Journal of Educational Statistics
Summer 1990, Vol. 15, No. 2, pp. 129–145

The Construction of Parallel Tests From IRT-Based Item Banks

Ellen Boekkooi-Timminga
University of Twente
Enschede, The Netherlands

Key words: *item response theory, item banks, test construction, parallel tests, information functions, linear programming*

The construction of parallel tests from IRT-based item banks is discussed. Tests are considered to be parallel whenever their information functions are identical. Simultaneous and sequential parallel test construction methods based on the use of 0–1 programming are examined for the Rasch and 3-parameter logistic model. Sequential methods construct the tests one after another; simultaneous methods construct them all at the same time. A heuristic procedure is used for solving the 0–1 programming problems. Satisfactory results are obtained, both in terms of the CPU-time needed and differences between the information functions of the parallel tests selected.

In testing situations there is a demand for parallel test forms. Large testing programs have a special need for interchangeable tests. Few algorithms for designing such tests have been developed. One exception in classical test theory is the matched random subtests method by Gulliksen (1950), which recently was algorithmized by van der Linden and Boekkooi-Timminga (1988).

The topic of this paper is the construction of so-called weakly parallel tests from item banks based on item response theory (IRT). Tests are defined to be weakly parallel if their information functions are identical (Samejima, 1977). Tests are strongly parallel if they have the same test length and if they have exactly the same test characteristic function (Lord,

---

The Netherlands Organization for Scientific Research (NWO) is gratefully acknowledged for funding this project. This research was conducted while Ellen Boekkooi-Timminga was supported by a PSYCHON-grant from NWO (560-267-001) awarded to W. J. van der Linden. I would like to thank Tom A. B. Snijders, University of Groningen, The Netherlands, for his remarks on an earlier version of Method 3. Also, I would like to thank Jos J. Adema and Wim J. van der Linden, University of Twente, The Netherlands, for their assistance during the preparation of this paper.

1980). An exact definition of the concept of information is given by Birnbaum (1968, Chapter 17). Here it is assumed that maximum-likelihood estimation is used for subjects' abilities so that the test information function is the sum of the item information functions.

In this paper, the procedures for parallel construction are based on 0–1 linear programming (e.g., Rao, 1985; Salkin, 1975; Taha, 1975; Wagner, 1972). Previous research on modeling test construction problems as mathematical programming problems has been carried out by Adema and van der Linden (1989), Boekkooi-Timminga (1986, 1987), Theunissen (1985, 1986), and van der Linden and Boekkooi-Timminga (1989). All methods for test construction described in these publications expect the test constructor to specify target test information function values for the test(s) to be constructed at some prechosen ability levels.

It seems obvious to determine parallel tests by selecting the tests one after another (sequentially) from an item bank using the same specifications. However, this approach does not give satisfactory test information functions (Boekkooi-Timminga, 1986, 1987); while constructing a test, it is not possible to take into account the tests to be constructed later. In Boekkooi-Timminga (1986, 1987) it was shown that by constructing all the tests at the same time (simultaneously) parallel tests with fairly similar test information functions could be obtained. However, the major problem of all methods examined so far was the amount of CPU-time needed, which is a feature inherent to 0–1 programming. Simultaneous test construction problems, in particular, turned out to be very difficult because of the large increase in the number of decision variables in the model.

In this paper simultaneous and sequential parallel test construction methods are examined using different IRT models and different limits on the accuracy required. First, the methods for constructing parallel tests are described. Next, the heuristic procedure applied in this paper is considered. Finally, the results of some experiments comparing the methods are outlined.

Four Methods of Constructing Parallel Tests

It is argued that in order to make computerized test construction practical, at least two conditions must be fulfilled. First, it should be an easy task for the test constructor to specify a target test information function. A procedure that meets this requirement was described by van der Linden and Boekkooi-Timminga (1989). Using this procedure, the shape of the test information function at some well-chosen points on the ability scale needs to be identified. Thus, only the ratio of information at one point to another is specified. This is much easier than specifying the exact test information function. Van der Linden and Boekkooi-Timminga described in detail how to elicit these specifications from the test constructor.

