A Review of Linear Programming and its Application to the Assessment Tools for Teaching and Learning (asTTle) Projects

Technical Report 5, Project asTTle, University of Auckland, 2000

Richard B. Fletcher[1]
Massey University (Albany)

This report reviews the international research literature on linear programming as applied to the issues of banking assessment items. It outlines the mathematical procedures needed to obtain feasible solutions to selections made by teachers and constraints imposed by the assessment developers. The various algorithms and heuristic procedures necessary for feasible solutions in an item bank of only 500 items with testlets are discussed and exemplified. The report recommends use of detailed item mapping, limiting the number of ability levels, use of the simultaneous selection of items and sets method, use of the *maximin* model, and use of the optimal rounding method in finding solutions.

Table of Contents [1]

Introduction

Item banking permits the test constructor to store a great deal of information about the items (e.g., item difficulty, item discrimination, response times, content attributes) which then can be used to develop tests according to detailed test specifications. A significant feature of item response theory (IRT) item banking is that items can be added to the item bank (once they are linked to the other items) or removed without impacting the measurement precision of other items. For the test development process, this is a great advantage as item pools can be enlarged over time. The payoff of having a larger item bank is the increased ability to select different sets of items and therefore reduce item exposure rates. IRT and item banking are major strengths that the asTTle projects will fully utilise.

In combination, IRT and computers have led to new innovations in test assembly. One practical example of the use of item banking is computer adaptive testing, where *individual* tests are constructed based on an examinee's responses to individual items (Lord, 1980). Ability is estimated after each item, and when the test is terminated at a predetermined criterion, the ability level is then assigned. The net result is a tailored test that accurately estimates the examinee's ability in less time, using fewer and different sets of items. Efficiency in trait estimation is the advantage of this ability testing approach.

There are difficulties, however, that have emerged from large-scale implementation of item banking. Firstly, the "best" items become overexposed, and therefore a small set of items tend to be selected into most tests. The net result is that tests do no vary from one another in terms of content. Secondly, item security can be problematic in that it has been possible for candidates to memorise the first few items in an adaptive test, and therefore the potential exists for the content of the item bank to be known and used by specialist test preparation

[1] Address for mailing: Richard Fletcher, School of Psychology, Massey University, Albany, Auckland, New Zealand. e-mail: R.B.Fletcher@massey.ac.nz

organisations. The above two issues can undermine utility of a test if too much is known about the item content beforehand. Ultimately, ability estimation is compromised and the testing process is invalidated. The issue facing test developers is to create a very large item bank to alleviate some of these problems.

Another example of the utility of item banking is optimal test design in the form of 0-1 linear programming (LP) that allows for *complete* tests to be assembled according to detailed test specifications while maximizing an objective function (Adema, Boekooi-Timminga, & van der Linden, 1991; Baker, Cohen, & Barmish, 1988; Timminga & Adema 1995, Thuenissen, 1985). Test assembly using existing item banks and LP is applicable to both classical test theory and IRT (Adema, 1990, 1992a, 1992b; Adema, Boekkooi-Timminga, & Gademann, 1992; Adema, Boekkooi-Timminga, & van der Linden, 1991; Adema & van der Linden, 1989; Armstrong, Jones, & Wang, 1994; Baker, Cohen, & Barmish, 1988; Berger & Veerkamp, 1996; Boekkooi-Timminga, 1987, 1990a, 1990b, 1993; de Gruijter, 1990; Stocking, Swanson, & Pearlman, 1991, 1993; Swanson & Stocking, 1993; Theunissen, 1985, 1986; Timminga & Adema, 1995, 1996; van der Linden, 1996, 2000; van der Linden & Boekkooi-Timminga, 1989). Furthermore, LP can be used to assemble tests using dichotomous (e.g., Baker, Cohen, & Barmish, 1988; de Gruijter, 1990; Stocking, Swanson, & Pearlman, 1991; Theunissen, 1985, 1986; van der Linden & Boekkooi-Timminga, 1989) or polytomous item formats (e.g., Berger, 1998; Berger & Mathijssen, 1997; and Fletcher & Hattie, in review).

Baker, Cohen, and Barmish (1988, p. 190) suggest that "mathematical programming represents a major addition to the tools of the test constructor because it provides an analytical rather than an ad hoc procedure for item selection under IRT." Such an approach to test assembly is a major strength of the asTTle project, as tests can be assembled to teacher specifications using items with known properties. The use of LP in this situation furnishes a teacher with an adaptive test that is comparable to other tests constructed from the same item bank. This is a major advance in classroom-based assessment.

The aim of this paper is to discuss linear programming (LP) in the context of the asTTle projects using IRT calibrated items, by providing practical examples for specifying objective functions and linear constraints. IRT test assembly is first addressed to overview this measurement model and its applicability to test construction within the asTTle projects. Secondly, LP is presented along with an explanation of some of the objective functions available to the test constructor. Thirdly, practical constraints for individual items are introduced along with a worked example to exemplify the main issues. Fourthly, set-based items are presented along with a practical example to illustrate this method. In sum, the paper will provide computer programmers with the relevant information and references to enable them to set up the constraints as outlined in the Assessment Tools for Teaching and Learning (asTTle) proposal.

Item Response Theory

For the purpose of this paper it is assumed that the one-parameter Rasch model is used as the basis of item calibration. The Rasch model is computationally the simplest of the current IRT models for dichotomously scored data (see Hambleton, 1989; Hambleton & Swaminathan, 1985; and Harris, 1996 for a discussion of the various unidimensional dichotomous IRT models). A critical assumption of the Rasch model (and all other unidimensional IRT models) is unidimensionality for each item; that is, one underlying latent ability should account for the item response. The Rasch model implies that the probability of a correct response to an item (dichotomous in this case) is a function of the item difficulty and the examinee's ability level (θ), and is denoted by the following equation:

$$P_i(\theta) = \frac{\exp(\theta - b_i)}{1 + \exp(\theta - b_i)} \quad (1)$$

where $P_i(\theta)$ is the item characteristic function and b_i is the item difficulty parameter.

An important feature of IRT models is the concept of item information that is the reciprocal of the standard error of measurement. Items with low standard errors have greater information and vice versa. Item information provides the test constructor with an indication of item measurement precision from which items can then be selected accordingly into the test on the basis of their information. For the Rasch model, item information function (IIF) is estimated using the following equation:

$$I_i(\theta) = P_i(\theta)[1 - P_i(\theta)] \quad (2)$$

The usefulness of IIF curves is that they can be added together. As a consequence, these specify the form of the test information function curve (TIFC). The TIFC is simply the sum of the IIF curves, and is computationally expressed:

$$I_t(\theta) = \sum_{i=1}^{n} I\ (\theta), \quad (3)$$

IRT Test Assembly

IIFs are the building blocks of IRT test assembly, and their use was first suggested by Birnbaum (1968) and Lord (1977, 1980), who indicated that their properties were useful in trying to match them to a target information function (TIF), such that they approximated the desired shape. The additive properties of IIF curves enable tests to be assembled according to the specified shape of the TIF. Accordingly, the TIF can be specified to cover a range of the trait continuum, or to provide information at certain ability points. For example, greater information may be required around a cut score on a test (i.e., mastery testing), whereas selecting information across a range of abilities would require TIF to be spread across a predetermined interval.

Lord (1980) outlined the following procedure for assembling tests under the IRT framework.

1. The TIF is specified to be a certain shape (see Figure 1 for an example of a hypothetical TIF over the ability range -1 to +1).
2. Items with information curves that can fill the difficult areas to fill under TIF are selected.
3. Item information function curves are added to determine their information contribution to the TIF.
4. Continue to add items so that the TIF is approximated at the specified ability levels.

