Open access • Journal Article • DOI:10.1177/014662168601000406

# Some applications of optimization algorithms in test design and adaptive testing



T.J.J.M. Theunissen

**Published on:** 01 Dec 1986 - Applied Psychological Measurement (Sage Publications)

**Topics:** Test functions for optimization, Optimization problem, Test design and Computerized adaptive testing

# Some Applications of Optimization Algorithms in Test Design and Adaptive Testing

**T. J. J. M. Theunissen**
**National Institute for Educational Measurement (CITO)**
**Arnhem, The Netherlands**

Some test design problems can be seen as combinatorial optimization problems. Several suggestions are presented, with various possible applications. Results obtained thus far are promising; the methods suggested can also be used with highly structured test specifications.

Even in a relatively small item bank of 500 items (calibrated with an item response model), the total possible number of tests, $2^{500}$, is so large that the fastest computers available would require many years for a complete enumeration of all possible tests. Due to physical constraints (e.g., the finite speed of signal transmission), this will always remain so. Many combinatorial problems simply cannot be completely enumerated within a practical period of time. Choosing from the set of all possible tests would therefore be equally impractical.

Nevertheless, tests can be constructed by selecting items from a pool of calibrated items. Two considerations generally guide the test construction process. First, the test constructor has a purpose in mind, which is usually translated into an objective that suits the purpose. For example, if the goal is to design a test for the selection of very gifted students for scholarships, only the 20% most difficult items from the pool of 500 might be used. This reduces the total number of tests to $2^{100}$, which is, however, still much too large for complete enumeration. Second, for various reasons the test constructor might decide to constrain his/her choices; for example, a test must be of a certain length, say 50 items. This would reduce the number of possible tests to $\binom{500}{50}$, which, although much less than $2^{500}$, is still too large for practical enumeration. Both restrictions together result in $\binom{100}{50}$ possible tests. Because this set is also still very large, the chance that any particular test will be chosen is extremely small. Therefore, obviously, if items must be selected according to some test specification and the choice has to be "best" on some criterion (which generally requires that one *particular* test be chosen), an additional choice mechanism must be used. Suggestions for such a mechanism are to be found in (discrete) optimization theory.

Recently, Theunissen (1985) demonstrated the theoretical possibility and practical feasibility of treating test design as a multidimensional Knapsack problem, or KP. Numerous optimization problems in various disciplines can be modeled as KPs. A general formulation of a KP is: Minimize

$$\sum_{i=1}^{n} w_i x_i \tag{1}$$

subject to

$$\sum_{i=1}^{n} p_i x_i \geq P \tag{2}$$

*APPLIED PSYCHOLOGICAL MEASUREMENT*
*Vol. 10, No. 4, December 1986, pp. 381–389*

*0146-6216/86/040381-09$1.70*

and

$$x_i \in (0,1) \qquad (i = 1, 2, \ldots, n) \quad , \tag{3}$$

where $x_i$ are variables indicating the absence or presence of objects,
$w_i$ are their weights,
$p_i$ are profits, and
$P$ is total profit.

Formulated this way, the KP tries to fill a knapsack with objects so that its total weight is minimal but its content value is never less than $P$. If Expression 3 is relaxed to $x_i \geq 0$, it is a general linear programming problem; if Expression 3 is relaxed to $x_i \geq$ (0, integer), it is an integer programming problem; as it stands, it is called a binary or zero-one problem. Expression 1 is called the objective function or target function; Expressions 2 and 3 are called the constraints. In this paper some meaningful interpretations of target functions and constraints will be presented in terms of psychometric ideas.

Because accuracy of measurement plays a key role in all types of tests, the concepts of item information and test information will play a part in the formulation of either psychometrically meaningful target functions or constraints, or both. The development below uses an important property of $I(\theta, u_g)$, the item information function, which is its relation to $I(\theta)$ (assuming maximum likelihood estimates of $\theta$), given by

$$I(\theta) = \sum_{g=1}^{n} I(\theta, u_g) \quad . \tag{4}$$

In many situations, interest is primarily focused not on the total test information function but on the information of the test at one or several important $\theta$ points on the scale, or at most the test information for a restricted $\theta$ range. In the former case, specification of a small number of points is usually sufficient (Theunissen, 1985).