Second, the amount of CPU-time needed to select the tests should be

small. The procedure proposed by Adema (1988), described in the next section, solves 0–1 linear programming problems in a reasonable amount of time. In this paper both the van der Linden and Boekkooi-Timminga (1989) approach and the Adema heuristic are applied. The 0–1 programming model formulation for each method is given below.

Notation

$i = 1, \ldots, I$	items in the item bank,
$t = 1, \ldots, T$	tests to be constructed,
$k = 1, \ldots, K$	ability levels to be considered,
x_i, x_{it}	decision variables indicating whether an item is (x_i or $x_{it} = 1$) or is not (x_i or $x_{it} = 0$) selected for the test,
r_k	relative amount of target test information at ability level θ_k,
$I_i(\theta_k)$	item information function value of item i at ability level θ_k,
$I_t(\theta_k)$	test information function value of test t at ability level θ_k,
N	number of items to be included in each test,
y	decision variable—multiplicative factor.

Sequential Test Construction

Two methods for sequentially selecting parallel tests are formulated. The first method uses a nonpartitioned item bank and the second a partitioned bank.

Method 1. Model (1)–(5), described below, is used for constructing the first test. By maximizing decision variable y in objective function (1), the total amount of test information obtained is maximized, subject to the constraints (2) that at least the specified proportions r_k are obtained. Actually, the shape of the test information function is enlarged by maximizing the multiplicative variable y. The obtained test information $I_t(\theta_k)$ is greater than or equal to $r_k y$ at all ability levels; $I_t(\theta_k) = r_k y$ at one ability level. The constraint in (3) guarantees that the test consists of N items. The ranges of the variables x_i and y are defined in (4) and (5).

$$\text{Maximize } y, \tag{1}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k) x_i - r_k y \geq 0 \qquad k = 1, \ldots, K \tag{2}$$

$$\sum_{i=1}^{I} x_i = N \tag{3}$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, I \tag{4}$$

$$y \geq 0. \tag{5}$$

For the other tests to be determined, the same model is used subsequently. However, the items included in previously constructed tests are excluded from further selection. This can be done by fixing their decision variables to 0 (and thus including some extra constraints) or by removing them from the item bank. The latter possibility is preferred, especially when many items have to be excluded, because CPU-time is gained.

The differences between the test information values of these sequentially constructed tests can be controlled. Because these differences can be positive as well as negative, the following two sets of constraints are added to the model:

$$\sum_{i=1}^{I} I_i(\theta_k)x_i - (1-p)I_t(\theta_k) \geq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, t^*, \tag{6}$$

$$\sum_{i=1}^{I} I_i(\theta_k)x_i - (1+p)I_t(\theta_k) \leq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, t^*, \tag{7}$$

where $t = 1, \ldots, t^*$ indicates the tests already constructed and p is the percentage by which the obtained test information values are allowed to differ from the values $I_t(\theta_k)$ of the tests already constructed.

Method 2. This method assumes that the item bank is partitioned into T subsets of the same size that are comparable in terms of their item information functions. The T subsets are $i = 1, \ldots, I/T; \ldots, i = ((t-1)I/T) + 1, \ldots, tI/T; \ldots; i = ((T-1)I/T) + 1, \ldots, I$. In the case of the Rasch model, the subsets are constructed by ordering the items on difficulty and subsequently dividing them over the subsets randomly. When 2- or 3-parameter logistic models are considered, the same strategy can be applied, ordering items on difficulty or discrimination parameter values. Then, the subsets may be less equal; however, the application of advanced cluster techniques will take too much time.

From each subset a test is selected; thus, in model (1)–(5), only the indices i have to be adapted. Because each test is selected from a different subset, one need not exclude items because of previous usage. The constraints in (6) and (7) can be included after a test has been constructed.

It is expected that by partitioning the item bank and subsequently selecting tests from these subsets, the amount of CPU-time needed will not be too large.

Simultaneous Test Construction

Two simultaneous test construction methods for parallel test construction are considered. Method 3 uses a nonpartitioned item bank, Method 4 a partitioned bank.

