

## Automated Test-Form Generation

Wim J. van der Linden and Qi Diao

CTB/McGraw-Hill

*In automated test assembly (ATA), the methodology of mixed-integer programming is used to select test items from an item bank to meet the specifications for a desired test form and optimize its measurement accuracy. The same methodology can be used to automate the formatting of the set of selected items into the actual test form. Three different cases are discussed: (i) computerized test forms in which the items are presented on a screen one at a time and only their optimal order has to be determined; (ii) paper forms in which the items need to be ordered and paginated and the typical goal is to minimize paper use; and (iii) published test forms with the same requirements but a more sophisticated layout (e.g., double-column print). For each case, a menu of possible test-form specifications is identified, and it is shown how they can be modeled as linear constraints using 0–1 decision variables. The methodology is demonstrated using two empirical examples.*

Test development is a comprehensive process involving the typical stages of item-bank design, item writing, field testing, and calibration, as well as test assembly and test-form generation. Guidelines for the various activities in these stages can be found, for instance, in Downing and Haladyna (2006) and Schmeiser and Welch (2006).

Any step to automation of these stages is welcome, especially when it leads to optimization of the results and reduction of the amount of work. An area where such automation has made important progress lately is test assembly. Automated test assembly (ATA) involves the selection of the items for one or more test forms from an item bank subject to sets of content and statistical specifications while optimizing their measurement properties. The automation has become possible through the application of the methodology of mixed-integer programming (MIP). The core of the application consists of the definition of 0–1 decision variables for the selection of the items, the modeling of the test specifications as a set of constraints with an objective function that is linear in the variables, and the use of a software package with an MIP solver to find the optimal solution, which is the combination of 0s and 1s for the variables for the items that identifies the optimal set in the bank. The use of MIP is extensively reviewed for a wide variety of test assembly problems in van der Linden (2005, chaps. 1–9). In addition, introductions to ATA and a few worked examples can be found in Cor, Alves, and Gierl (2008, 2009). For readers not familiar with the methodology, a brief example is given in the Appendix.

The same methodology can be used to design item banks and manage the process of item writing. The process then involves the definition of a design space for the items (i.e., the Cartesian product of all relevant item attributes) and the introduction

of integer variables for the design points that identify the numbers of items to be written for each of them. The program of test forms that the item bank has to serve defines the set of constraints to be imposed on the item bank optimization. The optimal solution is a blueprint for the item bank, which can be used to manage the item-writing process. For examples of this application, see Ariel, van der Linden, and Veldkamp (2006) and van der Linden (2005, chaps. 10–11).

The goal of this research was to assess the practicality of using the same methodology of MIP for automating the very last stage of test development—the generation of the actual test forms. Test form generation typically involves (i) the ordering of the previously selected items and (ii) formatting their appearance in the form. If the test is to be delivered on computer, its format simply is that of presenting the items in their optimal order on the computer screen, one item at a time. For informal paper-and-pencil testing (e.g., classroom testing), the formatting involves the organization of the form into a set of printed pages. For published test forms, the process is more complicated and likely to lead to the necessity of going back and forth between the processes of ordering the items and formatting the forms until the required layout is realized. Although modern testing technology has led to a large variety of modes of test administration (e.g., Drasgow, Luecht, & Bennett, 2006), it is believed that these three cases capture the essence of their variety.

The activities of item ordering and test-form formatting are usually controlled by several new specifications in addition to the content and statistical specifications that were imposed on the item-selection process. For instance, the item order may have to meet specifications that imply an increase or decrease of certain quantitative item attributes (such as item difficulty and expected response time) or require some categories of items to precede others (e.g., multiple-choice items before constructed-response items). These specifications are likely to conflict, and then an optimal compromise has to be found. The task becomes even more complicated when additional order restrictions are to be satisfied, such as that sequences of items with the same answer key beyond a certain length be excluded, or that the positions of the items in the form approximate their positions during the pretest. On top of this, test forms frequently have sets of items organized around common stimuli. Several of these specifications may then have to hold both with respect to the stimuli and the items in the form.

Unlike test forms delivered on a computer, paper forms need to be organized as a set of pages. One obvious constraint is to avoid page breaks in the middle of an item. Another is to keep item sets together with their stimuli to avoid back-and-forth thumbing. For published paper forms, such requirements can be expected to be stricter and accompanied by rules that control other layout issues, such as the possibility of double-column print. Also, for such forms, minimization of the total number of pages can lead to substantial savings, not only in the costs of printing but also of shipping and scanning of the returned forms. This holds especially when the forms are printed in signatures and the necessity of an extra signature for only a few last items can be avoided.

Automation of this rather complex process of test-form generation is welcome because it reduces the tedious labor involved in dealing with a nontrivial set of constraints while optimizing an objective function for the form. It also brings us closer

to the ideal of on-the-fly, fully automated test-form generation directly from a computerized item bank (e.g., Folk & Smith, 2002).

