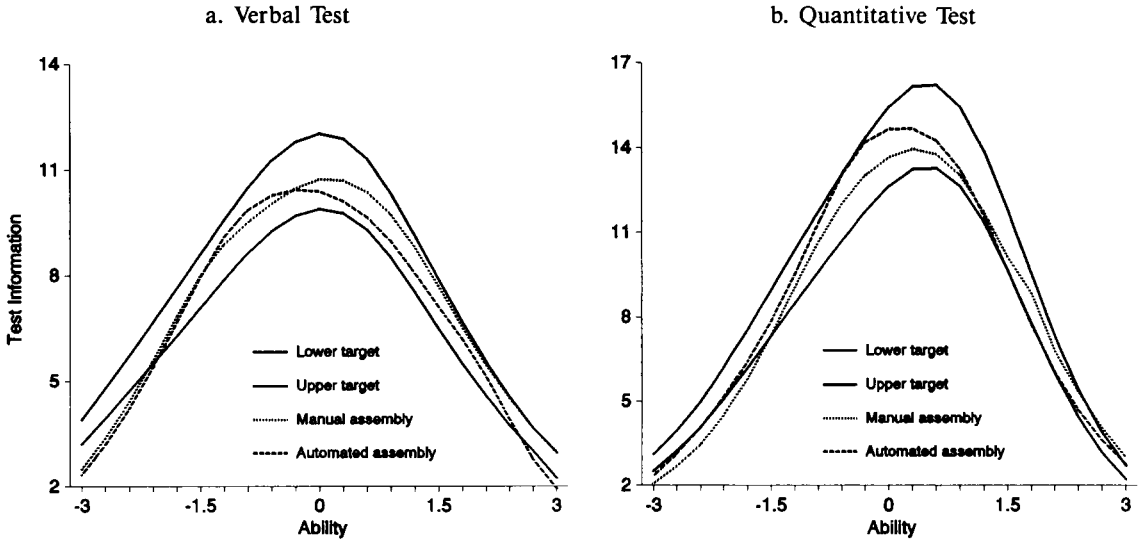
Manual Test Assembly

The test specialists constructed the tests based on target test information functions (TTIFs). The targets for each test were based on the attributes of previous editions of the same tests. Because the new test was required to be as parallel as possible to previous tests, both in terms of content and in terms of statistical properties, there were two targets for each test that represented the desirable range in which the resultant test information function (TIF) must lie. If the TIF was outside this range, it was considered to be less than optimally parallel to previous editions. The TTIFs for each test are shown as the lower and upper curves in Figure 1.

To develop the TTIFs for each test, the average of the TIFs for the previous six forms was computed. Then the variation of the six TIFs around this average was examined at different points on the trait metric. Most of the individual TIFs fell between approximately 10% below to 10% above the average curve at all trait levels. An arbitrary decision was made to create the lower and upper TTIFs for each test by computing the two curves representing this typical variation.

For the verbal test, the three sections were assembled in a serial fashion: The RC test was assembled

Figure 1
Upper and Lower TTIFs and the Resultant TIFs for the Manual and Automated Assemblies of the Verbal and Quantitative Tests



first because it was subject to the most constraints (because of the passages), had the least extensive item pool, and therefore was the most difficult to assemble. The CR section, which had a moderately extensive pool, was assembled next. The SC section, which had a reasonably extensive pool, was assembled last. Thus, any measurement deficits (in terms of test information) that existed after the assembly of the first two sections could be compensated for in the assembly of the final section.

During the test assembly, a test specialist selected an item or group of items to be included in the test, then evaluated the resultant partial TIF against the targets using a plot of the TIF and the TTIF provided by the item banking system. Difficult items were the rarest and were therefore usually selected first. Easy items were slightly less rare and were selected next. Items of middle difficulty were abundant, and therefore were usually selected last.

Simultaneously, the satisfaction of 43 nonstatistical constraints on item selection was monitored with a running tally of item properties in the form of a checklist. As with statistical properties, items that satisfied content considerations that were difficult to meet were typically selected first. At each item selection, an evaluation was also made to insure that a proposed item did not overlap with any item already included in the test. This process was repeated, with much backtracking, until the entire section was complete. The completed section was then turned over to the test specialist responsible for assembling the next section. The next test specialist attempted to remedy any measurement deficiencies in the partial test while satisfying the measurement and content constraints of the current section. It took approximately two professional work days to complete the initial assembly of the verbal test.

