LSAC RESEARCH REPORT SERIES

Simultaneous Assembly of Multiple Test Forms

Wim J. van der Linden
University of Twente, Enschede, The Netherlands

Jos J. Adema
PTT Telecom

Law School Admission Council
Computerized Testing Report 97-13
December 2005

A Publication of the Law School Admission Council

The Law School Admission Council (LSAC) is a nonprofit corporation whose members are more than 200 law schools in the United States and Canada. It was founded in 1947 to coordinate, facilitate, and enhance the law school admission process. The organization also provides programs and services related to legal education. All law schools approved by the American Bar Association (ABA) are LSAC members. Canadian law schools recognized by a provincial or territorial law society or government agency are also included in the voting membership of the Council.

This study is published and distributed by the Law School Admission Council (LSAC). The opinions and conclusions contained in these reports are those of the authors and do not necessarily reflect the position or policy of the Law School Admission Council.

Table of Contents

Executive Summary

In educational testing, often multiple test forms have to be assembled from the same item pool at the same time. A well-known example is a testing organization assembling several forms for administration at different time slots. In computerized testing, the problem of multiple-form assembly occurs if a set of units for a multistage testing system has to be assembled or an item pool has to be reorganized into a pool of testlets for use in a testlet-based adaptive test. These units or testlets have to be informative at different intervals of the ability variable measured by the item pool. In addition, they have to be assembled such that each examinee gets a set of items meeting the same set of specifications.

The problem of multiple-form assembly can be solved in a sequential (one form after the other) or a simultaneous way (all forms at the same time.) Sequential algorithms are fast but have difficulty balancing the item content between forms; simultaneous algorithms balance optimally between forms but generally are too slow to be of practical value. The present paper formulates a heuristic in which a simultaneous assembly problem is reformulated as a series of computationally less intensive two-form problems that can be solved as fast as in a sequential approach. At each step, one of these forms is a form needed. The other is a specially designed dummy or shadow form that is returned to the item pool and whose only task is to create a balance between earlier and later forms in the series. It is shown how the heuristic can be implemented using the technique of 0-1 linear programming.

The papers contain two empirical examples, both based on a former pool of 753 items from the Law School Admission Test (LSAT). In one example, a set of parallel forms was assembled to meet the same target derived from the current specifications for the LSAT. In the other example, the forms were required to meet the same set of content specifications in use for the LSAT but to differ systematically in difficulty. In both examples, all forms met their targets perfectly.

Abstract

An algorithm for the assembly of multiple test forms is proposed in which the multiple-form problem is reduced to a series of computationally less intensive two-form problems. At each step, one form is assembled to its true specifications; the other form is a dummy assembled only to maintain a balance between the quality of the current form and the remaining forms. It is shown how the method can be implemented using the technique of 0-1 linear programming. Two empirical examples using a former item pool from the LSAT are given—one in which a set of parallel forms is assembled and another in which the targets for the information functions of the forms are shifted systematically.

Introduction

In educational testing, often multiple test forms have to be assembled at the same time. A well-known example is a testing organization assembling several parallel forms of a test for administration at different time slots. Another example is the assembly of a pair of tests for an evaluation of an educational program in which a pretest-posttest design is followed. A third example is found in large-scale assessments where multiple test forms have to be used because the pool of items needed to cover the subject areas assessed is too large to administer all items to each student in the sample. Still other examples of simultaneous assembly of test forms are found in computerized testing, for example, when a set of units for a multistage test has to be assembled (Lord, 1980, chap. 9), or an item pool has to be reorganized into a pool of testlets for use in a testlet-based adaptive test (Wainer & Kiely, 1987).

All these examples of test assembly can be classified with respect to the differences between the content and statistical specifications that the individual forms are allowed to have. If parallel forms have to be assembled, both the content and the statistical specifications of the forms are required to be identical. On the other hand, if a pretest-posttest combination has to be assembled for an educational program that is expected to be successful, the content specifications of both forms are identical but the statistical specifications should require the posttest to be more informative at higher ability levels. In test assembly for educational assessments, the content specifications vary across test forms; in addition, it is desirable to match the statistical properties of the individual forms with the strata in the population they are administered to. When assembling units or testlets for computer-administered testing, the primary interest is in a systematic variation of the

The authors are indebted to Wim M.M. Tielen for his computational support. Portions of this paper were presented at the National Assessment Governing Board Achievement Levels Workshop, Boulder, CO, August 20-22, 1997.

statistical properties of the test units. As for the content specifications, the emphasis is not so much on the composition of the individual units but on the requirement that, whatever route through the system is taken, the set of test units administered to the examinee always satisfies the same set of specifications.