In the remainder of the paper, we present a menu of possible constraints that will be helpful when modeling test-form generation as an optimization problem and then discuss their use in the three cases distinguished above: computerized testing in which the items only need to be ordered, paper-and-pencil testing with additional page optimization, and published test forms with professional layout specifications. The paper concludes with two examples that demonstrate the use of the methodology.

### Test-Form Specifications

Let  $i = 1, \dots, n$  denote the items selected for the test form. In addition, we use  $r = 1, \dots, n$  to denote the positions available for the items in the test form.

The decision variables are defined as

$$x_{ir} = \begin{cases} 1 & \text{if item } i \text{ is assigned position } r, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We use these variables to model a menu of possible constraints, but first need to introduce two technical constraints that are always assumed to be part of the model.

#### Technical Constraints

Each item should be assigned only one position. Hence,

$$\sum_{r=1}^n x_{ir} = 1, \quad i = 1, \dots, n. \quad (2)$$

Likewise, each position should be assigned to only one item. Thus,

$$\sum_{i=1}^n x_{ir} = 1, \quad r = 1, \dots, n. \quad (3)$$

These constraints are not only necessary to avoid inconsistent assignments of items but also to guarantee the validity of some of the constraints introduced in the next sections.

#### Item Order

The items in the test form can be ordered both by quantitative (difficulty; expected response time; etc.) and categorical item attributes (content category; item format; etc.).

Let  $q_i$  denote the value of a positive-valued quantitative attribute for item  $i$  and suppose the test form has to display an increasing order of  $q_i$ . However, such a requirement is bound to conflict with an order to be imposed for another attribute. We therefore introduce a tolerance  $\delta_q \geq 0$  defined as an upper bound on the decrease

admitted between two subsequent items. The upper bound is guaranteed by the set of constraints

$$\sum_{i=1}^n q_i x_{ir} - \sum_{i=1}^n q_i x_{i(r+1)} < \delta_q, \quad r = 1, \dots, n-1. \quad (4)$$

A strict order for  $q_i$  is obtained if we set  $\delta_q = 0$ . Observe that the tolerance is indexed by  $q$  to allow for possible different weighing and/or scale differences between multiple quantitative attributes. Also, note the presence of the sums over  $i$  in these constraints. Because of (3), each position is assigned only to one of the items in the sum. We can therefore use this sum operation to simultaneously deal with all items for each position.

As for the ordering of the items by categorical attributes, let  $c = 1, \dots, C$  be the attributes in their required order. Besides, we will use  $V_c$  to denote the set of indices of the items with attribute  $c$ , and  $n_c$  for the size of this set. Finally, let  $W_c$  be the set of adjacent positions defined as  $(\sum_{k=1}^{c-1} n_k + 1, \dots, \sum_{k=1}^c n_k)$ , where the first sum is assumed to be equal to zero for  $c = 1$ . The order is ensured by the set of constraints

$$\sum_{i \in V_c} \sum_{r \in W_c} x_{ir} = n_c, \quad c = 1, \dots, C-1. \quad (5)$$

The same type of categorical constraint can be used to realize a test form with a section structure, for instance, one with sections of items for different chapters in a text book. The same type of constraint is required when the items should have positions in the test form comparable to their pretest positions (e.g., items in different thirds of the form should match the thirds during pretesting, etc.).

Observe that the constraints in (5) order the different categories strictly whereas those in (4) admit a tolerance. Our motivation for this choice is that it is easy to impose constraints on the selection of the items from the item bank to ensure the numbers of attributes  $n_c$  in (5) for the selected items. Also, for forms organized by topic, for instance, the occurrence of a geometry item in the middle of a set of algebra items is generally more annoying than minor deviations from their difficulty order.

### Answer Key Sequences

Test forms with longer sequences of identical answer keys are usually avoided because they may create uncertainty with the test takers or stimulate them to look for response strategies. Let  $a = 1, \dots, A$  denote the answer keys,  $V_a$  the set of item indices with key  $a$ , and  $n_a$  the maximum length of the sequence of items with this key. To avoid any sequences longer than  $n_a$ , the required set of constraints is

$$\sum_{i \in V_a} (x_{ir} + \dots + x_{i(r+n_a)}) \leq n_a, \quad r = 1, \dots, n - n_a + 1; \quad a = 1, \dots, A. \quad (6)$$

## Item Sets

We treat stimuli in item sets as another kind of entity to be assigned a position in the test form. Let  $s = 1, \dots, S$  be the index of stimuli for the test form. The items in the set for stimulus  $s$  are denoted as  $i_s = 1_s, \dots, n_s$ . It holds that  $\sum_s n_s = n$ . The total number of entities to be assigned a position is equal to  $n + S$ .