The assembly of the quantitative test was a simpler process because two separate sections were constructed simultaneously to be parallel, and the third was then added to complete the TIF. Items with approximately the same content and statistical properties were selected in pairs for the two parallel sections using frequency distributions of item difficulties and discriminations, as well as parallel content checklists. As with the verbal test, pairs that satisfied content and statistical categories that were

relatively rare were selected first. One of each pair was then randomly assigned to a section, making sure that a proposed item did not overlap with any item already assigned to that section. The test specialist responsible for the third section attempted to compensate for any measurement deficiencies in the two parallel sections, while simultaneously satisfying the measurement and content requirements of that section. There were 144 nonstatistical constraints on item selection across the three test sections. The three sections then were combined and their TIF was compared to the TTIFs. It took approximately one work day to complete the initial assembly of the quantitative test.

The statistical results of the manual assemblies for the two tests are shown as dotted lines in Figure 1. As this figure shows, the measurement requirements of the resultant tests were predominantly satisfied. Most of the nonstatistical specifications also were met. Two (out of 43) nonstatistical requirements were not satisfied for the verbal test. One constraint specified that between six and eight items of a particular type should be included, but only four were selected; a second constraint specified between two and four items but five were selected. Two (out of 144) nonstatistical specifications also were violated for the quantitative test. One constraint specified either one or two items were to be included, but none were chosen; a second constraint specified between zero and four items and six actually were selected.

Automated Assembly

The test specialists who participated in the automated test assembly were trained in the use of the computerized system as part of a larger project. The training activity began with two formal seminars. The first seminar focused on the rationale and theoretical formulation of the model and heuristic item selection algorithm employed. Although thorough knowledge of the technical details is unnecessary for the successful use of the software, it was felt that it would be helpful for test specialists to be aware of the foundations of both the model and the heuristic. The second seminar focused on demonstrating the actual use of the software and gave an overview of the entire process from test specifications to the production of the draft tests. Subsequent training was implemented by individualized instruction in the test specialists' offices working with the relevant item pools and the computer software.

The software was incorporated in the PC-based item banking system already being used by test specialists. A typical session began with the translation of written test specifications into "rules" for item selection. Each aspect of a test specification was associated with a particular code in a particular field of item identification contained in the item banking system. Rules were written in order to specify which code and which field in the database would be associated with a specification.

Then, for each rule, the test specialist entered the appropriate lower and upper bounds (which could be equal) on the desired number of items to satisfy a given rule, and a weight for that rule. The best procedure appeared to be to weight all rules equally for an initial execution and to only use differential weighting if the initial execution was unsatisfactory. Finally, the test specialist entered the total number of items for the test being assembled, as well as the total number of tests to assemble. If the statistical constraints to be used were based on IRT, the test specialist also identified the target curves and the trait levels at which optimization was to be accomplished.

All properties of items considered relevant to test assembly were coded in the item-banking system. Close observation and interaction with test specialists led to the realization that tests are assembled using not only the unique specifications for a particular test, but also using more general specifications that apply to a broader class of tests of which a particular test is a member. These additional specifications generally incorporated what are considered to be "good test construction practices" for assembling multiple-choice tests of this kind and are automatically taken into account by

experienced test specialists. However, for any automated method of test assembly to be effective, these types of more general constraints must be made explicit and incorporated into the test assembly problem.

The verbal test was assembled using 43 item properties; the quantitative test was assembled using 144 properties. Each item property had a lower and upper bound on the number of items with these properties that could appear in a test. If the bounds were not equal, then two constraints were added to the optimization problem. If the bounds were equal, then only a single constraint was added to the optimization problem. The total number of nonstatistical constraints for the optimization problem for the verbal test was 75 based on the 43 item properties. The total number of nonstatistical constraints for the quantitative test was 282 based on the 144 item properties.