Method 3. The 0–1 programming model is formulated below in (8)–(15). Expressions (8), (9), (10), (14), and (15) are simultaneous versions of the objective function and constraints in model (1)–(5). By maximizing y the lower bounds to the test information function values will be close to one another. The constraints in (11) stipulate that no items be included in more than one test. The maximum difference allowed between the obtained test information values of the tests is constrained in expressions (12) and (13). This difference is given in percentages, p, of the obtained test information values of the other tests.

$$\text{Maximize } y, \tag{8}$$

subject to

$$\sum_{i=1}^{I} I_i(\theta_k)x_{it} - r_k y \geq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, T \tag{9}$$

$$\sum_{i=1}^{I} x_{it} = N \qquad t = 1, \ldots, T \tag{10}$$

$$\sum_{t=1}^{T} x_{it} \leq 1 \qquad i = 1, \ldots, I \tag{11}$$

$$\sum_{i=1}^{I} I_i(\theta_k)x_{it'} - (1-p)\sum_{i=1}^{I} I_i(\theta_k)x_{it} \geq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, T-1 \quad t' = t+1, \ldots, T \tag{12}$$

$$\sum_{i=1}^{I} I_i(\theta_k)x_{it'} - (1+p)\sum_{i=1}^{I} I_i(\theta_k)x_{it} \leq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, T-1 \quad t' = t+1, \ldots, T \tag{13}$$

$$x_{it} \in \{0, 1\} \qquad i = 1, \ldots, I \quad t = 1, \ldots, T \tag{14}$$

$$y \geq 0. \tag{15}$$

Method 4. As in Method 2, it is assumed that the item bank is partitioned into T comparable subsets. From each subset a test is selected simultaneously. The decision variables x_i for $i = 1, \ldots, I/T; \ldots, i = ((t-1)I/T) + 1, \ldots, tI/T; \ldots; i = ((T-1)I/T) + 1, \ldots, I$ denote the items to be included in test $1, 2, \ldots, T$, respectively. The model for Method 3 can easily be adapted for this case:

$$\text{maximize } y, \tag{16}$$

subject to

$$\sum_{i=((t-1)I/T)+1}^{tI/T} I_i(\theta_k)x_i - r_k y \geq 0 \qquad k = 1, \ldots, K \quad t = 1, \ldots, T \tag{17}$$

$$\sum_{i=((t-1)I/T)+1}^{tI/T} x_i = N \qquad t = 1, \ldots, T \tag{18}$$

$$\sum_{i=((t'-1)I/T)+1}^{t'I/T} I_i(\theta_k)x_i - (1-p) \sum_{i=((t-1)I/T)+1}^{tI/T} I_i(\theta_k)x_i \geq 0 \quad k = 1, \ldots, K \quad t = 1, \ldots, T-1 \quad t' = t+1, \ldots, T \tag{19}$$

$$\sum_{i=((t'-1)I/T)+1}^{t'I/T} I_i(\theta_k)x_i - (1+p) \sum_{i=((t-1)I/T)+1}^{tI/T} I_i(\theta_k)x_i \leq 0 \quad k = 1, \ldots, K \quad t = 1, \ldots, T-1 \quad t' = t+1, \ldots, T \tag{20}$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, I \tag{21}$$

$$y \geq 0. \tag{22}$$

Note that the constraints in (11) are left out in this model.

Computational Procedure

The parallel test construction problems above are formulated as 0–1 linear programming problems. A 0–1 linear programming problem is a linear programming problem in which the decision variables are restricted to 0–1 values.

Solving a 0–1 linear programming problem optimally involves the following two steps (e.g., Taha, 1975; Williams, 1978): First, compute the *relaxed 0–1 linear programming problem.* This problem is obtained by dropping the 0–1 constraints on the decision variables; thus, $x_i \in [0, 1]$. Doing so, a regular linear programming model is obtained that can be solved quickly, for instance, using the well-known simplex algorithm. The objective function value, y_{LP}, of the relaxed problem solution is an upper bound for the objective function value, y_{0-1}, of the original 0–1 linear programming problem. The number of fractional decision variable values obtained never exceeds the number of constraints in the model (Dantzig, 1957).

