CONTRIBUTIONS TO TEST-ITEM BANK DESIGN AND MANAGEMENT

Adelaide Ariel

ISBN: 90-365-2249-8
Druk: PrintPartners Ipskamp B.V., Enschede

CONTRIBUTIONS TO
TEST-ITEM BANK DESIGN AND MANAGEMENT

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Twente, op gezag van
de rector magnificus, prof. dr. W. H. M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen op
vrijdag 23 september 2005
om 13.15 uur

door

Adelaide Ariel
geboren op 13 mei 1973
te Padang

Dit proefschrift is goedgekeurd door
de promotor prof.dr. Wim J. van der Linden,
de assistent promotor dr.ir. Bernard P. Veldkamp

Contents

1 Introduction

Psychological tests or examinations are tools to measure abilities, attitudes or skills of a person. Generally, tests are used to assess or to evaluate someone's performance in a specific subject. In educational measurement, one could think of, for example, tests for assessing someone's proficiency in a foreign language (Dutch as a Second Languange, NT2; Test of English as a Foreign Language, TOEFL), tests for higher education admission programs (Graduate Record Examination, GRE; Graduate Management Admission Test, GMAT; The Law School Admission Test, LSAT), or licensure tests for professionals (Certified Public Accountants, CPA; The United States Medical Licensing Examination, USML). In a broader sense, tests can be used to reveal someone's potential (e.g., IQ tests, personality tests, health tests), or to help evaluating and developing new psychological theories.

It is essential to standardize tests to enable fair comparison and evaluation. In practice, if a test possesses desirable content characteristics, it will be employed several times until there is a need to replace it. If this is the case, new items will be written and pretested. It is important to ensure that the old and the new version of the test are parallel.

The concept of standardized tests has emerged in the development of item response theory, IRT, (e.g., Hambleton, Swaminathan, & Rogers, 1991; Lord, 1980; Rasch, 1960; van der Linden & Hambleton, 1997). For this theory, different tests can be constructed, and even individualized to suit different ability levels, and still function as the same yardstick, as long as the items are from the same domain. IRT facilitates a single platform by parameterizing the items. As a result, tests constructed under IRT yield scores that are comparable between test takers.

The possibility to construct different tests has stimulated the practice of item banking. An item bank is a collection of items that are written, pretested, and calibrated. They might be replenished on a regular basis. It is important to ensure that the bank contains enough items for all content categories so that content specifications for the tests can be met. From an item bank, tests are assembled according to test specifications by using either mathematical programming techniques or heuristic methods.

This thesis deals with the need to maintain and to manage item banks. Items are not obtained for free; much efforts and resources are invested to develop and to calibrate items. Therefore, it is preferable to optimize item bank usage, without compromising the quality of the tests.

1.1 Item Response Theory (IRT)

IRT uses probabilistic models to relate the response of a person to an item through a function of the ability (or trait level, θ) of the person. Parameters in IRT models are divided into ability and item parameters. Various models exist to measure a single ability (unidimensional IRT) and multiple abilities (multidimensional IRT). In this thesis, we use the three-parameter logistic (3PL) unidimensional IRT. In the 3PL model, the probability of a response of an examinee to an item is explained by three item parameters, namely parameter for item discrimination, item difficulty, and guessing. The probability of an examinee answering item i correctly is defined as follows:

$$P_i(\theta) = c_i + (1 - c_i)\frac{\exp(a_i\theta + b_i)}{1 + \exp(a_i\theta + b_i)} \tag{1.1}$$

where a_i, b_i, and c_i are the discrimination, the difficulty, and the guessing parameters of item i, and θ is the ability of the examinee. Item discrimination indicates how well an item can distinguish low-ability from high-ability examinees. The higher the value of this parameter, the better the item separates low and high abilities. Item difficulty shows the difficulty level of an item, whereas item guessing denotes a probability of an examinee answering an item correctly through guessing.

Quantitative information can be obtained through a specific relationship between the item parameters and ability value of the examinee. This relationship is the item information function (IIF). In the 3PL model, an item can only give maximum information at one ability point. The higher the value of the item discrimination parameter, the higher the maximum item information will be. Figure 1.1 illustrates Equation 1.1 at item difficulty equal to zero ($b = 0$) with various discrimination and guessing values, and their item information function. Simply, an item that is too easy or too difficult for an examinee will provide little information. Measurement accuracy is directly related to item information.

1.2 Item Banking

Within the framework of IRT, one can estimate item parameters and ability independently. Specifically, item parameters are not group-dependent and ability

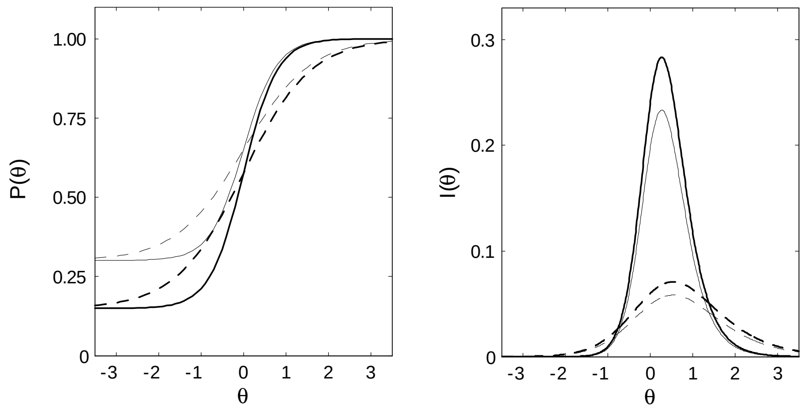

Figure 1.1: Probability curves for answering item i correctly (left) and their respective item information function (right), solid line represents high discrimination and dashed line low discriminacy whereas bold line low guessing and thin line high guessing parameter.

estimations are not test-dependent. Item parameter estimates obtained in different groups of examinees will be the same, except for sampling errors, and ability estimates obtained from different set of items will be the same, except for measurement error (Hambleton, Swaminathan, & Rogers, 1991). Because of these features, it is possible to design and construct tests to different specifications.

With these attractive benefits, for testing programs, it becomes practical to collect and to store items for present and later use. An item bank serves as a resource from which items are selected for their tests. An item bank normally contains specific additional information on the items such as their categorical attributes (e.g., item type), quantitative attributes (e.g., item parameters, word count, expected response time), and other information such as the date of administration and the frequency of use. A complete classification of item attributes is presented in van der Linden (2005, Chapter 2).

An item bank is not a static entity. No single item can be used forever; items may show parameter drift, become obsolete, or simply become known to candidate examinees (when they are used frequently). In testing programs, with tests administered on a continual basis, items are produced and pretested to gradually replace items in the bank (Way, Steffen, & Anderson, 1998). This process undoubtly causes fluctuations in the quality of the items in the bank. When items are retired temporarily or permanently, there is a challenge to produce new items with similar quality to replace them.

The task of developing new items is time consuming and the quality varies. It is therefore important to manage item usage carefully. To regulate item usage, we focus on the test assembly algorithms since they select which item will be in the test. To some extent, item usage is also influenced by the composition of the other items in the bank. We address the topic of test assembly and item-bank design in the following section.

1.3 Test Assembly

One major approach in test assembly is to use item information as a main criterium (Birnbaum, 1968) for selecting items. An important condition in this case is that the items fit the IRT model. It is desirable to obtain tests with high information in order to achieve measurement precision. Of course, the desired precision level depends on the goal of the tests. For example, for licensure tests, it is crucial to measure precisely near a cut-off ability level, to yield accurate pass-fail decisions.

In many modern testing programs, content specifications are also important; for example, test length, and the distribution of the items with respect to sets of categorical attributes. It is also important to consider bounds on quantitative attributes (e.g., word count or expected response time) of the items, or to exclude items that cue the answer of the other items. All of these requirements should be satisfied when the tests are assembled.

They can be addressed straighforwardly by using mathematical programming methods. Test content and the statistical requirements are the test constraints. The goal of mathematical programming is to find a set of solutions that meet a given set of constraints. Furthermore, mathematical programming methods aim to obtain an optimal solution, that is, the best solution with respect to the objective set by the methods. A mixed integer programming (MIP) model is suitable to formulate test assembly problem (Adema, 1990; van der Linden & Boekkooi-Timminga, 1989). Decision variables in such model are binary to denote whether or not an item is selected for the test. Various objectives can be used in test assembly. For example, maximizing the lower bound of test information (maximin approach, van der Linden & Boekkooi-Timminga, 1989), minimizing the deviation from target test information (literatures), matching observed-score distribution (van der Linden & Luecht, 1998), and maximizing test reliability (Adema & van der Linden, 1989; Armstrong, Jones, & Wang, 1998). A more comprehensive coverage of possible test assembly problems is given in van der Linden (2005).

The use of mathematical programming offers attractive possibilities to analyze the model. We can observe, for example, the impact of adding more constraints on the model, or whether some constraints appear to be difficult to satisfy (Timminga, 1998; Timminga & Adema, 1996; Huitzing, 2003). When the problem cannot be solved within a reasonable time, heuristic methods can be useful alternatives. Examples of the application of heuristic methods to test assembly problem are weight deviation methods (Luecht, 1998; Smith, 2005; Swanson & Stocking, 1993) and Monte Carlo method (Belov & Armstrong, 2005).

1.3.1 Paper-and-Pencil (P&P) Tests

Also known as linear tests, paper-and-pencil (P&P) tests are designed to serve a specific range of abilities at once. It means that a group of examinees will get the same test regardless of their ability levels. Though it is possible for P&P tests to use a computer for delivery, its common format is a booklet in which test questions are placed in a certain order.

It is often necessary to have several versions of the test. For example, when the test is offered at different testing sessions, or when the test is intended to observe the impact of some treatments (known as a pretest-posttest design, van der Linden, 2005). Also, different versions may be required to be parallel to a reference test. Often, these versions are assembled in advance to allow test specialists to scrutinize the content before they are administered.

The assembly of one test form may be simple. However, the assembly of multiple parallel tests requires a more careful approach. If the tests are assembled sequentially, the quality of the subsequent tests will decrease even though the item bank contains enough items to support parallel tests. In the long run, this practice may result in a serious depletion of some item bank section because best items are always selected first. Practitioners realize that the tests should be assembled *simultaneously*, to ensure uniform item usage in the bank. However, such task demands an advanced software and computer power which were simply not available in the early 1990s. The use of heuristic methods became an alternative and practical way to solve parallel test assembly (e.g., Ackerman, 1989; Adema, 1992; Armstrong, Jones, & Wu, 1992; Boekkooi-Timminga, 1990; Swanson & Stocking, 1993; van der Linden & Adema, 1998).

Test assembly is achieved by directly selecting items from an item bank. Fluctuations in the quality of the items in the bank will greatly influence the quality of the tests assembled from it. Optimal item-bank design for P&P tests anticipates this fluctuation (van der Linden, Veldkamp, & Reese, 2000). These authors proposed the use of an item-pool blueprint as a tool to keep item bank quality uniform over time. Their strategy was to observe and then to identify which items were really needed in the bank. In Chapter 5 of this thesis, a simulation study was used to observe the potential of an item-pool blueprint approach to manage an item bank.

1.3.2 Computerized Adaptive Tests (CAT)

With the development of computers, it is possible to tailor the test to different ability levels. A computerized adaptive test (CAT) is a test that adapts automatically to the ability of the examinee. It works like the practice of oral examiners where test questions are given sequentially, based on their impression of the examinee's ability in answering the previous questions.

In CAT, items are selected sequentially, according to the response of the examinee to the previous items. One popular selection criterium in CAT is maximum test information at each of ability estimates (e.g., Kingsburry & Zara, 1991; Lord, 1980; Owen, 1975; Weiss, 1982). As a result, only items with the highest information (at that ability estimate) will be chosen by the algorithm. Practitioners often critize this selection algorithm, mainly because items with the highest information are selected in the early stages when the ability estimation is still unstable. If a CAT is constrained, i.e., it has to meet certain test specifications, care must be taken to ensure that items are selected to satisfy these specifications. Stocking and Swanson (1993) developed a heuristic method to select items for constrained CAT. An exact method, called a shadow-test approach, has been proposed by van der Linden (2000). In a shadow-test approach, a complete test that meets the whole set of constraints is assembled at every stage of ability estimation, and one item from this test will be administered to the examinee. In so doing, a test that is optimal with respect to the objective and meets all test requirements can be obtained.

Experience with CAT has shown that usually only a small percentage of items in an item bank is selected. Unbalanced item-bank usage will lead to two serious problems. First, items of high exposure will be soon known to future examinees. Second, it is a waste of resources to invest in writing, producing, and pretesting items that appear almost never to be used for actual tests.

To overcome these problems, item-selection algorithms are modified in such a way that no item will be used more frequently than at a maximum allowable rate. Such algorithms are known as item-exposure control algorithms (e.g., Revuelta & Ponsoda, 1998; Stocking & Lewis, 1998; Sympson & Hetter, 1985; van der Linden & Chang, 2003; van der Linden & Veldkamp, 2004; Veldkamp, Eggen, Verschoor, & Maroujenkov, 2004).

However, many exposure control algorithms are less successful in promoting the use of underexposed items. A more effective approach would then be to limit the presence of popular items in a bank. Realization of these ideas requires a master bank to be equally divided into several operational banks which may share some of the items (e.g., Ariel, Veldkamp, & van der Linden, 2004; Stocking & Swanson, 1998). Mills and Steffen (2000) observed that such methods are also effective to maintain uniform test quality over time.

Unbalanced item-bank usage in CAT raises the idea of designing item bank effectively. Unlike its P&P counterpart, for a CAT bank it is very difficult to decide beforehand which items are needed in the bank. Veldkamp and van der Linden (2000) used a computer simulation to determine the number and the ideal composition of the items with respect to their categorical and quantitative attributes. A more tractable approach is presented in van der Linden, Ariel, & Veldkamp (in press), where a CAT pool is assembled as a set of linear tests. This idea is based on the use of the shadow test approach in constrained CAT.

1.3.3 Multistage Tests (MST)

In a multistage test (MST), the advantages of CAT and P&P testing are combined. Sets of items (testlet) are administered sequentially. Examinees proceed to a testlet with a higher or with a lower difficulty level based on their (provisional) ability estimate. For examinees, it is possible to review their answers before they move to the next stage. The number of stages and difficulty levels are predetermined, and the number of different paths possible for the examinees is much smaller than for CAT. Also, for test specialist, the task of evaluating and reviewing actual test content becomes manageable.

Operational MST programs require a large number of testlets to build several versions of the tests. This strategy is essential to protect the security of the test content, an issue that always has to be addressed in high stakes testing (e.g., licensure testing). Test assembly for MST is similar with parallel test assembly for P&P tests.

To anticipate uncertainty in the quality of the future assembly, it is necessary to maximize the use of items presently available in the bank. Optimal item-bank usage can be realized by assembling a maximum number of testlets, to avoid the accumulation of high quality items in a small number of testlets. Not all testlets are operational; some testlets are returned to the item bank, and will be employed to maintain uniform quality of the item bank over time. Such practice mimics a dummy test approach (Adema, 1990; van der Linden & Adema, 1998). These authors have successfully shown that parallel tests can be obtained by assembling the tests sequentially with the help of a dummy test.

To find the maximum number of testlets from a given bank, one can adopt popular numerical methods to locate a root of a mathematical function, such as the *bisection* method (e.g., Atkinson, 1989). The bisection method starts with an interval which is guaranteed to contain the number to be found. For example, for a bank of 1000 items with a test length of 50, one can start with interval [0,20]. The interval is bisected iteratively and, the midpoint is used to determine which half of the interval should be kept, by examining whether this midpoint gives feasibility to the model. In that example, the midpoint is 10. If the model is feasible for 10 number of testlets, then we keep [10,20]; otherwise we keep [0,10]. And so on, until the length of the interval is equal to one, and both values are evaluated to decide which one is taken. This searching method is potentially faster and does not need a sophisticated guess.

Alternatively, the maximum number of testlets can be estimated as follows. First, an initial number is used to solve the model. If the model is infeasible, the number is lowered by one and the model is solved again. The process is continued until the model is feasible. In this traditional method, a good guess is needed to improve the searching time.

Note that iterative methods are used only to find the maximum number of testlets. For the real assembly, the transformation of item bank to testlet banks should be done simultaneously, to avoid unbalanced item usage. With the fast development in computer and software, the assembly of maximum number of testlets is doable (Ariel, Veldkamp, & Breithaupt, 2004; Wu, 2001). In Chapter 4, we illustrate the transformation of item bank to testlet banks.

1.4 Overview of the Thesis

The focus of this thesis is on item bank management. Various tools to assemble and to manage item banks effectively are presented. A short overview is given in this section.

In Chapter 2, we propose alternative methods to assemble several item banks from a master bank. These item banks can be mutually exclusive (no item overlap), and the banks should possess similar quality to support fair testing. Empirical examples show that overlap between item banks is more effective than non-overlap in improving item usage, especially if they contain few popular items. However, forcing the use of underexposed items might result in reduced measurement precision.

One reason for unbalanced item-bank usage in adaptive testing may be an undesirable correlational structure between the content attributes of the items and their statistical information in a CAT bank. To minimize this effect, in Chapter 3 we explore the idea of a CAT bank as a set of linear tests. It is motivated by the use of a shadow-test approach in selecting items for constrained CAT. In shadow-test approach, a linear test is assembled at every ability estimation, to ensure that test specifications are met, but only one item from it is administered. The results show that the resulting CAT bank is effective in equalizing item usage while the error measurement is uniformly low.

As for a multistage testing design, a large number of testlets (short linear tests) is needed to provide parallel versions of the tests. The problem is identical to that of assembling a maximum number possible parallel linear tests. In Chapter 4, we illustrate optimal transformation of an item bank to a testlet bank. In this study, different weights are used to highlight the importance of some testlets with respect to their difficulty level.

A dynamic process of adding into and removing items from an item bank is unavoidable. This process can affect the quality of the bank, and it threatens the quality of the tests. In Chapter 5, a simulation study is conducted to observe what may happen to the quality of an item bank over several testing periods. Scenarios for different factors that may influence the quality of the bank (initial bank, management level, and item production) are used in the simulation study. The results show that the use of an optimal item-pool blueprint is effective for maintaining the quality of item banks.

The results of the simulation study in Chapter 5 also indicate that the accumulation of unselected items may deteriorate the quality of an item bank. In Chapter 6, various potential methods to analyze the causes of unselected items are given, and the application of one method is illustrated in an empirical study. The results of the simulation study suggest that there are items that always have to be selected, thus, they are critical for the assembly of feasible tests. The presence of critical items may trigger a strong interaction between their attributes and other items' attributes. This possible condition reinforces the need of a balanced composition of item banks.

2

Constructing Rotating Item Pools

Abstract

Preventing items in adaptive testing from being over- or underexposed is one of the main problems in computerized adaptive testing. Though the problem of overexposed items can be solved using a probabilistic item-exposure control method, such methods are unable to deal with the problem of underexposed items. Using a system of rotating item pools, on the other hand, is a method that potentially solves both problems. In this method, a master pool is divided into (possibly overlapping) smaller item pools which are required to have similar distributions of content and statistical attributes. These pools are rotated among the testing sites to realize desirable exposure rates for the items. In this paper, a test assembly model for the problem of dividing a master pool into a set of smaller pools is presented. The model was motivated by Gulliksen's (1950) matched random subtests method. Different methods to solve the model are proposed. An item pool from the Law School Admission Test (LSAT) was used to evaluate the performances of computerized adaptive tests from systems of rotating item pools constructed using these methods.
Key words: Computerized adaptive testing, item pool design, matched random subtests method, mathematical programming, rotating item pools, test assembly.

This paper has been published as: Ariel, A., Veldkamp, B.P., & van der Linden, W.J. (2004). Constructing rotating item pools for constrained adaptive testing. *Journal of Educational Measurement*, 41, 345-359.

2.1 Introduction

In paper-and-pencil (P&P) testing, the same set of items is administered to a population of examinees. A disadvantage of this testing format is its inability to deal with a broad range of abilities in the population. This disadvantage is remedied by computerized adaptive testing (CAT) where each examinee takes a test with items selected to match their ability estimates during the test. In doing so, CAT emulates an important aspect of oral examination, namely the practice of an examiner who chooses a more difficult question if the examinee responds correctly and an easier question if (s)he responds incorrectly. A CAT algorithm implements this practice by updating the examinee's ability estimate, $\hat{\theta}$, and choosing the next item to be optimal at this estimate.

Maximum-information and Bayesian item selection are commonly used criteria to select items (Hambleton, Swaminathan, & Rogers, 1991; van der Linden & Pashley, 2000). When an item is selected to maximize information at the current ability estimate, the algorithm prefers items for which both the difference between the current ability estimate $\hat{\theta}$ and the item difficulty parameter b_i is small and the item discrimination parameter a_i is high (Veerkamp & Berger, 1999). These items are of high quality but, unfortunately, for most item pools, their number is low. For a population of examinees, these items are selected more often and the high exposure rate of these items makes them vulnerable to security breaches. Typically, the other items are hardly selected at all and the resources invested in writing and calibrating them have futile effect (Veldkamp, 2001).

Various methods have been proposed to solve this problem, including methods of item-exposure rate control, item pool design, and rotating item pools. Sympson and Hetter (1985) introduced a probabilistic approach to control the exposure rate of every item in the pool (for a review of this method, see van der Linden, 2003). McBride, Wetzel and Hetter (1997) advocated an algorithm that reduced the exposure rates of items most popular at the beginning of the test. Veldkamp and van der Linden (2000) suggested to calculate a blueprint for the item pool such that distribution of the exposure rates for the items in the pool tends to be even. Stocking and Swanson (1998) introduced a method of rotating item pools with the same goal of even exposure rates. The item pools in this method are assembled from a master pool by splitting it into several smaller pools. The item pools are randomly rotated during operational testing. The goal of more uniformly distributed item-exposure rates is realized by assigning items with higher exposure rates to a smaller number of pools and items with lower rates to a larger number of pools. Unfortunately, studies of this method are rare. It is the purpose of this paper to propose some methods for assembling systems of rotating item pools from a given master pool and evaluate their properties.

In this paper, several methods to construct the system of rotating item pools are presented. All methods are based on a two-stage assignment process, in which items are first assigned to interim sets of (closely) parallel items and then from these sets to item pools. This process is optimized using the idea underlying the matched random subtests method introduced by Gulliksen (1950) to split a test into parallel subtests to the largest possible split-half reliability, which is always a lower bound to classical test reliability. Interestingly, the same idea can be generalized to the problem of assembling a system of rotating items pools. To illustrate how to put our methods into practice, several examples using an item pool from the Law School Admission Test (LSAT) are given. A constrained CAT algorithm was applied to evaluate the performance of the systems of rotating item pools constructed by these methods.

2.2 Current Methods to Construct Rotating Item Pools

Way (1998) observes that probabilistic item-exposure control may be inadequate to guarantee a secure CAT program. His suggestion is that a system of rotating item pools may be a more promising approach to prevent item compromise. Way, Steffen, and Anderson (1998) discuss several examples of strategies of managing rotating item pools and using such systems to enhance the security of the computerized testing.

For systems of rotating item pools to be effective, the availability of automated item-selection procedures to assemble such pools from a master pool is crucial. These procedures should have the following properties. First, each pool should have similar distributions of content and statistical attributes to support uniform measurement quality to examinees. Second, the composition of the pools should support uniform usage of the items. Third, the pools should have enough items to allow item selection for the adaptive tests to be constrained with respect to all the specifications to be imposed on the test.

Stocking and Swanson (1998) demonstrate a method to construct rotating item pools. In their method, the items in the master pool are assigned to pools by their weighted deviations model (WDM). The first step in this method is to calculate an average (nonoverlapping) pool from the master pool. The actual pools are then assembled to minimize the differences between these pools and the average pools with respect to (1) the item attributes in the constraints to be imposed on the CAT and (2) item information at selected θ levels. In doing so, the deviations are weighed to get a single objective function. The resulting pools are expected to have similar distributions of content and statistical attributes and, hence, to support each test administration equally well.