The automated assemblies also were subject to statistical constraints in terms of the resultant TIF, as were the manual assemblies. The TIFs were required to lie at or above the lower TTIFs and at or below the upper TTIFs at selected values on the trait metric. The number of trait levels at which these constraints are enforced determines the number of constraints added to the automated test assembly problem. The actual trait levels used for the final test assembly can be selected based on some prior knowledge of the strengths and weaknesses of the entire item pool, or through an iterative process when intermediate solutions are seen to be less than satisfactory. For the verbal test, the constraints on the TIF were specified at 7 trait levels (θ = -2.1, -1.5, -.9, 0.0, .9, 1.5, 2.1), thus adding 14 constraints to the verbal solution. The total number of items to be selected served as a final constraint, bringing the total number of constraints for the verbal test to 90. For the quantitative test, the constraints were specified at 11 trait levels (θ = -2.4. -1.8, -1.5, -1.2, -.9, -.6, -.3, 0.0, .9, 1.8, 2.4), adding 22 constraints for a total of 305 constraints on the solution, including a constraint on the total number of items in the test. All the constraints for both the verbal and quantitative tests were weighted equally at 1.0.

The automated item selection algorithm was implemented on a 386 PC with a 80387 math coprocessor, running at 20 megahertz. Extensive audit trails were created during program execution, making it possible to follow the progress of the algorithm after the completion of an assembly.

Results

It took 10 minutes to assemble a 70-item verbal test from a pool of 1,538 items, subject to 90 constraints; and it took 8.5 minutes to assemble a 65-item quantitative test from a pool of 853 items, subject to 305 constraints. These times include both the time required for the algorithm and the time required to retrieve relevant data from the database. The latter components of the total time are implementation-based and, although they are important in terms of practical applications, are less important when comparing properties of algorithms. The amount of time directly attributable to the algorithm was 2.6 minutes (out of 10 minutes) for the assembly of the verbal test and 2.4 minutes (out of 8.5 minutes) for the assembly of the quantitative test, as compared to two or three days for the manual test assemblies.

The results of the automated assembly in terms of test information are shown as dashed lines in Figure 1. The TIFs lie predominantly within the lower and upper TTIFs within the range of interest on the trait (θ = -3.0 to +3.0). Although different from those for the manual test assembly process, these results were judged acceptable by test specialists. These results were obtained with complete satisfaction of all nonstatistical constraints specified for the algorithm. As noted earlier, the manual assemblies failed to satisfy two nonstatistical constraints for each test. Thus, the automated assembly performed slightly better than the manual assembly in terms of assembling a test with all of the desired properties. The audit trail showed that the automated assembly algorithm mirrored the manual assembly process in many important ways. For example, in both the verbal and the quantitative tests

the algorithm first selected items that had relatively scarce properties (i.e., properties that occurred less often in the pool than their desired proportion in the test would suggest). This applied both to content-related constraints and to measurement properties. Both the automated and the manual methods selected easy and difficult items early in the process because these items were relatively scarce in the pool.

However, there were some significant differences in how the automated item selection algorithm and the manual assembly proceeded. For the verbal test, the automated algorithm focused early on SC items because, for this particular pool, these items were scarce relative to RC and CR items (counter to the normal expectation of test specialists). As the expected deviations for SC-related constraints were reduced, the algorithm selected more CR items. RC items were most abundant in the pool, and were avoided until later in the assembly.

An important difference between the automated and the manual assemblies was that the algorithm shifted attention to different constraints and test sections as the expected deviations changed. For example, after selecting several SC items the SC-related deviations were reduced and it therefore shifted to CR items. As the CR-related deviations decreased the SC-related deviations increased relatively, and it therefore shifted back to SC items. This kind of alternating behavior occurred because the algorithm works on an item-by-item basis unless some other order is imposed. By contrast, in the manual process the test specialists select all the needed items for a section before shifting to a different section.

For the verbal test, the automated method selected early those items that measured well at the lowest and the next-to-highest trait levels. The TTIF was the most difficult to meet at these two points, in the sense that the expected information of the pool diverged the most from the test specifications at these two points.

The assemblies of the quantitative test followed similar patterns. Both methods began with PS items because these items were most scarce. However, the automated method began to shift to DS items as the PS-related deviations decreased, but the test specialists completed the PS sections before beginning work on the DS section. As with the manual assembly, the algorithm alternated between the two parallel PS sections when it was selecting PS items.