In this paper, some possible applications of discrete or combinatorial optimization theory in test design problems will be presented. First, an isomorphism between KPs and certain test design problems will be demonstrated. Because large numbers of calibrated items may result in excessive demands on computer time, suggestions regarding approximation algorithms are presented below. Because item collections may have quite complex structures and the possibility exists that these structures must be mirrored in the test to be designed, it may be useful to have the means to do so; suggestions on this matter are also given. Finally, a different optimization technique will be presented that is useful in some circumstances, together with several possible applications and some results.

## Some Test Design Problems as KPs

Consider the following continuous KP: Maximize

$$\sum_{i=1}^{n} w_i x_i \tag{5}$$

subject to

$$\sum_{i=1}^{n} p_i x_i \leq P \tag{6}$$

and

$$0 \leq x_i \leq 1 \qquad (i = 1, 2, \ldots, n) \quad , \tag{7}$$

and assume the $x_i$ ordered as in

$$\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \ldots \frac{w_n}{p_n} \quad ; \tag{8}$$

then a theorem by Dantzig (1957) says that the optimal solution to this KP is

$$x_i = 1 \qquad (i = 1, 2, \ldots, r) \tag{9}$$

$$x_i = 0 \qquad (i = r + 2, \ldots, n) \tag{10}$$

$$x_{r+1} = \left(P - \sum_{i=1}^{r} p_i\right) \Big/ p_{r+1} \tag{11}$$

and $r$ is the largest index for which

$$\sum_{i=1}^{r} p_i \leq P \quad . \tag{12}$$

If $P$ is the maximum number of items, $w_i$ the item information at *one* specified $\theta$ point for all items, all $p_i$ are equal to 1, and the ordering is as in Equation 8, Dantzig's theorem says that the optimal result is found by ranking all items according to their information at the specified $\theta$ point and taking the first $P$ items, which of course is in accordance with item response theory for the case where a test needs to be designed, with test information maximized at one $\theta$ point and at most of length $P$.

The KP as defined by Equations 1–3 is called a one-dimensional KP, because there is only one constraint, Equation 2. If several constraints are imposed on the assignment of objects to one knapsack, the problem is called a multidimensional KP. Test design problems as multidimensional KPs are treated in Theunissen (1985) and Timminga (1985). In Theunissen (1985) the test information desired is specified for $m$ $\theta$ points and the purpose is to construct a test according to that specification with the minimum number of items (e.g., for reasons of economy). In matrix notation: Minimize

$$\mathbf{V}^T\mathbf{X} \tag{13}$$

subject to

$$\mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{14}$$

and

$$x_i = 0 \text{ or } 1 \qquad (i = 1, 2, \ldots, n) \quad , \tag{15}$$

where $\mathbf{V}$ is an $n$ vector with all elements equal to 1,
$\mathbf{X}$ is an $n$ vector with elements $x_i$,
$\mathbf{b}$ is an $m$ vector with elements being the test information specification at $m$ $\theta$ points, and
$\mathbf{A}$ is an $m \times n$ matrix where the elements of each one of the $m$ rows are the item information function values for all items for the corresponding $\theta$ point and $n$ is the number of items (for example, in an item bank).

(The target $\mathbf{V}^T\mathbf{X}$ is thus a minimum-length test that fulfills the test specification $\mathbf{A}\mathbf{x} \geq \mathbf{b}$.)

In Timminga (1985), again test information desired is specified for $m$ $\theta$ points, but the purpose now is to construct a test so that the information function has minimal distance (approaching from above) to the specified information function (a quadratic criterion is not used, since this leads to quadratic programming, for which computer programs are relatively scarce). Another restriction used is to fix the number of items (taking care that a feasible set of solutions remains). In formal notation: Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_i a_{ij} \tag{16}$$

subject to

$$\sum_{i=1}^{n} x_i a_{ij} \geq b_j \tag{17}$$

$$\sum_{i=1}^{n} x_i = p \tag{18}$$

and

$$x_i = 0 \text{ or } 1 \qquad (i = 1, 2, \ldots, n) \qquad (j = 1, 2, \ldots, m) \quad , \tag{19}$$

where $a_{ij}$ is the item information for item $i$ at $\theta$ point $j$,
$b_j$ is the specified test information at $\theta$ point $j$, and
$p$ is the number of items to be selected.

(The target is thus a minimal deviation from one of the test specifications, in this case $\sum_{i=1}^{n} x_i a_{ij} \geq b_i$.)