Two versions of the Stocking-Swanson method exists, one with nonoverlapping and another with overlapping item pools. The former was presented above. However, to get more uniformly distributed exposure rates, a system of overlapping item pools is more efficient. This system allows us to reduce the exposure rate of more popular items by assigning them to a smaller set of pools and to increase the usage of less popular items by assigning them to a larger set. Overlapping pools are assembled by first calculating the required numbers of pools the items should figure in and then assigning the items to the pools until these numbers are realized. For empirical results with these methods, see Stocking and Swanson (1998).

2.3 New Methods for Constructing Rotating Item Pools

The new methods proposed for constructing rotating item pools are all based on techniques of constrained combinatorial optimization. The objective functions in the optimization problems focus on the values of the item parameters in the pools. The goal is to give the pools identical distributions of parameters. At the same time, constraints are introduced to match the pools in terms of content attributes and to control the overlap between the pools.

All methods were motivated by Gulliksen's (1950) matched random subtests method. Gulliksen's method was proposed to split a test into two halves that are statistically as closely parallel as possible. The split-half reliability calculated from these halves is a lower bound to the classical test reliability. Gulliksen's method has two stages. In the first stage, the items are assigned to pairs of items that have minimal differences between their parameter values. In the second stage, the items are assigned to test halves. A formalization of Gulliksen's method as a problem of constrained combinatorial optimization is given in van der Linden and Boekkooi-Timminga (1988).

We apply the same method to the problem of splitting a master pool into a set of smaller pools for use as rotating item pools in CAT. To illustrate the method, its stages for the case of nonoverlapping pools are briefly described: First, interim sets of items are assembled, each of a size equal to the number of (nonoverlapping) pools to be constructed. Second, items in the same interim set are assigned to different pools. If the composition of these pools is required to meet certain constraints to support CAT, the assignment is subject to these constraints.

Figure 2.1 illustrates the general process of dividing a master pool into four nonoverlapping pools. In this figure, to get four nonoverlapping pools, the n items in the master pool are assigned to n/4 interim sets, each consisting of four different items.

The two stages are explained in more detail as follows.

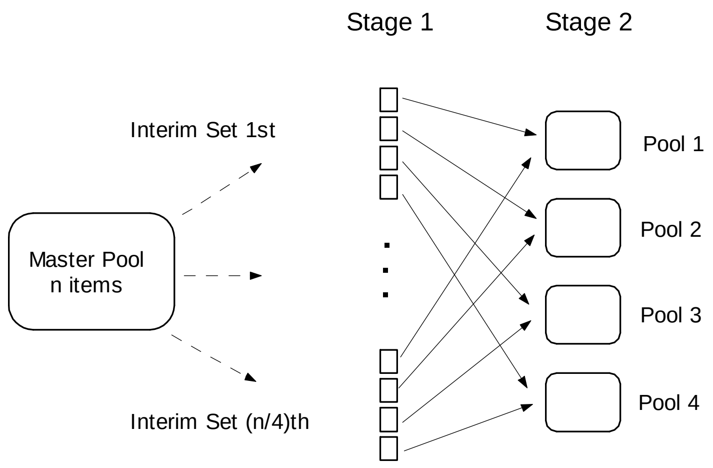

Figure 2.1: The process of dividing a master pool into four nonoverlapping pools

2.3.1 Stage 1: Assigning Items to Interim Sets

In the first stage, the items in the master pool are assigned to interim sets. For notational simplicity, we formulate the optimization problem for interim sets consisting of two items. Generalization to larger interim sets is straightforward.

A metric δ_{ij} is used to represent the differences between items i and j in interim sets. If the goal is to minimize the differences between the values of the items for the a and b parameters in the sets, a possible metric for these differences is

$$\delta_{ij} = |a_i - a_j| + \omega\,|b_i - b_j|\,, \tag{2.1}$$

where ω is a parameter that can be used to correct for differences between the scales of the two parameters. Other types of metric and larger numbers of item parameters are possible.

The problem of assigning items to interim sets can be formulated as a 0-1 mathematical programming problem with decision variables, $x_{ij}, i \neq j = 1, ..., I$, which are equal to 1 if item i and j are chosen in the same set and are equal to 0 otherwise, where I represents the number of items in the master pool. The objective function is

$$\min \sum_{i=1}^{I-1} \sum_{j=i+1}^{I} \delta_{ij} x_{ij} \tag{2.2}$$

subject to

$$\sum_{\substack{i \\ i<j}} x_{ij} + \sum_{\substack{i \\ i>j}} x_{ji} = 1 \ ,\forall j. \tag{2.3}$$

The constraint in (2.3) is to guarantee that every item is assigned to an interim set only once (van der Linden and Boekkooi-Timminga, 1988).

2.3.2 Stage 2: Assigning Items to Pools

In the second stage, the items in the interim sets are assigned to the item pools. Different assignment models for the assignment of items to nonoverlapping and overlapping pools are formulated.

Nonoverlapping Pools

The general idea in generating the item pools is to make them as similar as possible. In the mathematical programming model below, the objective function minimizes the differences between the total information in the pools at several ability values, while constraints are introduced to assign every item exactly once.

The model can be formulated as

$$\min z \tag{2.4}$$

subject to

$$\sum_i I_i(\theta_k) y_{is} - \sum_j I_j(\theta_k) y_{jp} \leq z, \forall \theta_k, s, p, s \neq p, i \neq j \tag{2.5}$$

$$\sum_i I_i(\theta_k) y_{is} - \sum_j I_j(\theta_k) y_{jp} \geq -z, \forall \theta_k, s, p, s \neq p \tag{2.6}$$

$$\sum_{i \in Q_r} y_{is} = 1, \forall s \tag{2.7}$$

$$\sum_s y_{is} = 1, \forall i \tag{2.8}$$

$$y_{is} \in \{0, 1\} \tag{2.9}$$

where i and j are indices for the items, Q_r is the rth interim set, s and p indicate item pools, θ_k is ability level k, $I_i(\theta_k)$ is the information about θ_k in item i, and y_{is} is a decision variable that is equal to one if item i is assigned to pool s and equal to zero otherwise.

In (2.5)-(2.6), the difference between the total information in the item pools at the ability values θ_k are constrained to be in the interval $(-z, z)$. The size of this interval is minimized in (2.4). The constraints in (2.7) require items in the same interim set to be assigned to different pools. The constraints in (2.8) guarantee that all items are assigned once, whereas the constraints in (2.9) define the decision variables to be 0-1. This model can be modified to allow for the case of overlapping pools. Moreover, the model can be extended to allow for additional constraints on the test required to deal with such issues as test content and word count. We will address these topics below.

Overlapping Pools

Overlapping item pools are obtained by changing the constraint on number of times an item is assigned to a pool in (2.8). This constraint is then replaced by

$$\sum_{s} y_{is} \leq n^{(r)}, \forall i \tag{2.10}$$

$$\sum_{s} y_{is} \geq n_r, \forall i \tag{2.11}$$

with $n^{(r)}$ and n_r denoting the maximum and minimum number of item replications admitted, respectively. Unpopular items should have larger values $n^{(r)}$ and n_r and popular items should have lower values.

Whether or not an item is popular is usually known only after conducting a CAT simulation. Based on items performance in the CAT algorithm, we then can decide what values $n^{(r)}$ and n_r should have. However, we can also decide on these values using the values of the items for the discrimination parameter (see empirical examples below).

Additional Content Constraints

The model previously described does not allow for possible additional content constraints on the CAT yet. Suppose, for example, that the CAT has to be constrained with respect to item type and word counts. In order to ensure that the pools have comparable distributions of these item attributes, the model in Stage 2 has to be extended with the following constraints:

$$\sum_{i \in V_m} y_{is} \geq n_m, \forall s \tag{2.12}$$

$$\sum_{i \in V_m} y_{is} \leq n^{(m)}, \forall s \tag{2.13}$$

$$\sum_{i} w_i y_{is} \geq n_w, \ \forall s \tag{2.14}$$

$$\sum_i w_i y_{is} \leq n^{(w)},\ \forall s \tag{2.15}$$

where V_m is the set of items of type m in the master pool, w_i is the word count of item i, n_m and n_w are the lower bounds on the number of items of type m and the word count in the test, respectively, and $n^{(m)}$ and $n^{(w)}$ are the upper bounds on these attributes.

2.4 Algorithms for Solving the Models

Various algorithms for solving the previous two types of models are presented. We first discuss the methods for assigning items to interim sets.

2.4.1 Assigning Items to Interim Sets

For a system of rotating nonoverlapping item pools, the number of parallel items in every interim set is equal to the number of pools to be created. Various methods for assigning items to interim sets are given.

Sequential Assignment

A simple method would be to use a greedy heuristic. This heuristic has been applied to test assembly problems in combination with the Weighted Deviation Model (Stocking & Swanson, 1993) and in combination with the Normalized Weighted Absolute Deviation Heuristic (Luecht, 1998). When this heuristic is applied to solve the model, the interim sets are constructed sequentially, that is, items with the smallest value for (2.1) are selected until all items have been used. It is obvious that this approach is easy to implement, but does have some drawbacks. For each subsequent item, the value of (2.1) will be larger and the quality of the interim sets will therefore deteriorate. Therefore a method based on simultaneous assembly of all interim sets is proposed.

Simultaneous assignment

If the number of pools to be assembled increases, the combinatorial optimization problem involved in their assembly becomes larger and we may soon reach a point at which a heuristic is needed to solve the model. In the empirical example below, we used a heuristic method known as simulated annealing. Simulated annealing is an iterative Monte Carlo method. At each step a candidate for a new solution is generated by randomly modifying the previous solution, after which it is decided whether this candidate is or is not accepted. An attractive feature of simulated annealing is that it accepts a worse solution with a nonzero probability. Typically, the probability becomes smaller after some iterations. This feature allows the algorithm to backtrack if it gets stuck in a bad path. The algorithm is terminated as soon as no further improvement can be made. Examples of a earlier applications of this heuristic to test assembly problems are given in van der Linden, Veldkamp and Carlson (2004) and Veldkamp (1999). A more theoretical explanation of simulated annealing can be found, for example, in van Laarhoven and Aarts (1987).

A simple but effective implementation of simulated annealing for the current problem is to generate new interim sets from randomly selected old sets. We initialized the algorithm solution by assigning all items to all interim sets. This initialization is possible because the final solution does not depend on the initial solution. (If the number of items in the master pool is not a multiplicity of the number of pools, a dummy interim set has to be created to which the remaining items are assigned.) Candidate interim sets were then constructed by randomly swapping items between sets. The metric in (2.2) was used to evaluate the candidate sets against the old sets. If the new value for the objective function was smaller, the candidate sets were accepted with probability one. If it was larger, the decision was made with a probability. At the next iteration step, new candidates were generated from the current interim sets.

2.4.2 Assigning Items to Pools

Two methods for assignment of items from interim sets to item pools are discussed.

Random Assignment Method

Because in the first stage of the method the interim sets are constructed to be as similar as possible, it may be possible to assign items in the same interim set randomly to item pools. If items in all interim sets are assigned, we may still have a dummy set that had to be created because the number of items in the master pool was not a multiplicity of the number of pools. The items in this set can also be assigned randomly. However, in the empirical example below, the information functions for each pool was calculated and the items were assigned to the pools with the smallest values for these functions.

Mathematical Programming

A more precise method is to solve the model in (2.4) until (2.9) and the additional constraints in (2.12) and (2.14) using a mathematical programming algorithm. Use of such algorithms is recommended especially when sequential assignment results in quick deterioration of the value of the objective function for the interim sets. In the empirical study below we used one of the standard algorithms in the mathe-

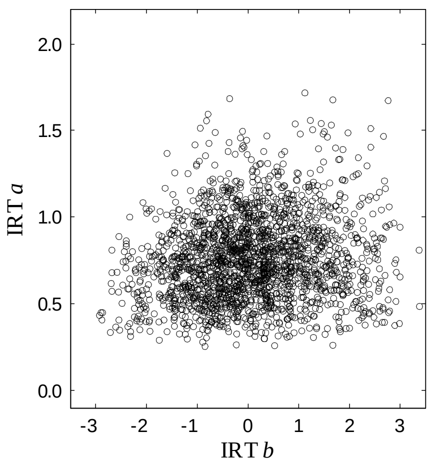

Figure 2.2: A scatter plot of item discrimination (IRT a) and item difficulty (IRT b) parameter

matical programming software packages AIMMS (Paragon Decision Technology, 2003).

2.4.3 Choosing the Number of Overlapping Pools

If overlapping pools are to be constructed, the bounds on the number of times an item can be assigned, $n^{(r)}$ and n_r, have to be specified for each item. The choice of these numbers can either be based on the values of the item parameters or on the actual exposure rates of the items. The first method is based on the discrimination parameter of the items. Since items with higher a_i values have a larger chance of being selected, such items should be given low values for $n^{(r)}$ and n_r. On the other hand, items with low a_i have a low probability of being selected and should be given higher values for $n^{(r)}$ and n_r. The second method is based on the actual exposure rates of the items. Items with high exposure rates are given low values for both $n^{(r)}$ and n_r, while items with low exposure rates are given high values.

2.5 Empirical Example

A previous pool of 2,131 items from the LSAT fitting the 3-parameter logistic model (Hambleton, Swaminathan & Rogers, 1991) was used as the master pool. Figure 2.2 shows the distribution of the values for the a_i and b_i parameters for the items in the pool. We ignored the c_i parameter because its variability was small and this parameter typically does hardly have any impact on the composition of the item pools.

The items in the master pool were of nine different types, leading to 20 content constraints. Because word counts of the items were available we used constraints to match the word counts of the pools as well. All constraints appeared relatively easy to satisfy. In this study, we first classified the items in the master pool based on their type and then solved the (extended) model in (2.1)-(2.3) per item type. The benefit of this approach was that the model in (2.4)-(2.15) reduced proportionally and (2.12)-(2.13) could be ignored. Observed that when the number of item of a certain type was small, the number of nonoverlapping pools produced would be limited since each pool had to satisfy (2.12)-(2.13).

Following Stocking's (1994) recommendation that the size of a CAT item pool size be at least 12 times the length of the adaptive test (which was 24 in our study), at most four nonoverlapping pools could be produced. The same restriction did not apply for the case of overlapping pools, however, because it allowed for the duplication of some items to meet (2.12)-(2.13). As a result, we were able to assemble one system of nonoverlapping pools with four pools and two different systems of overlapping pools with six and eight pools.

All item pools in this study were assembled using an objective function based on the metric in (2.1), with $\omega = 1$. All results were evaluated through a CAT simulation study. The CAT algorithm in these studies was based on a constrained CAT with shadow tests approach (van der Linden, 2000). In this approach, prior to the selection of an item first a full test meeting all constraints is assembled. From this test, the most informative item is administered to the examinee. After the ability estimate is updated, the shadow test is reassembled keeping all items already administered. The process continues until the required number of items administered is reached. The shadow tests were calculated using the AIMMS (2003) optimization software package. Test length was fixed at 24 items and the values of the examinees for the ability parameter were randomly drawn from the standard normal distribution. Abilities were estimated by the method of maximum likelihood estimation (MLE). As long as the simulated responses for an examinee were all correct or all incorrect, the ability estimate for the examinee was set equal to 3 and -3, respectively. In each condition, 1,000 examinees were simulated and ability estimation was always initialized at $\widehat{\theta} = 0$.

To evaluate the performances of the methods, the exposure rates of the items and the bias and mean squared error (MSE) functions for the ability estimators were calculated. No method of probabilistic item-exposure control was used; the effects of such methods would have confounded the evaluation of impact of the pool assembly methods on the exposure rates of the items. In a practical application, however, the use of additional exposure control may be necessary.

2.5.1 Nonoverlapping Item Pools

The performances of all four possible combinations of the methods of sequential and simultaneous assignment of items to interim sets and random assignment and mathematical programming were compared. Each combination was evaluated using CAT simulations. The simulations were based on the full systems of rotating pools, each with four nonoverlapping pools. Because each pool had approximately 500 items, each evaluation was based on approximately 2,000 items. In addition, CAT administrations directly from the master pool were simulated. The results for CAT from the master pool served as a point of reference for our evaluation of the results in the other four conditions. In the mathematical programming method, item information was controlled at $\theta_k =$-1,0, and 1.

Figure 2.3 shows the item exposure rates, bias, and MSE for CAT from all item pools assembled by the four combinations of methods as well as directly from the master pool. For each method, the rates of only 600 of the items (out of the total of 2,000 items) are given; the rates of all other items were equal to zero. The highest exposure rate for an item in CAT from nonoverlapping pools was less than 0.3. However, for CAT from the master pool the highest rate was equal to 1.0. Figure 2.3 also reveals that the number of items with nonzero rates is larger for CAT from nonoverlapping pools. These results show that, compared with CAT from the master pool, CAT from nonoverlapping pools improved the exposure rates of the items equally well for all four methods used to construct these pools.

Both the bias and the MSE functions are lower for CAT from the master pool than from nonoverlapping pools. The reason is the fact that during CAT from the master pool all items were available for selection for each of the examinees where with CAT from the nonoverlapping pools each item had to selected from a smaller set. For all practical purposes, the differences are small though. The differences between the functions for the four conditions with rotating item pools were also negligible.

2.5.2 Overlapping Item Pools

Unlike the previous study, only the second stage was used to construct overlapping pools. The upper and lower bounds $n^{(r)}$ and n_r to the overlap in (2.10)-(2.11) were determined using the two methods presented below. For each method, systems of six and eight overlapping item pools were created from the master pool.

The first method used to determine the upper and lower bounds on the item overlap between pools was based on the values of the items in the master pool for the discrimination parameter. These values were in the interval [0.2,1.7]. This

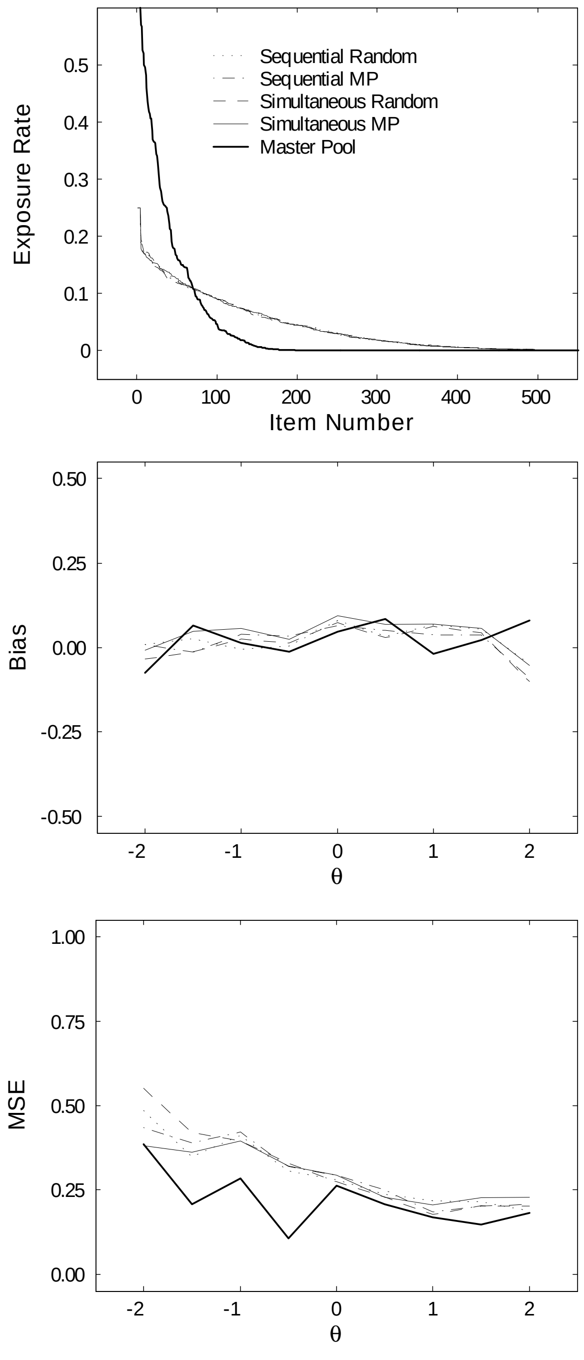

Figure 2.3: Item exposure rate, bias, and MSE for nonoverlapping pools.

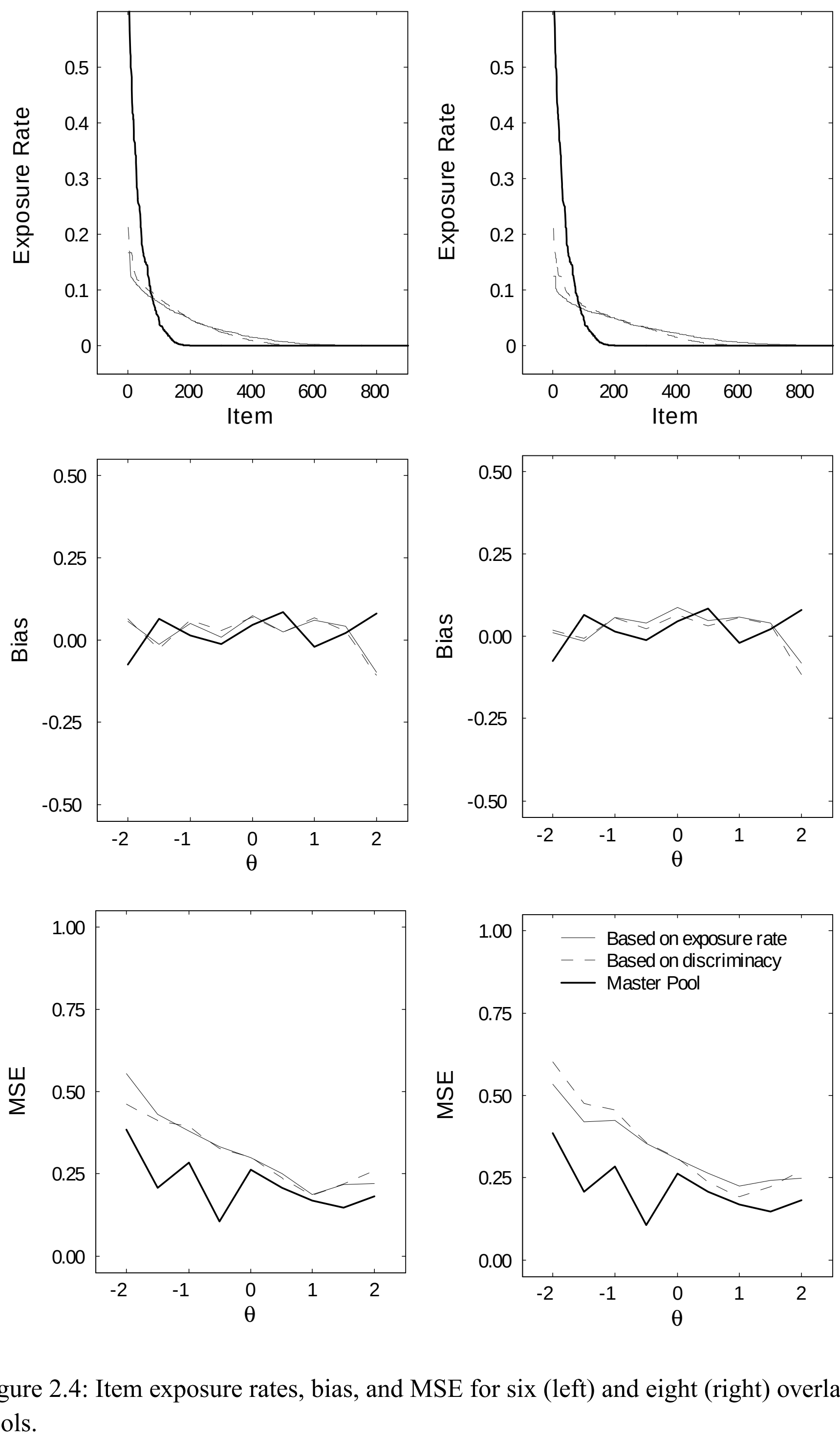

Figure 2.4: Item exposure rates, bias, and MSE for six (left) and eight (right) overlapping pools.

interval was divided into equally wide intervals, six and eight intervals for the cases of six and eight overlapping pools, respectively. Items with the highest value for the a_i parameter (the first interval) were assigned only once ($n^{(r)}$ and n_r are equal to 1), items in the second interval twice ($n^{(r)}$ and n_r are equal to 2), etc.