It is convenient to use additional decision variables

$$z_{sr} = \begin{cases} 1 & \text{if stimulus } s \text{ is assigned position } r, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

In addition to (2), we need the technical constraints

$$\sum_{r=1}^{n_s} z_{sr} = 1, \quad s = 1, \dots, S, \quad (8)$$

whereas (3) needs to be complemented as

$$\sum_{i_s=1}^{n_s} x_{i_s r} + \sum_{s=1}^n z_{sr} = 1, \quad r = 1, \dots, n + S. \quad (9)$$

Generally, if a stimulus receives position  $r$ , its items should be assigned positions  $r + 1, \dots, r + n_s + 1$ . This feature is guaranteed by the following set of constraints

$$\sum_{i_s=1}^{n_s} \sum_{k=r+1}^{r+n_s+1} x_{i_s k} - n_s z_{sr} = 0, \quad s = 1, \dots, S; r = 1, \dots, n + S - n_s. \quad (10)$$

Observe that these constraints also impose the reverse requirement; they lead to the assignment of the items from a set with stimulus  $s$  to positions  $r + 1, \dots, r + n_s + 1$  only when  $s$  is assigned position  $r$ .

Stimuli typically have only categorical attributes. An order on such attributes can be imposed using constraints analogous to (5). Suppose that  $c = 1, \dots, C$  now represent content categories for the stimuli in their required order,  $V_c^{(\text{stim})}$  is the set of the indices for the stimuli in category  $c$ , and  $n_c^{(\text{stim})}$  is the size of  $V_c^{(\text{stim})}$ . Besides,  $W_c^{(\text{stim})}$  now denotes the set of adjacent positions defined by  $\{\sum_{k=1}^{c-1} n_k^{(\text{stim})} + s, \dots, \sum_{k=1}^c n_k^{(\text{stim})} + s - 1\}$ , where the first sum is assumed to be equal to zero for  $c = 1$ . The following set of constraints imposes the required order:

$$\sum_{s \in V_c^{(\text{stim})}} \sum_{r \in W_c} z_{sr} = n_c^{(\text{stim})}, \quad c = 1, \dots, C - 1. \quad (11)$$

Analogous to (4), the items in the sets are ordered with respect to a (positive-valued) quantitative attribute  $q$  by

$$\sum_{i_s=1}^{n_s} q_{i_s} x_{i_s r} - \sum_{i_s=1}^{n_s} q_{i_s} x_{i_s(r+1)} < \delta_q, \quad r = 1, \dots, n + S; \quad s = 1, \dots, S. \quad (12)$$

### Computer Forms

In this mode of testing case, a computer screen needs to be fed one item at a time. The items only need to be ordered with respect to quantitative and categorical attributes. If the form has an item-set structure, the stimuli have to be ordered as well.

As an example, suppose the items have to be ordered with respect to two quantitative attributes,  $q^{(1)}$  and  $q^{(2)}$  and one categorical attribute with categories  $c = 1, \dots, C$ . The following model uses an objective function based on the minimax criterion to bring the two quantitative orders as closely as possible to their respective ideals:

$$\text{minimize } y \quad (13)$$

subject to

$$\sum_{i=1}^n q_i^{(1)} x_{ir} - \sum_{i=1}^n q_i^{(1)} x_{i(r+1)} < \delta_1 y, \quad r = 1, \dots, n - 1, \quad (14)$$

$$\sum_{i=1}^n q_i^{(2)} x_{ir} - \sum_{i=1}^n q_i^{(2)} x_{i(r+1)} < \delta_2 y, \quad r = 1, \dots, n - 1, \quad (15)$$

$$\sum_{i \in V_c} \sum_{r \in W_c} x_{ir} = n_c, \quad c = 1, \dots, C - 1, \quad (16)$$

$$y \geq 0. \quad (17)$$

The quantitative order constraints in (14) and (15) have a new (real-valued) decision variable  $y \geq 0$  as a common factor in their right-hand sides. The tolerances  $\delta_1$  and  $\delta_2$  are to weigh the relative importance of each of the two quantitative attributes and/or to adjust for scale differences between them. (Without the latter, the attribute with the largest range of values would always be dominant.) The minimization of  $y$  in (13) assigns the items to positions that reflect their ideal order for the two attributes as closely as possible, subject to the relative weights.

To make this basic model more realistic, an appropriate selection from the earlier constraints in (5)–(14) should be added to it. In addition, the model should always contain the technical constraints in (2) and (3).

## Paper Forms

Paper forms are more complicated because we need to observe the page breaks and want to optimize the use of space.