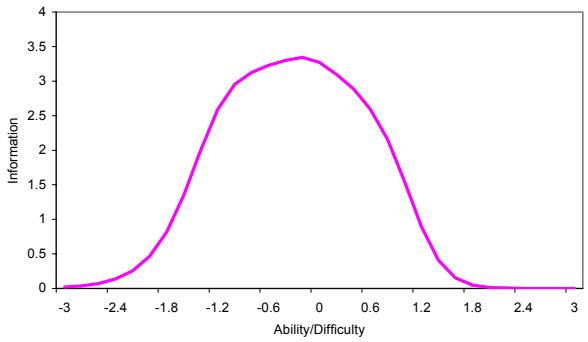

Figure 1. A hypothetical TIF specified over the ability range -1 to +1.

A major limitation of this method is that it becomes difficult to achieve when the item bank is large (Thuenissen, 1985). For example, an item bank of 500 items will result in 2,500 possible tests that can be assembled, and enumerating all possible tests is difficult, if not impossible. As tests are usually assembled to meet detailed specifications, then manually selecting items using IIFs to fit a TIF is an implausible task. One cannot be sure that the optimal set of items, which meets all of the specifications, has indeed been selected. Simultaneously, taking all the test requirements into consideration when assembling tests under the IRT framework is a laborious manual task, or computationally intractable. Test assembly methods that can utilize IRT measurement properties, and which incorporate complex test specification, are required, and are of great advantage to test constructors.

Linear Programming

One method that can maximize the psychometric properties of IRT information curves and fully implements Birnbaum's (1968) and Lord's (1977, 1980) IRT test assembly method, while meeting complex test

specifications, is mathematical linear programming (LP) in the form of 0-1 binary [integer] programming (e.g., Adema & van der Linden, 1989; Boekkooi-Timminga, 1989, 1990; de Gruijter, 1990; Theunissen, 1985, 1986; van der Linden & Boekkooi-Timminga, 1989). Adapted from the area of operations research, LP maximizes or minimizes an objective function while satisfying a series of linear constraints, and is therefore suited to test assembly problems.

The primary aim of LP models is to maximize or minimize an objective function (e.g., maximizing test reliability, minimizing the number of items in the test, and/or maximizing information at a given point of the ability scale) so that test specifications (e.g., test length, test information, item format, and item content) in the form of optimal constraints (e.g., the number of items in the test equal 45; mean p-value equals 0.25; items 3 and 5 should not appear together) are met. In a 0-1 LP model, the decision variables may take either 0 or 1. Accordingly, items that are selected into the test are assigned a 1 and those omitted from the test, a 0.

$$x_i = \{0 \text{ or } 1\}, \qquad i = 1,2,\ldots, I \tag{4}$$

where x_i is the decision variable and i is the number of items in the item bank.

LP provides the test constructor with a method for selecting items that meet the objective function while meeting a series of complex linear constraints, and can ensure that both content and statistical specifications are simultaneously satisfied (Fletcher & Hattie, in review). LP is a major advance in test assembly process and will fully realize the objective of the asTTle projects in that it will present teachers with a set of items that matches, or close approximates, their test specifications.

Objective Functions

LP involves an explicit statement about what the objective function of the test should be. In terms of the asTTle projects, this will likely entail teachers making an initial choice about the range of the ability they wish the test to cover. From an LP programming perspective, however, this means that the objective function used in each of the asTTle assessments will need to be flexible in its ability to realise feasible solutions and to satisfy the teachers' demands. In other words, the objective function must provide teachers with a set of items that meets, or closely approximates, their test specifications (e.g., range of ability). Thus, it is not desirable for teachers to be told there is no solution.

Timminga and Adema (1995) suggested six types of objective functions for achievement testing (see also van der Linden & Boekkooi-Timminga, 1989). The first two are based on information being maximized at a specified point of the ability scale (θ_f) (i.e., mastery testing), and the last four are based on a broader range of ability being tested (i.e., non-mastery). For example, information may be required to be greater across the ability range of θ_k (k = 1, ..., K). For the asTTle projects, the teachers will be asked to make choices about the range of ability the test should cover. Generally, their selection should be of three to five ability levels, as this is sufficient to provide satisfactory results (Timminga, 1995; cited in van der Linden, 1994). The six objective functions are:

1. Maximize the information at θ_k

$$\text{maximize} \sum_{i=1}^{n} I_i(\theta_k) x_i \tag{5}$$

   where I_i is the item information for item I_i, and n is the items in the item bank.
2. Minimize the deviation between the cut score and the β_i (item difficulty) of the item selected

$$\text{minimize} \sum_{i=1}^{n} |\theta_k - \beta_i| x_i \tag{6}$$

3. Exact target information functions that delineate the TIF should be as close as possible to the specified information values. In actuality, obtaining an exact TIF will be an improbable task, and therefore a decision is required as to how the deviation for the target values should be reduced. Timminga

and Adema (1995) suggest three methods that can be used to achieve this objective:

a. Minimize the sum of the positive deviations from the target values

$$\text{minimize} \sum_{i=1}^{n} \sum_{k=1}^{k} I_i(\theta_k) x_i \quad (7)$$

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i \geq T(\theta_k), \qquad k = 1,2,\ldots,K \quad (8)$$

where $T(\theta_k)$ is the target value.

The model suggests that an objective function and a set of constraints are required to obtain a set of values that are closely approached. Thus, minimizing the positive deviations from the target values is achieved when the values in the constraint (Equation 8) are at least reached. A limitation of this approach is that not all points of the TIF may be reached, and that at some ability points the accuracy is greater than for others (Timminga and Adema, 1995).

b. Minimize the sum of the absolute deviations

$$\text{miminize} \sum_{k=1}^{k} (y_k + u_k), \quad (9)$$

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i - y_k + u_k = T(\theta_k), \qquad k = 1,2,\ldots,K \quad (10)$$

$$y_f, u_f \geq 0, \qquad k = 1,2,\ldots k \quad (11)$$

In this model, two decision variables y_k (positive deviation ≥ 0) and u_k (negative deviation ≥0) are introduced. As the sum of y_k and u_k are minimized, and both are equal to or greater than zero, it means that when one is zero, the other is equal to the deviation from the TIF.

c. Minimize the largest deviation from the TIF.

minimize y

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i - T(\theta_k) \leq y, \qquad k = 1,2,\ldots,K \quad (12)$$

$$T(\theta_k) - \sum_{i=1}^{n} I_i(\theta_k) x_i \leq y, \qquad k = 1,2,\ldots,K \quad (13)$$

With this objective function, the decision variable y ($y \geq 0$) is the largest deviation from the TIF.

4. Maximum information is required at all ability points. This objective function was originally specified by Theunissen (1985) and is formulated to be the same as (2). However, as information is required at more than one ability point, and as LP deals with one objective function at a time, then these have to be re-specified for each θ level. Thus, the decision variable y ($y \geq 0$) specifies the amount of information to be obtained at the specified ability levels. The model is specified:

maximize y

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i \geq y, \qquad k = 1,2,\ldots,K \quad (14)$$

5. Relative target information values specify that the shape of the TIF be approximated at some of the ability points. This objective function was developed by van der Linden and Boekkooi-Timminga (1989) as the *maximin* model, and specifies the shape of the TIF rather than the exact height. Thus, the *maximin* model is a more flexible approach to test construction. The decision variable y ($y \geq 0$) denotes the degree to which information is maximized, and constant r_k denote the relative shape of the TIF across the specified θ points. The model is denoted:

maximize y

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i \geq r_k y, \qquad k = 1,2,\ldots,K \quad (15)$$

6. Minimizing the test length while the TIF values are required to be equal to or greater than the target values. With this model, one should not set the test length as a constraint as this will be in conflict with the objective

function, and will result in infeasibility. The objective function is to:

$$\text{minimize} \sum_{i=1}^{n} x_i \tag{16}$$

subject to:

$$\sum_{i=1}^{n} I_i(\theta_k) x_i \geq T(\theta_k), \qquad k = 1,2,\ldots,K \tag{17}$$

The objective functions outlined above each have their potential uses in the construction of achievement tests. The *maximin* model (van der Linden & Boekkooi-Timminga, 1989) appears to be the most flexible of the above objective functions, and seems more appropriate for the asTTle projects in that it only requires the TIF to be approximated.