Second, the optimal 0–1 solution is determined. Here, a *branch-and-bound method* is used because these methods have proved to be most successful for integer programming problems. Generally, branch-and-

bound methods perform a tree search starting from the relaxed solution. During this search several linear programming problems are solved. These problems are obtained by fixing (bounding) decision variables with fractional values (branching variables) to 0 or 1 in the original relaxed 0–1 linear programming problem. When a 0–1 solution is obtained or when a solution with an objective function value worse than the best 0–1 solution so far is found, the present node is fathomed because $y_{0\text{-}1}$ cannot be improved further. Consequently, the search tree is backtracked and the last bounded decision variable is constrained in the other direction. This process of branching, bounding, and backtracking is continued until all nodes have been fathomed. The 0–1 solution with the largest $y_{0\text{-}1}$ is the optimal solution. A clear description of this procedure is given, for instance, by Williams (1978, pp. 146–152).

In comparison to linear programming, 0–1 linear programming is very complex because of the large number of linear programming problems that must be solved. Increasing the number of constraints generally results in an increase of CPU-time for obtaining linear programming solutions and a decrease for obtaining 0–1 linear programming solutions (Murtagh, 1981; Williams, 1978). This decrease is explained by the fact that nodes can be fathomed more quickly.

It is commonly conjectured that for 0–1 programming problems, no fast algorithms exist (Lenstra & Rinnooy Kan, 1979). Much research has been carried out in this area to develop approximations. A comprehensive review of this research is given in O'hEigeartaigh, Lenstra, and Rinnooy Kan (1985). However, many of the heuristic procedures aim at special types of problems and are not applicable to test construction.

One heuristic approach is to round the fractional decision variables in the relaxed solution. However, no satisfactory results may be obtained because rounding the fractional decision variable values might result in violating the constraints.

Recently, a quick heuristic procedure, especially suitable for test construction problems, was proposed by Adema (1988). Adema adapted the above described branch-and-bound procedure in two ways: First, after the relaxed solution is obtained, the algorithm fixes a number of *decision variables* to 0 or 1 using their reduced costs as a criterion. The reduced cost, d_i, of a decision variable indicates the amount by which the objective function value, y, decreases per unit increase in this variable, provided the decision variable values of the other variables are not changed (Murtagh, 1981; Williams, 1978). Decision variables with fractional values have reduced costs of zero. All items with a large expected increase (small d_i) or decrease (large d_i) of y are set to 0 and 1, respectively. Two rules are used for fixing the decision variables:

$$\text{If } y_{LP} - h_1 y_{LP} < d_i, \text{ then } x_i = 0, \tag{23}$$

and

$$\text{if } y_{LP} - h_1 y_{LP} < -d_i, \text{ then } x_i = 1, \tag{24}$$

where h_1 ($h_1 < 1$) is a help variable whose value is chosen to be close to 1. Fixing these variables reduces the size of the search tree.

Second, using the fact that y_{LP} is an upper bound for $y_{0\text{–}1}$, Adema *initialized* $y+$ (i.e., the true lower bound of the optimal 0–1 objective function value) by $y_+ = h_2 y_{LP}$ ($0 \ll h_2 < 1$; $h_1 \geq h_2$) instead of $y_+ = -\infty$. Then, the first 0–1 solution having a $y_{0\text{–}1}$ between $h_2 y_{LP}$ and y_{LP} is accepted. This option should not be used when y_{LP} is equal to zero, as no solution can be obtained. It is possible that no solution is found if h_1 or h_2 is too large; then, these values should be adapted.

Experiments

Several experiments were carried out to compare the four methods for parallel test construction. The computer program LANDO (Centre for Mathematics and Computer Science) was used to solve the problems; the program is based on the branch-and-bound algorithm developed by Land and Doig (1960). It was adapted according to the previously described Adema heuristic, accepting 0–1 solutions that did not deviate more than 5% ($h_2 = .95$) from the relaxed objective function value (this value can be chosen to be smaller if the item bank is larger and/or when the Rasch model is considered). Furthermore, a value of $h_1 = .999$ for the help variable in (23) and (24) was used. The program was implemented on a DEC-2060 mainframe computer.