In linear programming, the maximization of a function is equivalent to the minimization of the negative of the same function. Rao (1984, section 3.2) presented some examples. However, some test design problems are more naturally formulated as minimization problems (as in Equations 13–15) and others as maximization problems (for instance, when information is to be maximized at several $\theta$ points, given a certain test length). The most natural formulation will be presented each time. Each programming problem can be formulated in standard form by turning it into a minimization form and, where necessary, by changing constraints to equalities by adding or subtracting so-called slack variables.

A number of results using Equations 13–15 are presented in Theunissen (1985). A substantial number of results using Equations 16–18 can be found in Timminga (1985). Timminga presented one set of results that corresponds to (part of) Table 2 in Theunissen (1985). These results are presented in Table 1. The same problem identification as in Theunissen's Table 2 is used. Timminga (1985) found that for the one-parameter logistic (Rasch) model, the number of necessary items becomes larger using Equations 16–18, while the selected items differ markedly. For the three-parameter model, the results are very similar for both sets of equations.

Table 1
Number of Items Selected (n) and Number of Iterations Required (IT) from Item Banks of Size N Using Equations 16-19 and a Multidimensional KP with Specified Information Functions

| Data Set | N | Model | Parameter | Information Function | n | IT |
|---|---|---|---|---|---|---|
| 1A | 300 | 1-par | a = 1<br>b = N(0,1)<br>c = 0 | I(0) $\geq$ 16 | 65 | 66 |
| 1B | 300 | 1-par | a = 1<br>b = N(0,1)<br>c = 0 | I(-1) $\geq$ 4<br>I(0) $\geq$ 4<br>I(1) $\geq$ 4 | 30 | 106 |
| 1E | 300 | 1-par | a = 1<br>b = N(0,1)<br>c = 0 | I(-2) $\geq$ 4<br>I(-1) $\geq$ 4<br>I(0) $\geq$ 4<br>I(1) $\geq$ 4<br>I(2) $\geq$ 4 | 35 | 97 |
| 2B | 300 | 3-par | a = U(0.5;1.5)<br>b = N(0,1)<br>c = 0.2 | I(-1) $\geq$ 1<br>I(0) $\geq$ 16<br>I(1) $\geq$ 4 | 58 | 77 |
| 2C | 500 | 3-par | a = U(0.5;1.5)<br>b = N(0,1)<br>c = 0.2 | I(-1) $\geq$ 1<br>I(0) $\geq$ 16<br>I(1) $\geq$ 4 | 52 | 69 |
| 2D | 1000 | 3-par | a = U(0.5;1.5)<br>b = N(0,1)<br>c = 0.2 | I(-1) $\geq$ 1<br>I(0) $\geq$ 16<br>I(1) $\geq$ 4 | 48 | 72 |

Numerous algorithms for the solution of these types of combinatorial optimization problems have been developed (for details see Syslo, Deo, & Kowalik, 1983; Taha, 1975). These algorithms try to find an exact solution. Unfortunately, none of them is efficient in the sense of Papadimitriou and Steiglitz (1982), where an efficient algorithm is defined as one that requires a number of computational steps that grows as a polynomial in the size of the input. Integer programming is known as N(on)-P(olynomial)-complete (Papadimitriou & Steiglitz, 1982), which means that a correct solution may require an exponential amount of time, although occasions where computer time requirements were really excessive have not yet been encountered. However, as the number of items in a bank grows and the number of constraints grows (e.g., due to complex subdomain specifications), it might be very useful to have approximation algorithms that are much faster and produce solutions that are near optimal.

## Approximations

Approximation algorithms for one-dimensional KPs are numerous, but are rather trivial in the context of the test design problems as specified in this paper. Attempts at approximation of multidimensional KPs with test design problems of the kind specified above have been made by Boomsma (1986) and Verstralen (Theunissen & Verstralen, 1986).

Consider the following maximization problem

(which can always be translated back to a minimization problem as in Equations 13–15): Maximize

$$\mathbf{V}^T\mathbf{X} \tag{20}$$

subject to

$$\mathbf{Ax} \leqslant \mathbf{b} \tag{21}$$

and

$$x_i = 0 \text{ or } 1 \qquad (i = 1, 2, \ldots, n) \tag{22}$$

where the symbols have the same meaning as in Equations 13–15. Then a theorem by Everett (Salkin, 1975) states that if $\boldsymbol{\lambda}$ is an $m$-column vector of Lagrange multipliers, and $\mathbf{x}^0$ solves for the following Lagrangian function: Maximize

$$L(X) = \mathbf{V}^T\mathbf{X} - \boldsymbol{\lambda}\mathbf{Ax} \tag{23}$$

subject to

$$x_i = 0 \text{ or } 1 \quad , \tag{24}$$

$\mathbf{x}^0$ will also solve Equations 20–22, with $\mathbf{b}$ replaced by $\mathbf{x}^0$.