In order to formulate a basic example, a new index  $p = 1, \dots, P$  is introduced to denote the pages in the test form, where  $P$  should be chosen large enough to contain the solution. Suppose that the net space available for items on page  $p$  consist of  $L_p$  lines. Let  $l_i$  be the number of lines needed to print item  $i$  (including all graphics, white space, etc.). In addition, we need new decision variables

$$z_p = \begin{cases} 1 & \text{if any item is assigned to page } p, \\ 0 & \text{otherwise,} \end{cases} \quad (18)$$

and

$$x_{irp} = \begin{cases} 1 & \text{if item } i \text{ is assigned position } r \text{ on page } p, \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

As objective, we choose to minimize the number of pages, with limits  $L_p$  for the amount of space available per page. Because of these limits, minimizing the number of pages automatically results in maximum use of space on the printed pages.

As the optimization model can have no further objectives, we replace decision variable  $y$  in the bounds on the quantitative orders in (14)–(15) by a common constant  $d$ . The idea is to run the model for a series of well-chosen values for  $d$  until a feasible solution for a satisfactory low value is obtained.

The following model is given as an example:

$$\text{minimize } \sum_{p=1}^P z_p \quad (20)$$

subject to

$$z_{p+1} - z_p \leq 0, \quad p = 1, \dots, P - 1, \quad (21)$$

$$\sum_{i=1}^n \sum_{r=1}^n l_i x_{irp} \leq z_p L_p, \quad p = 1, \dots, P, \quad (22)$$

$$\sum_{r=1}^n \sum_{p=1}^P x_{irp} = 1, \quad i = 1, \dots, n, \quad (23)$$

$$\sum_{i=1}^n \sum_{p=1}^P x_{irp} = 1, \quad r = 1, \dots, n, \quad (24)$$

$$\sum_{i=1}^n \sum_{p=1}^P p x_{irp} - \sum_{i=1}^n \sum_{p=1}^P p x_{i(r+1)p} \leq 0, \quad r = 1, \dots, n-1, \quad (25)$$

$$\sum_{i=1}^n \sum_{p=1}^P q_i^{(1)} x_{irp} - \sum_{i=1}^n \sum_{p=1}^P q_i^{(1)} x_{i(r+1)p} \leq \delta_1 d, \quad r = 1, \dots, n-1, \quad (26)$$

$$\sum_{i=1}^n \sum_{p=1}^P q_i^{(2)} x_{irp} - \sum_{i=1}^n \sum_{p=1}^P q_i^{(2)} x_{i(r+1)p} \leq \delta_2 d, \quad r = 1, \dots, n-1, \quad (27)$$

$$\sum_{r=1}^R \sum_{p=1}^P (r_i - r_i^{(\text{pre})}) x_{irp} \leq \delta_r \quad i = 1, \dots, I, \quad (28)$$

$$\sum_{r=1}^R \sum_{p=1}^P (r_i^{(\text{pre})} - r_i) x_{irp} \leq \delta_r \quad i = 1, \dots, I, \quad (29)$$

$$\sum_{r=1}^n x_{irp} = 0, \quad (i, p) \in V_{ip}, \quad (30)$$

$$z_p \in \{0, 1\}, \quad p = 1, \dots, P, \quad (31)$$

$$x_{irp} \in \{0, 1\}, \quad i = 1, \dots, n; r = 1, \dots, n; p = 1, \dots, P. \quad (32)$$

Clearly, the objective function in (20) minimizes the number of pages used. The constraints in (21) stop the selection of any higher page numbers as soon as a page has been unused. The constraints in (22) limit the available space to  $L_p$  lines per page. They also keep the variables for the assignment of items to positions and pages consistent: due to the presence of variable  $z_p$  in their right-hand side, the bound on the number of lines available per page is automatically set to zero if none of the items are assigned to the page.

The next two sets of constraints in (23) and (24) are new versions of the earlier technical constraints in (2) and (3) required to assign unique positions to the items and vice versa. Because of (24), the sums in (25) return the page number to which positions  $r$  and  $r + 1$  are assigned, respectively. These constraints thus keep the positions and page numbers consistent by ensuring that no lower positions are assigned to higher page numbers.

The constraints in (26) and (27) are those in (14) and (15) with the decision variable in the upper bounds replaced by a constant,  $d$ , whereas (28) and (29) are required when the positions of the items in the new form should not differ more than  $\delta_r$  positions from their positions during pretesting,  $r_i^{(\text{pre})}$ .

The constraints in (30) can be used to avoid the assignment of specific items to specific pages. The combinations of items and pages for which this should hold are defined as the set  $V_{ip}$  of ordered pairs  $(i, p)$ . These constraints are helpful, for instance, when the test form has item sets and we do not want these sets to begin with a stimulus on a right-hand page to prevent thumbing back and forth between the stimulus and the items toward the end of the set.

The last two sets of constraints define the range of the decision variables. Observe that the 0-1 nature of the variables for the items automatically avoids page breaks in the middle of an item.

Finally, to become realistic, the model should be extended with the earlier constraints on answer key sequences, categorical orders, specific orders of items within sets, etc., whenever appropriate.