Practical Constraints

For the asTTle projects, teachers will not only specify an objective function but will also make choices about characteristics of the type of test they desire. These choices will then be transferred into an objective function and series of practical linear constraints. Teachers will therefore be setting test specifications, albeit limited ones. In practice, however, test specifications are often complex, and practical constraints are required within LP in order to fully realize the test constructors' demands. Test specifications in LP are expressed as a series of linear constraints. Before setting practical constraints, it is advisable to consider the qualities of the item bank and its ability to meet them. For example, it would not make sense to ask for six deep processing items at θ_3 in the close reading curricular area, when in fact there were only three items. Such a constraint would lead to no solution being found to the test construction model, and as teachers will not know how to deal with such infeasibility, then constraints will need to be set within reasonable limits.

Teachers will certainly need to know something about the structure of the item bank to enable them to understand the choices they can make. For the computer programmers, however, mapping out the item bank will prove invaluable when writing the linear constraints and will help in identifying potential infeasibility problems and in finding out how to deal with these (see Tables 1, 2, and 4 for examples of hypothetical item bank structures). In general, though, the constraints should be reasonable so as to allow for a feasible solution to be obtained, and, therefore, knowing the qualities of the item bank will help to overcome this type of issue.

Practical Constraints for Simultaneous Selection of Items

The importance of setting accurate and reasonable practical constraints cannot be overstated, as these are essential components in LP test assembly because they are the means by which test specifications are fully met. Constraints should be closely checked before any attempt is made to solve the model. Some simple examples of practical test constraints that are likely to be used in the asTTle projects are given below.

1. Constraining the number of items allowed to be selected into the test. In other words, the sum of the items from $i = 1, 2, \ldots, I$ must be equal to N

$$\sum_{i=1}^{I} x_i = N \tag{18}$$

2. Limiting the number of items with certain characteristics (e.g., multiple-choice, open-ended questions, deep or surface learning). If S_d is a set of deep cognitive processing items, then one can limit the number of these items in the test using the equation below:

$$\sum_{i=\in S_d} x_i = N_d \tag{19}$$

3. Limiting the sum of certain item attributes. For example, the sum of the administration times (t_i) for item i, should be less than or equal to b_j. For this type of constraints, one should not use equality signs as these can lead to no solution being found. The inequality sign therefore provides some flexibility in designation of items into the test.

$$\sum_{i=1}^{I} t_i x_i \le b_j \quad (20)$$

4. Select all or none of the items in a subset. If, for example, the items in a set are consecutively numbered 1–7, then the equation that denotes selecting all or none of the items is:

$$\sum_{i=1}^{7} x_i = 7x_i \quad (21)$$

5. One can also specify the proportion of items to be selected from one subset in relation to another subset. For example, one might wish to have three times the items from subset F_1 as from subset F_2. The equation that denotes this is:

$$\frac{1}{3}\sum_{I_i \in F_1} x_i = \sum_{I_i \in F_2} x_i \quad (22)$$

6. The inclusion of an item into the test requires the item to take on a fixed value ($x_i = 1$). If this constraint is chosen then a modification to the objective function should be made to take into account the pre-selected items.
7. The exclusion of an item into the test requires the item to take on a fixed value ($x_i = 0$), or one can ensure that these items do not enter the test assembly model. The latter of these two options is suggested as this will speed up the selection process because fewer items are considered from the item bank (Timminga & Adema, 1995). In other words, leaving items out of the test assembly model will speed up the solution time.

An application of some of the practical constraints discussed above is presented below, highlighting some possible constraints that may be used in the asTTle projects. In the following example, the item bank is hypothesized to consist of 448 items (see Tables 1 & 2) with each item consecutively numbered from 1–448. Tables 1 and 2 represent the total item bank but the items are presented in separate tables for ease of interpretation when examining the practical constraints. Table 1 presents the items bank structure for the closed-choice items (multiple-choice, "fill in the blanks" and "true or false"), and Table 2 outlines the item bank structure for the open-ended items. Table 3 presents the number of items at each θ_k level ($K = 1, 2, ..., 7$).

A test constructor may, for example, want a test that meets a set of tests specifications to maximize information, such that:

1. The TIF should be approximately equal at $\theta_3 = -1.0$, $\theta_4 = 0.0$, $\theta_5 = +1.0$ (Equation 23) (see Figure 1).
2. Ten items must be from the personal reading curricular area (Equation 24).
3. Five closed-choice (C-C) items in the personal reading curricular area must be at the deep level of processing (Equation 25).
4. Ten items must be from the close reading curricular area (Equation 26).
5. Five C-C items in the close reading curricular area must be at the deep level of processing (Equation 27).
6. The test should contain 39 C-C items (Equation 28).
7. The test should contain one open-ended item (Equation 29).
8. The test administration time should take no more than 2,400 seconds (40 minutes) (Equation 30).

To meet these specifications the following model was constructed:

maximize y

subject to

$$\sum_{i=1}^{448} I_i(\theta_k) x_i - y \ge 0, \qquad k = 1,2,3 \quad (23)$$

$$\sum_{i=1}^{56} x_i = 10 \quad (24)$$

$$\sum_{i=1}^{3} x_i + \sum_{i=7}^{9} x_i + \sum_{i=13}^{15} x_i + \sum_{i=19}^{21} x_i + \sum_{i=25}^{27} x_i + \sum_{i=31}^{33} x_i + \sum_{i=37}^{39} x_i = 5 \quad (25)$$

$$\sum_{i=57}^{98} x_i = 10 \quad (26)$$

$$\sum_{i=57}^{59} x_i + \sum_{i=63}^{65} x_i + \sum_{i=69}^{71} x_i + \sum_{i=75}^{77} x_i + \sum_{i=81}^{83} x_i + \sum_{i=87}^{89} x_i + \sum_{i=93}^{95} x_i = 5 \quad (27)$$

$$\sum_{i=1}^{42} x_i + \sum_{i=57}^{98} x_i + \sum_{i=113}^{145} x_i + \sum_{i=169}^{210} x_i + \sum_{i=225}^{266} x_i + \sum_{i=281}^{322} x_i + \sum_{i=337}^{378} x_i + \sum_{i=393}^{434} x_i = 39 \quad (28)$$

$$\sum_{i=43}^{56} x_i + \sum_{i=99}^{112} x_i + \sum_{i=155}^{168} x_i + \sum_{i=211}^{224} x_i + \sum_{i=267}^{280} x_i + \sum_{i=323}^{336} x_i + \sum_{i=379}^{392} x_i + \sum_{i=393}^{434} x_i = 1 \quad (29)$$

$$\sum_{i=1}^{448} t_i x_i \leq 2400 \quad (30)$$

$$x_i \in \{0,1\}, \qquad i = 1,2,\ldots,448 \quad (31)$$

$$y \geq 0 \quad (32)$$

Practical Constraints for Simultaneous Selection of Sets of Items

For the asTTle projects the item bank is likely to be composed of set-based items (or *testlets*). That is, groups of items are linked to a common stimulus and as a result are set-bound. The implication is that if certain items are selected then the associated stimulus should also be selected, or vice versa. The challenge for the test constructor is to ensure that items related to a stimulus are selected, or that if the stimulus is selected then some or all of its associated items are selected into the test. Boolean constraints (e.g., if Item 1 is selected, then its related stimulus must be selected) are therefore needed to overcome the problem of simultaneous selection of items and stimuli. Van der Linden (2000) suggests six methods for selecting set-based items and their associated stimuli. Only one method is presented below (simultaneous selection of items and sets), as this method was shown to be computationally efficient and more accurate in its ability to match the TIF. Furthermore, van der Linden (2000) suggests that this method is most suited to new test assembly issues or new item banks, and is thus most appropriate to the asTTle projects. (For a further discussion, and the implications of the other methods, see van der Linden, 2000).