Boomsma (1986) investigated an algorithm that systematically varies $\boldsymbol{\lambda}$, until a vector $\mathbf{x}^0$ is found which either gives $\mathbf{b} = \mathbf{x}^0$, or approximates it as closely as possible. Several methods for finding $\boldsymbol{\lambda}$ have been suggested (see Salkin, 1975). Boomsma investigated this algorithm from various points of view. Some intermediate results concerning computer time requirements will be presented below.

It is known, however, that the quality of the solutions of this algorithm (quality in the sense of approaching the optimal solution) depends strongly on the density of the constraint equations, $S$, defined as

$$S = \frac{\sum_{j=1}^{m} \left( \frac{b_j}{\sum_{i=1}^{n} a_{ij}} \right)}{m} \quad . \tag{25}$$

If $S$ approaches 1, the approximations are very good. If $S$ is small, the algorithm does not work well. Closer inspection of these results suggests that (for the Rasch model) $S$ is dependent on the distribution of difficulties in relation to the test specification. If a uniform test information function is specified (e.g., for a diagnostic test) for a wide $\theta$ continuum and the difficulties are distributed as, for example, $N(0,1)$, many $a_{ij}$ values will be low, and $S$ will be large. Equation 25 also suggests that test specifications with high values of information for the selected $\theta$ points (high $b_j$) will increase $S$; the algorithm might work better for large than for small tests, given the same item bank. Further investigation by Boomsma is in progress.

Verstralen (Theunissen & Verstralen, 1986) has developed a simple and fast algorithm that, so far, reaches the optimum or comes very close to it (although it is not yet known whether this will always be the case). The algorithm consists of the following steps:

1. Using the same indices as in Equations 13–15, apply Dantzig's theorem by sorting the indices $i$ for each $b$.
2. Find the largest value $b_j$ ($j = 1, 2, \ldots, m$).
3. Remove the first (rank ordered) index $i$ from all rows $j = 1, \ldots, m$.
4. Add item $i$ to the test.
5. Compute new $b_j$, $b'_j = b_j - a_{ij}$ for all $j$. Go back to Step 2 if any $b_j > 0$, else stop.

Further refinements (such as backtracking) of this algorithm led to more frequent optimal solutions (for details, see Theunissen & Verstralen, 1986).

### Structured Optimal Item Selection

It may often be necessary to define a structure on the item bank; an example is when sampling from content subdomains is considered desirable. If the number of items is fixed, this can simply be done by adding constraints of the following form:

$$\sum_{i=1}^{r} x_i = n_1 \tag{26}$$

$$\sum_{i=r+1}^{s} x_i = n_2 \tag{27}$$

$$\sum_{i=s+1}^{t} x_i = n_3 \quad , \tag{28}$$

where $n_1 + n_2 + n_3 = N$ (the number of items in the test), $t$ is the number of items in the bank, and it is assumed that the items are previously ordered into three subdomains. If the number of items for the test to be designed is not fixed, proportional

drawing of items from subdomains can be done by restrictions of the following type:

$$a \sum_{i=1}^{r} x_i = b \sum_{i=r+1}^{s} x_i \quad . \tag{29}$$

The ratio of $a$ to $b$ determines the proportions.

Frequently, it may be useful to structure the selection process in more detail. A simple case where this might be necessary is the situation where the bank is filled with a set of items that contains subsets, the members of which differ only in trivial detail (e.g., simple arithmetic items that only differ in numerical detail). In such cases it is useful to specify conditional constraints of the form

$$\text{if } x_i > 0 \text{ then } x_j = 0, \quad x_i \text{ and } x_j = 0 \text{ or } 1 \quad , \tag{30}$$

meaning that when item $i$ is selected, item $j$ must not be selected. Formulating Equation 30 is equivalent to

$$x_i = 0 \quad \text{or} \quad x_j = 0 \quad . \tag{31}$$

By introducing the binary variable $q$, $q = 0$ or 1, Equation 31 can be written as constraints, as follows:

$$x_i = 1q \tag{32}$$

$$x_j = (1-q)1 \quad . \tag{33}$$

Occasionally, it may be necessary (e.g., for diagnostic purposes; see McArthur & Choppin, 1984) to direct the item selection procedure in a more complex and subtle manner. Such specifications can be captured in Boolean formulas. In a Boolean formula, any expression between brackets is called a clause; '+' stands for logical 'or' where $(a + b)$ means either $a$ or $b$ or both; '.' stands for 'and'; $\bar{a}$ stands for 'not $a$'; exclusive 'or' is designated by $(a.\bar{b} + \bar{a}.b)$; and the implication of $b$ by $a$ is formulated as $a.\bar{b} = 0$ or, alternatively, $\bar{a} + b = 1$. A restricted example may demonstrate the point. Suppose the test specification is as follows (assuming an item bank consisting of seven items, or seven types of items).

1. Of three items $x_1$, $x_2$, and $x_3$, one must be chosen.
2. If $x_5$ is chosen, neither $x_2$ nor $x_7$ may be chosen.
3. If $x_1$ is chosen, $x_7$ must also be chosen.
4. If items $x_2$ or $x_3$ are chosen, $x_6$ must also be chosen.
5. $x_4$ and $x_5$ are mutually exclusive.
6. $x_6$ and $x_7$ are mutually exclusive.

These conditions can be presented in 4 clauses:

$$C1 = (x_1.\bar{x}_2.\bar{x}_3 + \bar{x}_1.x_2.\bar{x}_3 + \bar{x}_1.\bar{x}_2.x_3) = 1 \tag{34}$$

$$C2 = x_5.(x_2 + x_7) = 0, \quad \text{or} \quad (\bar{x}_5 + \bar{x}_2.\bar{x}_7) = 1 \tag{35}$$

$$C3 = x_1.\bar{x}_7 = 0, \quad \text{or} \quad (\bar{x}_1 + x_7) = 1 \tag{36}$$

$$C4 = (x_2 + x_3).\bar{x}_6 = 0, \quad \text{or} \quad (\bar{x}_2.\bar{x}_3 + x_6) = 1 \tag{37}$$

To fulfill all demands, the following is necessary: C1.C2.C3.C4 = 1. Some (Boolean) algebraic manipulation yields the expression

$$C1.C2.C3.C4 = x_1.\bar{x}_2.\bar{x}_3.x_7 + \bar{x}_5 + \bar{x}_1.x_2.\bar{x}_3.\bar{x}_7.\bar{x}_5 + \bar{x}_1.\bar{x}_2.x_3.\bar{x}_7 = 1 \quad . \tag{38}$$

For this expression to be equal to 1 (Boolean), at least one of the terms of this sum must be equal to 1 (and of course, the expression must be satisfiable, i.e., internally consistent). $x_1.\bar{x}_2.\bar{x}_3.x_7.\bar{x}_5 = 1$ means that the item selection $x_1.x_7.x_4$ is a possibility ($x_4$ because of specification 2). $\bar{x}_1.x_2.\bar{x}_3.\bar{x}_7.\bar{x}_5 = 1$ means that the item selection $x_2.x_6.x_4$ is a possibility. $\bar{x}_1.\bar{x}_2.x_3.\bar{x}_7 = 1$ implies the item selection $x_3.x_6$. Since the choice between $x_4$ and $x_5$ is indifferent, there are two item selection possibilities, $x_3.x_6.x_4$ or $x_3.x_6.x_5$. Each of these selections can be evaluated with respect to their information value at one or more specified $\theta$ points. If necessary, the highest value selection is chosen. In adaptive testing, the selection corresponding best with a person's current $\hat{\theta}$ may be chosen. In this way, very detailed and complex test specifications can be made and added as (potential) variables to the binary programming problem.

## Discussion

Another possible application of KPS lies in the area of two-stage testing procedures (Lord, 1980, chap. 9). Assume the availability of a pool of calibrated items, a small subset of which is used as the routing test. Then, for Rasch-calibrated items,

compute all possible $\theta$ estimates that are possible with the (short) routing test. Divide the range of values in a number of (sequential) segments, the number being equal to the number of desired second-stage tests (generally few; see Lord, 1980). For each segment, specify for the $\theta$ points within it the second-stage test information such that the relative efficiency is as desired. Subsequently, for a fixed number of items use Equations 16–18, or more efficiently, use the minimum number of items from Equations 13–15.