The problem of the assembly of parallel test forms was addressed earlier in Ackerman (1989); Adema (1990, 1992); Armstrong, Jones, and Wang (1994); Armstrong, Jones, and Wu (1992); Boekkooi-Timminga (1987, 1990); Luecht (1998); and van der Linden and Boekkooi-Timminga (1988). Both Ackerman and Luecht use a greedy heuristic that builds up the test forms sequentially, spiraling item selection across forms. Ackerman's heuristic has a second stage in which items are swapped between forms to minimize the differences that have remained between their information functions. Armstrong et al. (1992) follow a two-stage approach. In the first stage, network flow programming is used to create stocks of items for the various content categories in the test forms. In the second stage, items are distributed over forms and then swapped between forms to minimize the remaining differences. The technique of 0-1 linear programming was used by van der Linden and Boekkooi-Timminga to create parallel test forms by matching them item by item. The methods by Boekkooi-Timminga and Adema will be discussed later in this paper.

It is the purpose of this paper to present a method for assembling multiple test forms that should be able to deal with any of the above examples of test assembling and produce good results no matter what constraints on the contents of the test forms or what statistical requirements have to be satisfied. The only assumption made is that the forms are assembled from the same pool. If this assumption were also to be dropped, the problem would be reduced to a series of independent, single-form assembly problems. Problems of the last type exist, for example, in psychological testing when a test battery has to be assembled for a job selection problem and each test is required to measure a different facet of the success criterion.

The remaining part of the paper is organized as follows: The problem of multiple-form assembly will first be formalized as an instance of constrained optimization in which one of the test specifications is assumed to take the form of an objective function and the other specifications formulate the constraints under which the objective function is optimized. Then, a general method for the assembly of multiple forms will be given. It is explained how the method can be implemented using the technique of 0-1 linear programming (LP). In this technique, 0-1 variables are used to denote the decision whether or not to select the items from the pool for the test form. The objective function and constraints are then formulated in the variables. An algorithm or heuristic is used to find the set of values for the variables that meets all constraints and has an optimal value for the objective function. For a description of the technique as it can be applied to test assembly problems, see van der Linden (1998). The paper is concluded by two empirical examples—one in which a set of parallel test forms was assembled and another in which the target information functions were spread along the ability scale.

Optimization Models for Test Assembly

It is assumed that the items in the pool are represented by decision variables x_i, $I = 1, ..., I$ denoting whether ($x_i = 1$) or not ($x_i = 0$) the items are to be included in the test. These variables are used to model the test assembly problem as an objective function to be optimized under a system of constraints. Experience with a large variety of test assembly problems has shown that most objective functions and constraints can be formulated as linear expressions and (in)equalities, respectively.

In IRT-based test assembly, the model typically has the following minimal form.

1. The objective function minimizes the distances between the test information function and a target function at a series of θ values.

2. One or more constraints are present to fix the length of the test and/or sections at prespecified numbers of items.

3. Several constraints are used to model the test specifications that deal with categorical item attributes. Examples of categorical item attributes are item content and format, whether or not the items have graphics, and cognitive classifications. The distinctive feature of categorical attributes is that each of them introduces a partition of the item pool with different classes of items each associated with different levels of the attribute. The constraints in this category typically specify the distribution of the items over the partitions to have a special form.

4. Several constraints may be present to model test specifications that deal with quantitative item attributes. These attributes are parameters or coefficients with numerical values, such as item *p*-values, word counts, and (expected) response times. The constraints in this category usually require sums or averages of the values of these attributes to be in certain intervals.

5. Some constraints may be needed to deal with possible dependencies between the test items in the pool. For example, certain items may have to be administered as a set related to the same text passage whereas others are not allowed to appear in the same form because they have clues to each other.

Exemplary Model

Following is an example of a model illustrating a few examples of the constraints in each of the above categories. The objective function minimizes the sum of the (positive) differences between the test information function and target values at a series of values θ_k, $k = 1, ..., K$. Because information functions have a well-behaved continuous form, only a few θ values are necessary. Practical experience has shown that a choice of three to five values is generally sufficient. The values of the information function of item I at these points are denoted as $I_i(\theta_k)$; the target values as $T(\theta_k)$. This objective is used here only as an example. In practice it can be met if a new form has to be assembled that has to be parallel to a previous form. An approach in which the distances between the information function and the target are minimized from below, or from both sides, is also possible. Other examples of targets for information functions will be given later. For convenience, only one categorical item attribute (cognitive level) is used, with levels $h = 1, ..., H$, each corresponding with a different subset of items in the pool, C_h. For each subset, the number of items in the test has to be between $n_h^{(l)}$ and $n_h^{(u)}$. Likewise, one quantitative attribute is used, which is chosen to be the expected response time on each of the items in the pool by a typical examinee in the population for which the test is assembled. These response times are denoted by r_i, and it is required that the total response time needed not exceed the amount of time allotted, $r^{(u)}$. Finally, as an example of a dependency between the test items in the pool, it is assumed that some items are not allowed to be in the same test form. The set of index values of these items is denoted as V_E.