The second method was based on the empirical exposure rates of the items. Using the exposure rates in the first study, the items were divided into two sets. The criterion was an exposure rate below or above 0.5. The items with larger exposure rates were assigned only once, the items with smaller rates were assigned to all item pools.

The estimated exposure rates as well as the bias and MSE functions for both methods are given in Figure 2.4. For items that were assigned to multiple pools, the exposure rates were cumulated across pools. The exposure rates were slightly more favorable than in Figure 2.3 but showed the same pattern. Also, both for the case of six and eight overlapping pools, the method for setting the bounds $n^{(r)}$ and n_r based on empirical exposure rates outperformed the method based on the discriminacy parameter. Also, the differences between the two methods were larger for the case of eight than six pools. However, these more favorable exposure rates were obtained at the costs of slightly higher bias and MSE.

The differences in results between the methods for setting bounds on item overlap follow directly from the criterion on which they were based. In the method based on empirical exposure rates, every pool contains limited number of good items because these items are assigned to only one pool. As a result, CAT algorithm is forced to choose considerable numbers of worse items and the errors in the ability estimates increase. For the method based on discriminacy values, only the items in the highest category are assigned once. Because the item pools have larger numbers of good items, the estimation errors are smaller but the worse items remain hardly used at all.

2.6 Discussion

The methods for constructing item pools presented in this paper are intended to optimize or to maximize the use of test items in CAT. It was shown that CAT from rotating item pools that do not overlap can improve the usage of items. As shown in Figure 2.3, the item exposure rates for CAT from nonoverlapping item pools were substantially better than for CAT directly from the master pool. The differences found between the exposure rates and statistical quality of the ability estimates of the four combinations of methods used to assemble these pools were generally negligible, though. This finding seems to suggest that best strategy in real-life applications is to choose a method from these four that is easy to implement under local constraints.

Though the use of nonoverlapping item pools did not dramatically increase the exposure rates of the less frequently items, the use of overlapping pools was more successful in this respect. Especially the results obtained when the method for setting the bounds $n^{(r)}$ and n_r was based on empirical exposure rate were promising. Key to these results seems to be the fact that larger numbers of popular items were not allowed to be assigned to more than one pool. As a result, the CAT algorithm was forced to select larger numbers of less popular items, which thus obtained larger exposure rates.

The experimental results also showed that improving the exposure rates of the items tends to results in larger errors for the ability estimates. This trade-off should not come as a surprise: More uniform item exposure rates can only be obtained by imposing more severe constraints on the item selection. These constraints result in a lower value for the objective function optimized during item selection, that is, in less information about the examinees' abilities in the test. However, it is the opinion of the authors that the size of the increase of these errors in the present studies was still acceptably low and could easily have been compensated, for example, by a small increase in the length of the test. The best possible way to deal with these errors is topic of further research.

The empirical study shows that implementation of the methods proposed in this paper is straightforward. In theory, the methods can be applied to construct systems with large numbers of rotating item pools. The only possible obstacle is limited availability of some types of items in the master pool. Though it seems attractive to duplicate such items and assign them to overlapping pools, hence there is virtually no bound on the number of overlapping pools possible, we should be aware of the possible adverse effects of this measure on the exposure rates of some of the items as well as a possible increase in measurement error.

3
Assembling CAT's Pool as a Set of Linear Tests

Abstract

Test-item writing efforts typically results in item pools with an undesirable correlational structure between the content attributes of the items and their statistical information. If such pools are used in computerized adaptive testing (CAT), the algorithm may be forced to select items with less than optimal information, that violate the content constraints, and/or have unfavorable exposure rates. Though at first sight somewhat counterintuitive, it is shown that if the CAT pool is assembled as a set of linear test forms, undesirable correlations can be broken down effectively. It is proposed to assembly such pools using a mixed integer programming model with constraints that guarantee that each test meets all content specifications and an objective function that requires them to have maximal information at a well-chosen distribution of ability values. An empirical example with a previous master pool from the Law School Admission Test (LSAT) yielded a CAT with nearly uniform bias and mean-squared error functions for the ability estimator and item-exposure rates that satisfied the target for all items in the pool.
Key words: Assembling multiple test forms, computerized adaptive testing, item pool design, mixed integer programming, shadow test approach.

This paper has been accepted for publication in *Journal of Educational and Behavioral Statistics.*

3.1 Introduction

Ideally, an adaptive testing program should offer each examinee a short test that is informative at his or her ability level and meets all content specifications. Also, particularly in high-stakes testing, to avoid risking the security of its items, the selection of the items in the test should not be restricted to a small subset of the pool. Item-selection algorithms have been developed to realize these goals in the best possible way. For example, it is now common to have CAT algorithms select items that maximize Fisher's information in the test or minimize the posterior variance of the examinee's ability (van der Linden & Pashley, 2000), balance the test content by spiraling item selection over subsets of item in the pool (Kingsbury & Zara, 1991), minimizing weighted deviations from bounds on content categories (Stocking & Swanson, 1993) or realizing constraints on test content through a shadow test approach (van der Linden, 2000), and control item exposure rates by adopting a random component in the selection procedure (Sympson & Hetter, 1985).

Nevertheless, though these methods are potentially powerful, their actual success is ultimately constrained by the composition of the item pool. For example, ability estimation may be less accurate than anticipated if several of the most informative items in the pool happen to have attributes that are excluded by some of the content specifications. Also, the items in the pool may have the majority of the required content attributes but several of them may miss one or two important attributes. If such cases occur, the item pool looses its efficiency because the CAT algorithm has to compromise between the necessity to select tests that meet all content specifications, have high information at the examinees' ability level, and do not overexposure a small set of items.

Observe that it is not necessarily the marginal distributions of the content or statistical attributes of the items in the pool that may make an item pool less efficient, but the fact that items with certain *combinations* of attributes are scarce. This observation is consistent with the notion that the problem of optimal assembly of an item pool should be viewed as an instance of *combinatorial* optimization (Nemhauser & Wolsey, 1999). Intuitively, an optimal item pool would consist of a maximal number of combinations of items that (1) meet all content specifications for the test and (2) are most informative at a series of ability levels reflecting the shape of the distribution of the ability estimates for a population of examinees. Observe that the second requirement is not only necessary to make the adaptive test informative for each examinee but also to yield exposure rates for the items in the pool that are below a realistic target value.

Though it may seem a step backward to more rigid traditional group-based paper-and-pencil testing, it is instructive to compare these two requirements with the practice of assembling linear test forms in testing programs that have such a format. In such programs, rather than assembling one individual test form at a time, it is common to assemble a set of parallel test forms from a mastery pool at the beginning of a new planning period to serve all of its testing sessions and locations. If the program is item response theory (IRT) based, these forms are parallel in that each of them meets the same content specifications and the same target for the test information function. The set of these forms can be viewed as the operational "item pool" during the planning period of the program.

If the same pool of items were to be used for a CAT version of the program, its composition as a set of tests would guarantee the pool to have a sufficiently large number of combinations of items meeting the content specifications. In fact, because the CAT algorithm would mix and match between the different tests, the actual number is much larger than the number of individual tests in the pool. However, because these combinations would tend to be optimal at ability levels near the center of the population distribution, the CAT algorithm would only be able to produce ability estimates with the same accuracy for examinees from the center of the population distribution if it compromised the content of the test and/or the exposure rates of some of the items in the pool. Intuitively, the need to compromise would be minimal if the set of individual tests in the pool kept its composition but had maximum information at a distribution of ability levels approximating the shape of the population of examinees.

In this paper, the idea of an item pool consisting of a set of fixed tests optimal at a distribution ability levels reflecting the population of examinees is elaborated for the case of CAT based on the shadow test approach (van der Linden, 2000; van der Linden & Reese, 1998). In this approach, the items in the CAT are selected not from the pool but from shadow tests assembled before the selection of each new item. Shadow tests are tests that (1) meet all of the content specifications for the adaptive test, (2) contain all items already administered to the examinee, and (3) are optimal at the examinee's current ability estimate. The item that is actually administered is the most informative item in the shadow test among those that the examinee has not seen before. Adaptive testing with shadow tests guarantees test administrations that always meet all content specifications (i.e., are feasible with respect to the content constraints) and have optimal ability estimation. The assembly of shadow tests in real time has become possible by the speed of modern computers and the use of software packages for linear programming (LP), such as the CPLEX package used in the empirical example below (ILOG, 2002).

The process of CAT with shadow tests is illustrated graphically for a fictitious examinee in Figure 3.1. As usual, θ denotes the ability parameter in the IRT model, for example, the three-parameter logistic (3PL) model in (1) below, and $\widehat{\theta}$ is an estimate of the examinee's value for the ability parameter. The first shadow test is assembled to be optimal at the initial ability estimate, $\widehat{\theta}_0$. The first item administered is the item in the shadow test optimal at $\widehat{\theta}_0$. After the response to the first item, the ability estimate is updated and the second shadow test is assembled to be optimal at this updated ability estimate, $\widehat{\theta}_1$, retaining the item that was already administered. The process is continued until the last free item in the shadow test, which was selected to be optimal at $\widehat{\theta}_n$, is administered. Observe that if the CAT were administered from a pool of linear tests, the set of shadow tests in Figure 3.1 would not be a subset of the tests in the pool. For one thing, the tests in the pool are disjoint whereas the set of shadow tests shows overlap. More importantly, the set of shadow tests would be the result of the process of mixing and matching between the individual tests in the item pool by the CAT algorithm discussed earlier. It is exactly this process that is expected to lead to less violation of the target for item-exposure rates and/or less loss of accuracy of ability estimation if the distribution of ability values at which the tests in the pool are optimal reflects the population of examinees.

The process in Figure 3.1 allows us to make the notion of "an item pool reflecting the population of examinees" more precise. Each shadow tests is optimal at a series of estimates of approaching the examinees' true values of θ. The choice is therefore between the distribution of all $\widehat{\theta}$s during the adaptive tests or the distribution of the true value of θ in the population of examinees. As a matter of fact, because the distribution of the actual exposure rates of the items depends on the former but the accuracy of the ability estimator on the latter, this choice amounts to one between the relative weight put on item security and the accuracy of the ability estimator. Our personal choice would be to guarantee item security and choose the distribution of $\widehat{\theta}$, if necessary with slight increase in test length to compensate for loss of accuracy in ability estimation. In the examples in this paper, the consequences of the choice between the two distributions is examined for an item pool assembled from a previous master pool from the Law School Admissions Test (LSAT).

3.1.1 Earlier Methods

The method proposed in this paper differs from the methods studied in van der Linden, Veldkamp and Reese (2000) and Veldkamp and van der Linden (2000). The major difference is that the methods in these references are methods for item pool *design*, whereas the current method is a method for the *assembly* of an item pool from a mastery pool. In an item pool design problem, no actual items are available yet, but an optimal blueprint for an item pool is produced, where a blueprint is defined as a plan for the distribution of numbers of items over the space with all possible combinations of the relevant statistical and content attributes of the items. In van der Linden, Veldkamp and Reese, an integer-programming method is used to calculate an optimal blueprint for an item pool for a testing program with linear forms that minimizes the item-writing costs. In Veldkamp and

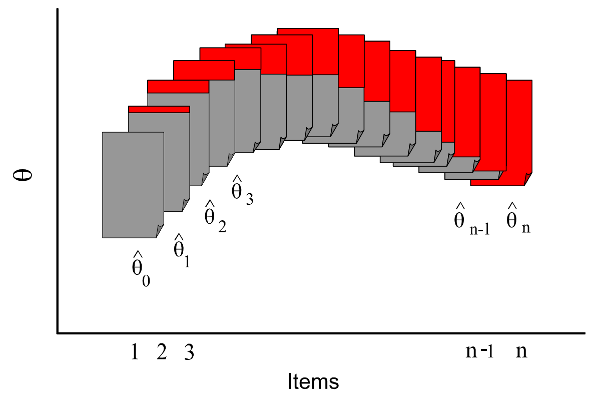

Figure 3.1: Graphical representation of item-selection process in CAT with shadow tests for a fictitious examinee (darker area represents items in shadow tests already administered, lighter area items reassembled at new ablity estimate).

van der Linden, an optimal blueprint for a CAT item pool is found by simulating adaptive test administrations over the design space.

The goal of our current research was to find a method for the assembly of an optimal CAT pool from a given master pool. Previous literature on this problem does not seem to exist. However, the problem of a method for assembling systems of rotating item pools from a mastery pool has been addressed earlier (Ariel, Veldkamp & van der Linden, 2004; Stocking & Swanson, 1998). These methods can be used to find an optimal split of a master item pool into a set of (possibly overlapping) smaller pools such that rotating these pools among the examinees would result in favorable item-exposure rates.

3.2 Item Pool Assembly Method

The method is formulated for the distribution of estimates $\widehat{\theta}$ for the population of examinees. The distribution is denoted as $g(\widehat{\theta})$. A method for the distribution $g(\theta)$ is obtained if θ is substituted for $\widehat{\theta}$. The question of how to get an estimate of $g(\widehat{\theta})$ or $g(\theta)$ will be addressed later.

3.2.1 Method

The method has the following steps:

1. Determining the required item pool size, m. It is assumed that m is a multiple of the length of the adaptive test, n. The item pool is thus assembled to consist of m/n different linear tests.

2. Choosing an ability interval, $[\theta_l, \theta_u]$, that is large enough to neglect possible parts of $g(\widehat{\theta})$ not covered by it.

3. Partitioning the interval $[\theta_l, \theta_u]$ into a series of m/n adjacent intervals. The sizes of intervals should be chosen such that each interval contains an equal portion of $g(\widehat{\theta})$. Because we have m/n intervals, each interval should contain a portion of $g(\widehat{\theta})$ equal to n/m. The midpoint of the fth is denoted as θ_f. This partitioning results in a set of points θ_f with a distribution that reflects $g(\widehat{\theta})$.

4. Assembling a set of m/n tests from the master pool, one for each point θ_f, $f = 1, ..., F = m/n$, using the model for test assembly in the next section. The model requires each test to meet the content constraints and to have maximum information at its point θ_q. The set of all tests constitutes the CAT item pool.

3.2.2 Models

In the empirical example below, the 3-parameter IRT model for the items in the master pool is:

$$p_i(\theta) = \Pr(U_i = 1 \mid \theta) = c_i + (1 - c_i)\{1 + \exp[-a_j(\theta - b_j)]\}^{-1}, \qquad (3.1)$$

where U_i is a binary variable for the response of the examinee to item i, $\theta \in (-\infty, \infty)$ represent the examinee's ability level, and $a_i \in [0, \infty)$, $b_i \in (-\infty, \infty)$, and $c_i \in [0, 1]$ are the discriminating power, difficulty, and guessing parameter for item i, respectively (Birnbaum, 1968).

To present the model for the simultaneous assembly of multiple tests, the following notation is needed. The individual items in the master pool are denoted by $i = 1, ..., I$ and the individual tests to be assembled by $f = 1, ..., F$. The model is formulated using decision variables x_{if} which take the value one if item i is assigned to test f and the value zero otherwise. The model will have constraints to prevent each item from being assigned to more than one test. As for the content specifications, constraints for only one set of categorical and one quantitative attribute are formulated; in a real-life application, more constraints of these types will be needed to represent the total set of content specifications. Generally, categorical constraints require tests to meet upper bounds $n_c^{(u)}$ and lower bounds $n_c^{(l)}$ on the numbers of items with content attributes $c = 1, ..., C$. The sets of items in the master pool that have these attributes are denoted by V_c. Quantitative constraints require tests to meet upper bounds $n_q^{(u)}$ and lower bounds $n_q^{(l)}$ on the sums of the values of the items on a quantitative attribute q (eg, expected response time or word count). Finally, Fisher's information in test f at θ_f is denoted as $I_i(\theta_f)$.

The general shape of the model is:

$$\text{maximize } y \tag{3.2}$$

subject to

$$\begin{aligned}
&\sum_{i=1}^{I} I_i(\theta_f)x_{if} \geq y && f=1,...,F\\
&\sum_{i=1}^{I} x_{if} = n && f=1,...,F\\
&\sum_{i\in V_c} x_{if} \leq n_c^{(u)} && f=1,...,F,\ c=1,...,C\\
&\sum_{i\in V_c} x_{if} \geq n_c^{(l)} && f=1,...,F,\ c=1,...,C\\
&\sum_{i=1}^{I} q_i x_{if} \leq n_q^{(u)} && f=1,...,F\\
&\sum_{i=1}^{I} q_i x_{if} \leq n_q^{(l)} && f=1,...,F\\
&\sum_{f=1}^{F} x_{if} \leq 1 && i=1,...,I\\
&x_{if} \in \{0,1\} && i=1,...,I,\ f=1,...,F
\end{aligned} \tag{3.3}$$

The first set of constraints in (3.3) defines a lower bound y to the information in each of the tests at the points θ_f. The objective function in (3.2) maximizes this lower bound. As a result, the tests are assembled to approximate a uniform distribution of information over θ_f, $f = 1,...,F$, at maximum height. The second set of constraints in (3.3) defines the length of the tests. The next four sets are needed to require both the numbers of items with categorical attributes and the sums of the values of the items on the quantitative attribute to be between upper and lower bounds. The two last sets of constraints guarantee that items are assigned to one test at most and the decision variables take only admissible values, respectively.

3.2.3 Solution

A solution to the optimization problem in (3.2)-(3.3) can be calculated using one of the commercially available software packages for linear programming with an optimizer for mixed integer problems. The authors' favorite is CPLEX, of which a new version with remarkably improved power has been released recently (ILOG, 2002). A solution to the model in (3.2)-(3.3) consists of a set of values for the decision variables x_{if} and y; the former denote the items that have been assigned to the tests, the latter is the maximum value possible for the master pool given the constraints in the model.

It is important that the tests be assembled simultaneously, i.e., in one optimization run. If they are assembled sequentially, each next test would tend to be less informative at its θ_f than the tests assembled earlier at theirs. Due to the nature of the decision variables, the model in (3.2)-(3.3) does assemble all tests simultaneously. The first to model the problem of simultaneous test assembly using this type of variable was Boekkooi-Timminga (1987).

It should be observed that, for a given master pool, the number of variables x_{if} increases exponentially in the number of tests to be assembled. If the number gets too large, it is recommended to run the optimizer until a solution is obtained with a value for the objective function differing from an upper bound to its optimum only by a small percentage. A reasonable upper bound can always be found by relaxing the integer constraints on x_{if} in the last set of constraints in (3.3), that is, replacing the domain of these variables by the interval [0,1]. Alternatively, an implementation of the model with the "dummy-test" approach proposed in van der Linden and Adema (1998) can be used.

3.3 Empirical Study

The goal of the empirical part of this study was to assess the impact of the distribution of the ability levels θ_f on the quality of the adaptive test. In particular, the difference in impact between sets of tests assembled at a distributions of θ_f matching $g(\theta)$ and $g(\widehat{\theta})$ was studied. As a baseline, adaptive testing from an item pool for a traditional paper-and-pencil program with tests with parallel information functions was used. More in particular, the following conditions were present:

Condition 1 A set of tests assembled using the method proposed in this paper to be optimal at a distribution of θ_f chosen to reflect $g(\widehat{\theta})$;

Condition 2 A set of tests assembled using the method proposed in this paper to be optimal at a distribution of θ_f chosen to reflect $g(\theta)$;

Condition 3 A set of parallel tests each assembled to have an information function reflecting $g(\theta)$.

Each of these pools was evaluated using CAT simulations with and without item-exposure control. The quality of the CAT administrations was determined using the bias and mean-squared error (MSE) functions of the ability estimators and the actual distribution of the item-exposure rates in the pool. It was not necessary to record the number of times the content constraints on the test were met.

A sufficient condition for all shadow tests to be feasible is to have at least one feasible combination of items in the pool (van der Linden & Reese, 1998), and this condition is automatically met for item pools consisting of a set of tests each assembled to meet the content specifications.

The differences in quality of measurement between the conditions were expected to be smaller near the middle of the ability distribution than in the tails. As for the tails, we expected the pools in Condition 1 and 2 to outperform the one in Condition 3, but hoped for small differences between the pools in Condition 1 and 2. In practice, an estimate of the distribution of ability estimates, $g(\widehat{\theta})$, for a population of examinees can easily be obtained by running CAT simulations. It is more difficult, however, to infer a good estimate of the true ability distribution, $g(\theta)$, from a set of response data. Small differences between Condition 1 and 2 suggest that we can use $g(\widehat{\theta})$ instead of $g(\theta)$.

3.3.1 Method

The master item pool was a previous pool from the LSAT with $I = 5,316$ items. This size of pool is huge for the test assembly problem involved in the method in this paper. However, because the pool was a previous real-life mastery pool, it was decided to leave it intact. The sum of the information functions of all items in the pool is depicted in Figure 3.2. As the figure reveals, the pool had a slight tendency to be on the difficult side for the $N(0,1)$ population assumed in this study.

The length of the adaptive test was set equal to 50 items, which is half the length of the current paper-and-pencil version of the LSAT. The bounds in the right-hand sides of the constraints in (3.3) were reduced proportionally. The paper-and-pencil version of the LSAT does have an item-set structure with several specifications for the content of the common stimuli, which was ignored in this study though. The total number of categorical and quantitative constraints needed to model the content specifications of the LSAT in (3.3) was 96.

The size of the operational CAT pools to be assembled from the LSAT pool was set to 500. Given the length of the adaptive test, the pools therefore had to consist of ten individual tests each. This number of tests involved a version of the model in (3.2)-(3.3) with $10\text{x}5,316 + 1 = 53,161$ variables.

The true ability distribution $g(\theta)$ was assumed to be $N(0,1)$. For the first condition, the distribution of $g(\widehat{\theta})$ was obtained by simulating CAT administrations from the master item pool for 10,000 examinees randomly sampled from $g(\theta)$. The simulations were based on the shadow-test algorithm described above. The distributions of $g(\theta)$ and $g(\widehat{\theta})$ are shown in Figure 3.2. As expected, the estimate of $g(\widehat{\theta})$ was less peaked and, consequently, had more mass in its tails.

For the first condition, the interval [θ_l, θ_u]=[-2.0, 2.0] was partitioned into ten intervals using $g(\widehat{\theta})$ according to Step 3 in the description of method above.

The intervals had midpoints θ_f that were equal to $\{$-1.606, -1.018, -0.656, -0.361, -0.109, 0.143, 0.404, 0.696, 1.055, 1.627$\}$. For the second condition, the same method was used to partition the interval $[\theta_l, \theta_u]$=[-2.0, 2.0] into ten intervals using $g(\theta)$. These intervals had midpoints θ_f equal to $\{$-1.593, -0.991, -0.648, -0.373, -0.122, 0.122, 0.373, 0.648, 0.990, 1.593$\}$.

For the third condition, the pool of parallel tests with the shape of their individual test information function matching the shape of $g(\theta)$ was assembled using a modified version of the model in (3.2)-(3.3). The modification was the replacement of the first set of constraints in (3.3) by a larger set

$$\sum_{i=1}^{I} I_i(\theta_k)x_{if} \geq g(\theta_k)y \quad f = 1, ..., F, k = 1, ..., K, \tag{3.4}$$

where the test information functions were controlled at the values θ_k, $k = 1, ..., K$. These sets of constraints require the test information function to have a relative shape determined by the weights $g(\theta_k)$ but maximizes its height. In this study, we chose to control the test information functions at $\theta_1 = -1$, θ_2=0, and θ_3=1. This choice was motivated by our ample experience that, as information functions are smooth, well-behaved functions, control at only a few θ values is sufficient to get desired results.