For items calibrated by a three-parameter model, compute the lowest and highest possible $\theta$ estimate and divide the range in equal parts. For each segment, specify $\theta$ points, including both borderline points and proceed as above. If a person's $\theta$ estimate coincides with one of these borderline values, he or she is randomly routed to a more or less difficult second-stage test. Advantages of this procedure are that the second-stage levels are optimally adapted to the initial $\theta$ estimates and that there is no need to find cutting points on the routing test.

Another possible application is text-banking for reading comprehension tests or listening comprehension tests for native and foreign languages. These tests generally consist of subtests, each one accompanying a particular text (written or spoken). The information function for each subtest can be stored. After $\theta$ points are specified, the subtest information function value for these $\theta$ points can be determined, and subtests can be treated as items in the previous programming problem. For the three-parameter model, it might be more efficient to store polynomial approximations of the subtest information functions. All advantages that derive from item banks also hold for text banks (e.g., adaptive testing).

If, as Lord (1980, section 11.11) suggests, the effectiveness of a mastery test is often to be evaluated by test information, several issues have a straightforward solution. If masters and nonmasters are to be separated at $\theta = \theta_0$, optimal item selection is obviously a simple application of Dantzig's theorem. If a common upper bound $\theta_2$ and a common lower bound $\theta_1$ (common for a group of test content specialists) for nonmastery and mastery, respectively, are defined, optimal item selection is a two-dimensional KP (i.e., when $\theta_1$ and $\theta_2$ are far apart). Treating this KP as a minimization problem (as in Equations 13–15) will automatically give the minimum test length with specified standard errors of estimation for $\theta_2$ and $\theta_1$ (using the information function), given a particular pool of calibrated items. It is interesting to note that if $\theta_2$ and $\theta_1$ are relatively close together, persons with $\hat{\theta}$ between $\theta_2$ and $\theta_1$ belong to the group of most accurately measured persons. Still, these persons are disregarded. Intuitions concerning the cost of passing a nonmaster or failing a master could (at least partly) be mirrored in the specification of the length of the confidence intervals for $\theta_2$ and $\theta_1$ respectively, at least if they are not too close together (because if they are, majoring the higher information value for either $\theta_2$ or $\theta_1$ would imply majoring the other).

Finally, as a rather obvious application, Knapsack test design is very appropriate for designing parallel tests, not only in the sense of tests having identical test information functions but by specifying the constraints of the KP in sufficient detail.

Occasionally, it may be useful to have a relatively large number of tests of equal length and of slowly increasing difficulty, where no pair of tests has an item in common. An example is a classroom situation where it is desired to present each student with a test adapted to his or her (known) level of competence, or when it is desired to measure progress in a student over a period of time (with $\theta$ confidence intervals that are non-overlapping). For this type of test design, an optimization problem with some pleasant characteristics is available. (Applying Dantzig's theorem here for each test would not guarantee non-overlap of items.)

Suppose $N$ $\theta$ points are specified for the $N$ tests under consideration, that each test consists of $t$ items, and that there are $n$ items in the bank. The purpose is to find $N$ non-overlapping tests, each with maximum information for its specified $\theta$ point. Items and abilities can now be regarded as nodes in a bipartite graph $G(V,E)$, where $V$ consists of a subset $V_1$ of $n$ items, and $V_2$ is a subset of $N \times t$ ability specifications, with $t$ equal specifications for

each test. The set $E$ has as elements the edges connecting all elements of $V_1$ with those of $V_2$. A weight $a_{ij}$ is associated with each edge, where $a_{ij}$ is the information value of item $x_i$ at a particular $\theta_j$ ($j = 1, 2, \ldots, N \times t$).