The model is as follows:

$$\text{minimize} \sum_{k=1}^{K} \sum_{i=1}^{I} I_i(\theta_k)\, x_i \qquad \text{(objective function)} \qquad (1)$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k) x_i - T(\theta_k) \geq 0, \quad k = 1, ..., K \qquad \text{(information target)} \qquad (2)$$

$$\sum_{i=1}^{I} x_i = n, \qquad \text{(test length)} \qquad (3)$$

$$\sum_{i \in C_h} x_i \leq n_h^{(u)}, \quad h = 1, ..., H, \qquad \text{(cognitive level)} \qquad (4)$$

$$\sum_{i \in C_h} x_i \geq n_h^{(l)}, \quad h = 1, ..., H, \qquad \text{(cognitive level)} \qquad (5)$$

$$\sum_{i=1}^{I} r_i x_i \leq r^{(u)} \qquad \text{(allotted time)} \qquad (6)$$

$$\sum_{i \in V_E} x_i \leq 1 \qquad \text{(mutually exclusive items)} \qquad (7)$$

$$x_i = 0, 1, \quad I = 1, ..., I. \qquad \text{(definition of } x_i\text{)} \qquad (8)$$

As is evident from these equations, all variables in the model are 0-1, and each expression is linear in the variables. Optimization models of this type are known as 0-1 linear programming (LP) models. They can be solved for optimal values of their variables using one of the standard software packages available for LP. A choice of algorithms and heuristics is also offered in the test assembly package ConTEST (Timminga, van der Linden, & Schweizer, 1996). If the model has a special structure, efficient implementation of algorithms may be possible (for an example, see Armstrong & Jones, 1992).

Differences Between Objective Functions and Constraints

The model in Equations 1–8 allows us to discuss more precisely the differences between the examples of multiple-form assembly problems given earlier. Objective functions are generally chosen to deal with the statistical features of test forms, constraints to deal with the content of the test. If this convention is followed, each of the examples differs in the use of its objective function and/or constraints. The problem of assembling parallel forms involves both identical objective functions and identical constraints across all forms. In the pretest-posttest problem discussed earlier, the target for the objective function of the posttest has values $T(\theta_k)$ shifted relative to those for the pretest, but the constraints for the two tests are identical. In educational assessments, it may be desirable for each form to have an objective function tailored to the distribution of a stratum of the population it is assigned to. In addition, the sets of constraints for the test forms are allowed to vary provided all items in the pool are placed in a form and there is enough overlap between forms to estimate covariances between subscores. Test units for use in computerized adaptive testing have objective functions with respect to targets covering different intervals on the ability scale. They usually have no overlap in content but are to be assembled such that units at the same stage meet the same subset of constraints.

Definition of Problem

The first approach to the problem of assembling multiple test forms that comes to mind is to assemble the forms in a sequential fashion, each time removing the items already selected from the pool and adapting the model to fit the next form. The case of assembling parallel forms is used to demonstrate that this method has two serious disadvantages. First, if these forms are assembled one after the other, the value of the objective function for the solution to the model tends to deteriorate over forms because the items with the best values for their attributes are likely to be selected first. As a consequence, the forms cannot be parallel. The second disadvantage is the possibility of unnecessary infeasibility of the problem at a later stage in the assembly process. This phenomenon can be illustrated using the data in Table 1, which describes the levels of only a few of the items in a larger pool on two of the attributes used in the test specifications, labeled Attribute 1 and Attribute 2. For simplicity, these attributes are assumed to represent features that the items either have or do not have (e.g., use of graphics in the stem). Suppose two test forms have to be assembled such that each form has to have at least two items with Attribute 1 and one item with Attribute 2. In a sequential procedure, the selection algorithm might pick both Item 2 and Item 3 for the first test because they have large contributions to the target. However, as a consequence of this choice, a second form satisfying the same set of constraints is no longer possible. In a simultaneous approach, a sound algorithm would always assign Item 2 to one test form and Item 3 to the other, and thus prevent the problem. (In fact, it is the presence of more complicated examples of such attribute structures, often not immediately obvious, that makes manual assembly of multiple test forms a notoriously difficult process. Once a feasible solution is found, test assemblers may feel inclined to stop because of feelings of relief rather than the certainty that an *optimal* feasible solution has been found.)