The method outlined below is based on an item bank with set-based items. This method can incorporate discrete items, but for the sake of simplicity only set-based items are fully discussed. As van der Linden (2000, p. 229) states

> The key feature of this method is that separate decision variables for the selection of items and stimuli are defined. The variables are used to model the constraints imposed on the section of items and stimuli. Special logical constraints are added to keep the selection of items and stimuli consistent, that is, to prevent the selection of items without the selection of stimuli, or the selection of stimuli without the selection of items.

Set-based items can be conceptualized at four levels (although there could be more): individual item, stimuli, item sets, and total test (van der Linden, 2000). Table 4 shows the attribute and constraint levels for items as specified by van der Linden (2000). It is important to note that constraints must be delineated at the same level as the attribute level, or higher. Understanding the levels of the items in terms of attribute and constraint levels will make setting the constraint a straightforward task.

The notation used is similar to that used by van der Linden (2000). Items are arranged in item sets, S (s = 1, 2, ..., S) (if the item bank were to contain discrete items, then the items might be arranged into S-1 sets (s = 1, 2, ..., S), with each set having a common stimulus, and with the final set being the discrete items. The only change to the constraints below would be on the selection of quantitative and categorical stimuli attributes, and the number of item sets; thus, the summation for these would be over S-1). Items within each set are denoted i_s = 1, 2, ..., I_s. Specifically, for this example, each curricular area has seven stimuli, with each stimulus having seven associated items hypothesized to be at the different θ levels (θ_1 - θ_7). Accordingly, for Stimulus 1 in the personal reading curricular area, the associated items are the first item at each of the seven θ levels as laid out in Table 5 (e.g., items 1, 8, 15, 22, 29, 35, and 43); for Stimulus 2 in the reading area

the associated items are 2, 9, 16, 22, 29, 35, and 44, and so on. Sets are selected into the test if $z_s = 1$. Alternatively, $z_s = 0$ indicates that the set is not selected. Item i_s is denoted $x_{is} = 1$ when it is selected into the test, and $x_{is} = 0$ when it is not in the test.

The *maximin* objective function (van der Linden & Boekkooi-Timminga, 1989) is specified such that information for item i_s at θ_k, denoted as I_s (θ_k), is specified by the target values $T(\theta_k)$ ($k = 1, 2, ..., K$) for each value of the TIF at θ_k. Thus, the TIF at each θ_k is required to fall between upper and lower bounds $[T(\theta_k)\text{-}y, T(\theta_k) + y]$. To control the size of the bounds, $y \geq 0$ is an actual numerical value that defines the width of the interval. The objective function is therefore to minimise y. Further notation is needed to fully explicate the model and is based on those suggested by van der Linden (2000), such that:

1. q_i is the value of item i_s and on quantitative attribute q. For example, the quantitative attribute for item i_s could be item difficulty.
2. r_s denotes the value of stimulus s on quantitative attribute r. For example, the quantitative attribute may be time taken to read a passage of writing.
3. C_g is defined as the set of indices of items with the value g on categorical attribute C, $g = 1, 2, ..., G$. If the categorical attribute is level of processing then the associated items would be those at the deep or the surface level.
4. D_h is a set of stimuli indices with value h on categorical attribute D, $h = 1, 2, ..., H$.
5. n defines the upper (u) and lower (l) bounds for the numbers of items in subsets from the item bank.

The following model formally outlines the *maximin* objective function for items sets, and some practical possible constraints that can be used. Equations 33 and 34 set the TIF to a relative shape and to fall within a certain range. Equations 35 and 36 set the length of the test. Equation 37 sets the number of sets to be selected. The number of items to be selected from an item set is given in Equations 38 and 39. Items to be selected on the basis of their quantitative or categorical attributes are denoted in Equations 40 to 43. Similarly, quantitative and categorical stimulus attributes are defined in Equations 44 to 47. Equations 48 to 50 are definitions of the decision variables.

Minimize y

subject to:

$$\sum_{s=1}^{S}\sum_{i_s=1}^{I_s} I_{i_s}(\theta_k)x_{i_s} + y \geq T(\theta_k), \qquad k = 1,2,\ldots,K, \tag{33}$$

$$\sum_{s=1}^{S}\sum_{i_s=1}^{I_s} I_{i_s}(\theta_k)x_{i_s} - y \leq T(\theta_k), \qquad k = 1,2,\ldots,K, \tag{34}$$

$$\sum_{s=1}^{S}\sum_{i_s=1}^{I_s} x_{i_s} - n_s^{(l)} \geq 0 \tag{35}$$

$$\sum_{s=1}^{S}\sum_{i_s=1}^{I_s} x_{i_s} - n_s^{(u)} \leq 0 \tag{36}$$

$$\sum_{s=1}^{S} z_s = m \tag{37}$$

$$\sum_{i=1}^{I_s} x_{i_s} - n^{(l)} z_s \geq 0, \qquad s = 1,2,\ldots,S, \tag{38}$$

$$\sum_{i=1}^{I_s} x_{i_s} - n^{(u)} z_s \leq 0, \qquad s = 1,2,\ldots,S, \tag{39}$$

$$\sum_{s=1}^{S}\sum_{i=1}^{I_s} q_{i_s} x_{i_s} \geq q^{(l)}, \tag{40}$$

$$\sum_{s=1}^{S}\sum_{i=1}^{I_s} q_{i_s} x_{i_s} \leq q^{(u)}, \tag{41}$$

$$\sum_{s=1}^{S}\sum_{i_s \in C_g} x_{i_s} \geq n_g^{(l)}, \qquad g = 1,2,\ldots,G, \tag{42}$$

$$\sum_{s=1}^{S}\sum_{i_s \in C_g} x_{i_s} \leq n_g^{(u)}, \qquad g = 1,2,\ldots,G, \tag{43}$$

$$\sum_{s=1}^{S} r_s z_s \geq r^{(l)}, \tag{44}$$

$$\sum_{s=1}^{S} r_s z_s \leq r^{(u)}, \tag{45}$$

$$\sum_{s=1}^{S} z_s \geq n_h^{(l)}, \qquad h = 1,2,\ldots,H, \tag{46}$$

$$\sum_{s=1}^{S} z_s \leq n_h^{(u)}, \qquad h = 1,2,\ldots,H, \tag{47}$$

$$y \geq 0 \tag{48}$$

$$x_{i_s} \in \{0,1\}, \quad i_s = 1,2,\ldots,I_s \quad s = 1,2,\ldots S, \tag{49}$$

$$z_s \in \{0,1\}, \quad s = 1,2,\ldots,S, \tag{50}$$

To provide a worked example for selecting set-based items using some of the above constraints, a hypothetical set of test specifications, using item sets in Table 4, are specified, such that:

1. The information at $\theta_1 = -2$ and $\theta_3 = 0$ means that the TIF should fall between a series of lower and upper bounds such that the interval is minimized (Equations 51 & 52).
2. The test must contain no more than 40 items (Equations 53 & 54).
3. Ten sets must be selected (Equations 55).
4. Six sets must come from the personal and the close reading curricular areas (Equation 56).
5. There must be no more than five items per set (Equations 57 & 58).
6. The test can take no longer than 40 minutes to complete (Equations 59 & 60).
7. At least five items $\geq \theta_3$ at the deep processing level from the personal reading area must be selected (Equations 61 & 62).