The quantitative pool measured better at high trait levels, relative to the test specifications, than it did at low trait levels. For this reason, the automated method focused early on easy items, unlike the manual assembly procedure. As the deviations associated with the lower trait levels decreased, the algorithm began to select items of higher difficulty.

When the tests assembled automatically were compared with the tests assembled by test specialists, there was little overlap of the actual items selected. For the entire verbal test, only 12 of 70 items (all associated with a single reading passage) were selected by both methods; the automated and manual assemblies of the quantitative test only had 7 of 65 items in common. Evidently, the large item pools were rich enough to support the assembly of a number of parallel tests without compromising either statistical or content constraints.

As a final evaluation on the forms assembled automatically, each form was given to test specialists for review. These test specialists normally assembled the final forms manually but were not involved in the manual assembly for this application.

The comments made by the test specialists fell into two distinct categories. One category concerned items that individual test specialists considered to be of poor substantive quality for a variety of reasons. Items with poor *statistical* properties were, for the most part, not included in the pools available for this application. The elimination of the problematic items identified on a substantive basis by test specialists could be handled in a similar manner. Eliminating them from the pool would guarantee that they would not be selected for inclusion in a test assembled by the heuristic. Incorporating

them appropriately in the constraints guarantees that they are only selected when absolutely necessary. The former solution requires no additional item coding; the latter solution requires additional item coding, but is probably closer to what test specialists do in practice.

A second category of comments from test specialists concerned the assembled tests and the classification of items represented by the assemblies. Some subcategories of classifications of items were not previously identified as important and therefore were not part of the item coding or the specification of constraints. In addition, some items were identified previously as having only a single property (for example, a verbal item on the abolitionist movement) when they should have been identified as having two properties (e.g., the role of women in the abolitionist movement). Thus, this exercise proved useful in eliciting from test specialists a more complete list of unique specifications for the tests. A more detailed classification and coding of items as well as the specification of additional constraints would be required to satisfy these additional specifications.

Discussion and Conclusions

Subsequent to the application described here, the Swanson & Stocking (1993) weighted deviations model and heuristic algorithm were successfully applied to the assembly of 14 other tests or test sections from item pools ranging in size from 300 to over 5,000 items and subject to from 30 to over 500 constraints. A number of these tests or test sections used conventional rather than IRT-based item statistics in the assembly process.

Much of what has been learned from using the weighted deviations model and heuristic algorithm has less to do with methods of automated item selection as applied to test assembly and more to do with item-banking systems in general, item classification schemes, and quality control. For example, the process of ensuring that a pool is "clean" (i.e., all items are correctly classified with respect to both content and statistical properties) is difficult, particularly when pool sizes are quite large. Although items with missing or incorrect classifications may be easy to correct, they may be difficult to detect before using test assembly software. In addition, because the weighted deviations model allows the specification of constraints on stimulus material as well as items, stimulus material must be identified as being uniquely different from items. Although this was generally true for all assemblies attempted thus far, it is not always the case.

Test specialists were generally pleased with the quality of the draft tests produced. Some were pleased to be relieved of the basically clerical task of constructing a draft test and to be able to concentrate their energies and expertise on producing a final form from a good first draft. An unanticipated benefit identified by a number of test specialists was the use of the software for pool management. When multiple forms are simultaneously assembled, the software attempts to insure that each form meets all content and statistical constraints. The results of a multiform assembly are useful in determining the quality of the item pool as well as pinpointing areas in which the pool is weak. This information, in turn, allows more precise management of future item writing and pretesting based on the number of draft tests that can be assembled from the current pool.

There are a number of possible areas of improvement in the implementation of the weighted deviations model and heuristic process. Two different kinds of logic that are sometimes found in written test specifications have been considered but not yet implemented. The first is "if-then" logic; for example, if an item of type 1 has been selected, then an item of type 2 also must be selected. The second is "either-or" logic; for example, either select an item of type 1 or an item of type 2, but not both.