TABLE 1
Example of unnecessary infeasibility in sequential test assembly

	Item 1	Item 2	Item 3	Item 4	Item 5
Attribute					
1	x		x	x	x
2		x	x		
Contribution to Target	0.35	0.71	0.84	0.29	0.45

Note. An "x" indicates that the item has the attribute.

Both disadvantages were already noted in Boekkooi-Timminga (1990). Her solution was to remodel the problem using different decision variables. Suppose the individual (parallel) test forms are denoted by $f = 1, \ldots, F$. The new decision variables, x_{if}, are defined such that $x_{if} = 1$ indicates that item I is assigned to test form f and $x_{if} = 0$ otherwise. Hence, each item is assigned directly to a test form and all assignments take place simultaneously. For the model used in Equations 1–8 the result would be as follows:

$$\text{minimize} \sum_{f=1}^{F} \sum_{k=1}^{K} \sum_{i=1}^{I} I_i(\theta_k) x_{if} \qquad \text{(objective function)} \qquad (9)$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k)x_{if} - T(\theta_k) \geq 0, \quad k = 1, ..., K, \quad f = 1, ..., F \qquad \text{(information target)} \qquad (10)$$

$$\sum_{i=1}^{I} x_{if} = n_f, \quad f = 1, ..., F, \qquad \text{(length of forms)} \qquad (11)$$

$$\sum_{i \in C_h} x_{if} \leq n_h^{(u)}, \quad h = 1, ..., H, f = 1, ..., F, \qquad \text{(cognitive level)} \qquad (12)$$

$$\sum_{i \in C_h} x_{if} \geq n_h^{(l)}, \quad h = 1, ..., H, f = 1, ..., F \qquad \text{(cognitive level)} \qquad (13)$$

$$\sum_{i=1}^{I} r_i x_{if} \leq r^{(u)}, \quad f = 1, ..., F, \qquad \text{(allotted time)} \qquad (14)$$

$$\sum_{f=1}^{F} x_{if} \leq 1, \quad I = 1, ..., I, \qquad \text{(no overlap)} \qquad (15)$$

$$\sum_{i \in V_e} x_{if} \leq 1, \quad f = 1, ..., F, \qquad \text{(mutually exclusive items)} \qquad (16)$$

$$x_{if} = 0, 1, \quad I = 1, ..., I, f = 1, ..., F. \qquad \text{(definition of } x_{if}) \qquad (17)$$

Observe that Equation 15 has been added to prevent each item from being assigned to more than two forms. Also, the total number of constraints has gone up because all constraints in Equations 9–14 and 16 now are in force F times, and Equation 15 entails I new constraints. More important, however, the number of variables has gone up by a factor F. Therefore, unless problems with a special structure are met (e.g., network-flow problems) or satisfactory heuristics are possible, only smaller problems can be solved. Larger problems may quickly result in memory management problems and/or prohibitively large computation times. If so, the methods presented in the next section are helpful.

Basic Method

As just outlined, the basic problem with a sequential approach to the assembly of multiple forms is an unbalanced assignment of items to forms. On the other hand, the approach does have the advantage of the smallest number of decision variables and constraints needed. The simultaneous approach in the previous section elegantly solves the problem of unbalancedness but its price is a larger number of variables and constraints. The method in this paper, of which a version for the assembly of weakly parallel test was already proposed in Adema (1990), does provide the balancing of test content and at the same time minimizes the increase in the numbers of variables and constraints considerably.

Basically, the method reduces any multiple-form assembly problem to a series of computationally less intensive, two-form problems. At each step, one form is assembled according to the true specifications. The other form is a dummy assembled according to specially adapted specifications; its only task is to balance between the contents of the current form and the later forms. As soon as both forms have been assembled, the items selected for the dummy are returned into the pool, and the process is repeated. The reason for putting the items for the dummy back into the pool is not only to increase the number of items to choose from; these items are likely to have good attribute values and can therefore be expected to improve the quality of later forms.

To present the method two different sets of decision variables will be used—one set of variables x_i, $I = 1, ..., I$, to denote whether ($x_i = 1$) or not ($x_i = 0$) item I will be assigned to the form assembled and another set of variables z_i, $I = 1, ..., I$, for the same decision with respect to a dummy form. A choice from the following two objective functions is proposed:

1. An objective function that can be used if the relative efficiencies of the test forms have to be controlled, but more information is always welcome. An example is a pair of forms for use in pretest-posttest study where each form is required to be equally informative over a different interval on the ability scale, but at the same time it holds that the more informative the two tests are, the better.