Observe that the model is infeasible for values of  $d$  chosen too small. Such outcomes are reported immediately by the solver. It makes sense therefore to begin with a low value for  $d$  and increase the value until a feasible solution is obtained. In fact, it may be profitable to run the model for a few extra values and check the results. Because a tradeoff exists between  $d$  and the value of the objective function in (20), a solution for a slightly larger value of  $d$  than necessary may result in the removal of one extra page (or even an entire signature when this page is critical).

### Published Forms

Published test forms usually require the satisfaction of additional professional printing specifications. For instance, some of the pages in the test form may need embellishments or instructions. Such features are easily realized by absorbing their space requirements into the measures of the numbers of lines available per page in the right-hand sides of (22),  $L_p$ .

Also, some of items may require printing in double columns. In order to deal with such features, we assume that the measures  $l_i$  of the size of the items are already for double-column layout.

If the whole form has to have this layout, we simply apply the previous model in (20)–(32) using  $p$  to denote individual columns rather than pages. If only a subset of the items has to be printed in double columns, it makes sense to first form a block of items with this feature and then position the block among the regular items in the form. Again, such blocks can be prepared using the same style of optimization.

The decision variables are now for the assignment of positions to the items in a column of the block,

$$x_{irh} = \begin{cases} 1 & \text{if item } i \text{ is assigned position } r \text{ in column } h, \\ 0 & \text{otherwise.} \end{cases} \quad (33)$$

$i = 1, \dots, n_d$ ,  $r = 1, \dots, n_d$ , and  $h = 1, 2$ , where  $n_d$  is the total number of items that require double-column print.

Generally, we want a block of minimum length. This is created by the solution of the model

$$\text{minimize } y \quad (34)$$

subject to

$$\sum_{i=1}^{n_d} \sum_{r=1}^{n_d} l_i x_{ir1} \leq y, \quad (35)$$

$$\sum_{i=1}^{n_d} \sum_{r=1}^{n_d} l_i x_{ir2} \leq y, \quad (36)$$

$$\sum_{h=1}^2 \sum_{r=1}^{n_d} x_{irh} = 1, \quad i = 1, \dots, n_d, \quad (37)$$

$$\sum_{h=1}^2 \sum_{i=1}^{n_d} x_{irh} = 1, \quad r = 1, \dots, n_d, \quad (38)$$

$$\sum_{h=1}^2 \sum_{i=1}^{n_d} h x_{irh} - \sum_{h=1}^2 \sum_{i=1}^{n_d} h x_{i(r+1)h} \leq 0, \quad r = 1, \dots, n_d - 1, \quad (39)$$

$$x_{irh} \in \{0, 1\}, \quad i = 1, \dots, n_d; \quad r = 1, \dots, n_d; \quad h = 1, 2. \quad (40)$$

The combination of the objective function and the first two constraints amounts to the application of a minimax criterion to the length of the two columns. The criterion ensures two columns of approximately the same minimal length. The constraints in (37) and (38) are again required to assign unique positions to items and vice versa. Finally, analogous to (25), the set of constraints in (39) keeps the assignment of

positions and columns consistent by ensuring that the right-hand column does not receive any lower positions than the left-hand column.

Of course, additional constraints may have to be added to the basic model in (34)–(40) to order the items in the columns appropriately.

The main assembly of the test form then uses the model in (20)–(32) with the block as a regular item. If the block has to be longer than one page, we should reformulate the model for the case of four ordered columns (the first two for one page and the last two for the next), with a constraint on the length of the first two columns and (34)–(36) replaced by a minimax criterion simultaneously over the size of last two columns and the difference between the actual size and the space available for the first two columns.

### Alternative Approach

So far, the entire process of test-form production was assumed to consist of separate steps of item selection and test-form formatting. However, in principle, it is possible to integrate the two steps into a single step with only one optimization model. The decision variables are then used to assign a selection of the items in the pool directly to the available positions in the test form. The necessary change of variables only involves a redefinition of the ranges of the indices in (1). That is, for an item bank with  $I$  items, we still have

$$x_{ir} = \begin{cases} 1 & \text{if item } i \text{ is assigned position } r, \\ 0 & \text{otherwise,} \end{cases} \quad (41)$$

but now with indices  $i = 1, \dots, I$  and  $r = 1, \dots, n + 1$ . The extra position  $r = n + 1$  is a dummy assigned to all items that do not make it to the test form.

These variables are used to formulate a model that includes both the constraints for the usual item selection process (see the Appendix) and the appropriate selection from the test-formatting constraints in the preceding sections. The presence of the dummy position requires some special attention to some of these constraints. For instance, the constraint that all items be assigned a unique position in (3) should hold for the regular positions but not for the dummy.