Minimize y

subject to:

$$\sum_{s=1}^{56}\sum_{i_s=1}^{56} I_{i_s}(\theta_k) x_{i_s} - y \geq T(\theta_k), \qquad k = 1,2,\ldots,k \tag{51}$$

$$\sum_{s=1}^{56}\sum_{i_s=1}^{56} I_{i_s}(\theta_k) x_{i_s} - y \leq T(\theta_k), \qquad k = 1,2,\ldots,k \tag{52}$$

$$\sum_{s=1}^{56}\sum_{i_s=1}^{56} x_{i_s} - 39 \geq 0 \tag{53}$$

$$\sum_{s=1}^{56}\sum_{i_s=1}^{56} x_{i_s} - 40 \leq 0 \tag{54}$$

$$\sum_{s=1}^{56} z_s = 10 \tag{55}$$

$$\sum_{s=1}^{7} z_s + \sum_{s=8}^{14} z_s = 6 \tag{56}$$

$$\sum_{i=1}^{392} x_{i_s} - 3z_s \geq 0, \qquad s = 1,2,\ldots,S \tag{57}$$

$$\sum_{i=1}^{392} x_{i_s} - 5z_s \leq 0, \qquad s = 1,2,\ldots,S \tag{58}$$

$$\sum_{s=1}^{56}\sum_{i=1}^{56} q_{i_s} x_{i_s} \geq 2340, \tag{59}$$

$$\sum_{s=1}^{56}\sum_{i=1}^{56} q_{i_s} x_{i_s} \leq 2400, \tag{60}$$

$$\sum_{s=3}^{7}\sum_{i_s \in CR_{deep}} x_{i_s} \geq 5, \qquad g = 1,2,\ldots,G, \tag{61}$$

$$\sum_{s=3}^{7}\sum_{i_s \in CR_{deep}} x_{i_s} \leq 7, \qquad g = 1,2,\ldots G, \tag{62}$$

$$y \geq 0, \tag{63}$$

$$x_{i_s} \in \{0,1\}, \quad i_s = 1,2,\ldots I_s, \quad s = 1,2,\ldots S, \tag{64}$$

$$z_s \in \{0,1\}, \quad s = 1,2,\ldots S \tag{65}$$

where $i_s \in CR_{deep}$ denotes a set of items at the deep processing level in the close reading curricular domain.

Solving Binary 0-1 and Integer LP Models

The solving of 0-1 and integer LP test assembly models is computationally complex and, in terms of computing time, very demanding (Timminga & Adema, 1995). In other words, the time taken to search for the optimal solution can be prohibitive, and hence the practical benefits of LP are lost. A major concern for the asTTle projects is to ensure that feasible solutions are obtained in reasonable computing time. There are, however, trade-offs between optimality and a relative degree of precision in obtaining a solution to the test assembly problem. As optimality may not be guaranteed, finding solutions as close to optimal as possible is required. In order to ensure that a feasible solution can be obtained close to optimality, heuristics is required. The use of heuristics facilitates close-to-optimal solutions but does affect the precision of the test obtained. The strength of heuristics is that it allows for complex test specification to be realized in practical computing time, and therefore places LP within the realm of the asTTle projects.

LP Algorithms and Heuristics

The main algorithms used to solve linear programming problems are the *simplex* method and the *branch-and-bound* method.

The simplex method is an iterative selection process that achieves optimality by finding successive basic feasible solutions. The procedure moves from one feasible region to another to improve the value of the objective function, and when the objective function cannot be improved any further the solution is optimal and the process is halted. In other words, when a set of items that provides a solution that maximizes the objective function by satisfying all the constraints is found, and no set of items can better it, then it is considered the best possible solution (see Ignizio & Cavalier, 1994; Sutton, 1993; Wolsey, 1998, for a more detailed discussion of the simplex algorithm).

The branch-and-bound method for solving 0-1 LP problems refers to a class of algorithms (see Adema, 1992a; Ignizio & Cavalier, 1994; Sanders, Theunissen, & Bass, 1996; Sutton, 1993; and Wolsey, 1998, for a more detailed discussion of the branch-and-bound algorithm). Basically, the branch-and-bound algorithm starts by calculating a solution to a relaxed model – the relaxed model being real values taking the place of integer variables. For maximization or minimization problems, the value obtained for the objective function for a relaxed solution is an upper or lower bound to the objective function value of the integer solution to be obtained.

Once the relaxed solution is obtained, two new problems are created; one for the variable with a fractional value in the solution, which is fixed at 0, and the other for the same variable fixed at 1. The two problems are then solved. Once a problem is solved, its solution is checked to determine if its value is greater than the best integer solution found thus far. The problem with the best objective function value is split again using one of the other variables with a fractional solution. If the objective function value is no better, then backtracking to the previous solution takes place. If the solution is an integer, then a comparison with the best integer solution found thus far takes place, and the best one is retained. The process continues until all the variables are assigned a value of either 1 or 0, and the objective function value with the best integer value is determined. Timminga et al. (1996, p. 15) suggest that the branch-and-bound algorithm can be

> represented as a search in a tree where each node is a problem with partially fixed variables and the two branches leaving the nodes are the problems associated with fixing another variable. Fortunately it is not necessary to search the whole tree for a solution. As soon as a node produces a solution with a value for the objective function worse than the one at hand, all

problems farther along the branch can be skipped.

A full branch-and-bound search on a large LP problem requires a great deal of time to reach a solution, and thus, to use it as the sole minimization method would be unacceptable in the asTTle projects. To facilitate test assembly using large item banks, test developers can employ heuristics. In general, heuristics facilitates varying degrees of precision depending on the process chosen and the constraints used. (For a more complete discussion of the uses and implications of various heuristics, see Timminga et al., 1996). Stocking, Swanson, and Pearlman (1991) presented the following heuristics for facilitating feasible solutions, based on the work of van der Linden and Boekkooi-Timminga (1989).

1. *Crude linear rounding.* The initial solution to the relaxed problem is obtained, and the decision variables are allowed to take on non-integer values $0 \leq x_i \leq 1$, and the findings are rounded to 0 or 1. A problem with this approach is that it may not reach optimality, or satisfy all of the constraints.
2. *Improved linear rounding.* This approach differs from *crude linear rounding* in that the decision variables are ranked in descending order, and the first n (n = the number of items) are re-rounded to 1 and selected into the test. Again, optimality is not guaranteed, nor may all the constraints be satisfied.
3. *Optimal rounding.* First the relaxed linear solution is obtained ($0 \leq x_i \geq 1$), then all variables equal to 0 or 1 are fixed. Optimality is then achieved by applying the branch-and-bound method. Again, this method does not guarantee an optimal solution to the problem.
4. *First 0-1 solution.* Branch-and-bound methods are used to search for a global solution by discarding many local solutions. After the first 0-1 integer solution is found, the method stops. If a solution to the constraints exists, it is found, although it may not be optimal.
5. *Second 0-1 solution.* This solution is similar to the *first 0-1 solution*, except that the method terminates after the second integer solution is found. Likewise, optimality is not guaranteed.

The application of the simplex and the branch-and-bound methods for solving test assembly problems will facilitate an optimal solution, if it exists. The main limitation of these approaches in terms of the asTTle projects is the computing time needed to reach optimality. Heuristics greatly increases the ability of the test constructor to assemble tests using LP methods, and therefore makes LP feasible to test assembly problems. Of the heuristics outlined above, van der Linden & Boekooi-Timminga (1989) suggest that the *optimal rounding* method provides excellent results that are close to optimal in reasonable computing time. The drawback of heuristics is that they lead to varying degrees of precision when the final solution is obtained, and this should be borne in mind when examining results.