2. An objective function that can be used for the assembly of multiple forms, each of which has a target for its information function with an absolute shape relative to the ability scale in use for the item pool. This case is almost exclusively met when sets of parallel forms have to be assembled. In such applications, it would actually hinder if some of the forms were more informative at certain ability values than dictated by the common target because of the impact of the differences on the observed-score distributions.

For either case a test assembly model will be given in the next sections. Also, each model will be illustrated by an empirical example later in this paper.

Targets for Relative Efficiencies

Let $R_f(\theta_k)$, $k = 1, \ldots K$, denote the relative efficiencies for test form f; that is, these numbers denote the relative heights of the test information function that is maximized as a target in the test assembly. The model for assembling form f plus its associated dummy form is

maximize y (objective function) (18)

subject to

$$\sum_{i=1}^{I} I_i(\theta_k)x_i - R_f(\theta_k)y \geq 0, \quad k = 1, \ldots, K, \qquad \text{(target for form f)} \qquad (19)$$

$$\sum_{i=1}^{I} I_i(\theta_k)z_i - \sum_{g=f+1}^{F} R_g(\theta_k)y \geq 0, \quad k = 1, \ldots, K, \qquad \text{(target for dummy)} \qquad (20)$$

$$\sum_{i=1}^{I} x_i = n_f, \qquad \text{(length of form f)} \qquad (21)$$

$$\sum_{i=1}^{I} z_i = \sum_{g=f+1}^{F} n_g, \qquad \text{(length of dummy)} \qquad (22)$$

$$\sum_{i \in C_h} x_i \leq n_{hf}^{(u)}, \quad h = 1, \ldots, H, \qquad \text{(cognitive level)} \qquad (23)$$

$$\sum_{i \in C_h} x_i \geq n_{hf}^{(l)}, \quad h = 1, \ldots, H, \qquad \text{(cognitive level)} \qquad (24)$$

$$\sum_{i \in C_h} z_i \leq \sum_{g=f+1}^{F} n_{hg}^{(u)}, \quad h = 1, \ldots, H, \qquad \text{(cognitive level)} \qquad (25)$$

$$\sum_{i \in C_h} z_i \geq \sum_{g=f+1}^{F} n_{hg}^{(l)}, \quad \mathrm{h} = 1, \ldots, H, \qquad \text{(cognitive level)} \qquad (26)$$

$$\sum_{i=1}^{I} r_i x_i \leq r^{(u)}, \qquad \text{(allotted time)} \qquad (27)$$

$$\sum_{i=1}^{I} r_i z_i \leq \sum_{g=f+1}^{F} r_g^{(u)}, \qquad \text{(allotted time)} \qquad (28)$$

$$x_i + z_i \leq 1, \quad I = 1, \ldots, I, \qquad \text{(no overlap)} \qquad (29)$$

$$\sum_{i \in V_E} x_i \leq 1, \qquad \text{(mutually exclusive items)} \qquad (30)$$

$$\sum_{i \in V_E} z_i \leq 1, \qquad \text{(mutually exclusive items)} \qquad (31)$$

$$x_i \in \{0,1\} \quad I = 1, ..., I. \qquad \text{(definition of } x_i\text{)} \qquad (32)$$

$$z_i \in \{0,1\} \quad I = 1, ..., I. \qquad \text{(definition of } z_i\text{)} \qquad (33)$$

In Equation 19, lower bounds $R_f(\theta_k)y$ are imposed on the information function for form *f*. Their common factor, *y*, is maximized in Equation 18. Since the length of form *f* is fixed at n_f items in Equation 21, the solution will satisfy the inequalities in Equation 19 close to equality. The result is an information function that has a form dictated by the efficiency parameters $R_f(\theta_k)$ and has a maximum height because of Equation 18. The constraints in Equation 20 ensure that the same will happen for the dummy form. In Equation 22 the length of the dummy is set equal to the sum of the lengths of all remaining forms. Because the test length is constrained from above, the constraints in Equations 19–20 can be expected to be satisfied close to equality at optimality. The constraints related to the various cognitive levels in Equations 23–26 as well as to the allotted times in Equations 27–28 are adapted accordingly. The constraint needed to prevent items from overlapping between the test forms has now to be formulated, as in Equation 29. Finally, the constraints to deal with dependencies between the test forms are now repeated for the dummy test in Equation 31.