Two item banks, each consisting of 100 items drawn from the following distributions, were considered: (a) the Rasch model, with difficulty parameters $b \sim N(0, 1)$; and (b) the 3-parameter model, $b \sim N(0, 1)$, discrimination parameters $a \sim U(0.5, 1.5)$, and guessing parameters $c \sim U(0, 0.25)$. In all experiments two parallel tests were selected. As diagnostic tests were to be constructed, the relative target information values were equal: $r_k = 1, 1, 1$ at the ability levels $\theta = -1, 0, 1$. Furthermore, the desired test length was $N = 10$ in all cases.

In this section, first, the influence of varying $p\%$ in (12) and (13) on CPU-time is examined for Method 3. Then, the results from the various parallel test construction methods are compared.

The solutions for the relaxed and 0–1 problems are discussed below. The best way to consider the relaxed solutions is as upper bounds for the optimal solutions.

The Influence of Varying p *for Method 3*

The results from the experiments in which the maximum accepted difference in test information values between the tests were varied are sum-

marized in Table 1. The values $p = 0.5, 1.0, 2.0, 5.0$ and $\infty\%$ (where $\infty\%$ means leaving out (12) and (13)) were considered. In the table the obtained objective function values y_{LP} for the relaxed problem and $y_{0\text{-}1}$ for the 0–1 problem, the actual test information function values, and the CPU-times needed are given. Furthermore, the relative differences of y_{LP} and $y_{0\text{-}1}$ from y_{LP} as well as the differences in test information between Tests 1 and 2 in percentages from the values of Test 1 are summarized. From the CPU-times for the *relaxed problem* it can be seen that there is not much variation. No clear trend toward lower CPU-times can be observed as the allowed differences in test information increase; only for $p = \infty\%$ do CPU-times turn out to be significantly lower, which is obvious because less constraints have to be taken into account. For the Rasch model, CPU-times tend to be higher, because Rasch item information functions are looking more alike, making it more difficult to choose between the individual items. In the case of the *0–1 solution* for the 3-parameter model, both large CPU-times and large variations in CPU-times are noted. For the Rasch model the CPU-times and their variances are much smaller. If $p = \infty\%$, the CPU-times for the 0–1 problems are remarkably lower.

It is noted that the differences in test information values for the Rasch model are, for the most part, much smaller than the maximum acceptable difference. For the 3-parameter model the differences are more in accordance with the maximum allowed percentage. If $p = \infty\%$, and thus only the lower bounds for the test information functions are considered, the differences are extremely small for the Rasch model; in the case of the 3-parameter model they are larger. This is to be expected because the item information functions vary more for the 3-parameter model.

Results for the Methods

In Tables 2 and 3 the results of the experiments for the four methods are summarized for the Rasch and 3-parameter model, respectively. For Methods 1 and 2 the second test was determined in two ways: without ($t = 2a$) and with ($t = 2b$) inclusion of constraints (6) and (7). For Methods 2 and 4, the item banks were partitioned into two subsets. This was done by ordering the items in the bank either on item difficulty (Ob) or on item discrimination (Oa), and subsequently dividing the items randomly into the two different subsets. For all methods, excluding the construction of Tests $2a$ for Methods 1 and 2, the maximum allowed difference between the actual test information values was $p = 1\%$.