Infeasibility

An issue that can occur in LP is infeasibility. That is, there is a problem in finding a feasible solution that fully meets the test constraints. Although not an uncommon problem, there are methods for overcoming such issues. Swanson & Stocking (1993) note that studies reported in the literature report relatively small item banks of fewer than 1,000 items, and with as few as 50 constraints. In practice, however, item banks tend to be much larger with many more complex constraints, and therefore the probability of finding a feasible solution increases. The message is clear, the larger the item bank, the lower the potential for infeasibility.

Timminga and Adema (1995, p. 422) suggest that "the basic cause of infeasibility is that the set of test requirements is in contradiction with the characteristics of the item bank." An important consideration in the test development phase therefore is the identification of the test specifications and how these relate to the structure of the item bank. Test specifications need to be thoroughly reviewed prior to the application of LP methods in order to decrease the probability of infeasibility. If, however, infeasibility is encountered, then practical

solutions are required to solve the problem. The cause of infeasibility is often difficult to detect, as computer software is unable to identify the problem in the constraints. More often, the problem can be manually traced to impractical constraints on the item bank. For example, if the item bank has 10 items relating to surface learning, and the test specification demand that, 12 items be surface learning, then it is impossible to obtain a solution. In this obvious example, the test constructor can identify the problem and rectify the constraints.

Making changes to the right-hand side of a constraint is one method for increasing the solution space to produce feasible solutions. It is important to understand that changes, for example, in the right-hand side in Equation 15, will not increase the feasible region, as these are unrelated solution space. Changing the values in Equations 35 and 36 will, however, assist in increasing the solution space, and therefore may result in a feasible solution being obtained. Some manipulation (i.e., increase or decrease) of *n* in the right-hand side is needed to facilitate any increase in the solution space.

Another approach to increasing solution space is to examine closely the equality and inequality signs. In general, the use of inequality signs can result in the increase of the solution space if the cause of infeasibility is due to that constraint. Replacing inequality with equality signs can create infeasibility as there may not be an exact solution to such a constraint. Timminga and Adema (1995) suggest that equality signs should be used only where constraints are applied to integer-valued variables. With item sets, one should attempt to increase the number of items associated to a common stimulus to reduce potential infeasibility problems. Thus, for set-based items, and for the asTTle projects, the implication is to have a large item bank with many items associated with each stimulus to assist in reducing issues of infeasibility with set-based items.

As previously stated, knowing the qualities if the item bank can help to identify sources of infeasibility. If, for example, a constraint is too severe, then it should be either modified or deleted in such a way as to increase the feasible region. Combining constraints will also enlarge the solution area. For example, if no solution could be found to the model specified above (i.e., Equations 22 to 32), then Equations 23 and 25 can be combined so that:

$$\sum_{i=1}^{56} x_i + \sum_{i=57}^{98} x_i = 20 \qquad (66)$$

Combining these two constraints results in the same number of items being selected, but it allows more items to be drawn from a curricular area that is best able to provide information that meets the TIF. It may be that one curricular area can provide 12 items, and the other area can provide 8 items to then meet all the test specifications. Test specificity is somewhat compromised in this example, but not to a great extent. The same approach can also be applied to combining constraints on item sets to avoid infeasibility. In general, the problem of combining constraints is that test specificity may be compromised. As a consequence, if constraints are combined, then the final solution should be examined to determine its acceptability in terms of the original test specifications.

Problems may also be encountered if the objective function and one or more of the constraints are in conflict. For example, if items are designated not to be in the test (assigned 0, and left out the problem), then modification of the objective function or the constraints is required to take this into account.

Although the literature on infeasibility and test assembly problems is limited, Timminiga and Adema (1995) suggest the following analytical strategy to overcome infeasibility.

1. Check that the constraints are compatible with the item bank.
2. Solve the relaxed LP problem. If infeasibility occurs, then check the constraints using some of the approaches outlined above to modify the constraints.
3. If the relaxed problem can be solved, then solve the 0-1 LP problem.
4. If the 0-1 LP problem has a solution, the test is specified.
5. If the relaxed problem cannot be solved, then check the constraints.

6. If the 0-1 LP is still infeasible, then a heuristic method should be applied to the problem.
7. If infeasibility continues with the 0-1 LP problem, then the test constructor should modify the test specifications.

In sum, problems with infeasibility will be generally be traceable to incompatibility with the item bank or to constraints that are in conflict with one another or with the objective function. Most infeasibility can be overcome by knowing the structure of the item bank and by setting reasonable test specifications. Timminiga and Adema (1995) point out that if the above procedures are followed, then the test constructor should always be able to assemble a test.

Conclusions

The review presented above suggests that LP is highly applicable to the asTTle projects, as it provides a search mechanism that will furnish teacher-specified tests with a high degree of precision. Linear programming simultaneously satisfies both content and statistical attributes that will be specified by the teacher through the use of an objective function and a series of linear constraints. As the teacher choices will be limited (difficulty, curricular area, level of processing, etc.), then writing these in LP form should not be too problematic.

An issue for the computer programmers of the asTTle projects will be to set an objective function that can conform closely to the teachers' test specifications in terms of the range of ability. In general, three to five ability points should be sufficient for the range of TIF. With set-based items, setting the objective function should not be too problematic, given that that *maximin* model (van der Linden, 2000; van der Linden & Boekkooi-Timminga, 1989) will allow for the TIF, and will be specified to fall between an upper and lower bound. The task of the computer programmers will be to determine the width of this interval to alleviate potential infeasibility. Finding a set of values that will decrease the potential of infeasibility, but still maximize information, will be essential to the success of the asTTle projects. Setting these values will be dependent on the quality of the items and the amount of information they have. One method for determining the upper and lower bound on the TIF would be to draw repeated random samples of 40 items to determine the upper and lower values of the TIF, which could then serve as the values for the TIF in all tests.

The practical constraints presented above should provide the computer programmers with the relevant information to be able to formulate the teachers' choices (i.e, curricular area, item format, administration time, level of processing, and item difficulty) into linear constraints. The set-based items approach outlined by van der Linden (2000) will allow for the constraints to be designated between upper and lower bounds. As with the TIF, the use of upper and lower bounds on the practical constraints will provide the computer programmers with some flexibility when dealing with possible infeasibility issues.

Solving LP problems should not be problematic for the asTTle projects, as heuristics are available that should provide close-to-optimal results in reasonable computing time. Van der Linden and Boekkooi-Timminga (1989) state that as the size of the item bank increases, the amount of computing time needed to find an exact solution increases. In the future, items are likely to be added to the asTTle item bank, and therefore the adoption of heuristics, such as the optimal rounding heuristic, will provide accurate tests (to be generated quickly) that meet detailed specifications.

Recommendations

1. Have a detailed map of the item bank showing the quantitative and qualitative aspects of the items, and how these relate to the curricular areas (see Tables 1, 2, and 4, and also Timminga, van der Linden, & Schweizer, 1996).
2. Although there are seven ability levels to be covered across the asTTle projects, the teachers should be constrained to choose no more than five ability levels, as increases on this number will likely result in difficulty in approximating the TIF or result in infeasibility.

3. As the asTTle project will use mainly set-based items, the method for the simultaneous selection of items and sets outlined by van der Linden (2000) is recommended. This method is shown to provide a more accurate approximation of the objective function in reasonable computing time using a large number of constraints (e.g., 24 stimuli with 5–12 items per stimuli (498 items) with over 200 constraints across 2 tests; the LP models were solved in 1–5 minutes).
4. For set-based items, the TIF should be specified to be a relative shape (the *maximin* model developed by van der Linden & Boekkooi-Timminga, 1989, and van der Linden, 2000). This approach is the most flexible approach, given that the TIF can be set to fall between lower and upper bounds. Determining the width of the interval will be a critical aspect of the computer programming.
5. For solving the binary 0-1 and integer problems, the optimal rounding method is more flexible in its approach to finding feasible solutions in reasonable computing time. This method will be invaluable for finding accurate and fast solutions to the teachers' choices, as the size of the item bank for the asTTle projects is likely to increase in the coming years.