For each of the conditions, the CAT simulations were repeated twice. The simulations were also based on the shadow-test algorithm. In the first set, 1,000 CAT administrations were simulated for each value θ =-2.0, -1.5, ..., 2.0. This set was used to evaluate the bias and MSE functions of the ability estimator. The same number of simulations was used at each of the θ values was to get estimates of these functions with uniform accuracy along the scale. In the second set, 10,000 CAT administrations were simulated with θ values randomly sampled from $N(0, 1)$. This set was used to evaluate the distributions of the actual exposure rates of the items in the three pools for the true population in this study.

The ability estimator was the expected a posteriori (EAP) estimator with uniform prior over [-4.0, 4.0]. For each simulated examinee, the ability estimator was initialized at $\widehat{\theta} = 0$. In the conditions with item-exposure control, the method with probabilistic item-ineligibility constraints in van der Linden and Veldkamp (2004) was used with a target value for the exposure rates set at .25.

3.3.2 Results

For the first two conditions, the assembly of the pools took approximately 83 seconds on a PC with an Intel Pentium II 860 MHz processor with 128 MB of memory. For the third condition, it turned out to be difficult to reach optimality in realistic time. The process was therefore stopped when a solution with a value for the objective function differing no more than 5% from the relaxed solution

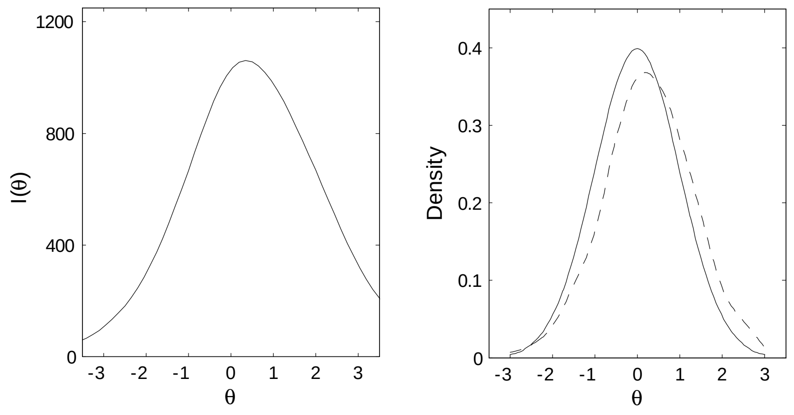

Figure 3.2: Pool information (left), the distribution of $g(\theta)$ and $g(\widehat{\theta})$ (right), where dashed line represents $g(\widehat{\theta})$.

was found. This result was reached in approximately three hours. The difference in solution times between Condition 1-2 and 3 can only be due to the presence of the additional constraints in (3.4); all other CPLEX settings were identical. Apparently, if a problem reaches this size (over 50,000 variables), the reduction of the feasible area introduced by (3.4) was too large for CPLEX to handle. It is very well possible that different settings or a slight reformulation of the model could remove this effect. However, as demonstrated by the results in Figure 3.3, a solution differing less than 5% from optimality works well for all practical purposes.

Figure 3.3 shows the information functions for the 10 tests and for the entire item pools in the three conditions. Each of the individual tests had its peak approximately at the required θ value. The only difference is a slight tendency of the peaks toward the lower end of the scale to be to the right of their required θ values. This tendency is believed to be a consequence of the fact that the master pool was slightly on the difficult side for the N(0,1) ability distribution (see Figure 3.2). Observe that the peaks for the curves for Condition 1 and 2 follow the ability distribution in the chosen population, whereas those for Condition 3 were all near the middle of the ability distribution. Also, the peaks for Condition 1, which were based on $g(\widehat{\theta})$ instead of $g(\theta)$, showed a slightly larger spread. All results were thus nearly exactly as required.

It is interesting to note that, in spite of the differences between the first three plots in Figure 3.3, the last plot shows approximately the same shape of the total information functions for the three pools. It is thus possible that items pools, which are nearly identical on the surface, at a deeper level have an entirely different composition. The important difference between the three pools is the distribution of the individual tests of which they are composed. For Condition 1 and 2, this distribution guarantees numbers of items with both maximum information and the required combination of attributes available at θ values in proportion to the density of the examinees in the population. The pool for Condition 3 lacks this feature.

The quality of the three pools was further assessed using CAT simulations. The bias, MSE functions, and exposure rates are given in Figure 3.4. Both for the CAT with and without exposure control, the bias functions for Condition 1 and 2 had never a difference larger than .009 over the full range of θ values. Both functions also showed a slight inward trend typical of a Bayesian ability estimator. The MSE functions for Condition 1 and 2 differed never more than .005. Also, they were flat for all practical purposes, except for a slight upward trend at the lower end of the scale, which is believed to be a consequence of the fact that the master pool was on the difficult side and the lower end of the distributions of the tests in the pools for these two conditions were influenced by this. The curves for Condition 3 coincide with those for Condition 1-2 for the θ values in the middle of the scale. In fact, the only meaningful differences are between Condition 1-2 and 3 at the lower and upper ends of the scale, where both the bias and MSE in Condition 3 became approximately twice as large as in Condition 1-2.

In Figure 3.4, the item-exposure rates for the three pools with and without exposure control were nearly indistinguishable. For the condition with exposure control, each item always had an exposure rate below the target value of .25. This result should not be ascribed to the composition of the item pools, but to the power of the exposure-control method used, which guarantees effective control of all items (for details, see van der Linden & Veldkamp, 2004). However, the fact that these results were obtained introducing only small random variation in the bias or MSE functions was due to the composition of the pools. As each pool consisted of a sufficiently large number of items with the required combinations of attributes, once the exposure-control method had rejected an item for administration to an examinee, it was always able to find another one with the same combination of attributes that had its maximum information at a point not too far from the item rejected. As demonstrated by the less uniform shape of the bias and MSE functions for Condition 3 in Figure 3.4, the only difference between the item pools was that for this condition both the rejected item and its substitute tended to be closer to the center of the ability distribution than desirable.

3.4 Conclusion

Though the method item pool assembly introduced in this paper is based on simple traditional principle, its impact on the composition of the item pool and, hence, on the properties of the adaptive test in the empirical study was remarkably positive. The method succeeded in breaking down effectively the undesirable correlational

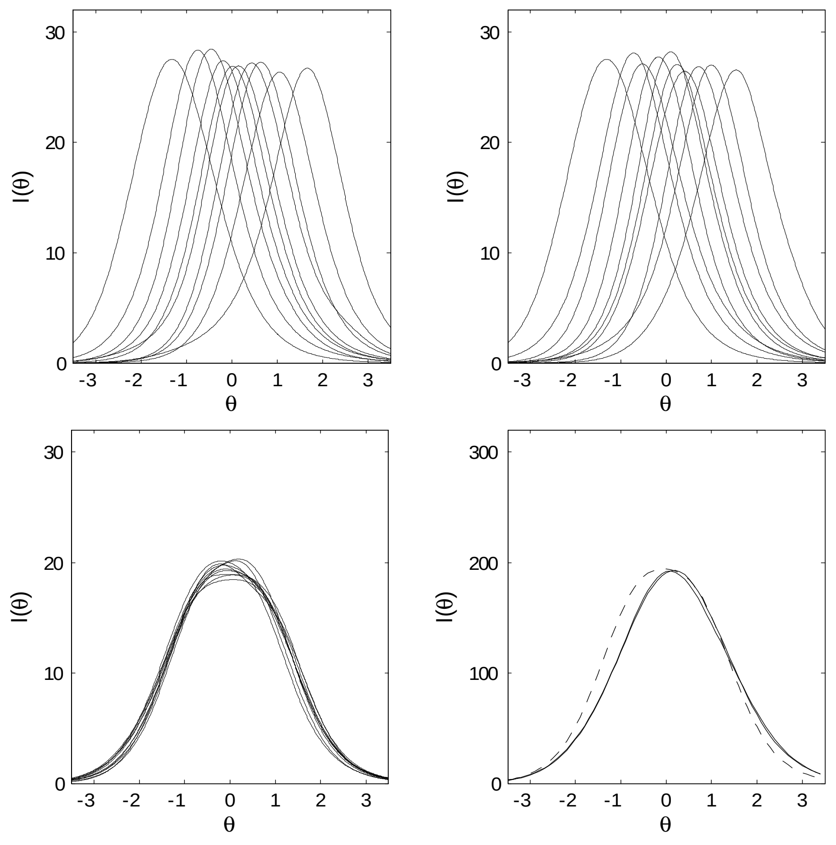

Figure 3.3: Test information curves for Condition 1 (top left), Condition 2 (top right), and Condition 3 (bottom left), and pool information for each condition (bottom right, where dashed line represents Condition 3).

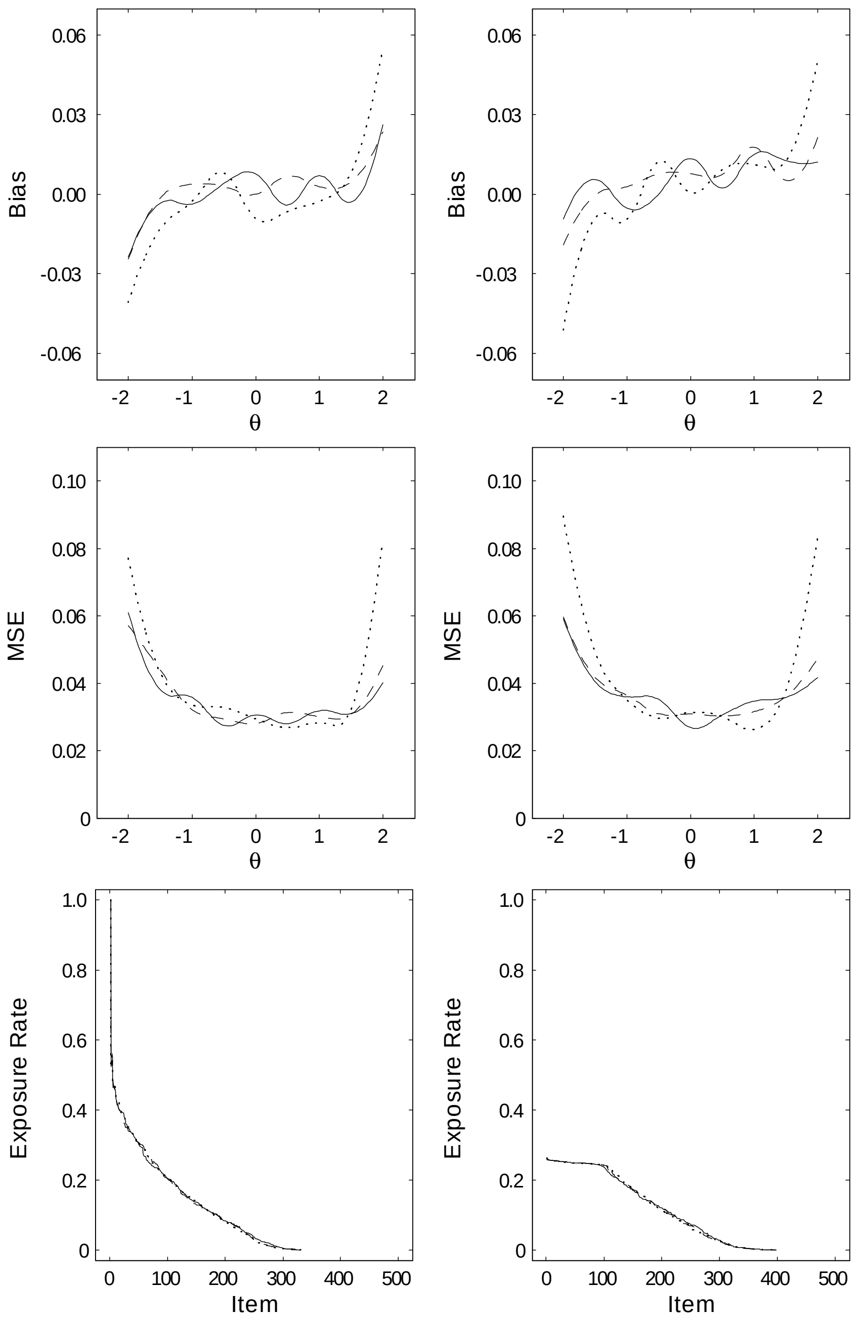

Figure 3.4: Bias, MSE, and item exposure rate for each condition, without exposure control (left) and with exposure control (right), where solid line represents Condition 1, dashed line Condition 2, and dots Condition 3.

structure between the points of maximum information and the content attributes of the items typically found in operational CAT pools. As a result, the CAT algorithm was never hampered by any lack of items with required combinations of attributes along the θ scale and able to move more freely from the location of the initial item to true ability level of the examinee. The nearly uniform MSE functions in Figure 3.4 witness this effect. A second result is that the item-exposure control mechanism had no difficulty replacing items rejected for an examinee with items of comparable statistical quality. The fact that, except for small random variation, the curves in Figure 3.4 have the same shape for the conditions with and without exposure control identical witnesses this second result.

In the empirical example we hardly found any differences between the pools for Condition 1 and 2 assembled for an application of the method with $g(\widehat{\theta})$ and $g(\theta)$, respectively. This result suggests that in practical applications there is no need to estimate $g(\theta)$; instead we can use the much easier to estimate distribution $g(\widehat{\theta})$.

Finally, in a sense, the method proposed in this paper might only seem to have shifted the problem of item pool design from the operational pool to the master pool. In our empirical study, the master pool did not impose any severe constraint on the assembly of the operational pools. This does not necessarily hold for real-life situations with less-well managed master pools. However, we do *not* recommend using our method for solving problems with the composition of master pools. Such an application would involve the imposition of the constraints in (3.2)-(3.3) derived from the content specifications for the adaptive test on the master pool. Rather, we believe these constraints should be imposed directly on where they belong, namely the algorithms or the pool of authors used to generate the items.

4 Optimal Testlet Pool Assembly

Abstract

Computerized multistage testing (MST) designs require sets of test questions (testlets) to be assembled to meet strict, often competing criteria. Rules that govern testlet assembly may dictate the number of questions on a particular subject, or may describe desirable statistical properties for the test, such as measurement precision in the range of the passing score. In a multistage testing (MST) design, testlets of differing difficulty levels must be created. Statistical properties for assembly of the testlets can be expressed using item response theory (IRT) properties of questions. The testlet test information function (TIF) value can be maximized at a specific point on the IRT ability scale. In practical MST designs, parallel versions of testlet are needed, so sets of testlets with equivalent properties are built according to specifications for that testlet. In this project, we study the use of a mathematical programming solution to simultaneously assemble testlets to ensure equivalence and fairness to candidates who may be administered different testlets, and to ensure the best use of the available item pool.
Key words: Mathematical programming, multistage testing, security, testlet, testlet assembly problem.

This paper has been submitted for publication.

4.1 Introduction

Operational computer-based testing (CBT) programs normally require a large available item pool. To provide uniform tests continually, items in the pool should also have stable psychometric and categorical/quantitative attributes. This might be costly; writing, banking and pretesting new items requires resources and quality is variable. In addition, an existing item pool can be expected to change at various times during a testing program (Way and Steffen, 1998).

One solution to quality control is to manage the use of items in the pool by assembling testlets or ”bundles of items” (Wainer and Kiely, 1987). In a testlet, items are not used individually. Instead, items are assembled and administered in clusters. The testlet approach combines some advantages of paper and pencil tests (e.g., the opportunity to build to content specifications and control item usage) with some advantages of computerized adaptive testing (CAT). An additional benefit of testlet assembly is the possibility of predicting and managing the security of test content. Monitoring of exposure can be accomplished at the assembly stage, in addition to tracking the administration history of content over time.

There have been a variety of models that use testlets, for example multistage testing (MST) (Lord, 1980) and computer-adaptive sequential testing (CAST) (Luecht, Brumfield, and Breithaupt, 2002; Luecht and Nungester, 1998). MST models assign a group of testlets to form a panel where the content difficulty selected for the examinee depends on performance during administration (adaptation). A thorough discussion over MST design, routing decision, and final score calculation is beyond the scope of this paper. In this study we use Figure 4.1 merely as an illustration and in the next paragraph we described only a general rule in MST. In Figure 4.1, a panel design consists of three stages and seven testlets. The difficulty levels of testlets in this example are classified as moderate, average, and difficult. Lines connecting the testlets in Figure 4.1 represent the possible pathways an examinee may take. In this panel, there are seven possible paths.

Only one testlet per stage is seen by each examinee. Every examinee who is administered any panel will start at Stage 1 where the testlet consists of items with average difficulty. At the next stage, depending on their responses toward items in the previous testlet, the examinees are routed to take either a moderate, an average, or a difficult testlet at Stage 2. A subsequent routing decision is repeated to select a testlet at Stage 3. Notice that only content difficulty vary among the paths; all possible paths should lead to identical test content.

Particularly for a testing program with a large volume of candidates every year, much more panels are required to support continuous testing and to minimize content exposure. Direct implication of such policy is that a large number of testlets has to be assembled (Luecht, 2000). The focus of this paper is on how to transform a discrete-item pool into a testlet pool. It is important to build this testlet

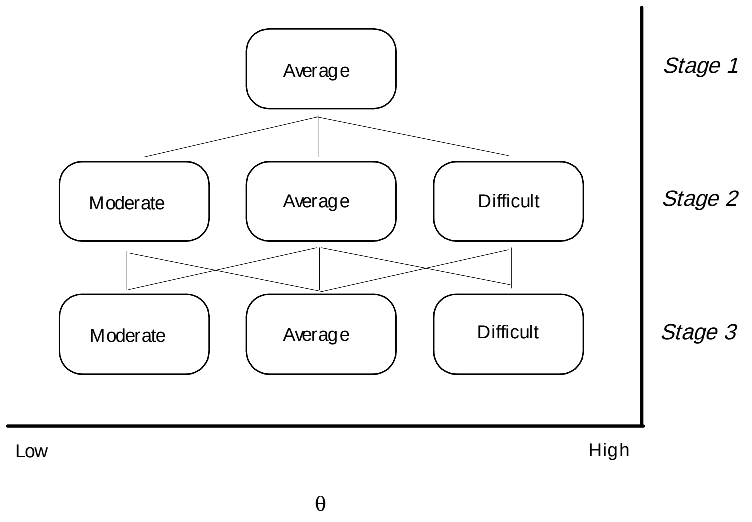

Figure 4.1: An example of a multistage testing (MST) design

pool by assembling testlets simultaneously. To construct testlets sequentially, or one by one, would be an easily programmed solution. However, a separate assembly approach would lead to a deterioration in the quality of the testlets formed later in the test assembly problem, compared to those assembled first. This is because fewer test questions (items) remain in the pool, and usually items with desirable properties are chosen early. Building testlets simultaneously will guarantee that items are equally distributed with respect to their psychometric attributes, thus the resulting test forms are expected to have uniform quality. Another important factor to consider is a balance of content as specified in the test blueprint. This balance of content must be assured for every path seen by candidates. One solution is to demand every testlet to meet both psychometrical and test requirements, and to enforce every testlet to be parallel in content. These concerns, and others, are represented as key principles in the design of testlets model. Discussion of these principles follows in the next section.

4.2 Principles of Constructing Testlets

The mandate of most testing programs relies on ensuring that candidate scores represent a measure across the content domains specified by the test blueprint.

Ensuring that any possible path through the MST will provide required content coverage equal for all candidates is essential. How to build the testlets with regard to their difficulty level is important as well. As indicated earlier, automated testlet assembly should be designed to produce parallel panels. In this section, we describe how these key considerations were operationalized using an optimization programming approach.

4.2.1 Content Constraints

Content and other constraints are incorporated into the testlet assembly process to be certain that the examinee experience is uniform. For example, the number of test questions, the bounds on the number of items with certain item types, the number of words, or the number of times any item has been used in previous testing events may be considered as content constraints when testlets are built. In our study, these constraints have to be applied at the testlet level so that once testlets appear together in a panel every possible path is balanced for content. The content balancing process can become laborious and is sometimes impractical when the testing program offers an MST with different content at each stage. Standardizing content at the testlet level also allows us to easily build parallel versions of routes and panels.

4.2.2 Difficulty Levels

In MST, items are selected into a testlet with respect to a specified range on the ability scale (moderate, average and difficult are the ranges in our study). The relationship between the ability level of candidates and the difficulty profile of items or testlets is strictly defined by IRT models (Hambleton, Swaminathan & Jones, 1991). Difficulty of testlets is implicit in the assembly process when a test information function (TIF) is used to identify items assigned to testlets.

The choice of appropriate theta values guides selection of items for a desired TIF, or difficulty range for any testlet. Meanwhile, the choice of targets will have implications for content exposure and precision of test scores (Luecht & Burgin, 2003). If possible, statistical targets for optimizing the TIFs would be selected to equalize exposure of all testlets in the build.

Formal approaches to the selection of TIF targets are not readily available from the existing literature. This is not surprising since the MST approach to test design in operational programs is a relatively recent development. In general, the choice of TIF targets depends on the specific objective of the testing program. For example, in the framework of licensure exams, some principled strategies to determine appropriate theta values have been proposed by Luecht (2000) and Luecht & Burgin (2003). According to Luecht and Burgin, the choice for the TIF should reflect three goals: precise measurement for critical decisions, appropriate information value with respect to the available item in the pool, and balance of the item (testlet) exposure rates. Obviously, the exposure of testlets depends on the choice of a cut-score for routing to moderate, average, or difficult testlets at subsequent stages, and the distribution of ability in the population. These issues are outside the scope of this paper, and to maintain generality we will use theta values range between -1 and 1.

4.2.3 Testlet Parallelism

A general procedure to construct tests based on item information is suggested by Birnbaum (1968). In that procedure, single items are selected from a calibrated item bank to fill the desired shape of TIF. This information function is inversely related to the asymptotic score variance; hence it is possible to control measurement error along the ability scale. Luecht and Nungester (1998) explicitly mention the use of target information to effectively control the measurement precision.

As previously explained, it is important to construct the largest possible testlet pool while maintaining parallelism and high quality throughout (Luecht, Brumfield, and Breithaupt, 2002). Paralellism, or statistical equivalency in this case can be defined by the IRT properties of the testlet questions and by content required by the test blueprint.

4.3 Simultaneous Assembly of Testlets

4.3.1 Model

In this study, a general model for the assembly process defined by van der Linden and Boekkooi-Timminga (1989) was adapted. In their assembly model, parallel tests are built by maximizing the lower bound of the testlet information at the specific target ability point (maximin approach). This lower bound value is called a relative target and is represented as y. It is possible to adjust the relative target (in this case testlet information) of different testlets by using weights to make the best possible use of the item pool (van der Linden, 2003). For instance, some item difficulty level might overlap for testlets of high difficulty and of moderate difficulty level as they share some theta value. In this sense, the IRT characteristics of items would be similar, and items would be eligible for moderate or difficult testlets. If no item-overlap is allowed among testlets, a higher weight can be given to the information target used to construct testlets of higher priority, say, difficult testlets.

In some case, it may be desirable to emphasize moderate testlet targets to ensure even use of the item bank across the ability scale, and as a consequence ensure useful feedback for candidates whose ability is low. Weights on the TIFs of testlets of different difficulties can be adjusted to meet these goals, when optimization and simultaneous assembly are used.

The linear programming solution for the assembly problem will also include many other constraints, for example item enemies (i.e. certain items are not allowed to appear on the same test if they cue other items or are close variations of a single item). There may also be rules for forced inclusion of items on key topics, or limits on the number of items related to similar content.