From the CPU-times needed to solve the relaxed problems (LP) in Tables 2 and 3, it can be seen that approximately equal times are required for the Rasch and 3-parameter model. However, Method 3 required a larger amount of time for the Rasch model (see also Table 1). Also, for the time needed to determine the 0–1 solutions, very few differences in

TABLE 1
Varying the maximally accepted difference between test information values for Method 3, Model 2[a]

				Test information			CPU-time[b] in seconds	
t	%	y_{LP}	y_{0-1}	$I(-1)$	$I(0)$	$I(1)$	*LP*	0–1
				Rasch model				
1	0.5	1.962	1.931 (1.6%)	1.978	2.473	1.939	43.0	82.9
2				1.985 (0.4%)	2.471 (0.1%)	1.931 (0.4%)		
1	1.0	1.962	1.947 (0.8%)	1.961	2.480	1.960	42.6	79.9
2				1.947 (0.7%)	2.470 (0.4%)	1.967 (0.4%)		
1	2.0	1.962	1.950 (0.6%)	1.950	2.456	1.955	32.8	52.3
2				1.956 (0.3%)	2.488 (1.3%)	1.969 (0.7%)		
1	5.0	1.962	1.951 (0.6%)	1.966	2.474	1.951	37.5	94.7
2				1.960 (0.3%)	2.474 (0.0%)	1.958 (0.4%)		
1	∞	1.962	1.959 (0.2%)	1.959	2.484	1.964	16.8	32.7
2				1.959 (0.0%)	2.485 (0.0%)	1.964 (0.0%)		

				3-parameter model				
1	0.5	2.433	2.333 (4.1%)	2.420	2.996	2.333	22.1	89.3
2				2.418 (0.1%)	3.007 (0.4%)	2.339 (0.3%)		
1	1.0	2.433	2.332 (4.2%)	2.408	2.999	2.332	17.9	146.4
2				2.430 (0.9%)	3.004 (0.2%)	2.340 (0.3%)		
1	2.0	2.433	2.340 (3.8%)	2.340	3.123	2.493	21.3	372.6
2				2.376 (1.5%)	3.103 (0.6%)	2.491 (0.1%)		
1	5.0	2.433	2.359 (3.0%)	2.403	2.920	2.359	18.6	59.1
2				2.411 (0.3%)	3.032 (3.5%)	2.366 (0.3%)		
1	∞	2.433	2.328 (4.3%)	2.453	2.959	2.349	14.1	29.9
2				2.328 (5.1%)	3.048 (3.0%)	2.420 (3.0%)		

[a] $h_1 = .999$; $h_2 = .95$.

[b] CPU-time includes reading the input and writing the output file.

TABLE 2
Results for Methods 1–4 for the Rasch model[a]

Method	t	y_{LP}	y_{0-1}	Test information			CPU-time in seconds	
				$I(-1)$	$I(0)$	$I(1)$	*LP*	0–1
1	1	1.964	1.960 (0.2%)	1.960	2.490	1.967	2.5	3.4
	2*a*	1.961	1.958 (0.2%)	1.958 (0.1%)	2.479 (0.4%)	1.963 (0.2%)	2.4	3.4
	2*b*	1.961	1.951 (0.5%)	1.951 (0.5%)	2.476 (0.6%)	1.967 (0.0%)	2.5	4.0
2 *Ob*[b]	1	1.962	1.954 (0.4%)	1.964	2.477	1.954	1.6	2.1
	2*a*	1.962	1.925 (1.9%)	1.996 (1.6%)	2.479 (0.1%)	1.925 (1.5%)	1.6	2.2
	2*b*	1.962	1.960 (0.1%)	1.960 (0.2%)	2.481 (0.2%)	1.961 (0.4%)	1.4	2.3
3	1	1.962	1.947 (0.8%)	1.961	2.480	1.960	42.6	79.9
	2			1.947 (0.7%)	2.470 (0.4%)	1.967 (0.4%)		
4 *Ob*	1	1.962	1.942 (1.0%)	1.942	2.473	1.975	10.4	18.4
	2			1.958 (0.8%)	2.468 (0.2%)	1.956 (1.0%)		

[a] $h_1 = .999$; $h_2 = .95$.
[b] *Ob*: item bank ordered on item difficulty.