References

Adema, J. J. (1990). The construction of customized two-stage tests. *Journal of Educational Measurement, 27*(3), 241–253.

Adema, J. J. (1992a). Implementations of the branch-and-bound method for test construction problems. *Methodika, 6*, 99–117.

Adema, J. J. (1992b). Methods and models for the construction of weakly parallel test. *Applied Psychological Measurement, 16*(1) 53–63.

Adema, J. J., & van der Linden, W. J. (1989). Algorithms for computerized test construction using classical item parameters. *Journal of Educational Statistics, 14*(3), 279–290.

Adema, J. J., Boekkooi-Timminga, E., & Gademann, A. J. R. M. (1992). Computerized test construction. In M. Wilson (Ed.), *Objective Measurement: Theory into Practice* (Vol. 1, pp. 261–273). Norwood, New Jersey: Ablex.

Adema, J. J., Boekkooi-Timminga, E., & van der Linden, W. J. (1991). Achievement test construction using 0-1 linear programming. *European Journal of Operational Research, 55*, 103–111.

Armstrong, R. D., Jones, D. H., & Wang, Z. (1994). Automated test construction using classical test theory. *Journal of Educational Statistics, 19*(1), 73–90.

Baker, F. B., Cohen, A. S., & Barmish, B. R. (1988). Item characteristics of tests constructed by linear programming. *Applied Psychological Measurement, 12*(2), 189–199.

Berger, M. P. F. (1998). Optimal design of tests with items with dichotomous and polytomous response formats. *Applied Psychological Measurement, 22*(3), 248-258.

Berger, M. P. F., & Mathijssen, E. (1997). Optimal test designs for polytomously scored items. *British Journal of Mathematical and Statistical Psychology, 50*, 127–141.

Berger, P. F., & VeerKamp, W. J. J. (1996). A review of selection methods for optimal test design. In G. Engelhard & M. Wilson (Eds.), *Objective Measurement: Theory into Practice* (Vol. 3, pp. 437–455). Norwood, New Jersey: Ablex.

Birnbaum, A. (1968). Some latent trait models. In F. M. Lord & M. R. Novick (Eds.), *Statistical Theories for Mental Test Scores*. Reading, MA: Addison-Wesley.

Boekooi-Timminga, E. (1987). Simultaneous test construction by zero-one programming. *Methodika, 1*(2), 101–112.

Boekooi-Timminga, E. (1990a). The construction of parallel tests from IRT-based item banks. *Journal of Educational Measurement, 15*(2), 129–145.

Boekooi-Timminga, E. (1990b). A cluster-based method for test construction. *Applied Psychological Measurement, 14*(4), 341–354.

Boekooi-Timminga, E. (1993). Computer assisted test construction. *Social Science Computer Review, 11*(3), 292–300.

de Gruijter, D. N. M. (1990). Test construction by means of linear programming. *Applied Psychological Measurement, 14*(2), 175–181.

Fletcher, R. B. & Hattie, J. A. (In review). Automated Test Assembly for Polytomous Items: The Application of Linear Programming to Sports Psychology Test Construction.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of Item Response Theory*. London: Sage.

Harris, D. (1996). Comparison of 1-, 2-, and 3-parameter models. *Educational Measurement: Issues and Practice, 15,* 157–163.

Ignizio, J. P., & Cavalier, T. M. (1994). *Introduction to Linear Programming*. Englewood Cliffs, New Jersey: Prentice-Hall.

Lord, F. M. (1952). A theory of test scores. *Psychometric Monograph, 7*.

Lord, F. M. (1977). Practical applications of item characteristic curve theory. *Journal of Educational Measurement, 14*, 117–138.

Lord, F. M. (1980). *Applications of Item Response Theory to Practical Testing Problems*. New Jersey: Lawrence Erlbaum.

Sanders, P. F., Theunissen, T. J. J. M., & Baas, S. S. (1996). The optimization of decision studies. In G. Engelhard & M. Wilson (Eds.), *Objective Measurement: Theory into Practice* (Vol. 3, pp. 301–312). Norwood, New Jersey: Ablex.

Stocking, M. L., Swanson, L., & Pearlman, M. (1991). *Automated item selection using item response theory*. (Research Report 91-9). Princeton, NJ: Educational Testing Service.

Stocking, M. L., Swanson, L., & Pearlman, M. (1993). Application of an automated item selection method to real data. *Applied Psychological Measurement, 17*(2), 167–176.

Sultan, A. (1993). *Linear Programming: An Introduction with Applications.* Boston: Academic Pres, Inc.

Swanson, L., & Stocking, M. L. (1993). A model and heuristic for solving very large item selection problems. *Applied Psychological Measurement, 17*(2), 151–166.

Theunissen, T. J. J. M. (1985). Binary programming and test design. *Psychometrika, 50*(4), 411–420.

Theunissen, T. J. J. M. (1986). Some applications of optimization algorithms in test design and adaptive testing. *Applied Psychological Measurement, 10*(4), 381–389.

Timminga, E., & Adema, J. J. (1995). Test construction from item banks. In G. H. Fischer & I. W. Molenaar (Eds.), *Rasch Models: Foundations, Recent Developments and Applications*. New York: Springer-Verlag.

Timminga, E., & Adema, J. J. (1996). An interactive approach to modifying infeasible 0-1 linear models for test construction. In G. Engelhard & M. Wilson (Eds.), *Objective Measurement: Theory into Practice* (Vol. 3, pp. 419–436). Norwood, New Jersey: Ablex.

Timminga, E., van der Linden, W. J., & Schweizer, D. A. (1996). CONTEST (Version 2) [Computer Program]. Groningen: ProGAMMA.

van der Linden, W. J. (1996a). Assembling tests for the measurement of multiple traits. *Applied Psychological Measurement, 20*(4), 373–388.

van der Linden, W. J. (1994). Optimum design in item response theory: 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). Optimal assembly of tests with item sets. *Applied Psychological Measurement, 24*(3), 225–240.

van der Linden, W. J., & Boekkooi-Timminga, E. (1989). A maximin model for test design with practical constraints. *Psychometrika, 54*(2), 137–247.

Wolsey, L.A. (1998). *Integer Programming.* New York: John Wiley & Sons.