Note that the coefficients in the constraints have been made form-dependent to allow for differences in specifications between the forms. Also, apart from the change of variables, the main changes in the constraints for the dummy are in their right-hand side coefficients; these have been made larger to enforce adequate balancing of test contents between forms.

Absolute Targets

To realize this type of target, the efficiency parameters $R_f(\theta_k)$ are replaced by target values $T_f(\theta_k)$ as follows.

$$\text{minimize } y \qquad \text{(objective function)} \qquad (34)$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k)x_i - T_f(\theta_k) \leq y, \quad k = 1, ..., K, \qquad \text{(target for form f)} \qquad (35)$$

$$\sum_{i=1}^{I} I_i(\theta_k)x_i - T_f(\theta_k) \geq -y, \quad k = 1, ..., K, \qquad \text{(target for form f)} \qquad (36)$$

$$\sum_{i=1}^{I} I_i(\theta_k)z_i - \sum_{g=f+1}^{F} T_g(\theta_k) \leq y, \quad k = 1, ..., K, \qquad \text{(target for dummy)} \qquad (37)$$

$$\sum_{i=1}^{I} I_i(\theta_k)z_i - \sum_{g=f+1}^{F} T_g(\theta_k) \geq -y, \quad k = 1, ..., K, \qquad \text{(target for dummy)} \qquad (38)$$

The idea in Equations 35–36 is to constrain the differences between the values of the test information functions and their targets to the interval [-*y*,*y*]. The objective in Equation 19 is to make this interval as small as possible. The same idea is repeated for the dummy test in Equations 37–38.

Minimax/Maximin Principle

Both the objective functions in Equations 18–20 and 34–38 are of the minimax/maximin type. In the former, a series of lower bounds is maximized until no further improvement is possible without violating one of them. The latter implies minimization of the largest deviation of a test information function value from its target. The maximin/minimax principle was introduced in test assembly in van der Linden and Boekkooi-Timminga (1989). Application of the principle is a convenient way to unify targets for different test forms into a single objective function. The series of lower bounds or deviations over which the optimization takes places is defined over the same grid of θ values for all forms. This requirement poses no problem; the grid can always be defined on the union of the sets of θ values needed for each form, setting to zero target values for the θs that are too far out.

Relaxed Decision Variables

If the problem is still too large to be solved in realistic time, an effective reduction of the combinatorial complexity involved in 0-1 LP can be realized by relaxing the decision variables for the dummy test, that is, by replacing Equation 33 by

$$z_i \in [0,1], \quad I = 1, ..., I. \qquad \text{(relaxed variables)} \qquad (39)$$

This measure may result in a slightly less effective form of content balancing between the various test forms, but since the number of 0-1 variables is halved, the effect on the branch-and-bound step generally used in the algorithms and heuristics in this domain can be expected to be dramatic. As some of the variables are now real valued, the problem becomes an instance of mixed integer linear programming (MILP).

Examples

Two different examples are given, both based on the same previous pool of 753 items from the Law School Admission Test (LSAT). In the first example, three different test forms with an identical target for the information functions were assembled. This common target was chosen to be exactly between the upper and lower limits for the information function used in the assembly of the LSAT. In the other example, the target for one form was kept the same whereas one of the two other forms had a target shifted .60 to the left and the other a target shifted .60 to the right. These three targets thus indicated different levels of difficulty of the test.

In both examples, the forms had to meet the same set of constraints. The constraints represented the specifications of the LSAT with respect to such item attributes as content, answer key, gender and minority orientation, and word count. The only constraints left out were those dealing with the item set structure in two sections of the test; the topic of assembling tests with item sets is dealt with elsewhere (van der Linden, 2000). The total number of constraints was equal to 115.

The total test length was set equal to 75 items, with the items being in three different sections: Analytic Reasoning, Logical Reasoning, and Reading Comprehension. (The full LSAT duplicates one of the sections).

The models for both examples were solved using the First Acceptable Integer Solution Algorithm in the ConTEST software (for a description of the algorithm, see Timminga, van der Linden, & Schweizer, 1996, sect. 6.6). The algorithm uses the value of the objective function in the solution of the fully relaxed model to calculate a bound to the value of the objective function in the original problem at which the search for a further improvement on the current solution is stopped. In the present examples, the relaxed versions of the models had solutions with a value for the objective function equal to .00. Therefore, it was decided to stop each search after 30 minutes. The values for the objective functions at this point were between .215 and .555 for all test forms (see the scales in Figures 1–3).