TABLE 3

Results for Methods 1–4 for the 3-parameter model[a]

Method	t	y_{LP}	y_{0-1}	Test information $I(-1)$	$I(0)$	$I(1)$	CPU-time in seconds *LP*	0–1
1	1	2.681	2.615 (2.5%)	2.746	3.458	2.615	2.2	2.4
	2*a*[b]	2.230	2.092 (6.2%)	2.092 (23.8%)	2.631 (23.9%)	2.211 (15.5%)	2.4	3.9
	2*b*	not feasible						
2 *Ob*[c]	1	2.286	2.279 (0.3%)	2.279	3.085	2.294	1.2	1.6
	2*a*	2.530	2.520 (0.4%)	2.520 (10.6%)	3.194 (3.5%)	2.546 (8.6%)	1.3	1.4
	2*b*	2.302	2.275 (1.2%)	2.275 (0.2%)	3.089 (0.1%)	2.305 (0.5%)	1.3	11.6
Oa[d]	1	2.325	2.234 (3.9%)	2.234	3.279	2.428	1.4	1.6
	2*a*	2.493	2.456 (1.5%)	2.527 (13.1%)	3.093 (5.7%)	2.456 (1.2%)	1.3	1.4
	2*b*	2.256	no 0–1 solution				1.3	1.8
3	1	2.433	2.332 (4.2%)	2.408	2.999	2.332	17.9	146.4
	2			2.430 (0.9%)	3.004 (0.2%)	2.340 (0.3%)		
4 *Ob*	1	2.286	2.279 (0.3%)	2.279	3.085	2.294	8.8	30.8
	2			2.298 (0.8%)	3.070 (0.5%)	2.279 (0.7%)		
Oa	1[b]	2.325	2.208 (5.0%)	2.208	3.019	2.442	8.1	106.4
	2[b]			2.228 (0.9%)	3.023 (0.1%)	2.457 (0.6%)		

[a] $h_1 = .999$; $h_2 = .95$.

[b] $h_1 = .99$; $h_2 = .93$.

[c] *Ob*: item bank ordered on item difficulty values.

[d] *Oa*: item bank ordered on item discrimination values.

CPU-time between the Rasch and 3-parameter model could be noted for Methods 1 and 2 (Tests 2*a*). However, for Methods 3 and 4 (see also Table 1) the Rasch model was faster. In general, it was seen that Methods 3 and 4 were much slower than 1 and 2. Furthermore, Methods 2 and 4 turned out to be quicker than their nonpartitioned counterparts. For the 3-parameter model, Method 2 had difficulties in determining Tests 2*b*: When the item bank was ordered by difficulty, it took a considerable amount of time to select the test, and when the item bank was ordered on discrimination, no 0–1 solution was found.

From the objective function values y_{LP} of the relaxed solutions for the Rasch model, it can be seen that they were all equal to 1.962 except for Method 1. For Methods 3 and 4, this implies that one of the tests had a lower bound equal to 1.962, whereas the other lower bound was slightly larger than or equal to 1.962. Method 1 showed a large discrepancy between the y_{LP} values for both tests.

Next, the y_{LP} values for the 3-parameter model were compared. By summing the y_{LP} values of Tests 1 and 2*b* for Methods 1 and 2, and by taking two times the y_{LP} values for Methods 3 and 4, these values could be compared. Thus, the values 4.588, 4.581, 4.866, 4.572, and 4.650 were obtained for Methods 2(*Ob*), 2(*Oa*), 3, 4(*Ob*), and 4(*Oa*), respectively. The largest objective function value was found for Method 3. The results for Methods 2 and 4 were slightly worse.

Comparing the results for the Rasch and the 3-parameter model, it is observed that the differences between y_{LP} and y_{0-1} values were remarkably smaller for the Rasch model (less than 1%, except for one case). The differences between the test information function values of Tests 1 and 2 were also much smaller for the Rasch model, even for Tests 2*a* (less than 1%, except for one test). For the 3-parameter model these differences were much larger; for Method 1 percentages of about 25% were found (Tests 2*a*). In two cases small changes in help variables h_1 and h_2 (used in (23) and (24)) were required for the 3-parameter model, because no solution could be obtained otherwise (see Table 3). For the 3-parameter model no solution was found for Method 1, Tests 2*b*. Method 1 selected tests that were far from parallel, Methods 3 and 4 gave very parallel tests, and Method 2 (Tests 2*b*; item bank ordered on difficulty) performed slightly worse than 3 and 4. For the Rasch model approximately the same results were obtained for all methods.