Appendix One – Item Bank Structure

Appendix Table 1

Item Bank Structure for the Closed-Choice Items

Curricular Area	Items in each Area	C-C Items	C-C Items at θ_1	C-C Items at θ_2	C-C Items at θ_3	C-C Items at θ_4	C-C Items at θ_5	C-C Items at θ_6	C-C Items at θ_7	C-C Items at Deep Processing	C-C Items at Surface Processing
Personal Reading	1–56	1–42	1–6	7–12	13–18	19–24	25–30	31–36	37–42	1–3, 7–9, 13–15, 19–21, 25–27, 31–33, 37–39	4–6, 10–12, 16–18, 21–24, 28–30, 34–36, 40–42
Close Reading	57–112	57–98	57–62	63–68	69–74	75–80	81–86	87–92	93–98	57–59, 63–65, 69–71, 75–77, 81–83, 87–89, 93–95	60–62, 66–68, 72–74, 78–80, 84–86, 90–92, 96–98
Expressive Writing	113–168	113–154	113–118	119–124	125–130	131–136	137–142	143–148	149–154	113–115, 119–121, 125–127, 131–133, 137–139, 143–145, 149–151	116–118, 122–124, 128–130, 134–136, 140–142, 146–148, 152–154
Poetic Writing	169–224	169–210	169–174	175–180	181–186	187–192	193–198	199–204	205–210	169–171, 175–177, 181–183, 187–189, 193–195, 199–201, 205–207	172–174, 178–180, 184–186, 190–192, 195–197, 202–204, 207–210
Transitional Writing	225–280	225–266	225–230	231–236	237–242	243–248	249–254	255–260	261–266	225–227, 231–233, 237–239, 243–245, 249–251, 255–257, 261–263	228–230, 234–236, 230–232, 246–248, 251–253, 258–260, 264–266
Exploring Writing	281–336	281–322	281–286	287–292	293–298	299–304	305–310	311–316	317–322	281–283, 287–289, 293–295, 299–301, 305–307, 311–313, 317–319	284–286, 290–292, 295–297, 301–303, 308–310, 314–316, 320–322
Thinking Critically	337–392	337–378	337–342	343–348	349–354	355–360	361–366	367–372	373–378	337–339, 343–345, 349–351, 355–357, 361–363, 367–369, 373–375	340–342, 345–347, 352–354, 358–360, 364–366, 370–372, 376–378
Processing Information	393–448	393–434	393–398	399–404	405–410	411–416	417–422	423–428	429–434	393–395, 399–401, 405–407, 411–413, 417–419, 423–425, 429–431	396–398, 402–404, 408–410, 414–416, 420–422, 426–428, 433–434

Note: C-C denotes closed-choice items.

Appendix Table 2
Item Bank Structure for the Open-Ended Items

Curricular Area	Items in each Area	O-E Items	O-E Items at θ_1	O-E Items at θ_2	O-E Items at θ_3	O-E Items at θ_4	O-E Items at θ_5	O-E Items at θ_6	O-E Items at θ_7	O-E Items at Deep Processing	O-E Items at Surface Processing
Personal Reading	1–56	43–56	43–44	45–46	47–48	49–50	51–52	53–54	55–56	43, 45, 47, 49, 51, 53, 55	44, 46, 48, 50, 52, 54, 56
Close Reading	57–112	99–112	99–100	101–102	103–104	105–106	107–108	109–100	111–112	99, 101, 103, 105, 107, 109, 111	100, 102, 104, 106, 108, 110, 112
Expressive Writing	113–168	155–168	155–156	157–158	159–160	161–162	163–164	165–166	167–168	155, 157, 159, 161, 163165, 167	156, 158, 160, 162, 164, 166, 168
Poetic Writing	169–224	211–224	211–212	213–214	215–216	217–218	219–220	221–222	223–224	211, 213, 215, 217, 219, 221, 223	212, 214, 216, 218, 220, 222, 224
Transitional Writing	267–280	267–280	267–268	269–270	271–272	273–274	275–276	277–278	279–280	267, 269, 271, 273, 275, 277, 279	268, 270, 272, 274, 276, 278, 280
Exploring Writing	281–336	323–336	323–324	325–326	327–328	329–330	331–332	333–334	335–336	323, 325, 327, 329, 331, 333, 335	324, 326, 328, 330, 332, 334, 336
Thinking Critically	337–392	379–392	379–380	381–382	383–384	385–386	387–388	389–390	391–392	379, 381, 383, 385, 387, 389, 391	380, 382, 384, 386, 388, 390, 392
Processing Information	393–448	435–448	435–436	437–438	439–440	441–442	443–444	445–446	447–448	435, 437, 439, 441, 443, 445, 447	436, 438, 440, 442, 444, 446, 448

Note: O-E denotes open-ended items.

Appendix Table 3
Hypothetical Number of Items at each θ across the Three Levels based on an Item Bank of 448 Items

	θ_1 = -3	θ_2 = -2	θ_3 = -1	θ_4 = 0	θ_5 = +1	θ_6 = +2	θ_7 = +3
Level 1	very easy	easy	average	hard	very hard		
Level 2		very easy	easy	average	hard	very hard	
Level 3			very easy	easy	average	hard	very hard
	64 items	64 items	64 items	64 items	64 items	64 items	64 items

Note: θ_2 -θ_6 share the same items due the overlapping of the grade/ability levels.

Appendix Table 4
Item Attribute and Constraint Levels for Set-Based Items

Level	Attribute Level	Constraint Level
Item	Can be quantitative or categorical; e.g., content, processing level, item parameters, item format, word count, response time.	Inclusion/exclusion of items with certain attribute values into the test; e.g., “the first five items should be surface processing”.
Stimulus	Can be quantitative or categorical; e.g., curricular area, processing level, reading passage.	Inclusion/exclusion of stimuli with specific attribute values into the test; e.g., “no stimulus should be longer than 250 words”.
Item Set Level	Mainly quantitative; e.g., number of items in the set.	Control the distribution of categorical attributes. These are usually set between lower and upper bounds; e.g., “each item set should have at least three items at the deep processing level”.
Test Level	Mainly quantitative; e.g., test length, deviation of the test information function and the TIF.	Distributions or functions of items or stimulus attributes; e.g., “no more than three open-ended questions should be in the test”.

Appendix Table 5
Hypothetical Item Bank with Set-based Items

Curricular Area	Items in each Area	C-C Items at θ_1	C-C Items at θ_2	C-C Items at θ_3	C-C Items at θ_4	C-C Items at θ_5	C-C Items at θ_6	C-C Items at θ_7	C-C Items at Deep Processing	C-C Items at Surface Processing
Personal Reading (PR)	1–49	1–7	8–14	15–21	22–28	29–35	36–42	43–49	1–4, 8–11, 15–18, 22–25, 29–32, 36–39, 43–46	5–7, 12–14, 19–21, 26–28, 33–35, 40–42, 47–49
Close Reading (CR)	50–98	50–56	57–63	64–70	71–77	78–84	85–91	92–98	50–53, 57–60, 64–67, 71–74, 78–81, 85–88, 92–95	54–56, 61–63, 68–70, 75–77, 82–84, 89–91, 96–98
Expressive Writing (EW)	99–147	99–105	106–112	113–119	120–126	127–133	134–140	141–147	99–102, 106–109, 113–116, 120–123, 127–130, 134–137, 141–144	103–105, 110–112, 117–119, 124–126, 131–133, 138–140, 145–147
Poetic Writing (PW)	148–196	148–154	155–161	162–168	169–175	176–182	183–189	190–196	148–151, 155–158, 162–165, 169–172, 176–179, 183–186, 190–193	152–154, 159–161, 166–168, 173–175, 180–182, 187–189, 194–196
Transitional Writing (TR)	197–245	197–203	204–210	211–217	218–224	225–231	232–238	239–245	197–200, 204–207, 211–214, 218–221, 225–228, 232–235, 239–242	201–203, 208–210, 215–217, 222–224, 229–231, 236–238, 243–245
Exploring Writing (EXW)	246–294	246–252	253–259	260–266	267–273	274–280	281–287	288–294	246–249, 253–256, 260–263, 267–270, 274–277, 281–284, 288–291	250–252, 257–259, 264–266, 271–272, 278–280, 285–287, 292–294
Thinking Critically (TC)	295–343	295–301	302–308	309–315	316–322	323–329	330–336	337–343	295–298, 302–305, 309–312, 316–319, 323–326, 330–333, 337–340	299–301, 306–308, 313–315, 320–322, 327–329, 334–336, 341–343
Processing Information (PI)	344–392	344–350	351–357	358–364	365–371	372–378	379–385	386–392	344–347, 351–354, 358–361, 365–368, 372–375, 379–382, 386–389	348–350, 355–357, 362–364, 369–371, 376–378, 383–385, 390–392