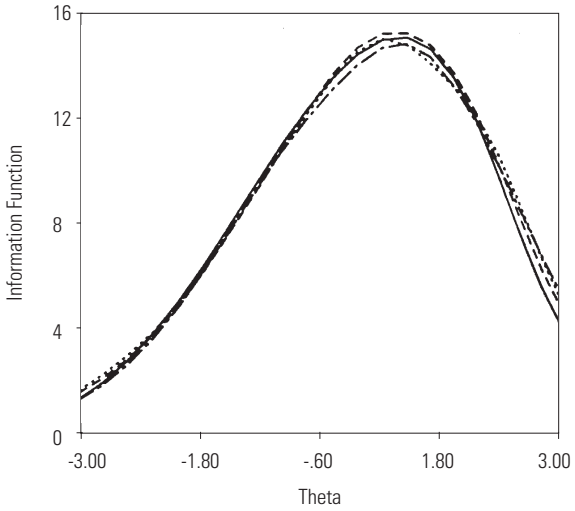

FIGURE 1. *Information functions for the parallel test forms assembled in the first example (solid line represents target)*

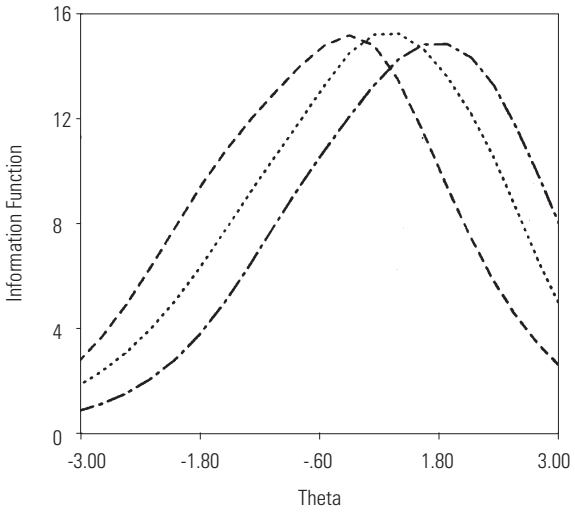

FIGURE 2. *Information functions for the test forms with different difficulty levels assembled in the second example*