As for the items actually selected, it was seen that they had difficulty values close to 0 for the Rasch model. For the 3-parameter model there was a trend toward selecting items with discrimination values larger than 1, difficulty parameters varying between -1 and 1, and guessing parameters smaller than 0.12.

For Methods 1 and 3, the item banks were also ordered on difficulty and discrimination, respectively. The results, which are not included in the tables, show no improvement over the results for nonordered item banks.

Generalizability of the Results

In the above examples, only two item banks and one required type of test were considered. The main point is that less CPU-time is required when fewer items with desired properties are available. For instance, the CPU-times would have been lower if item banks with $b \sim U[-3, 3]$ were used, or if target values $r_k = 4, 1, 1$ for $\theta_k = -2, 0, 2$ were specified.

Discussion

In this paper four parallel test construction methods have been discussed. The tests were selected from item banks on the basis of their information functions using 0–1 linear programming. Methods 1 and 2 and Methods 3 and 4 select the tests sequentially and simultaneously, respectively. Methods 2 and 4 assume that the item banks are partitioned into homogeneous subsets.

It is concluded that when the Rasch model fits the items, sequential test construction methods are to be preferred. Simultaneous and sequential methods are of equal accuracy; however, sequential methods require far less CPU-time. Care should be taken when Method 1 is used; it may provide good results as long as there are enough items at hand. When there are no restrictions to the maximally allowed difference between the information function values of the tests, simultaneous construction also gives accurate results in small amounts of time. For the 3-parameter model the use of Method 1 is absolutely inappropriate. For this model, the recommendation is to use Method 2 with (6) and (7) or Method 4.

Only the psychometric aspects of the tests have been considered in this paper. It is, however, possible to add all kinds of practical constraints, for instance, to control subject matter aspects. For a report on some of the possibilities, the reader is referred to van der Linden and Boekkooi-Timminga (1989). Although these authors considered the construction of one test at a time, most constraints can easily be generalized to the simultaneous test construction problems discussed here.

References

Adema, J. J. (1988). *A note on solving large-scale zero-one programming problems* (Research Report 88-4). Enschede, The Netherlands: University of Twente.

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

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

Boekkooi-Timminga, E. (1986). *Algorithms for the construction of parallel tests by zero-one programming* (Research Report 86-7). Enschede, The Netherlands: University of Twente.

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

Centre for Mathematics and Computer Science. *LANDO*. The Netherlands: Amsterdam.

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

Gulliksen, H. (1950). *Theory of mental tests*. New York: John Wiley & Sons.

Land, A. H., & Doig, A. (1960). An automatic method of solving discrete programming problems. *Econometrica, 28,* 497–520.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Computational complexity of discrete optimization problems. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.), *Discrete optimization I.* New York: North-Holland Publishing Company.

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

Murtagh, B. A. (1981). *Advanced linear programming: Computation and practice*. New York: McGraw-Hill.

O'hEigeartaigh, M., Lenstra, J. K., & Rinnooy Kan, A. H. G. (Eds.). (1985). *Combinatorial optimization: Annotated bibliographies*. New York: Wiley.

Rao, S. S. (1985). *Optimization: Theory and applications* (2nd ed.). New Delhi: Wiley Eastern Ltd.

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

Samejima, F. (1977). Weakly parallel tests in latent trait theory with some criticisms of classical test theory. *Psychometrika, 42,* 193–198.

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. (1986). Optimization algorithms in test design. *Applied Psychological Measurement, 10,* 381–390.

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

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

Wagner, H. M. (1972). *Principles of operations research: With applications to managerial decisions*. London: Prentice-Hall International.

Williams, H. P. (1978). *Model building in mathematical programming*. New York: Wiley.

Author

ELLEN BOEKKOOI-TIMMINGA is Research Associate, Department of Education, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands. She specializes in educational measurement.