The complete rules that govern the model are summarized as follows. The model seeks to:

$$\text{Maximize } y \tag{4.1}$$

subject to the following constraints:

$$\sum_{i=1}^{I} x_{it} * I(\theta_k) \geq \omega_t * y, \;\; t = 1...T \tag{4.2}$$

$$\sum_{i=1}^{I} x_{it} = n, \;\; t = 1...T \tag{4.3}$$

$$n_{(r)} \leq \sum_{i \in V_r} x_{it} \leq n^{(r)}, \;\; t = 1...T \tag{4.4}$$

$$\sum_{i \in V_c} x_{it} \geq 1, \;\; t = 1...T \tag{4.5}$$

$$\sum_{i \in E}^{I} x_{it} \leq 1, \;\; t = 1...T \tag{4.6}$$

$$\sum_{t}^{T} x_{it} \leq 1, \;\; i = 1...I \tag{4.7}$$

$$x_{it} \in \{0, 1\} \tag{4.8}$$

$$y \in \mathbb{R}^{+} \tag{4.9}$$

In Constraints 4.2 relative information is measured at k ability values using a nonzero weight, ω. Total number of items for each testlet is fixed in Constraints 4.3 while bounds on the number of items with respect to certain categorical attributes are specified in Constraints 4.4. The inclusion of critical content and the exclusion of items (item enemy) are addressed in Constraints 4.5 and Constraints 4.6, respectively. In this assembly, no item is shared among testlets (Constraints 4.7). The last two constraints define the binary decision variables to select items and a positive real number as a lower bound for the testlet information.

4.3.2 Testlet Assembly Algorithm

It is essential to assemble the testlets simultaneously, to prevent unequal use of good quality items across testlets. However, assembling parallel testlets simultaneously is a challenging numerical aspect of binary programming. The complexity is due to the presence of binary variables; the number of variables is enormously increased as a multiplication between the intended number of test forms and the number of available items in the pool. The problem of assembling parallel testlets is similar to the one of creating parallel linear tests. The latter has been addressed by several authors. Adema (1992) proposed combining heuristic and optimal methods, Boekkooi-Timminga (1990) suggested selecting tests simultaneously from comparable subpools of items, and van der Linden and Adema (1998) offered the dummy test approach to reduce computational difficulty while obtaining parallel solutions for all test forms.

Heuristic techniques are popular alternative methods used to approximate optimal solutions in a relatively small amount of time. Nevertheless, the quality of solutions resulting from heuristic methods cannot be guaranteed and constraint violations may occur (Swanson & Stocking, 1993). From this point of view, heuristic methods are not sufficient or appropriate when it is essential to build precisely parallel testlets that are optimal and meet exactly all problem constraints.

Exact methods guarantee that solutions will be optimal, given that a feasible solution exists for the data. A feasible solution can be defined as a set of testlets where all content rules and other assembly rules are met. One example of an exact method is a network-flow approach. Wu (2001) demonstrated this method that can be applied to solve the problem of assembling parallel tests simultaneously. However, to accommodate complex constraints such as relative test information and weights on content required, the network-flow should be combined with heuristic approach. One example is from the work of Armstrong, Jones and Kunce (1998) where Lagrange relaxation is integrated into their network-flow model.

The assembly solution in this study makes use of another exact method known as mixed integer programming (MIP). In the statement of the problem, binary variables are employed to indicate whether or not a given item will be included in a particular testlet. The MIP model contains linear constraints that must be resolved. To find solution for MIP problems, a branch and bound algorithm (Dakin, 1965; Garfinkel & Nemhauser, 1972) is widely used. The efficiency of this algorithm is determined primarily by the formulation of the problem, the size of the problem, and the data structure. Much time (and computer resources) might be required in searching for feasible and optimal integer solutions. An analysis can be prolonged when it is possible such a solution does not exist based for a given data set (coded item pool). Some modifications to the MIP analysis can speed up the algorithm. However, the appropriateness of these modifications depends entirely on the specific problem. A thorough investigation and some useful suggestions concerning

MIP problems and the branch and bound algorithm framed in the test assembly vernacular are available from Veldkamp (2001).

The above model can be solved using commercial modeling language for mathematical programming such as AIMMS (Paragon Decision Technology, 2003) or OPL Studio© 3.1 (ILOG, 2003), both of these use branch and bound algorithms provided by CPLEX 8.1 (ILOG, 2003) to generate a solution. Both AIMMS and OPL Studio© have a variety of algorithms available. The test developer can choose settings that influence the solution strategy, with some effect on processing time and the optimality of the solution.

4.3.3 Continuous Testing Programs

Testlet assembly solutions, especially for high stakes continuous testing program, must create solutions that consider the quality of future assemblies into account. If the pool used to build the testlets will change over time (e.g., by adding new items), the task of producing equally high quality and parallel test forms over time is challenging. A common response toward this problem is to divide the master pool into smaller-sized, parallel pools and use only a subpool for any particular assembly. Literature concerning the use of splitting a master pool into subpools is available from a variety of computerized testing authors (e.g., Mills & Stocking, 1996; Way, Steffen, & Anderson, 1998), while the methods for splitting the item pool are addressed in, for example, Ariel, Veldkamp and van der Linden (2004), and Stocking and Swanson (1998).

The subpool approach could lead to fewer testlets or even infeasibility if the master pool has an inadequate number of items in some content areas, or otherwise uneven item composition. One approach to this problem, illustrated in our study, is to create a maximum possible number of testlets from a given item pool. Once the capacity of the master pool is known, conservative selection of testlets over time forms the subpools of interest. Such approach allows for uniform testlets across multiple testing periods by allocating some testlets for the present administration, and reserving others for future builds. This idea is very similar to the dummy test approach proposed by van der Linden and Adema (1998).

4.4 Empirical Example

This section illustrates the implementation of the principle we have discussed. A subset of multiple-choice questions (MCQ) from the master pool of an operational high-stakes testing program forms the basis of our study.

In our examples, the pool consists of 1,066 coded MCQ items that were calibrated using the 3PL IRT model. The general shape of item information curve in this pool is illustrated in Figure 4.2. Complete coding of the items involved discrete coding for content, skill, and enemy conditions. Enemies are defined as items with minor variations in wording, or items that provide a cue for the correct answer of other items. We use the bounds on content and other rules for test requirements that have already been specified by test developers. In order to benefit from a standardized testlet format, every testlet in the same stage is bound to the same requirements.

At the second and third stages in this MST design, testlets will differ in difficulty level. In practice, the choice of ability points (difficulty) where one would maximize testlet information is made with consideration of a variety of factors. These factors include routing rules and expected testlet exposure rates (Luecht & Burgin, 2003). In our study, we chose ability values ranging from -1 to 1 to identify where on the theta scale information for a testlet will be optimized.

Specifically, three sets of theta values were used to locate the midpoints and left or right targets for maximum information. In our study, for all stages, testlet of average difficulty has target at $\theta = 0$. The second and third stages have testlets of moderate difficulty with a midpoint at $\theta = -1, 0, 1$ and difficult testlets with a midpoint at $\theta = 1$. One reason for using overlap theta values among testlets was to enable good coverage of content difficulty for candidates lower on the ability scale who may need to use feedback scores to prepare to re-take the examination. At the same time, the precision of the total test score at the passing point is tightly controlled. It is possible to meet both objectives by using different weights for testlet targets to distinguish the amount of information required along the ability scale.

To accommodate various theta values in combination with different weight, Constraints 4.2 were appropriately extended as follows (van der Linden, 2003).

$$\sum_{i=1}^{I} x_{it} * I(0) \geq \omega_t * y, \quad t \in \text{ average testlets} \tag{4.2a}$$

$$\sum_{i=1}^{I} x_{it} * I(-1) \geq \omega_t * y, \quad t \in \text{ moderate testlets} \tag{4.2b}$$

$$\sum_{i=1}^{I} x_{it} * I(0) \geq \omega_t * y, \quad t \in \text{ moderate testlets} \tag{4.2c}$$

$$\sum_{i=1}^{I} x_{it} * I(1) \geq \omega_t * y, \quad t \in \text{ moderate testlets} \tag{4.2d}$$

$$\sum_{i=1}^{I} x_{it} * I(1) \geq \omega_t * y, \quad t \in \text{ difficult testlets} \tag{4.2e}$$

In our case study, we examined the benefit of using weights and impact on testlet information. The above model, therefore, is implemented under two conditions, namely:

Condition 1	The same weights are applied for all testlets, i.e., all testlets will be maximized using the same ω_t value $(\omega_t = 1)$
Condition 2	Different weights are used, i.e., a higher weight is given given to difficult testlets $(\omega_t = 1.5)$ while the other testlets will have lower weight $(\omega_t = 1)$.

In the first condition, we expect to see the original composition of the testlets information from the given pool. Observe that, as illustrated in Figure 4.2, the difficulty level of the available items tends to be higher than 0. The effect of such composition is that difficult testlets might have lower information as items of high difficulty have to be fairly distributed among testlets (the effect would be stronger as the testlets share similar theta value). In the second condition, higher priority is given to difficult testlets through the use of higher weight. In this case, it is expected to obtain higher testlet information for difficult testlets.

As described earlier, the maximum number of testlets that can be produced from the item pool is determined by the combination of constraints and the item pool. Constraints refer to the item enemy rules, the content bounds, and other conditions related to key content. It might be important to note that the target weights used will have no effect on the number of testlets that can be built. Therefore, a rough estimate of the upper limit on the number of testlets can be conjectured based on item availability and bounds on testlet constraints.

The maximum number of feasible testlets from this pool can be represented as Max. The actual maximum number of testlets that are produced, M, is determined by the composition of the item bank and rules for items or enemy-sets. Our objective was to find M. We investigated the number of testlets possible by solving the model for $m = 1...Max$. Once infeasibility occurred, Max was lowered by one and the process is repeated until we reach a feasible solution. In that case we conclude that $M = Max$.

4.4.1 Results

The testlet assembly models were solved using AIMMS with CPLEX 8.1 as the solver. An integer feasible solution was found, on average, in 40 minutes on a fairly common desktop computer. Improvements on the initial solution required

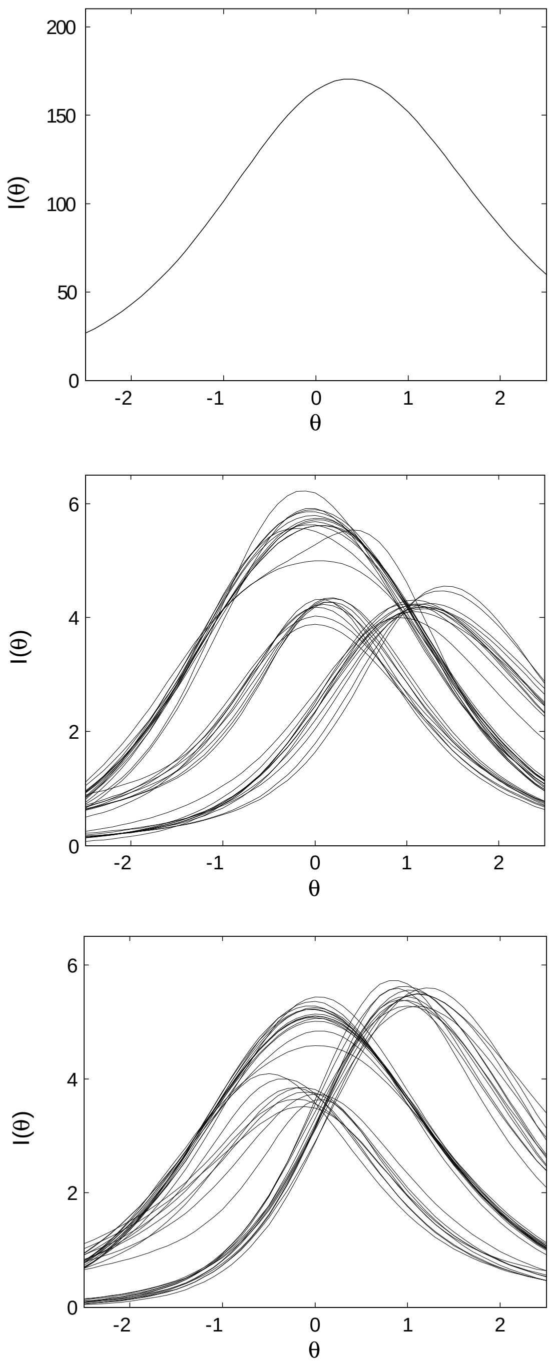

Figure 4.2: Item pool information (top) and testlet information for Condition 1 (middle) and Condition 2 (bottom)

more time, and depended primary on the content constraints required by the test developer. We terminated the program when the difference between the relaxed objective value and the obtained integer value was below 8% (this is a setting in the CPLEX routine termed the tolerance gap). This decision was based on a review of results for a similar assembly problem (van der Linden, Ariel, & Veldkamp, in press), where the gap criterion was in the same range (about 5%).

The maximum number of testlets (each having 25 items) from our pool of 1066 MCQs that we assembled initially was 35 testlets. These testlets used about 82% of the available items. It is possible to increase the number of feasible testlets into 42 (or assign about 98% of the items in the pool), if we relax the bounds on some content constraints (where items were scarce).

Recall, in one solution we wished to see the impact of different weighting for testlet targets. We used the same weights for all testlets, in one condition. In the second condition we applied a higher weight for difficult testlets to increase their total amount of IRT test information. These two formulations of the weights can be seen when the TIFs for testlets are compared (Figure 4.2).

Figure 4.2 (middle) illustrates weight $\omega = 1$ at the midpoint of the ability targets for all testlets. Figure 4.2 (bottom) depicts testlets with weight $\omega = 1$ for average and moderate, and weight $\omega = 1.5$ for high difficulty testlets. Our optimized solutions both meet required content constraints in the test blueprint.

Observe that every testlet of a given difficulty level in Figure 4.2 is closely parallel to others of the same difficulty in statistical characteristics. The TIFs are overlapping. One interesting finding is exemplified when we note the testlets do not always reach their highest information values at the target theta values used to build the testlets (e.g., see the moderate testlets in Figure 4.2). This outcome is a result of our model expression where relative target information is specified as the objective function.

In a MIP solution where relative targets are defined, the resulting testlet TIF shape depends, to a great extent, on item pool information. As shown in Figure 4.2 (top), item pool information peaks around 0.3. When the model has to optimize testlet information at several ability points, as in our case, the algorithm has to distribute the available items equally well to meet all testlet targets, given the available pool.

The results also suggest that weight usage is an effective means to acquire the desired amount of information for particular testlets. As shown in Figure 4.2, the use of a higher weight for high difficulty testlets increased their information. Of course, increasing the information for particular testlets would automatically reduce information for other testlets. Particularly when ability points are shared, the use of weights had the effect of shifting the position of the curves. This was logical because the algorithm has to supply the testlet of higher weight with more informative items, and while the ability points are shared and the lower bound on information has to be maximized, the algorithm will choose alternative items whose information values are lower at these ability points. The consequence of taking these 'alternative items' is that some curves for the (lower weight) testlets peaked at a different theta.

4.5 Discussion

This paper addresses one of the essential areas of any application of computerized MST design, namely the testlets assembly problem. The empirical example illustrates a mathematical programming model approach to practical operational concerns for large scale testing programs. Results can be generalized beyond the computerized MST design, and applied to a variety of psychometric models, including linear administration.

The approach was successful, in the sense that we used an example pool of items and created testlets that exactly met content and other rules for assembly. This mathematical programming technique is a straightforward treatment for the complex and representative problem, where competing constraints exist on testlets assembly.

An important feature of our case study solution is the analysis of maximum testlet construction from the pool. This approach has value as it can be applied prior to an individual assembly to protect test quality over time. This is a benefit of simultaneous assembly that can be used to reserve feasible solutions for later administrations. This feature will be important whenever uniform quality among testlets or tests is required for continuous administrations.

The general approach illustrated here offers a practical solution to a common testing program problem. Similar test assembly problems can be solved in a reasonable time, on common computer equipment using publicly available software to generate solutions using MIP algorithms. Individual settings can be tailored by the test developer for their program objectives and psychometric model. For example, size of build, length of tests, range and weights for targeted difficulty can be customized to best use available test content, while adhering to principles of parallelism and meeting the mandate of a testing program. This study offers some techniques to obtain testlets for MST, in the presence of practical inventory restrictions and constraint rules where test decisions depend on valid test scores.

5 Strategy for Managing Item Pools

Abstract

Item-pool management requires a balancing act between the input of new items into the pool and the use of items in tests assembled from the pool. A strategy for optimizing item-pool management is presented in which the notion of maintaining an optimal blueprint for the item pool is used to tune item production to test assembly. A simulation study for a previous item pool from the Law School Admission Test (LSAT) was used to evaluate the improvement of test assembly over time due to the use of this strategy.
Key words: Item-pool design, item-pool management, optimal design, test assembly.

This paper has been submitted for publication.

5.1 Introduction

An item pool is a collection of test items for a given domain usually stored in computer memory along with a list of codes for their attributes. All item attributes can be classified into three different categories, namely categorical attributes (e.g., item content, cognitive level, and item format), quantitative attributes (e.g., word counts, exposure rates, and item-parameter values), and logical attributes (e.g., relations of exclusion or inclusion in subsets of items) (van der Linden, 2005, chap. 2).

Although item pools are typically calibrated using an item-response theory (IRT) model, for example, the three-parameter logistic (3PL) model (Hambleton, Swaminathan, & Rogers, 1991), their quality depends only partially on the distribution of the item parameters. Key to the performance of a test-assembly algorithm is the match between the composition of the pool and the complete list of specifications for the tests assembled from it.

For example, in spite of their quality, highly discriminating items are not chosen if their content attributes are not the ones required by the test specifications. In an extreme case, if the pool misses one item with a specific combination of content attributes, it can even become impossible to select a valid test from the pool at all. Also, items in the pool with relatively scarce combinations of content attributes are easily overused. The latter problem creates potential item-security risk in computerized adaptive testing (CAT) and leads to the necessity of an adaptation of the item-selection algorithm to prevent items from being overexposed (e.g., Ariel, van der Linden & Veldkamp, 2004; Boyd, Dodd, & Fitzpatrick, 2002; McBride, Wetzel & Hetter, 1997; Revuelta & Ponsoda, 1998; Stocking & Lewis, 1998; Sympson & Hetter, 1985; van der Linden & Veldkamp, 2004).

Suppose that an item pool is optimal for a testing program. If items from the pool have been used for test production, ideally they are replaced by new items with the same statistical and content attributes. Otherwise, the pool may get depleted of sufficiently good items after some time. The strategy for item-pool management studied in this paper is based on this simple principle. In practice, however, a testing program seldom begins with an optimal item pool. The reason often is restrictions on the resources of the testing organization which forces it to begin test assembly while the pool has not yet been completed. Under such conditions, it is difficult to implement the principle. We will show how item-pool management can nevertheless be optimized with respect to it. Our basic idea is not to begin with an actual item pool but with a design of it in the form of an optimal blueprint. The blueprint is periodically updated allowing for the attributes of the new items that were added to the pool during the last planning period as well as the attributes of the items that were used in tests assembled from it. The updated blueprints are used to govern the production of the items needed to complete the pool. Each step in this process is optimized using integer programming (IP) modeling. The notion of a blueprint for an item pool will be made more precise below.

In a testing program with paper-and-pencil tests of a linear format, the tests are frequently assembled from a master pool (Way, Steffen, & Anderson, 1998). For a program with adaptive tests, the tests are usually not administered from a master pool but from an operational pool assembled from it (Veldkamp & van der Linden, 2000). Recently, it has been proposed to assemble this operational pool also as a set of linear tests (van der Linden, Ariel, & Veldkamp, 2005), but even then the adaptive test is administered from an operational pool and not directly from a master pool. The same practice occurs in linear on-the-fly testing and multistage testing, where a pool of exchangeable linear tests or subtests has to be maintained to guarantee uniform quality of testing for all examinees.

In the current paper, the focus is on the management of a pool for a program with linear tests assembled from the master pool. This includes the cases of a program with linear paper-and-pencil tests as well as adaptive, linear on-the-fly, or multistage testing from a pool of linear tests or subtests. Fluctuations in the quality of the master pool immediately translate into fluctuations in the quality of the sets of (sub)tests assembled from it. Optimizing item-pool management to guarantee uniform high quality of testing over time for these types of testing seems therefore important.

In the next two sections, we first explain the notions of item-pool design and an item-pool blueprint and then discuss a strategy for item-pool management based on the latter. In the last two sections, we present a simulation study with this strategy and discuss the results.

5.2 Designing the Item Pool

Our method of item-pool design is basically a nonstatistical extension of optimal design methods as they have been develop in statistics (e.g., Silvey, 1980) and used, for example, to optimize item calibration and ability estimation problems in IRT (e.g., Berger, 1997; van der Linden, 1994). A key notion in optimal design theory is that of a design space. For instance, if the problem is to design an experiment for estimating the parameters in a linear regression equation, the design space for this experiment is the set of possible values for the predictor variables. Typically, the full space is not considered but a subset of points that covers the space in sufficient detail. The design of the experiment is the distribution of the observational units over these design points. A design is optimal if the distribution maximizes the accuracy of the estimators of the parameters.

van der Linden, Veldkamp and Reese (2000) suggested to approach the problem of item-pool design from an optimal-design perspective. The design space in this approach is the Cartesian product of all (categorical and quantitative) item attributes addressed in the specifications of the tests that the item pool has to serve. The objective function that is optimized is not the accuracy of certain statistical parameters but a function that represents the costs of writing the items.

More specifically, suppose the item pool has to serve a program with parallel versions of a set of different tests $t = 1, ..., T$, The tests are from the same pool but differ in some of their specifications (e.g., an easier and a more difficult test or a system of T short tests for use in multistage testing). We use n_t to denote the number of parallel versions needed for test t. This description of a testing program is general. If the program has only one type of test, we set $T = 1$. Likewise, if only one version of test t is needed, we set $n_t = 1$.

Let $\mathcal{D}$ denote the design space for this item pool. The space is defined as the Cartesian product of the categorical and quantitative item attributes used in the specifications for the set of T tests. The individual points in $\mathcal{D}$ are denoted as $d = 1, ..., D$. Each point represents a possible combination of the attributes that an item can have,

$$d\text{=(attribute 1, attribute 2,...).}$$

An example of a design point is (magnetism, four-choice item, presence of a graph, calculus required, 120 words, $b_i = .50$, $a_i = 1.00)$. We will use the term *item blueprint* to denote a combination of attributes at a given design point. Item blueprints can be used to govern the item-writing process. They give item writers more specific instructions than the usual taxonomy, list of learning objectives, or description of a content domain. We expect item writers to be able to write items with the categorical attributes in the blueprint and with such quantitative attributes as word counts. But they will generally have difficulty producing items with prespecified IRT parameter values. We will return to this point later.

Let x_{dt} denote the number of items in the pool at design point d needed to assemble test t. How these numbers can be calculated is addressed in the next section. The total number of items needed in the pool at design point d for the planning period is

$$N_d = \sum_{t=1}^{T} n_t x_{dt}. \tag{5.1}$$

The total number of items needed (that is, the size of the item pool) is

$$I = \sum_{d=1}^{D} N_d. \tag{5.2}$$

We will refer to the array of numbers

$$(N_1, ..., N_D) \tag{5.3}$$

as the design of the item pool. Alternatively, since (5.3) can be viewed as the collection of item blueprints for the pool, we will also refer to it as an *item-pool blueprint*.

The choice of a finite set of points implies a discretization of the quantitative item attributes. This is not a problem as long as we choose a set of values that is typical of the range of the attribute. In fact, the reduction is the same as in IRT-based test assembly when we control a test information function over the range of θ for a population of students at a set of discrete values θ_k, $k = 1, ..., K$, where K typically is chosen to be in the range 3–5. Also, we do not need to address such logical item attributes as relations of exclusion between items ("enemy items"). Constraints on these attributes have to be imposed only when the tests are actually assembled. For these and other issues in item-pool design, see van der Linden (2005, chap. 10).

5.2.1 Design Model

The basic numbers x_{dt} can be calculated using integer programming (IP). In the IP model, x_{dt} are the integer decision variables that tell us how many items we need with the attributes at point $d = 1, ..., D$ for test $t = 1, ..., T$. The variables are used to formulate the sets of specification for the T tests as constraints in the model. As objective function, we use a cost function function based on the cost φ_d of writing an item with the set of attributes associated with design point $d = 1, ..., D$. A suggestion for a cost function is discussed later in this paper.