In the process of manual assembly, test specialists automatically check to make sure that an item being added does not overlap with items already included in the test. Items may overlap for a variety

of reasons—they may test the same or nearly the same topic, one item may give away the answer to another, they may represent an unfair repetition of a content property that is irrelevant to the purpose of the test (for example, a verbal test composed of only science items), they may represent an unfair overuse of difficult words, and so forth. Currently, overlap can only be incorporated in the weighted deviations model and heuristic through the specifications of constraints. In theory, it is possible to code items in sufficient detail so that the specifications of constraints alone can prohibit undesirable overlap within a test. In practice, this may not be possible.

In the context of using the weighted deviations model for item selection in adaptive testing (Stocking & Swanson, in press), the concept of overlap groups, in which each group contains items that cannot be selected for the same test, is used to prevent undesirable overlap. The formation of such groups depends on the detection of both word overlap and content overlap. The most obvious, but also the most arduous, method of overlap detection is to make all possible pairwise comparisons of items in the pool. This is probably not practical for large pools because the number of comparisons is $N \times (N - 1)/2$, where N is the number of items in the pool. Other methods include computerized morphological and thesaurus-based algorithms that, although fallible, will reduce the number of comparisons to be made.

The automated assembly of tests should result in very good tests that may be more parallel to each other than those constructed through a manual process. However, it seems unlikely that any automated process can ever produce a test that can be considered finished without any human intervention. The weighted deviations model and heuristic studied here can probably never capture that part of test assembly that is clearly more of an "art" than a "science." Based on this application, as well as the more comprehensive applications, the progress of automated test assembly methods appears to lie in improving item banking systems, classification schemes, and quality control measures.

References

Ackerman, T. (1989, March). *An alternative methodology for creating parallel test forms using the IRT information function.* Paper presented at the 1989 National Council on Measurement in Education annual meeting, San Francisco.

Adema, J. J. (1988). *A note on solving large-scale zero-one programming problems* (Research Rep. 88-4). Enschede, The Netherlands: Department of Education, University of Twente.

Adema, J. J. (1989). *Implementations of the branch and bound method for test construction problems* (Research Rep. 89-6). Enschede, The Netherlands: Department of Education, University of Twente.

Baker, F. B., Cohen, A. S., & Barmish, B. R. (1988). Item characteristics of tests constructed by linear programming. *Applied Psychological Measurement, 12,* 189–199.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick, *Statistical theories of mental test scores* (pp. 395–479). Reading MA: Addison-Wesley.

Boekkooi-Timminga, E. (1989). *Models for computerized test construction.* DeLier, The Netherlands: Academisch Boeken Centrum.

De Gruijter, D. N. M. (1990). Test construction by means of linear programming. *Applied Psychological Measurement, 14,* 175–181.

Lord, F. M. (1980). *Applications of item response theory to practical testing problems.* Hillsdale NJ: Erlbaum.

Stocking, M. L., & Swanson, L. (in press). A method for severely constrained item selection in adaptive testing. *Applied Psychological Measurement.*

Swanson, L., & Stocking, M. L. (1993). A model and heuristic for solving very large item selection problems. *Applied Psychological Measurement, 17,* 151–166.

Theunissen, T. J. J. M. (1985). Binary programming and test design. *Psychometrika, 50,* 411–420.

Theunissen, T. J. J. M. (1986). Some applications of optimization algorithms in test design and adaptive testing. *Applied Psychological Measurement, 10,* 381–389.

Van der Linden, W. J. (1987). Automated test construction using minimax programming. In W. J. Van der Linden (Ed.), *IRT-based test construction* (pp. 1–16). Enschede, The Netherlands: Department of Education, University of Twente.

Van der Linden, W. J., & Boekkooi-Timminga, E. (1989). A maximin model for test design with practical constraints. *Psychometrika, 54,* 237–248.

Wingersky, M. S. (1983). LOGIST: A program for computing maximum likelihood procedures for logistic test models. In R. K. Hambleton (Ed.), *Applications of item response theory* (pp. 45–56). Vancouver: Educational Research Institute of British Columbia.

Acknowledgments

This work was supported by Educational Testing Service through the Program Research Planning Council. Part of this paper was presented at the annual meeting of the National Council on Measurement in Education, Boston, 1990.

Author's Address

Send requests for reprints or further information to Martha L. Stocking, Educational Testing Service, Princeton NJ 08541, U.S.A. Internet: mstocking@rosedale.org.