From an optimization point of view, joint modeling of both types of problems is advantageous and generally leads to better results. That is, as the test form is generated directly from the bank, it may very well be possible to find a feasible paper form with fewer pages than when separating the stages of item selection and formatting. However, the size of the resulting model quickly becomes prohibitive. For a bank of 2,000 items, say, and a test form of 80 items—both not unusual numbers in test publishing—the number of variables is already equal to 160,000. Also, the numbers of constraints of certain types may quickly become quite large. Therefore, this alternative approach seems attractive only for combinations of smaller item banks and/or shorter test forms.

### Empirical Examples

The methodology is demonstrated for two different cases: a paper form and a computerized form. The test was a 31-item mathematics test from a large-scale testing

program. The form had already been selected manually by content specialists using the specifications in force for the program. The test was chosen only because it nicely illustrates a typical level of complexity involved in test-form generation; the choice of another example may have led to another selection from the earlier menu of possible constraints.

The two test forms had to be formatted using the following additional specifications:

1. The items in the forms belonged to five different reporting categories: numbers and operations (N&O); algebraic concepts (AC); geometry (G); measurement (M); and data analysis and probability (DA&P). The order in which the categories could appear was free. The only constraint was a maximum length for sequences of items from the same category. The maximum had to be set at four.
2. Likewise, the maximum length of sequences of items with the same answer key (A, B, C, and D) had to be set at three.
3. As the forms had to begin with relatively easy items, the  $p$ -values of the first five items were constrained to be greater than .6.
4. Each item had to be in a position in the forms differing not too much from its pretest position. The maximum difference had to be set at six.

In addition, the following specifications had to be met for the paper form:

1. For each item the number of lines required to print it was known. The total number of lines per page had to be constrained at 54 lines.
2. The number of pages for the paper form had to be minimized. Because the total number of lines for the 31 items was 601 lines (=11.13 pages), we already knew that the best solution could never be a paper form with fewer than 12 pages.

The optimization model for the paper form had the basic structure in (20)–(32) with the additional four sets of specifications above added. The model had a total of 14,430 variables and 9,302 constraints.

In order to make the results for the two examples comparable, the same model was used for the computerized form with the exception of the structure in (20)–(32). The removal of all variables and constraints related to page minimization led to a great reduction of the size of the model. The numbers of variables and constraints was now equal to 961 and 376, respectively. As there were no other clear objectives for this form, we maximized the sum of all variables  $x_{ir}$ . The choice did not have any undesirable impact on the solution because the number of variables with a value equal to one in the solution was already fixed by the technical constraints in (2) and (3).

Both models were run using the integer solver in *IBM ILOG OPL Version 6.3* (International Business Machines Corporation, 2009), which offers a convenient modeling environment for constrained optimization problems and uses the same solvers as in *CPLEX 12.1*. The model for the computerized form was run using the default settings. The running time was less than one second. The model for the paper form was run with the relative MIP gap tolerance set equal to .1, which means that it would stop as soon as it found a solution known to be within 10% from optimal. The running time was 206 s. The solution consisted of a paper form with 12 pages (see

Table 1  
*Summary of Results for Computerized Test Form*

Position	Item ID	Content Category	Answer Key	p-Value	Difference in Position
1	2	N&O	C	.61	2
2	4	DA&P	B	.91	3
3	6	M	C	.77	6
4	11	DA&P	D	.67	6
5	1	G	A	.75	-3
6	3	G	D	.39	-1
7	8	M	C	.50	1
8	7	M	D	.67	-1
9	10	G	A	.79	3
10	13	M	B	.33	6
11	12	N&O	A	.59	2
12	14	DA&P	B	.77	1
13	5	N&O	D	.89	-4
14	9	N&O	D	.69	-2
15	17	G	A	.63	3
16	15	DA&P	D	.74	1
17	19	N&O	C	.50	3
18	16	N&O	D	.87	-1
19	22	N&O	B	.46	6
20	18	AC	D	.84	-1
21	23	N&O	B	.85	6
22	25	G	D	.55	6
23	20	G	C	.63	2
24	27	DA&P	C	.82	6
25	21	M	A	.85	0
26	26	AC	B	.90	3
27	24	AC	D	.63	3
28	29	N&O	C	.75	6
29	28	DA&P	A	.57	6
30	30	AC	C	.67	6
31	31	AC	C	.43	5

Table 2), which, as already indicated, was the minimum number of pages possible given the total number of lines required by the items. This result illustrates an important feature of the implicit enumeration process required to solve MIP problems: the solution may already be found early in the process but, to guarantee its optimality, the iterative process has to be continued until all feasible solutions have been checked. Use of the gap tolerance parameter and checking the result helps to prevent unnecessary long processes.