Our example of a design model is for a program with IRT-based test assembly. Let $I_d(\theta_k)$, $k = 1, ..., K$ be the information at ability level θ_k in the response to an item at design point d. This quantity can be calculated from the values of the IRT parameters in the definition of point d. (If the 3PL model is used, we can adopt a common value for c_i, for example, an empirical average and ignore this parameters in our definition of $\mathcal{D}$.) Suppose the tests $t = 1, ..., T$ should have information functions that satisfy absolute target values $\mathcal{T}_{tk}$ at θ_k, $k = 1, ..., K$, for test t. To show an example of a model with constraints on categorical attributes, we use sets V_c, c=1,...,C, to denote the partition of the design space by the attribute, for example, a content classification (that is, if $d \in V_c$, then d represents a potential item for content category c). Finally, let q_d be the value of an arbitrary quantitative attribute q for an item at design point d.

A general version of the design model for a program with T tests is:

$$\text{minimize} \sum_{t=1}^{T} \sum_{d=1}^{D} \varphi_d x_{dt} \quad \text{(total costs)} \tag{5.4}$$

subject to

$$\sum_{d=1}^{D} I_d(\theta_k) x_{dt} \geq \mathcal{T}_{tk}, \quad k = 1, ..., K,\ t = 1, ..., T \tag{5.5}$$

$$\sum_{d=1}^{D} x_{dt} = l_t, \quad t = 1, ..., T \tag{5.6}$$

$$\sum_{d \in V_c} x_{dt} \leq n_{ct}^{\max}, \quad t = 1, ..., T, \ c = 1, ..., C \tag{5.7}$$

$$\sum_{d \in V_c} x_{dt} \geq n_{ct}^{\min}, \quad t = 1, ..., T, \ c = 1, ..., C \tag{5.8}$$

$$\sum_{d=1}^{D} q_d x_{dt} \leq b_{qt}^{\max}, \quad t = 1, ..., T \tag{5.9}$$

$$\sum_{d=1}^{D} q_d x_{dt} \geq b_{qt}^{\min}, \quad t = 1, ..., T \tag{5.10}$$

$$x_{dt} \in \{0, 1, ...\}, \quad d = 1, ..., D, \ t = 1, ..., T. \tag{5.11}$$

The constraints in (5.5) require the test information functions to satisfy the target values $\mathcal{T}_{tk}$. The constraints in (5.6) specify the lengths of the tests. In (5.7) and (5.8), the numbers of items with categorical attribute c are constrained by lower bounds $n_c^{\min}$ and upper bounds $n_c^{\max}$. Likewise, (5.9) and (5.10) impose lower bound $b_q^{\min}$ and upper bounds $b_q^{\max}$ on the quantitative attributes of the tests. We need to include as many constraints of the types in (5.7)–(5.10) as we have bounds on the categorical and quantitative attributes in the set of test specifications for the tests. The final set of constraints defines the decision variables as integer.

If the tests should be assembled to relative target values $\mathcal{R}_{tk}$, we can maximize the height of the test information function subject to the shape defined $\mathcal{R}_{tk}$ (for the notions of relative and absolute targets for information functions, see van der Linden, 2005, sect. 5.1.1). This is realized if (5.4) and (5.5) are replaced by

$$\text{maximize } \omega y - (1 - \omega) \sum_{t=1}^{T} \sum_{d=1}^{D} \varphi_d x_{dt} \tag{5.12}$$

subject to

$$\sum_{d=1}^{D} I_d(\theta_k) x_{dt} \geq \mathcal{R}_{tk} y \quad k = 1, ..., K, \ t = 1, ..., T \tag{5.13}$$

where y is a nonnegative (real-valued) variable and $0 < \omega < 1$ is a weight for the relative importance of the objectives of minimal item-writing costs and maximal test information. Variable y serves as a common factor in the lower bounds to the test information in (5.13) that is maximized in the objective function.

A solution to a design model is a set of values for the variables x_{dt} that meets all constraints and involves minimal costs. The values can be used to calculate the optimal blueprint for the item pool in (5.3). Solutions can easily be calculated using a general IP solver, for instance, the solver in CPLEX 9.0 (ILOG, 2004).

5.2.2 Cost Function

If direct estimates of writing the items for the combinations of attributes at the design points are available, they should be used in the objective function. Alternatively, we could use subjective estimates of these costs or an empirical estimate of a proxy variable.

A useful proxy to direct costs was recommended in van der Linden, Veldkamp and Reese (2000). Their class of functions was estimated from a previous item pool for the same testing program. Let η_d be the number of items in the previous pool at design point d. The proposed class of functions was

$$\varphi_d = f\left(\frac{1}{\eta_d + \delta}\right), \tag{5.14}$$

with $f(.)$ a monotonically increasing function and δ an arbitrary small constant needed to prevent infinite values at points with $\eta_d = 0$. Since $f(.)$ is monotone and the function is used in an optimization problem, any choice of function in this class leads to the same result.

The direct assumption on which this proxy is based is that items written more frequently in the past are easier to produce and therefore involve less costs. But the choice of (5.14) also has a hidden advantage: It allows us to capitalize on the empirical dependencies between the item attributes in the program, in particular on the dependencies between the IRT parameters and all nonstatistical attributes. As already observed, we expect item writers to be able to produce items with all attributes in the blueprints, except the values of the IRT parameters. Although item writers know how to tell a difficult from an easy item, it is typically hard for them to predict their exact difficulty. This observation holds more strongly for the discrimination parameter. Minimization of the function in (5.14) means preference of the items with combinations of IRT parameter values and the other attributes that occurred more frequently in the past and apparently are easier to write. This strategy may become more effective if the program has fixed item writers and we use these item writers as an attribute in our definition of $\mathcal{D}$. We then capitalize on the specific practice of the individual item writers (van der Linden, 2005, sect. 10.7.2)

If the size of the previous pool is relatively small, the cost estimates in (5.14) are instable and it is recommended to smooth the estimates over the quantitative attributes in the design space. A useful smoothing method is k-nearest neighbor regression, which replaces the estimate at point d in (5.14) by the average found in a small neighborhood of d. This method was used in the empirical example below.

5.3 Managing the Item Pool

When item pools are developed dynamically, with items added to the pool and taken from it to assemble tests continuously, the blueprint in (5.3) should be used as a tool for monitoring and permanent optimization of this developmental process of an item pool rather than in a one-shot approach. In this approach, at appropriate points of time, we compare the actual composition of the item pool with the optimal blueprint and use the comparison to calculate a projection of the additional numbers of items needed at the design points.

The updates consists of the following steps:

1. Establishing the sets of parallel versions of the tests to be assembled in the next planning period;
2. Updating the model to allow for possible chances in the test specifications for the next planning period;
3. Adapting the model to account for the current composition of the item pool (see below);
4. Updating the cost function, using the numbers of items written in the preceding period to reestimate the function in (5.14);
5. Running the model using an integer solver to determine the numbers of new items that have to be written for the next planning period.

The update in Step 3 of this procedure consists of a simple operation. Let p denote the planning period for which the update is needed, $\eta_d^{(p-1)}$ the numbers of items in the pool available at point d at the end of period $p-1$, and $x_{dt}^{(p)}$ the decision variables for the additional numbers of items in the update of the model in (5.4)–(5.11) for period p.

Optimal values for $x_{dt}^{(p)}$ can be calculated using the updated version of the model with the constraint

$$\sum_{t=1}^{T} x_{dt}^{p} \geq \eta_d^{(p-1)}, \quad d = 1, ..., D. \quad \text{(current item pool)} \tag{5.15}$$

added to it. This constraint is necessary to account for the attributes of the current items in the pool. (It thus also accounts for the actual values of the item difficulty and discrimination parameters produced by the item writers!)

The projection of the additional numbers needed for p follows from the following updates of (5.1) and (5.3):

$$N_d^{(p)} = \sum_{t=1}^{T} n_t^{(p)} x_{dt}^{p} \tag{5.16}$$

and

$$(N_1^{(p)}, ..., N_d^{(p)}). \tag{5.17}$$

5.4 Simulation Study

The goal of this study was to assess the impact of the management tool above on the quality of a testing program. The quality was measured both by the average costs per written item and the maximum number of tests that could be assembled from the pool. Since these two variables are also dependent on the quality of the initial item pool and the quality of item writing, these two factors were also varied in this study. For the evaluation of the measurement quality of the tests assembled in this study, see the end of this section.

The study was run in the form of a computer simulation of several scenarios for a fictitious testing program for one of the section of the Law School Admission Test (LSAT). The program was assumed to consists of six testing periods per year, with six versions of the test needed for each period. The size of the item pool was fixed at 350 items. The test-assembly model consisted of 40 constraints on such attributes as test length, content distribution, and word count. The test versions were not allowed to have any overlap in items. The set of test versions for each testing period was assembled simultaneously; that is, we first tried to assemble a set of six versions. If this problem was not feasible, we tried to assemble a set of five, and so forth.

The cost function was the one in (5.14) estimated from a previous pool of 2,436 items. The function was smoothed using the regression method referred to above. All items had been calibrated using the 3PL model but we omitted the guessing parameter in the definition of the design space and used the average value for this parameter in the previous pool to calculate the values of the item-information functions in the design model. The test-information function was controlled using the actual bounds for the function for the LSAT at $\theta = -1.2$, 0.0, and 1.2.

The size of each test was 50 items. Once an item was used in a test, it was permanently removed from the pool and replaced by a new item. Each new period thus began with a complete pool of 350 items. In principle, the maximum number of forms possible from the pool was thus seven per testing period. However, to make the study realistic we simulated attrition of the item pool due to items becoming outdated, showing parameter drift, detection of flaws in their formulation, and so on. Way, Steffen and Anderson (1998) report that the number of such items should be expected to be approximately 5% of the inventory. This percentage was used in our study.

Table 5.1. Eight different scenarios in simulation study.

Scenario	Initial Pool	Management	Item Writing
1	+	+	+
2	+	+	-
3	+	-	+
4	+	-	-
5	-	+	+
6	-	+	-
7	-	-	+
8	-	-	-

The design of the study was the fully balanced three-factor design in Table 5.1. Each factor had two levels which we classified as optimal $(+)$ and less than optimal $(-)$. More specifically, the levels were implemented as follows:

1. *Initial pool*. The optimal initial pool was assembled using a blueprint calculated according to the model in (5.4)–(5.11), whereas the less than optimal pool was randomly drawn from the previous pool.

2. *Management*. Optimal item-pool management was based on the dynamic version of the design model with a new update for each testing period, as proposed in this paper. For the conditions with less than optimal management, we sampled the blueprints for the new items from the old item pool. That is, for each new item needed, we sampled an item from it, classified its attributes, and used the list of attributes as the blueprint for the new item. This level of management simulates a continuation of the current practice without any explicit attempt to improve or even intervene.

3. *Item writing*. It was assumed that all content attributes in the item blueprints were realized but the authors had two levels of command as to the quantitative attributes. For both levels we perturbed the attribute values in the blueprint randomly, but used a level of random error for the less than optimal conditions approximately twice as large as for the optimal conditions. For example, the random error in the difficulty parameter was uniformly in [.15–.20] for the optimal level of item writing, but in [.20–.30] for the less than optimal level.

Each combination of levels of these three factors was implemented using the following steps:

1. The initial pool was assembled;

2. The set of tests for the first period was assembled simultaneously;
3. The items selected for a test were removed from the pool. Also, from the remaining items 5% was removed to simulate attrition;
4. The blueprint for the items to be added to the pool for the next period;
5. New items were written to this blueprint by randomly disturbing the values of the quantitative parameters in the blueprints.

The simulation of each scenario was repeated five times, and the results were averaged to get stable estimates of the impact of the scenario on the item-writing costs and the number of feasible tests for each period. It should be noted that each test assembled in this study was constrained by the actual bounds on the test-information function for the LSAT. Thus, the number of feasible tests in a scenario also evaluates the measurement quality made possible by the item pool.

5.4.1 Results

The results are presented in Table 5.2. Of course, the case with all three factors at optimal level (scenario 1) yielded the best results both for the number of feasible tests and the average cost per item.

Table 5.2. Number of feasible test forms and average costs per item in each planning period for the eight scenarios.

	Initial Pool: Optimal							
	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Period	Forms	Costs	Forms	Costs	Forms	Costs	Forms	Costs
1	6.0	.17	6.0	.17	6.0	.17	6.0	.17
2	6.0	.17	6.0	.50	4.6	.47	3.8	1.63
3	6.0	.17	6.0	.38	3.3	.47	2.0	1.73
4	6.0	.17	6.0	.50	2.2	.47	1.1	1.45
5	6.0	.17	6.0	.37	2.2	.49	1.0	1.56
6	6.0	.17	6.0	.42	1.6	.51	0.4	2.13
	Initial Pool: Less Optimal							
	Scenario 5		Scenario 6		Scenario 7		Scenario 8	
Period	Forms	Costs	Forms	Costs	Forms	Costs	Forms	Costs
1	5.0	.48	5.0	.48	5.0	.48	5.0	.48
2	3.3	.17	3.0	.39	3.4	.48	2.6	1.71
3	4.4	.19	4.0	.65	2.6	.47	1.8	1.09
4	5.2	.18	4.2	.66	2.0	.51	1.2	1.36
5	5.4	.18	4.4	.41	1.1	.45	1.0	1.26
6	6.0	.17	3.0	.53	1.1	.50	0.2	1.53

For the optimal pool, the main impact of optimal management but less than optimal item writing (scenario 2) was an increase in the item-writing costs, whereas, for the converse case of less than optimal management but optimal item writing (scenario 3), the main effect was a systematic decrease of the number of feasible tests. When the less than optimal levels of management and item writing were combined with a less than optimal initial item pool (scenarios 6 and 7) both effects occurred. A strong interaction effect between item-pool management and item writing was observed both for the optimal initial item pool (scenarios 1-4) and the less than optimal initial pool (scenarios 5-8). As a matter of fact, for these two cases, it was no longer possible to compensate bad management by good item writing (or bad item writing by good management), and both the average costs and number of feasible tests deteriorated quickly.

5.5 Concluding Comments

The previous results are from one study only, and we should be careful not to overgeneralize. Nevertheless, we believe they underscore the importance of high quality item-pool management. For a testing program, we expect the number of feasible tests per testing period to be the most important success parameter. As Table 5.2 shows this number was always considerably higher for the cases with optimal management (scenarios 1, 2, 5, and 6) than with less than optimal management (scenario 3, 4, 7, and 8). The strong interaction between item-pool management and item writing in Table 2 also points at the critical importance of good management. It turned out to be impossible to compensate bad management by good item writing.

Further, the results show that once the number of feasible tests start to decrease for a scenario, it tends to keep decreasing. This makes sense because of the cumulative nature infeasibility over time. If a set of tests is assembled, the good items in the pool disappear, while the bad items remain. As a result, the likelihood of a lower number of feasible tests in the next period increases.

The last results thus confirm the observation in (Way, Steffen, & Anderson, 1998) that item pools tend to decrease in quality over time , and, consequently, an increasing number of new items have to be produced to maintain the testing program. The management tool presented in this paper is one instrument that could be used to break this tendency.

6

Unraveling the Causes of Underexposure in Automated Test Assembly

Abstract

Not all items in the pool can be selected by test assembly algorithms; these items are referred to as underexposed items. The research on item pool management described in the previous chapter suggests that if the attributes of underexposed items are undesirable, they would still not be chosen in the next assembly. For item pool management, it would be of interest to know why some items in the pool are underexposed. In this paper, we explore some possible methods to unravel the causes of underexposed items, and one of the methods is used in the empirical study. For this case, the results indicate that there is a strong interaction between the attributes of items critical to feasibility and the attributes of other items. Critical items always have to be selected to yield feasible tests, therefore their attribute values caused underexposure of the other items.
Key words: Underexposure, test assembly, item pool management.

6.1 Test Assembly

Test assembly can be formulated as an instance of combinatorial optimization. The general formulation of an optimization problem consists of two components, namely a set of constraints and an objective function. Both components are linear functions of decision variables. The goal of optimization problems is to find values for decision variables, also refered to as solutions, that satisfy the constraints and optimize the objective function. In a typical automated test assembly (ATA) problem, to decide whether an item will be taken for the test or not, decision variables are binary. ATA can be modeled in various ways, but in general it takes the following form:

$$\text{Maximize} \sum_{i=1}^{I} c_i x_i \tag{6.1}$$

subject to the constraints:

$$\sum_{i=1}^{I} a_{ij} x_i \leq b_j, \quad j = 1, 2, ..., J \tag{6.2}$$

$$x_i \in \{0, 1\}, \quad i = 1, 2, ..., I \tag{6.3}$$

where c_i represents the contribution value of item i to the objective function (e.g., item information), a_{ij} the attribute value of item i with respect to attribute j (e.g., categorical attribute such as item type, quantitative attribute such as word count, and logical attribute such as item-enemy), and b_j the bound on the constraint related to attribute j (e.g., total number of item on a certain category, total word count, expected response time).

A feasible solution to this problem is a set of items that meets all constraints. It is possible to have more than one feasible solution. However, for an optimization problem, the main goal is to find an *optimal solution*, i.e., a feasible solution that yields the best value for the objective function.

Whether or not an item is selected for a test depends on several factors such as the attribute values of an item, the attribute values of the other items, and the methods used to assemble the test. An item might not be chosen if it has undesirable attributes value (e.g., a very high or very low difficulty parameter, too many or too few words) or if there are other item-combinations that perform better with regard to the objective of the test. Likewise, an item might be chosen if it has attribute values that are less frequent in the pool while this attribute is needed for the test. The test assembly method also plays a role. If the tests are assembled one by one, instead of simultaneously, the best items will be taken first, and some items that otherwise could have been taken will be left in the pool.

These factors can be related to each other, and therefore, it is often not possible to determine beforehand which item will be taken for the test.

6.1.1 Underexposed Items

Items that are seldom or never selected for the test are referred to as underexposed items. Little is known about these items in the pool. Research on item pool management (Ariel, van der Linden, & Veldkamp, 2004; Way, Steffen, & Anderson, 1998) suggest that if the attributes of underexposed items are undesirable, they would still *not* be chosen in the next assembly when new items have been added to the pool. Besides, if they were not chosen because other item-combinations performed better, they will still not be chosen in the next assembly when the new items have the same quality as the retired ones. In the simulation study, Ariel, van der Linden, and Veldkamp showed that the quality of the item pool tends to deteriorate even when desirable items are added into the pool. Recovery is only possible when desirable items are added continuously to the pool. This may be hard to do; in practice, we typically see a loss of quality over time that has to be compensated by the production of large number of items (Way et al., 1998).

For item pool management, it would be interesting to know why some items in the pool are underexposed. In this study, we will explore some possible methods to unravel the causes of underexposed items. One of the methods will be applied in the empirical study, and it will be shown how the results can be used to improve item pool management.

6.2 Some Methods to Analyze Item Pools

If there is no solution for the assembly problem, the problem is infeasible. (That is, a test that meets all specifications does not exist.) There are several possibilities why a solution does not exist. For example, it can be impossible because of conflicting constraints, errors in item-code, and deficiency in the item pool (Huitzing, 2003). Practical assembly problems are often confronted with numerous constraints. It is not easy to identify which constraints cause infeasibility.

In order to obtain one or more tests, assuming that there is no mistake in the model formulation, nor in item-coding, items in the pool have to be replaced or the test specifications need to be modified by relaxing some constraints. When changing the test specifications is impossible, it becomes necessary to improve the quality of the item pool. Analysing and evaluating an item pool is one step needed to reach this goal.

In test assembly, a group of items needs to be selected, instead of a single item. Whether an item will be selected also depends largely on the attribute values of other items in the pool. If the attribute values of the rest of the items are poor, even an item with superior attribute values might not be chosen. In other words, item pool composition plays a key role. The success of an automated test assembly is determined by the quality of item pool as well. If the item pool is poor, no method will be capable of assembling the tests.

Ideally, an item pool should contain enough items for each content category. It is also preferable for the item difficulty distribution to match the distribution of the examinees' abilities. Other quantitative attributes such as word count and expected response time should also be adequately represented by items in the pool, so that requirements on these attributes can be met. To know whether an item pool is ideal, test assembly can be utilized. The results of the assembly indicate whether a given pool is able to produce feasible tests.

In reality, item pools suffer from unbalanced item usage. Some items appear to be favored by a test assembly algorithm. (If a small number of items is highly exposed, it will threaten the security of the testing.) Another consequence is that the pool cannot yield the expected number of test forms. In many cases, it will be beneficial to evaluate the pool; to find out, for example, which items are critical to the feasibility, or why some items are not selected for tests, or why the pool cannot produce a certain number of tests. Based on this evaluation, attempts can be made to improve the quality of the pool. Various methods can be applied to analyze an item pool.

6.2.1 Infeasibility Analysis

Infeasibility analysis is a mathematical technique to diagnose why a feasible solution cannot be found for a certain ATA model. The main focus of this analysis is on finding a smallest set of constraints that together cause the infeasibility, but for which any proper subset gives a feasible model. This set is known as an irreducible infeasible set or IIS. To yield a feasible model, one or more appropriate constraints in an IIS should be changed or removed from the model. In many cases, an infeasible model has more than one IIS with possibly overlap of constraints among IISs.

It is not easy to find an IIS, especially if the model is complex, such as when its variables are integer or binary. A general approach to finding IISs is to solve the infeasible model iteratively by observing how the elimination of one constraint affects the infeasibility, until a smallest number of constraints that cause infeasibility is found. Such approach is called *filtering*; many variations of filtering exist. This approach can be burdensome because the order of the constraints eliminated from the model can influence the speed of finding an IIS. Another approach is called the additive method, in which the constraints in an infeasible model are ordered. Detailed explanations for both algorithms can be found in, for example Guieu and Chinneck (1999). For automated test assembly problem, several studies have been conducted to develop more practical methods to find IISs (e.g., Timminga, 1998; Huitzing, Veldkamp, & Verschoor, 2005).

Fundamental in infeasibility analysis are the constraints, and not the objective function, as they shape the feasible region. For an integer or binary model, it may be possible that the feasible region does not include integer values; the model is then infeasible. Or, some constraints may conflict so that a feasible region does not exist. Huitzing (2003) observed that one possibility why the feasible region cannot be formed is because of the lack of contributions of item-attributes to some constraints. In other words, solutions often cannot be found because the item pool cannot provide certain items sufficiently to satisfy the test specifications.

Infeasibility analysis can be employed to analyze an item pool. Particularly, infeasibility analysis is useful when applied to the portion of an item pool that remain after all feasible tests have been removed. It then helps us to find out which constraints in the test assembly are critical with respect to feasibility. Because constraints are governed by item-attributes, finding IISs from these items may tell us which item attributes are critical to feasibility. Thus, it may be possible to unravel the cause of underexposed items by applying infeasibility analysis to them.

6.2.2 Discriminant Analysis

Discriminant analysis is a statistical technique that can be employed to discriminate between groups, on the basis of some characteristics or variables. The basic quantity is a function known as a *canonical discriminant function*. It can be written as follows.

$$f_g = \alpha_{g0} + \sum_{j=1}^{J} \alpha_{gj} z_j \tag{6.4}$$

where f_g is the discriminant function for group g, α denotes the coefficients, z denotes the variables, and J is the number of variables. The number of functions is determined by the number of groups (minus one) or variables, whichever is smaller. Suppose we have more than two groups and variables. For the first function, α should be derived in such a way that the group means on the function are as different as possible. The same idea is also used to derive the coefficient α for the second function, given the condition that the first and the second function should not be correlated, and so on. In other word, the coefficients should make every function orthogonal. The literature on discriminant analysis provides some guidance on how to obtain the coefficients (e.g., Klecka, 1980; McLachlan, 1992).