This test design problem can then be formulated as a maximum weight matching in a bipartite graph, where a matching is a set of edges where no two edges have a node in common. If $y_{ij}$ is allowed to be associated with edges connecting items with θs, this problem may be formulated as: Maximize

$$\sum_{i,j} a_{ij} y_{ij} \tag{39}$$

subject to

$$\sum_{j=1}^{(N \times t)} y_{ij} = 1 \qquad (i = 1, 2, \ldots, n) \tag{40}$$

and

$$\sum_{i=1}^{n} y_{ij} = 1 \qquad (j = 1, 2, \ldots, N \times t) \quad . \tag{41}$$

Note that no integer constraints are added to Equations 39–41, with respect to $y_{ij}$. Surprisingly, this is not necessary, due to the special structure of this problem. (In the following it is assumed that both subsets are of equal size and the number of nodes is even; if not, add dummy nodes with zero-weight edges incident upon them.) If Equations 39–41 are written in matrix notation, taking into consideration the above assumption, the result is the constraint matrix **A** with element $A_{ij}$. An integer matrix **A** is called totally unimodular if every square, nonsingular submatrix **B** of **A** is unimodular. **B** is unimodular whenever det(**B**) = ±1 holds. Matrix **A** is in fact a node-edge incidence matrix of an undirected bipartite graph. In Papadimitriou and Steiglitz (1982, chap. 13), it is demonstrated that **A** in this context is always totally unimodular, and theorems and proofs of sufficiency are presented therein to show that this implies that solving Equations 39–41 with standard linear programming (LP) will automatically produce integer (binary) solutions. In Chapter 11 of the same text, a constructive proof is given by transforming this matching problem into another problem for which this LP result was already known.

Given the isomorphism between certain test design problems and certain discrete optimization problems (problems that have been very thoroughly studied), it should not be a surprise that algorithms developed for these optimization problems are also relevant for test design problems. Results presented by Theunissen (1985) were obtained by a general purpose integer programming problem, designed to obtain exact solutions. For Rasch-calibrated item banks, for KPs as in Equations 13–15, with the number of items variously 300 or 500 and the number of specified θ points ranging from 1 to 5, computer central processor (CPU) time requirements were from 41 to 130 seconds on a DEC-10 computer (problems 1A to 1E in Theunissen, 1985, Table 2). One of Verstralen's (Theunissen & Verstralen, 1986) approximation algorithms for Rasch-calibrated items, with the same set of problems, reached the optimum solution in all five cases, with CPU time ranging from 6 to 14 seconds on an Epson QX-10 personal computer. Boomsma's (1986) approximations of KPs as in Equations 13–15, for those cases where the exact or near-exact solution was obtained, typically took only approximately 2 seconds on a DEC-10 computer with problems of similar size.

These results suggest that making use of the special structure of test design problem formulated in this paper may lead to significant reduction in CPU time requirements. Both maximum matching and maximum weight matching can be done in polynomial time (at worst, time grows as $n^3$, where $n$ is the number of nodes). For general matching algorithms, very fast algorithms have been developed; in one case, for 500 nodes and 2,500 edges, .04 seconds on an Amdahl 470V/8 computer were necessary (see Table 3.6 in Syslo et al., 1983). An issue for future research is the relation between the performance of the exact algorithms and approximation algorithms under varying distributions of item parameters within an item bank.

## References

Boomsma, Y. (1986). *Item selection by mathematical programming*. Master's thesis, Twente University of Technology.

Dantzig, G. (1957). Discrete-variable extremum problems. *Operations Research, 5,* 266–277.

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

McArthur, D. L., & Choppin, B. H. (1984). Computerized diagnostic testing. *Journal of Educational Measurement, 21,* 391–397.

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial optimization: Algorithms and complexity*. Englewood Cliffs NJ: Prentice-Hall.

Rao, S. S. (1984). *Optimization: Theory and applications* (2nd ed.). New Delhi: Wiley Eastern.

Salkin, H. M. (1975). *Integer programming*. Reading MA: Addison-Wesley.

Syslo, M. J., Deo, N., & Kowalik, J. S. (1983). *Discrete optimization algorithms*. Englewood Cliffs NJ: Prentice-Hall.

Taha, H. A. (1975). *Integer programming*. New York: Academic Press.

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

Theunissen, T. J. J. M., & Verstralen, H. H. F. M. (1986). Algorithmen voor het samenstellen van toetsen [Algorithms for test design]. In W. J. van der Linden (Ed.), *Moderne methoden van toetsconstructie en -gebruik*. Lisse, the Netherlands: Swets & Zeitlinger.

Timminga, E. (1985). *Geautomatiseerd toetsontwerp: Itemselectie met behulp van binair programmeren* [Automated test design: Item selection by binary programming]. Master's thesis, Twente University of Technology.

## Author's Address

Send requests for reprints or further information to T. J. J. M. Theunissen, CITO, P.O. Box 1034, 6801 MG Arnhem, the Netherlands.