The two solutions that were found are summarized in Tables 1 and 2. As is easy to verify, both satisfied the bounds in each of the constraints. The fact that the two forms show several items with positions that are close is a common effect of these constraints. On the other hand, although the number of differences looks small, their

Table 2  
*Summary of Results for Paper Test Form*

Position	Item ID	Content Category	Answer Key	p-Value	Difference in Position	Page Number	No. of Lines
1	2	N&O	C	.61	2	1	15
2	7	M	D	.67	5	1	13
3	6	M	C	.77	6	1	25
4	4	DA&P	B	.91	1	2	52
5	1	G	A	.75	-3	3	25
6	8	M	C	.50	2	3	21
7	3	G	D	.39	-2	4	28
8	11	DA&P	D	.67	2	4	16
9	5	N&O	D	.89	0	4	10
10	12	N&O	A	.59	3	5	10
11	16	N&O	D	.87	6	5	41
12	9	N&O	D	.69	0	6	10
13	13	M	B	.33	3	6	10
14	14	DA&P	B	.77	-1	6	21
15	10	G	A	.79	-3	6	10
16	19	N&O	C	.50	4	7	10
17	18	AC	D	.84	2	7	15
18	17	G	A	.63	0	7	16
19	22	N&O	B	.46	6	7	10
20	21	M	A	.85	5	8	21
21	15	DA&P	D	.74	-4	8	12
22	23	N&O	B	.85	5	8	19
23	25	G	D	.55	5	9	23
24	20	G	C	.63	1	9	24
25	27	DA&P	C	.82	5	10	30
26	26	AC	B	.90	3	10	19
27	24	AC	D	.63	3	11	34
28	29	N&O	C	.75	6	11	10
29	28	DA&P	A	.57	6	12	15
30	31	AC	C	.43	6	12	21
31	30	AC	C	.67	5	12	15

combined effect on the optimal value for objective function for the paper form is substantial. If we would paginate the computerized form in Table 1 using the same criterion of maximal 54 lines per page, the result would be a total number of pages equal to 14 instead of 12.

As already noted, a paper version of the test form had already been prepared manually to satisfy the same specifications. As the item IDs in Tables 1 and 2 reflect their position in this original form, it is immediately possible to evaluate where the use of the optimization model has led to a different decision.

## Conclusion

The important steps of (i) ordering the items according to content categories, statistical parameters, and other relevant attributes and (ii) additional formatting of the final test form were still missing in the current trend toward automation of test-form production. This paper showed that the same methodology of MIP modeling and use of an appropriate solver from a standard LP software package that has become popular for selecting items from a bank can be used to automate these final steps as well. The examples showed the steps required to use the methodology in a practical application.

The solver used in the examples was a high-end commercial solver already in use by some of the testing companies. However, cheaper alternatives are rapidly becoming available. Two quite affordable solvers available as add-ins for *Microsoft Excel* have recently been evaluated for different types of test assembly in Cor et al. (2008, 2009). Also, a new, much more powerful version of the classical solver *lp\_solve* (version 5.5) is now available as freeware. For an evaluation of its use for different types of test assembly problems using the *lpSolveAPI* interface from *R* (also freeware), see Diao and van der Linden (2011).

## Appendix: Test Assembly as MIP Modeling

Each possible test from the bank of  $I$  items is labeled using the string of 0–1 variables  $x_i$ ,  $i = 1, \dots, I$ , which take the value 1 if item  $i$  is included in the test and the value 0 if it is not. The variables are used to formulate the test specifications as a constraint or the objective function for the selection of the test. Constraints impose an upper or lower bound on an attribute of the test whereas the objective function requires the attribute to take a maximum or minimum value possible for the item bank given the constraints. Because constraints offer precise control of the attributes, they are the main vehicles for formulating test specifications. The objective function is typically chosen for technical reasons, or for an attribute that is less important.

Test specifications can be formulated as three different types of constraints:

1. Quantitative constraints impose bounds on attributes that take numerical values, such as statistical parameters, word counts, test information, and item-exposure rates. In the standard model below, we use  $q_i$  as a generic symbol for the value of item  $i$  on a quantitative attribute and  $q_b$  for a bound on this attribute.
2. Categorical constraints have the distinctive feature of imposing bounds on the numbers of items from certain categories in the bank. Examples of these categories are content categories, cognitive level of the items, strategies required to solve the item, and item type or format. In the standard model, we let  $V_c$  be a generic symbol for the subset of items in the bank that belong to category  $c$ . A bound on the number of items from this set is denoted as  $n_c$ .
3. Logical constraints are required to specify the logical structure of the test, for example, the organization of the test into item sets with a common stimulus. Another example of a logical constraint is that of exclusion between items, for example, because they clue each other's solution. We use  $V_e$  to denote a set of items that has this relation of exclusion.