The distribution of group means with respect to the discriminant functions can be observed by plotting their values in the space spanned by these functions. In this way, one can easily examine group differences visually, if the number of functions is lower than four. In general, not all functions are useful, and in many cases, two or even one function is sufficient (Klecka, 1980).

As for the case of item pool analysis, selected and unselected items can be separated into different group, while the item attributes are used as variables. Note that the inclusion of many variables in the functions does not always guarantee discriminating power of them, since there are variables that are actually redundant and can complicate the analysis. Several procedures exist to determine which variables are potentially useful. One possible use of discriminant analysis is to reveal which variables are the most powerful discriminators. Therefore, it can be used to reveal which item attributes play important role in determining whether or not an item is selected. Furthermore, because the discriminant function could explain the difference between selected and unselected items, it may help us to reveal the causes of underexposed items.

6.2.3 Sensitivity Analysis

Sensitivity analysis is a method to observe how the variations in the input data change the optimal solution. This analysis is motivated by the fact that the data might change over time, for example, unpredictable quality of some input data.

The application of sensitivity analysis to item pool analysis is relatively straightforward. In this case, one needs to have a blueprint for the pool that is considered as ideal. This ideal pool does not necessarily have to be an existing pool. For example, one can use an item-pool blueprint (van der Linden, Veldkamp, & Reese, 2000; Veldkamp & van der Linden, 2000) as an ideal pool. An item-pool blueprint specifies which items should be in the pool, in order for the item pool to function optimally.

The composition of a given pool can then be compared to that of an ideal pool. Certainly, it is desirable to have a close resemblance between the two pools. A standard function for distance measurement can be used to observe the differences between a given pool, p, and an ideal pool, b, and can be formulated as follows:

$$\delta_{pb}(q) = q_p - q_b \tag{6.5}$$

where $\delta_{pb}(q)$ is a function of the difference between pool b and p for item-attribute q, q_b the number of item in pool b with respect to attribute q, and q_p the number of item in pool p with respect to attribute q.

In reality, a pool may be far from ideal $(\delta_{bp}(q) \neq 0)$. However, this comparison enables us to identify possible cause of underexposed items. A positive value of $\delta_{pb}(q)$ shows that, for attribute q, its number of items in the given pool is abundant, and test assembly algorithms will not have difficulty to select items to meet test specifications for attribute q. When the number of $\delta_{bp}(q)$ is much larger than expected, many items with attribute q will not be selected. On the other side, the negative value of $\delta_{bp}(q)$ indicates that the pool lacks items with attribute q, which makes such items become critical to feasibility. Veldkamp (2005) has illustrated the use of item-pool blueprint to identify critical features on item pool.

Sensitivity analysis calculates several variations for the value of $\delta_{pb}(q)$, and their impact on the optimal solution; in this case optimal can be seen as a minimum number of underexposed items in the pool. In a more advanced setting, sensitivity analysis can measure possible interaction (Saltelli, Tarantola, & Chan, 1998), not necessarily linear, among input parameters, in this case item-attributes.

6.2.4 Item-Attribute Analysis

One can also assess the quality of an item pool by means of optimal test assembly. The results of test assembly are, for example, a maximum number of feasible tests that the pool can produce and the test information value. The maximum number of feasible tests shows how many items in a given pool can be of use; thus it reveals whether the pool is efficient.

To analyze an item pool, it is important to observe which item attributes contribute to feasibility and to optimality, relative to other item attributes in that pool. This is because a feasible solution is constructed by a linear combination of item attributes values, not by a single combination of item attributes. When an item has an undesirable attribute value (or, an undesirable combination of attribute values), the question is whether other items in the pool can compensate for this undesirable value. To verify which attribute values are critical for feasibility or optimality, the test assembly problem can be solved by constraining item i $(i \in I)$ to be included to test t $(t \in T)$, namely by adding the following constraint to the test assembly model:

$$\sum_{t=1}^{T} x_{it} = 1. \tag{6.6}$$

On the other hand, an item may have an attribute value that cannot be easily compensated by other item attributes. This happens, for example, when the test specifications demand a certain number of items with such an attribute, but their number in the pool is low. In this case, such items are said to be critical to feasibility, and the success of the assembly depends on the availability of this kind of items. To investigate this possibility, the test assembly problem can be solved by constraining item i to be excluded from test t $(t \in T)$, namely by adding the following constraint to the test assembly model:

$$\sum_{t=1}^{T} x_{it} = 0 \tag{6.7}$$

The results of both approaches can be evaluated using feasibility and optimality criteria. They can be plotted as a function of the item attributes, to observe which attribute contributes significantly to each criterion. In the current study, we will use this analysis method.

6.2.5 Discussion

All methods discussed in this section attempt to analyze item pools. Although they differ systematically, they all examine combinations of item-attribute values to find out the causes of underexposed items. Actually, the strengths and the weaknesses of the pool should be evaluated based on combinations of item-attributes. For example, in the discriminant analysis, it is clear from Equation 6.4 that it is almost impossible to examine only one item attribute, as the discriminant function requires a (linear) combination of item attributes to judge which item-attribute is a strong discriminator.

There are also methods that can be used to judge which item-attribute plays an important role in determining whether or not an item will be selected. Example of such methods includes artificial learning methods, such as neural networks, which are trained to recognize the characteristics of an item pool that are needed to function efficiently. For the methods to learn well, wide variations on the quality of pools are necessary. One difficulty with the application of such methods is that it is required to extract some knowledge (Andrews, Diederich, & Tickle, 1995) to understand the mechanism underlying such methods, since most of them work as a black box.

6.3 Empirical Study

In this study, item-attribute analysis was used. On one hand, we would investigate what would happen if one item was forced to be taken for the assembly. Particularly, we would like to know how serious it would affect the assembly model. Two possibilities that could happen were (1) the model would be infeasible, or (2) the objective value would vary. On the other hand, it would be also necessary to compare the results by forcing one item to be out from the assembly model, to find out whether an item was critical for the feasibility.

As an illustration, the general model in (6.1)-(6.3) is detailed in the following test assembly model (van der Linden & Boekkooi-Timminga, 1989).

$$\text{Maximize } y$$

subject to:

$$\sum_{i=1}^{I} x_{it} * I_i(\theta_k) \geqslant y, \quad \forall t, \forall k \tag{6.8}$$

$$\sum_{i=1}^{I} x_{it} = n, \quad \forall t \tag{6.9}$$

$$n_{(r)} \leqslant \sum_{i \in C} x_{it} \leqslant n^{(r)}, \quad \forall t \tag{6.10}$$

$$q_{(r)} \leqslant \sum_{i=1}^{I} x_{it} * q_i \leqslant q^{(r)}, \quad \forall t \tag{6.11}$$

$$\sum_{t=1}^{T} x_{it} \leqslant 1, \quad \forall i \tag{6.12}$$

$$x_{it} \in \{0, 1\} \tag{6.13}$$

$$y > 0 \tag{6.14}$$

where lower bound on test information was maximized at every theta value for each test t (Constraints 6.8), each test had exactly the same number of items (Constraints 6.9), upper and lower bound for item categorical attributes C and quantitative attributes Q were defined (Constraints 6.10 and 6.11, respectively), no item was shared among the tests (Constraints 6.12), and decision variables were binary to denote whether an item i was selected for test t (Constraints 6.13). Categorical attributes used in the simulation study were item type and answer key, while quantitative attribute was word count.

In this research, the criterion for feasibility was the maximum number of tests that can be assembled. It was required to assemble all tests simultaneously, so that every single test would gain similar quality. These ways would ensure optimal item usage (the number of underexposed items would be low). With the advent of computer and computational softwares, it is doable to assemble a maximum number of tests simultaneously (Ariel, Veldkamp, & Breithaupt, 2004).

Tests were maximized at three theta values, namely at -1.2, 0, and 1.2. This model was solved in AIMMS 3.4 (Paragon Decision Technology) using *CPLEX 9* (ILOG) as an MIP solver. The operational item pool had 300 items, and four feasible test forms could be assembled with an optimal y value equal to 21.87. The test length was equal to 50 items. Based on this initial assembly, 200 items (50x4) were selected for the tests whereas 100 items were not. Based on the assembly results, the items in the pool were divided into two exclusive sets, to separate items that were taken for the tests from those that were not. Specifically, we assigned unselected items to K_1 and selected items to K_2. One interesting benefit of this approach was that underexposed items (K_1) was clearly identified. Two different conditions were examined in this study, each with different constraints added to the model.

Condition 1 Solve the assembly model by forcing the inclusion of item i, $i \in K_1$

$$\sum_{t=1}^{T} x_{it} = 1, \quad i \in K_1 \tag{6.15}$$

Condition 2 Solve the assembly model by forcing the exclusion of item i, $i \in K_2$

$$\sum_{t=1}^{T} x_{it} = 0, \quad i \in K_2 \tag{6.16}$$

6.4 Results

The maximum number of feasible test forms and the value of the objective function were used to observe the impact of constraining an item to be in or be out of the tests. Specifically, we examined the relationship between each of the item attributes (item discrimination, item difficulty, word count, and item type) and these criteria. Figure 6.1 shows the relationship between item attributes and the number of feasible forms, and Figure 6.2 between item attributes and the objective value.

In Figure 6.1, for Condition 1, where every item not taken was forced to be taken in the assembly, the assembly model was still feasible regardless of whatever kind of item included into the tests. However, for Condition 2, where every item taken was forced not to be in the assembly, the exclusion of some items cause infeasibility. From Figure 6.1 we can see immediately that the type of item played a role in determining whether the model was feasible. The fact that item type 4 was always chosen for the tests showed that this item type was critical in influencing the feasibility. The exclusion of this item type always resulted in infeasibility, i.e., a lower number of feasible test forms.

In Figure 6.2, *only* the objective values from four feasible test forms are shown, in order to provide a fair comparison. This is because, when the number of feasible forms is lower, the objective value becomes higher as the algorithm has more choices in selecting which items can contribute better to the objective function. In this study, the objective values from the infeasible problem (i.e., only able to produce three feasible test forms) was approximately 26.12.

The inclusion of items not taken in to the tests (Condition 1) deteriorated the objective value. The objective value showed a linear relationship with item difficulty; the easier the item, the lower the contribution to the objective value. However, the results indicated a relatively small (about 0.04) difference between the objective values. For Condition 2, both item discrimination and item difficulty influenced the objective value. When a high discriminating item was excluded from the tests, the objective value dropped to a lower level. The same also happened when an item of relatively high difficulty was excluded. Unlike Condition 1, the differences among the objective values were quite high (approximately 0.6).

6.5 Discussion

In this study, the optimal solution gives the highest value for the test information, because items with better quality are selected. However, the results for Condition 1 show that the inclusion of underexposed items can still produce a feasible solution. Although their inclusion deteriorates the objective value, the results showed that the differences from optimal solution were relatively small (optimal value is 21.87, while ’the worst value’ is 21.83). This may suggest that underexposed items can be used for the tests, for instance, to save the better items for later use.

On the other hand, there are items that appear to be critical for feasibility. Omitting one of them can lead to infeasibility, as was shown by the results for Condition 2. Because the test assembly algorithm always had to select critical items, their other attributes influenced which other items in the pool could be selected. For example, if critical items had a relatively high word count, items with low word count would have a higher chance to be selected.

The results indicated that the attribute values of critical items may have caused underexposure of the other items. Different reasons exist for why certain items become critical. For example, poor item writing practices, possible interaction within item attributes, or less optimal item pool management. To reduce the number of underexposed items, it seems reasonable to consider limiting the impact of critical items. For example, by ensuring that they are well represented in the pool. Intuitively, this requirement somehow resembles the necessity to balance between demand (test specifications) and supply (item pool composition). When there is no balance between demand and supply, there could be a strong interaction between those item attributes that are in surplus and those that are deficient. The task of item bank management is to take these dependencies into account and to balance the composition of the item pool.

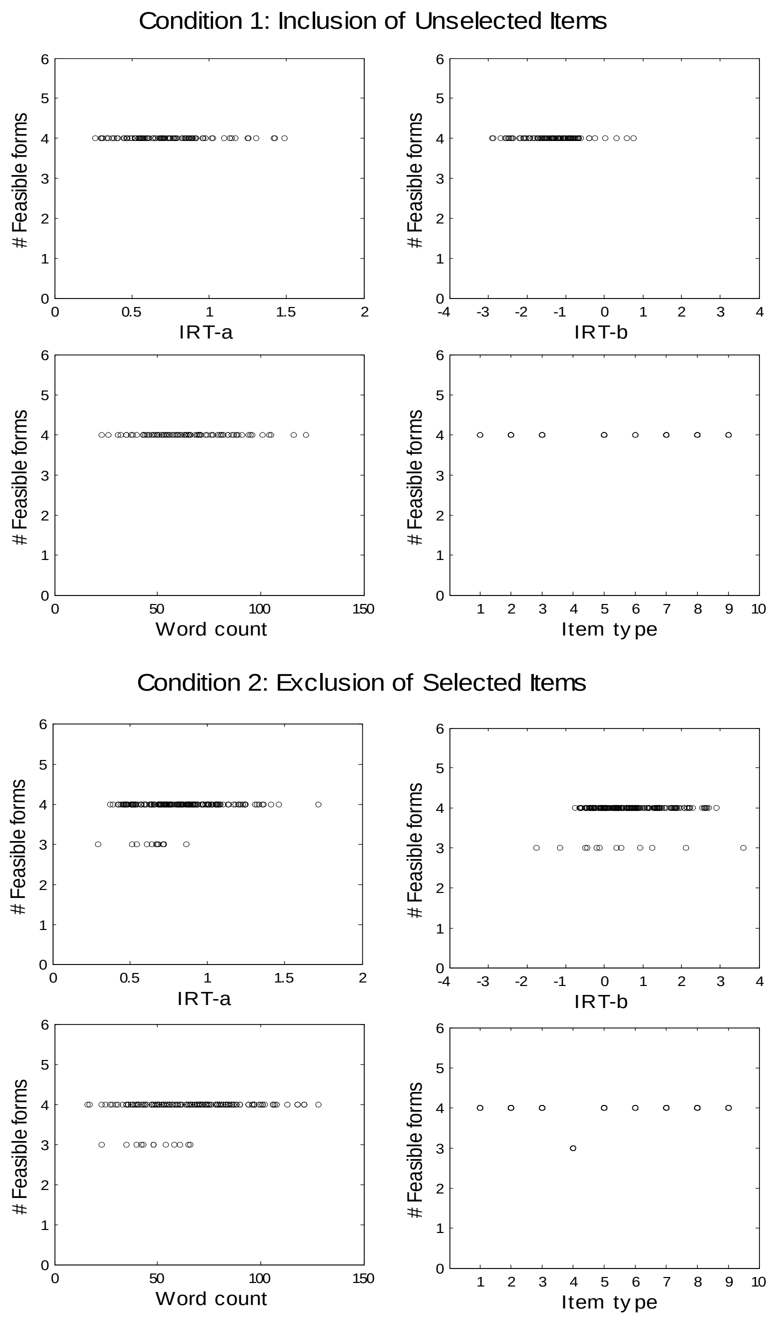

Figure 6.1: Plots of item attributes with respect to the number of feasible test forms.

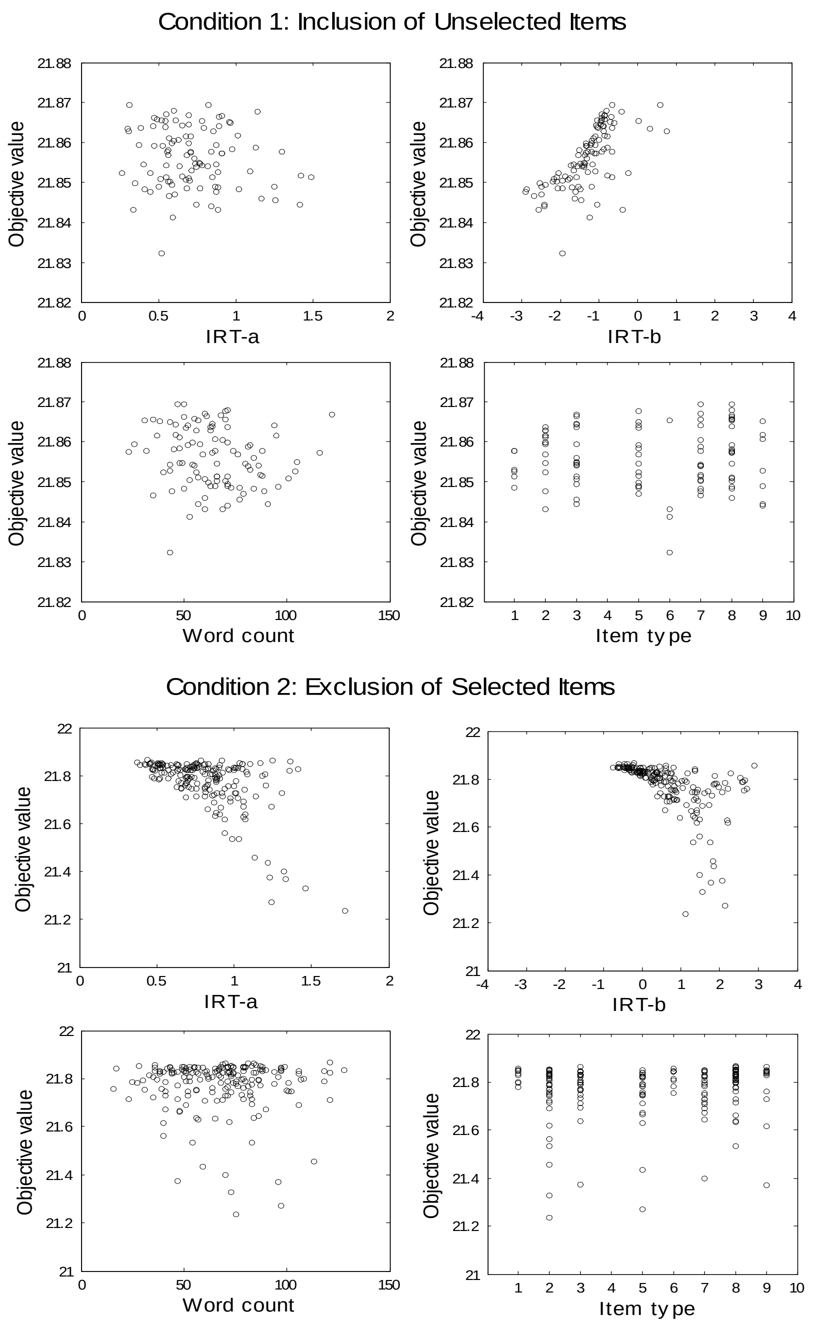

Figure 6.2: Plots of item attributes with respect to the objective value.

Summary

An item bank is a collection of items (test questions) along with their psychometric parameters, such as item-discrimination, item-difficulty, and item-guessing (for the 3PL IRT model), and other attributes such as word counts, categorical attributes, expected response times, and the dates of administration. From an item bank, tests are assembled according to test specifications. Mathematical programming techniques or heuristic methods can be employed to select items from the bank. Because of this, item bank is a valuable resource for any testing program whose tests are offered on a continual basis.

It is desirable to optimize item bank usage for two reasons. First, for security reasons. The continual use of portions of the bank will increase the exposure of its items and, in turn, threat the integrity of the item bank. Second, for efficiency reasons. Much effort and resources have been invested in writing, producing, pretesting, and calibrating the items, and therefore it is important to get a maximum return on the investment.

However, in practical testing situations, it is often impossible to use all items for the tests. Especially when the tests have to meet a certain quality level, for example, high information over a specific abilities range, only items qualified for this level will be selected. Forcing the use of items of lower quality will deteriorate the quality of the tests. The value of the test information function (TIF), the bias and the variance of the ability estimates are the common criteria to measure the quality of the tests.

The focus of item-bank management is on how to optimize item usage while ensuring that the tests' quality level will still be acceptable. This thesis proposes various methods to improve item-bank design and management.

Automated test assembly (ATA) can help test specialists to address some of these concerns. For paper-and-pencil tests, ATA methods focus, for example, on the assembling of multiple parallel test forms. For computerized adaptive tests (CAT), ATA methods are manipulated in such a way that the items will not be selected beyond a tolerable rate. (The last methods are known widely as item-

exposure control algorithms.). In Chapter 1, general test assembly methods for various testing formats are described.

However, many item-exposure control algorithms focus only on items with superior quality, as they tend to be chosen for the tests. In Chapter 2, we demonstrate the use of rotating item pools, as an alternative way to promote more uniform item usage. The basic idea underlying this approach is that the number of popular items should be limited in the pool, in order to promote the use of less popular items. In this approach, a master pool is divided into several parallel subpools. These pools are rotated among the testing sites to realize desirable exposure rates for the items. Gulliksen's method (1950), in which a test is split into two parallel halves, is adopted to ensure that the resulting subpools possess the same psychometric quality. Furthermore, we also demonstrate the possibility to construct overlapping subpools, in which some of the items are shared. The results show that overlapping subpools dramatically increase item usage. Though this benefit is obtained with larger errors for the ability estimates, it can easily be compensated, for example, by an increase in the length of the test.

To a large extent, the composition of an item bank determines whether its items can be used efficiently. If the item bank contains many flawed items, or items with less desirable attributes, no test assembly method will be capable of improving item usage. Especially for adaptive testing, where items are selected individually, the selection algorithms tend to magnify the effect of small differences in the value of the attributes of items. Even when this problem can be lessened through the use of exposure control algorithms, there will always be a trade-off between item exposure and measurement precision. Our resolution toward this problem is to design a CAT bank such that it contains only items that are needed. In Chapter 3 we show that a shadow-test approach can be utilized to construct a CAT bank. A shadow-test approach is an item-selection method for a CAT where a set of items (a linear test) is assembled at every ability estimation but only one item from it is administered. In this approach, the problem of selecting an adaptive test is viewed as the assembly of a series of linear tests, and therefore, it is reasonable to construct the CAT's pool as a set of linear tests. An empirical example with a master pool from the Law School Admission Test (LSAT) yielded a CAT with nearly uniform bias and mean-squared error functions for the ability estimator and item-exposure rates that satisfied the target for all items in the pool.

The need to strictly control the content and the quality of a test has given rise to the practice of multistage testing (MST). MST is adaptive testing with a predetermined number of paths. In this testing, instead of administering a single item, a set of items (testlet) is administered at every stage of ability estimation. The limited number of paths enables the test specialists to inspect carefully the content and the quality of the tests before they are administered. Testing programs with this format require a large number of testlets to provide parallel versions of the tests. This is important to secure the test.

In Chapter 4 we demonstrate the transformation of an item pool to a testlet pool. The assembly of testlet pool is similar to the assembly of a set of maximum numbers of parallel linear tests. Such an approach can promote more uniform item usage, and when the assembly is done simultaneously, every testlet will have a comparable quality. A portion of the testlet pool will be operational, while the rest can be used for, in combination with new pretested items, future test assembly. In a simulation study, we demonstrate the use of different weights to control the importance of some testlets, with regard to their difficulty level.

In general, operational testing programs suffer from fluctuations in the quality of their item banks. Fluctuations are unavoidable, as new items are added and some items are removed (either permanently or temporarily). Items are removed because they have been used frequently, or because they appeared to have desirable attributes for the tests. In addition, some items are removed if they show parameter drift or become obsolete. The fluctuation in the quality of an item bank makes the task of maintaining continuous testing of uniform quality hard to achieve. This situation has motivated us to develop a tool to maintain the quality of an item bank. In Chapter 5, we use a simulation study to observe the efficacy of an item-pool blueprint in maintaining the quality of a pool over several testing periods. The results indicate that an item-pool blueprint can be used effectively to maintain the pool. The simulation also reveals that once the quality of a pool deteriorates, extra efforts (in this study: both usage of an optimal item-pool blueprint and an excellent level of item writing) are needed to restore its quality.

The task of item-pool management is difficult mainly because it involves uncertainty in the quality of the future items. To some extent, the quality of items left in the pool can influence the quality of the future assembly. If an item pool is dominated by items that are not chosen in earlier assembly, assuming that the assembly was done optimally, the quality of the future tests would be worse if new pretested items could not compensate for the low quality of old items.