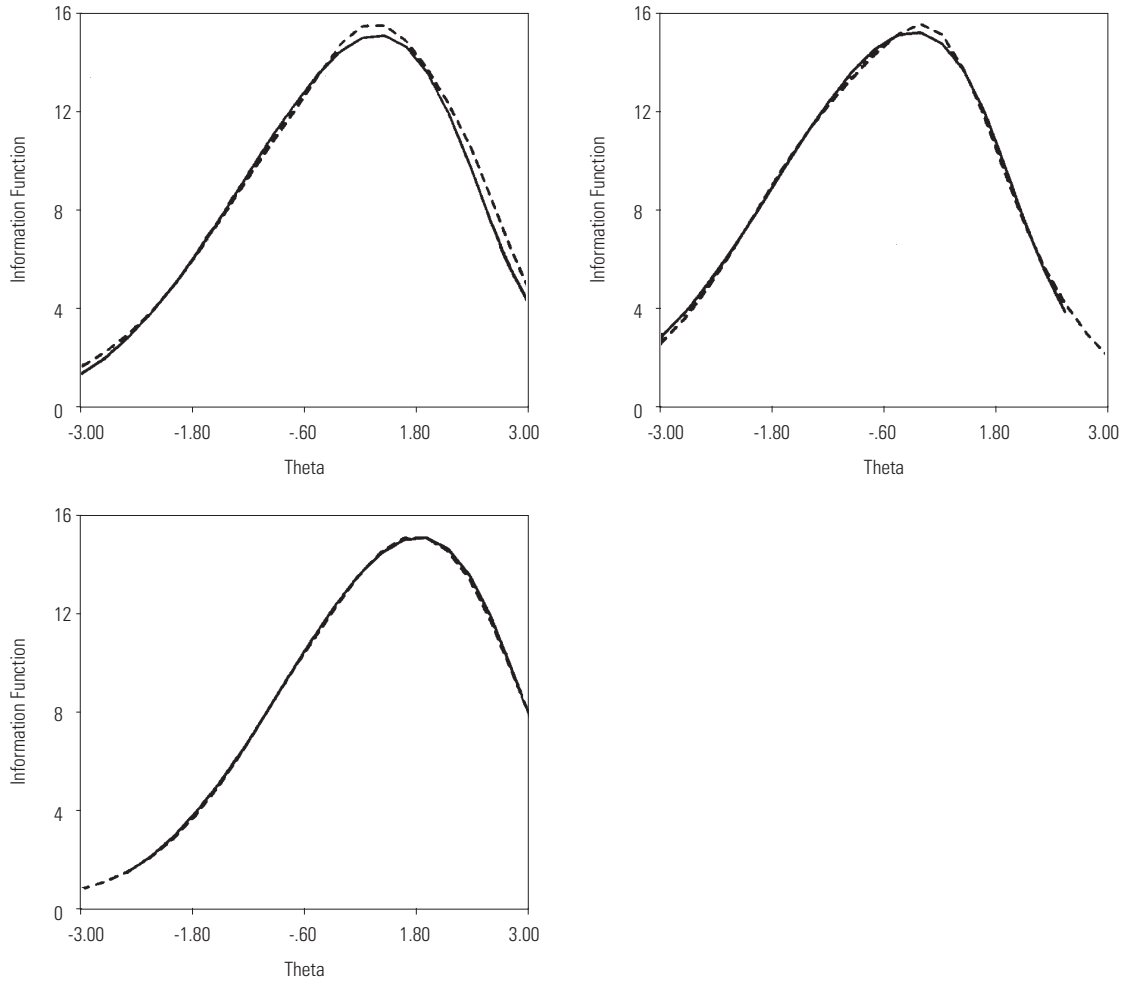

FIGURE 3. *Information functions and targets for the three test forms in the second example (solid lines represent targets)*

The information functions for the first example are shown in Figure 1. All functions are close to the target represented by the solid line in the graph. The information functions for all forms in the second example are shown in Figure 2. The functions show the systematic shifts along the scale wanted. The panels of Figure 3 show the information functions for the individual test forms along with their targets. Again, each function is close to its target, but the information for the most difficult test is closest. Apparently, the item pool has more difficult than easy items to chose from.

Concluding Remarks

The technique of 0-1 LP has already proved to be a useful aid in test assembly problems in which a single test form has to be assembled. This paper shows how the same technique can be extended to solve the problem of assembling multiple test forms with possibly varying specifications the same time. The example produced forms that not only met all of their constraints but also had excellent information functions. This result is not only due to optimization but also to the quality of the item pool from the LSAT. It should be noted that the quality of an item pool depends not so much on its size as on the degree to which the pool has items with attribute values that are "on target"; that is, they meet the constraints and allow for the right distribution of information on the ability scale. General recommendations on the size of item pools can therefore not be given.

References

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

Adema, J. J. (1990). The construction of customized two-staged tests. *Journal of Educational Measurement, 27,* 241–253.

Adema, J. J. (1992). Methods and models for the construction of weakly parallel tests. *Applied Psychological Measurement, 16*, 53–63.

Armstrong, R. D., & Jones, D. H. (1992) Polynominal algorithms for item matching. *Applied Psychological Measurement, 16* (4), 365–371.

Armstrong, R. D., Jones, D. H., & Wang, Z. (1994). Automated parallel test construction using classical test theory. *Journal of Educational Statistics, 19,* 73–90.

Armstrong, R. D., Jones, D. H., & Wu, I.-L. (1992). An automated test development of parallel tests. *Psychometrika, 57,* 271–288.

Boekkooi-Timminga, E. (1987). Simultaneous test construction by zero-one programming. *Methodika, 1*, 1101–1112.

Boekkooi-Timminga, E. (1990). The construction of parallel tests from IRT-based item banks. *Journal of Educational Statistics, 15*, 129–145.

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

Luecht, R. M. (1998). Computer-assisted test assembly using optimization heuristics. *Applied Psychological Measurement, 22,* 224–236.

Timminga, E., van der Linden, W. J., & Schweizer, D. A. (1996). ConTEST [Computer program and manual]. Groningen, The Netherlands: iec ProGAMMA.

van der Linden, W. J. (1998). Optimal assembly of educational and psychological tests, with a bibliography [Special Issue]. *Applied Psychological Measurement, 22* (3).

van der Linden, W. J. (2000). Optimal assembly of tests with item sets. *Applied Psychological Measurement, 24* (3), 225–240.

van der Linden, W. J., & Boekkooi-Timminga, E. (1989). A maximin model for test design with practical constraints. *Psychometrika, 54,* 237–247.

van der Linden, W. J., & Boekkooi-Timminga, E. (1988). A zero-one programming approach to Gulliksen's matched random subsets method. *Applied Psychological Measurement, 12*, 201–209.

Wainer, H., & Kiely, G. L (1987). Item clusters and computerized adaptive testing: A case for testlets. *Journal of Educational Measurement, 24*, 185–201.