Observe that each of these constraints can be applied at any level in the test (e.g., all items, a section, or multiple test forms). An example of a model for a single test from a bank of discrete items with specifications at different levels is

$$\text{optimize } \sum_{i=1}^I q_i x_i \text{ (objective)} \quad (\text{A1})$$

subject to the constraints at:

**Test level**

$$\sum_{i=1}^I x_i \geq n, \text{ (test length)} \quad (\text{A2})$$

$$\sum_{i \in V_c} x_i \geq n_c, \text{ for all } c, \text{ (categorical constraints)} \quad (\text{A3})$$

$$\sum_{i=1}^I q_i x_i \geq b_q, \text{ (quantitative constraints)} \quad (\text{A4})$$

**Item level**

$$\sum_{i \in V_1} x_i = n_1, \text{ (categorical constraints)} \quad (\text{A5})$$

$$\sum_{i \in V_0} x_i = 0, \text{ (categorical constraints)} \quad (\text{A6})$$

$$q_i x_i \leq b_q^{\max}, \text{ for all } i, \text{ (quantitative constraints)} \quad (\text{A7})$$

$$b_q^{\min} x_i \leq q_i, \text{ for all } i, \text{ (quantitative constraints)} \quad (\text{A8})$$

$$\sum_{i \in V_e} x_i \leq 1, \text{ for all } e, \text{ (logical constraints)} \quad (\text{A9})$$

### Definition of variables

$$x_i \in \{0, 1\}, \text{ for all } i. (\text{range of variables}), \quad (\text{A10})$$

where  $\geq$  indicates the choice of an equality or an inequality sign, and  $V_0$  and  $V_1$  are generic symbols for sets of items in the bank with a required or forbidden categorical attribute, respectively.

In specific applications, the model will have multiple versions of each of these constraints. For a real-world testing program, the total number of constraints easily runs into the hundreds. Also, the model has to be adjusted when it is to be used for the simultaneous assembly of a set of  $T$  test forms. The variables  $x_i$  are then replaced by  $x_{it}$ ,  $i = 1, \dots, I$  and  $t = 1, \dots, T$ , with  $x_{ij} = 1$  if item  $i$  is included in test form  $t$  and  $x_{ij} = 0$  otherwise. Besides, extra constraints are added to the model to control the inclusion of the same item in multiple test forms.

### References

- Ariel, A., van der Linden, W. J., & Veldkamp, B. P. (2006). A strategy for optimizing item-pool management. *Journal of Educational Measurement*, *43*, 85–96.
- Cor, K., Alves, C., & Gierl, M. (2008). Conducting automated test assembly using the Premium Solver Platform Version 7.0 with Microsoft Excel and the large-Scale LP/QP solver engine add-in. *Applied Psychological Measurement*, *32*, 652–663.
- Cor, K., Alves, C., & Gierl, M. (2009). Three applications of automated test assembly within a user-friendly modeling environment. *Practical Assessment, Research, & Evaluation*, *14*(14). Available online: <http://pareonline.net/getvn.asp?v=14&n=14>.
- Diao, Q., & van der Linden, W. J. (2011). Automated test assembly using lp\_solve version 5.5 in R. *Applied Psychological Measurement*, *35*. In Press, OnlineFirst DOI: 10.1177/0146621610392211.
- Downing, S. M., & Haladyna, T. M. (2006). *Handbook of test development*. New York, NY: Routledge.
- Dragow, F., Luecht, R. M., & Bennett, R. E. (2006). Technology and testing. In R. L. Brennan (Ed.), *Educational measurement* (4th ed., pp. 471–515). Westport, CT: Praeger.
- Folk, V. G., & R. L. Smith (2002). Models for delivery of CBTs. In C. N. Mills, M. Potenza, J. J. Fremer, & W. Ward (Eds.), *Computer-based testing: Building the foundation for future assessments* (pp. 41–66). Hillsdale, NJ: Lawrence Erlbaum.
- International Business Machines Corporation (2009). *IBM ILOG OPL, Version 6.3* [Software program and manuals]. Armonk, NY: Author.
- Schmeiser, C. B., & Welch, C. J. (2006). Test development. In R. L. Brennan (Ed.), *Educational measurement* (4th ed., pp. 307–353). Westport, CT: Praeger.
- van der Linden, W. J. (2005). *Linear models for optimal test assembly*. New York, NY: Springer.

### Authors

WIM J. VAN DER LINDEN is Chief Research Scientist, CTB/McGraw-Hill, 20 Ryan Ranch Road, Monterey, CA 93940; [wim\\_vanderlinden@ctb.com](mailto:wim_vanderlinden@ctb.com). His primary research interests include test theory, applied statistics, and research methods.

QI DIAO is a Research Scientist, CTB/McGraw-Hill, 20 Ryan Ranch Road, Monterey, CA 93940; [qi\\_diao@ctb.com](mailto:qi_diao@ctb.com). Her primary research interests include adaptive testing, linear programming, and item response theory.