In Chapter 6, an attempt is made to unravel the causes of underexposure of some items. Several methods are presented. The results of the simulation study show that, unless the test information has to meet a target exactly, the inclusion of underexposed items will still yield feasible tests. On the other hand, the results also show that there are critical items in a pool. Apparently, these items always have to be selected to yield feasible tests. The attribute values of these critical items determine whether or not other items can be selected for the tests. The results suggest that their presence should be optimized to prevent an undesirable interaction between attributes of the items in the pool.

Samenvatting

Een itembank is een database waarin een verzameling items (toetsvragen) wordt opgeslagen, met hun psychometrische parameters, zoals bijvoorbeeld het onderscheidingsvermogen, de moeilijkheid en de gokkans (voor het 3PL IRT model). Daarnaast worden andere kenmerken zoals het aantal woorden, categorische kenmerken, de verwachte responstijd en de datum van afname opgeslagen in de bank. Uit een itembank worden toetsen samengesteld op basis van de toetsspecificaties. Mathematisch programmerings algoritmen of heuristieken kunnen gebruikt worden om individuele items uit de bank te selecteren. De itembank is daarom erg waardevol voor een toetsprogramma waarbij toetsen op een continue basis aangeboden worden.

Er zijn twee redenen om het gebruik van de itembank te optimaliseren. In de eerste plaats zijn er veiligheids redenen. Vanwege de continue afname van de toetsen is het mogelijk dat sommige items bekend raken onder de kandidaten. Dit tast de integriteit van de itembank aan. Daarnaast zijn er redenen met betrekking tot de efficiëntie. Veel inspanningen en geld zijn gestoken in het schrijven, ontwikkelen, pre-testen en calibreren van de items en het is van belang om maximaal rendement te krijgen uit deze investeringen.

In de praktijk echter is het vaak onmogelijk om alle items in de bank te gebruiken in de toetsen. Helemaal als er bepaalde kwaliteitseisen aan de toets gesteld worden, zoals het geven van veel informatie rond een bepaald vaardigheidsniveau. In dat geval zullen slechts items geselecteerd worden die gekwalificeerd zijn voor dat doel. Geforceerde selectie van andere items zal de kwaliteit van de test verlagen. De waarde van de Toets Informatie Functie (TIF), de bias en de variantie van de vaardigheidsschatter zijn veel gebruikte criteria voor het meten van de kwaliteit van een toets.

De focus van itembank management ligt op het optimaliseren van het gebruik, terwijl de kwaliteit van de toetsen op een acceptabel niveau blijft. In dit proefschrift worden een aantal methoden voorgesteld voor het verbeteren van itembank design en itembank management.

Geautomatiseerde toetsconstructie (*Automated Test Assembly*, ATA) kan toets constructeurs helpen bij het oplossen van de hierboven geschetste dilemma's. Bij schriftelijke toetsen kunnen ATA methoden gebruikt worden om parallelle toetsen samen te stellen. Bij computer adaptief toetsen worden de methoden dusdanig aangepast dat ze er voor zorgen dat items niet al te vaak geselecteerd worden (deze methoden worden ook wel *exposure-control* methoden genoemd). In hoofdstuk 1 worden verschillende methoden voor toetsconstructie beschreven.

Veel *exposure-control* algoritmen zijn vooral gericht op de kwalitatief hoogwaardige items in de bank, omdat die het meest geselecteerd worden voor de toetsen. In hoofdstuk 2 laten we zien hoe de methode van roterende itembanken werkt, als een alternatieve methode voor het verkrijgen van een uniform gebruik van de items in de bank. Het uitgangspunt bij deze methode is dat de kwalitatief hoogwaardige items verspreid worden over de verschillende item banken, zodat het gebruik van minder goede items wordt gestimuleerd. Bij deze aanpak wordt een itembank opgedeeld in een aantal kleinere parallelle itembanken. Deze itembanken roteren over de verschillende toetslocaties om op die manier de gewenste *exposure rate* te realiseren. De methode van Gullikson (1950) wordt gebruikt om er voor te zorgen dat de psychometrische eigenschappen van de kleine itembanken met elkaar overeenkomen. Bovendien laten we in hoofdstuk 2 zien hoe je een itembank kunt opdelen in elkaar overlappende kleinere banken, die een aantal items gemeenschappelijk hebben. De resultaten laten zien dat er veel beter gebruik wordt gemaakt van de items in de bank als de bank opgedeeld wordt in kleinere (elkaar overlappende) itembanken. Er moet wel opgemerkt worden dat dit voordeel behaald wordt ten koste van een grotere fout bij het schatten van de vaardigheid. Dit kan echter gecompenseerd worden door de toets te verlengen.

De samenstelling van de itembank bepaalt in grote mate of de items efficiënt gebruikt kunnen worden. Als er veel slechte items in de bank zitten, of items met verkeerde kenmerken, dan kan geen enkele methode ervoor zorgen dat alle items gebruikt worden in het toetsproces. Dit effect wordt nog eens versterkt als de toets adaptief afgenomen wordt, omdat kleine verschillen in de waardes van de item parameters dan al kunnen leiden tot grote verschillen in *exposure rates*. Alhoewel dit effect verkleind kan worden door het gebruik van *exposure-control* methoden, blijft er een trade-off tussen exposure control en de grootte van de meetfouten. Dit probleem kan opgelost worden door alleen die items in de bank te stoppen die nodig zijn voor de toetsen. In hoofdstuk 3 laten we zien dat de *shadow-test approach* gebruikt kan worden om een itembank te bouwen. De *shadow-test approach* is een item selectie methode voor CAT, waarbij telkens een complete toets wordt geconstrueerd, waarvan vervolgens het beste item afgenomen wordt. Nadat het vaardigheidsniveau is geschat wordt weer een complete toets geconstrueerd en deze procedure wordt herhaald totdat het benodigde aantal items is afgenomen. In deze benadering wordt het het samenstellen van een adaptieve toets gezien als het maken van een serie van complete toetsen. Daarom klinkt het logisch om een itembank voor CAT te construeren als een verzameling complete toetsen. Om de methode te illustreren is hij toegepast op de Law School Admission Test. Dit resulteerde in een bijna uniforme bias en mean-squared error voor de vaardigheidsschatter en in *exposure rates* die lager waren dat de maximaal toelaatbare waarden.

De behoefte om de inhoud en de kwaliteit van een toets vooraf te controleren heeft geresulteerd in het multistage toetsen (MST). MST is te vergelijken met adaptief toetsen met een van te voren vastgelegd aantal paden. In plaats van het afnemen van een item, wordt een groepje items (testlet) afgenomen voordat er weer een schatting van het vaardigheidsniveau wordt gemaakt. Omdat er maar een beperkt aantal beslismomenten is en dus een gelimiteerd aantal paden, kunnen toetsspecialisten alle combinaties van te voren controleren. Voor deze soort van toetsen is een groot aantal testlets nodig, om parallelle versies te kunnen maken. Dit is belangrijk voor de geheimhouding van de toets. In hoofdstuk 4 wordt een methode gedemonstreerd voor het omzetten van een itembank in een testletbank. Het samenstellen van testlets is vergelijkbaar met het samenstellen van een maximaal aantal parallelle toetsen. Deze benadering zorgt voor een uniform gebruik van de items in de bank. Als bovendien het samenstellen van alle testlets simultaan gebeurt, dan hebben ze ook dezelfde kwaliteit. Een aantal testlets in de bank wordt gebruikt voor het samenstellen van toetsen, de overige testlet kunnen gebruikt worden in de toekomst. In een simulatiestudie laten we zien hoe er verschillende gewichten aan testlets toegewezen kunnen worden omdat sommige testlets belangrijker zijn dan andere, afhankelijk van hun moeilijkheid.

Operationele toetsprogramma's hebben te lijden onder fluctuaties in de kwaliteit van de itembank. Deze fluctuaties kunnen niet vermeden worden, omdat nieuwe items worden toegevoegd terwijl anderen (tijdelijk) worden verwijderd uit de bank. Sommige items worden verwijderd omdat ze te vaak gebruikt zijn. Andere worden verwijderd omdat hun itemparameters veranderd zijn. Deze fluctuaties in de kwaliteit van de bank maken goed itembank management een complexe taak. Daarom hebben we een instrument ontwikkeld om de kwaliteit van itembanken op peil te houden. In hoofdstuk 5 wordt een simulatiestudie gebruikt om de doelmatigheid van dit instrument te onderzoeken. De resultaten laten zien dat een blauwdruk voor een itembank effectief gebruikt kan worden bij het onderhouden van de bank. De simulatiestudie laat ook zien dat extra maatregelen (optimale blauwdruk en excellent niveau van itemschrijven) nodig zijn om de itembank te herstellen als de kwaliteit afneemt.

Item-bank management is lastig omdat je rekening moet houden met onzekerheid over de kwaliteit van toekomstige items. De kwaliteit van de items in de bank die niet gebruikt worden voor toetsconstructie hebben ook invloed op de kwaliteit van toekomstige toetsen. Als een itembank hoofdzakelijk bestaat uit items die bij een vorige selectie afgevallen zijn, dan zal de kwaliteit van toekomstige toetsen afnemen als de nieuwe items hier niet voor kunnen compenseren. In hoofdstuk 6 wordt een poging gewaagd om de oorzaken te achterhalen waarom sommige items niet geselecteerd worden. In een simulatiestudie wordt aangetoond dat het verplicht selecteren van deze items meestal toetsen oplevert die voldoen aan de eisen. Aan de andere kant laat de simulatiestudie ook zien dat er een aantal items in de bank zijn die een cruciale rol spelen bij itemselectie. Deze items moeten altijd geselecteerd worden. De eigenschappen van deze cruciale items bepalen of andere items wel of niet geselecteerd worden. De resultaten geven bovendien aan dat deze items optimaal gebruikt moeten worden om ongewenste interactieeffecten tussen eigenschappen van items in de pool te voorkomen.

References

Ackerman, T. A. (1989, March). *An alternative methodology for creating parallel tests forms using the IRT information function*. Paper presented at Annual Meeting of the National Council on Educational Measurement, San Fransisco.

Adema, J. J. (1990). *Models and algorithms for the construction of achievement tests*. Thesis, Enschede, The Netherlands: University of Twente.

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

Adema, J. J., & van der Linden, W. J. (1989). Algorithms for computerized construction using classical item parameters. *Journal of Educational Statistics, 14*, 279-290.

Andrews, R., Diederich, J, and Tickle, A. B. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems, 8*, 373-389.

Ariel, A., Veldkamp, B. P., & Breithaupt, K. (2004). *Optimal testlet pool assembly for multi-stage testing designs*. Paper submitted for publication.

Ariel, A., Veldkamp, B. P., & van der Linden, W. J. (2004). Constructing rotating item pools for constrained adaptive testing. *Journal of Educational Measurement, 41,* 345-359.

Armstrong, R. D., Jones, D.H., & Kunce, C.S. (1998). IRT test assembly using network-flow programming. *Applied Psychological Measurement*, *22*, 237-247.

Armstrong, R. D., Jones, D.H., & Wang, Z. (1998). Optimization of Classical Reliability in Test Construction. *Journal of Educational and Behavioral Statistics, 23*, 1-17.

Armstrong, R. D., Jones, D.H., & Wu, I. L. (1992). An Automated Test Development of Parallel Tests from a Seed Test. *Psychometrika, 57*, 271-288.

Atkinson, K. E. (1989). *An introduction to numerical analysis*. Singapore: Wiley.

Belov, D. I., & Armstrong, R. D. (2005). Monte carlo test assembly for item pool analysis and extension. *Applied Psychological Measurement, 29*, 239-261.

Berger, M. P. F. (1997). Optimal designs for latent variable models: A review. In J. Rost & R. Langeheine (Eds.), *Applications of latent trait and latent class models in the social sciences* (pp. 71-79). Muenster, Germany: Waxman

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord and M. R. Novick, *Statistical theories of mental test scores* (pp. 397-479). Reading, MA: Addison-Wesley.

Boekkooi-Timminga, E. (1987). Simultaneous test construction by zero-one programming. *Methodika, 1*, 101-112.

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

Boekkooi-Timminga, E. (1991, June). *A method for designing Rasch model based item banks*. Paper presented at the Annual Meeting of the Psychometric Society, Princeton, NJ.

Boyd, A. M., Dodd, B. G., & Fitzpatrick, S. J. (2002, April). *A comparison of exposure control procedures in CAT systems based on different measurement models for testlets using the verbal reasoning section of the MCAT*. Paper presented at Annual Meeting of the National Council on Educational Measurement, Chicago, IL.

Dakin, R.J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal, 8*, 250-255.

Garfinkel, R. S. & Nemhauser, G.L. (1972). *Integer programming*. Toronto: Willey

Guieu, O., & Chinneck, J. W. (1999). Analyzing infeasible mixed-integer and integer linear programs. *INFORMS Journal on Computing, 11*, 63-77.

Gulliksen, H. (1950). *Theory of mental tests*. New York: Willey.

Hambleton, R. K., Swaminathan H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. California, USA: Sage Publications.

Huitzing, H. A. (2003). *Infeasibility in automatic test assembly: Analysis, causes and solutions*. Thesis, Groningen, The Netherlands: University of Groningen.

Huitzing, H.A., Veldkamp, B.P., & Verschoor, A.J. (2005). Infeasibility in automated test assembly models: A comparison study of different methods. *Journal of Educational Measurement, 42*, 223-244.

ILOG, Inc. (2002). *CPLEX 8.1* [Computer program and manual]. Mountain View, CA: ILOG

Kingsbury, G. G., & Zara, A. R. (1991). Procedures for selecting items for computerized adaptive tests. *Applied Measurement in Education*, *2*, 359-375.

Klecka, W. R. (1980). Discriminant analysis. California: Sage Publications.

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.

Luecht, R. M. (2000, April). *Implementing the computer-adaptive sequential testing (CAST) framework to mass-produce high quality computer-adaptive and mastery test*. Paper presented in the National Council on Measurement in Education Symposium, N3, New Orleans, LA.

Luecht, R. M., Brumfield, T. & Breithaupt, K. (2002, April). *A testlet-assembly design for the uniform CPA examination*. Paper presented at the Annual Meeting of the National Council on Measurement in Education, New Orleans, LA.

Luecht, R. M., & Burgin, W. (2003, April). *Test information targeting strategies for adaptive multistage testing designs*. Paper presented at the Annual Meeting of the National Council on Measurement in Education, Chicago, IL.

Luecht, R. M., & Nungester, R. J. (1998). Some practical examples of computer-adaptive sequential testing. *Journal of Educational Measurement, 35*, 229-249.

McBride, J. R., Wetzel, C. D., & Hetter, R. D. (1997). Preliminary psychometric research for CAT-ASVAB: selecting an adaptive testing strategy. In W.A. Sands, B. K. Waters & J. R. McBride (Eds.), *Computerized adaptive testing: from inquiry to operation* (pp. 83-95). Washington DC: American Psychological Association.

McLachlan, G. J. (1992). Discriminant analysis and statistical pattern recognition. New York: Willey.

Mills, C. N., & Steffen, M. (2000). The GRE computer adaptive test: Operational issues. In W. J. van der Linden & C. A. W. Glas (Eds.). *Computerized adaptive testing: Theory and practice* (pp. 75-99). Boston, MA: Kluwer Academic Publishers.

Mills, C. N., & Stocking, M. L. (1996). Practical issues in large-scale computerized adaptive testing. *Applied Measurement in Education, 9*, 287-304.

Nemhauser, G. L., & Wolsey, L. A. (1999). *Integer and combinatorial optimization*. New York: Wiley.

Owen, R. J. (1975). A bayesian sequential procedure for quantal response in the context of adaptive mental testing. *Journal of the American Statistical Association, 70*, 351-356.

Paragon Decision Technology, B.V. (2003). AIMMS (Version 3.4) [Computer Software]. Haarlem, The Netherlands: Paragon Decision Technology.

Rasch, G. (1960). *Probabilistic models for some intelligent and attainment tests*. Copenhagen, Denmark: Nielsen and Lydiche.

Revuelta, J., & Ponsoda, V. (1998). A comparison of item exposure control methods in computerized adaptive testing. *Journal of Educational Measurement*, *35*, 311-327.

Sands, W. A., & Waters, B. K. (1997). Introduction to ASVAB and CAT. In W. A. Sands, B. K. Waters, & J. R. McBride (Eds.). *Computerized adaptive testing: From inquiry to operation* (pp. 3-10). Washington, D. C.: American Psychological Association.

Silvey, S. D. (1980). *Optimal design*. London: Chapman & Hall.

Stocking, M. L. (1994). Three practical issues for modern adaptive testing item pools. (ETS Research Report No. 93-2). Princeton, NJ: Educational Testing Service.

Stocking, M. L., & Lewis, C. (1998). Controlling item exposure conditional on ability in computerized adaptive testing. *Journal of Educational and Behavioral Statistics*, *23*, 57-75.

Stocking, M.L., & Swanson, L. (1993). A method for severely constrained item selection in adaptive testing. *Applied Psychological Measurement, 17*, 277-292.

Stocking, M.L., & Swanson, L. (1998). Optimal design of item banks for computerized adaptive tests. *Applied Psychological Measurement, 22*, 271-279.

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

Sympson, J. B., & Hetter, R. D. (1985). Controlling item-exposure rates in computerized adaptive testing. *Proceedings of the 27th annual meeting of the Military Testing Association* (pp. 973-977). San Diego, CA: Navy Personnel Research and Development Center.

Timminga, E. (1998). Solving infeasibility problems in computerized test assembly. *Applied Psychological Measurement, 7*, 201-210.

Timminga, E., & Adema, J. J. (1996). An interactive approach to modifying infeasibility 0-1 linear programming models for test construction. In G. Engelhard Jr. & M. Wilson (Eds.), *Objective measurement: Theory into practice (Vol. 3)*. Norwood NJ: Abex.

van der Linden, W. J. (1994). Optimum design in item response theory: Applications to test assembly and item calibration. In G. H. Fischer & D. Laming (Eds.), *Contributions to mathematical psychology, psychometrics, and methodology* (pp. 308-318). New York: Springer-Verlag.

van der Linden, W. J. (2000). Constrained adaptive testing with shadow tests. In W.J. van der Linden & C.A.W. Glas (Eds.), *Computerized adaptive testing: Theory and practice* (pp. 27-52). Boston, MA: Kluwer Academic Publishers.

van der Linden, W. J. (2003). Some alternatives to Sympson-Hetter item-exposure control in computerized adaptive testing. *Journal of Educational and Behavioral Statistics, 28*, 249-265.

van der Linden, W. J. (2003). MIP formulation and CPLEX implementation. Unpublished manuscript, University of Twente, The Netherlands.

van der Linden, W. J. (2005). *Linear models for optimal test design*. New York: Springer-Verlag.

van der Linden, W. J., & Adema, J. J. (1998). Simultaneous assembly of multiple test forms. *Journal of Educational Measurement, 35*, 185-198.

van der Linden, W. J., Ariel, A., & Veldkamp, B. P. (in press). Assembling a CAT item pool as a set of linear tests. *Journal of Educational and Behavioral Statistics.*

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

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., & Chang, H. H. (2003). Implementing content constraints in alpha-stratified adaptive testing using a shadow test approach. *Applied Psychological Measurement, 27*, 107-120.

van der Linden, W. J., & Glas, C. A. W. (2000). *Computerized adaptive testing: theory and practice*. Boston, MA: Kluwer Academic Publishers.

van der Linden, W. J., & Hambleton, R. K., Eds. (1997). *Handbook of modern item response theory*. New York, NJ: Springer-Verlag.

van der Linden, W. J., & Luecht, R. M. (1998). Observed-score equating and a test assembly problem. *Psychometrika, 63*, 401-418.

van der Linden, W. J., & Pashley, P. J. (2000). Item selection and ability estimation in adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Computerized adaptive testing: Theory and practice* (pp. 1-25). Boston: Kluwer.

van der Linden, W.J., & Reese, L.M. (1998). A model for optimal constrained adaptive testing. *Applied Psychological Measurement, 22*, 259-270.

van der Linden, W. J., & Veldkamp, B. P. (2004). Constraining item exposure in computerized adaptive testing with shadow tests. *Journal of Educational and Behavioral Statistics, 29*, 273-291.

van der Linden, W. J., Veldkamp, B. P., & Carlson, J. E. (2004). Optimizing balanced incomplete block designs for educational assessments. *Applied Psychological Measurement, 28*, 317-331.

van der Linden, W. J., Veldkamp, B. P., & Reese, L. M. (2000). An integer programming approach to item pool design. *Applied Psychological Measurement, 24*, 139-150.

van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: theory and applications*. Dordrecht: Reidel.

Veerkamp, W. J. J., & Berger, M. P. F. (1999). Optimal item discrimination and maximum information for logistic IRT models. *Applied Psychological Measurement, 23*, 31-40.

Veldkamp, B. P. (1999). Multiple objective test assembly problems. *Journal of Educational Measurement*, 36, 253-266.

Veldkamp, B. P. (2001). *Principles and methods of constrained test assembly*. Thesis, Enschede, The Netherlands: University of Twente.

Veldkamp, B. P. (2005, April). *Finding critical features in item pool maintenance*. Paper presented at Annual Meeting of the National Council on Educational Measurement, Montreal, Canada.

Veldkamp, B. P., Eggen, T. J. H. M., Verschoor, A., & Maroujenkov, K. (2004, April). *Item exposure control in CAT by active item pool management*. Paper presented at the Annual Meeting of the National Council on Measurement in Education, San Diego.

Veldkamp, B. P., & van der Linden, W. J. (2000). Designing item pools for computerized adaptive testing. In W.J. van der Linden & C.A.W. Glas (Eds.), *Computerized adaptive testing: Theory and practice* (pp. 149-162). The Netherlands: Kluwer Academic Publishers.

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

Way, W.D. (1998). Protecting the integrity of computerized testing item pools. Educational Measurement, Issues and Practice, 17, 17-27.

Way, W. D., & Steffen, M. (1998, April). *Strategies for managing item pools to maximize item security*. Paper presented at Annual Meeting of the National Council on Educational Measurement, San Diego, CA.

Way, W. D., Steffen, M., & Anderson, G. S. (1998). Developing, maintaining, and renewing the item inventory to support computer-based testing. In C. N. Mills, M. Potenza, J. J. Fremer, & W. Ward (Eds.), *Computer-based testing: Building* *the foundation for future assessments* (pp. 89-102). Hillsdale, NJ: Lawrence Erlbaum Associates.

Weiss, D. J. (1982). Improving measurement quality and efficiency with adaptive testing. *Applied Psychological Measurement, 6*, 473-492.

Wu, Ing-Long. (2001). A new computer algorithm for simultaneous test construction of two-stage and multistage testing. *Journal of Educational and Behavioral Statistics, 26*, 180-198.

Thank You

... is never sufficient to express my gratitude. For the past four years, I have been blessed to have an opportunity to work on a phd project in such a pleasant working atmosphere. I owe it to all of my colleagues at OMD Department, and I wish to sincerely thank them.

Some individuals I would like to thank in particular:

Prof. Wim J. van der Linden,
whose foresight and enthusiasm have always been a strong motivator

Dr. Bernard P. Veldkamp,
whose encouraging remarks and opinion have been invaluable to my research

Prof. Cees Glas, Dr. Rob Meijer, Prof. Don Mellenbergh,
Prof. Piet Sanders and Prof. Theo de Vries
for taking part in my graduation committee

Dr. Krista Breithaupt and the entire staff of the American Institute of CPA,
whose professional expertise had given me a valuable insight
during the summer internship

Lorette and Birgit,
for every effort you make to organize *gezellige* events

my roommate Anna, and Irene, Jonald, Leo
for both useful and useless conversations we had,
especially during the rainy days (literally)

the Ariels and the Huntings
for everything all of you do to show that you care and love me,
and most important,
my dearest Marcel
with whom I enjoy every single thing.