DOCUMENT RESUME

ED 314 502 TM 014 434

AUTHOR Adema, Jos J.
TITLE Implementations of the Branch-and-Bound Method for Test Construction Problems. Project Psychometric Aspects of Item Banking No. 46. Research Report 89-6.
INSTITUTION Twente Univ., Enschede (Netherlands). Dept. of Education.
PUB DATE Nov 89
NOTE 45p.
AVAILABLE FROM Bibliotheek, Department of Education, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.
PUB TYPE Reports - Evaluative/Feasibility (142)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Achievement Tests; *Computer Assisted Testing; Computer Software; *Item Banks; Item Response Theory; *Mathematical Models; Search Strategies; *Test Construction; Test Items
IDENTIFIERS *Branch and Bound Method; *Maximin Model

ABSTRACT

Item banks, large sets of test items, can be used for the construction of achievement tests. Mathematical programming models have been proposed for the selection of items from an item bank for a test. These models make automated test construction possible. However, to find an optimal or even an approximate optimal solution to a test construction model can be time consuming. This paper shows how test construction models, and in particular the Maximin Model, are solvable by the program MPSX/370 V2. This program offers the user several implementations of the branch-and-bound method, which can be used for solving test construction models. Several implementations are compared. The results show that test construction models are solvable in a practical amount of time if the user applies the options offered by the program in an intelligent way. An appendix describes the Extended Control Language (ECL) computer program. Five data tables and two figures are included. (Author/SLD)

Implementations of the Branch-and-Bound method for Test Construction Problems

Research Report 89-6

Jos J. Adema

department of EDUCATION

Division of Educational Measurement and Data Analysis

University of Twente

Project Psychometric Aspects of Item Banking No.46

Colofon:
Typing: L.A.M. Bosch-Padberg
Cover design: Audiovisuele Sectie TOLAB Toegepaste Onderwijskunde
Printed by: Centrale Reproductie-afdeling

Implementations of the Branch-and-Bound method for Test Construction Problems

Jos J. Adema

Implementations of the branch-and-bound method for test construction problems , Jos J. Adema - Enschede : University of Twente, Department of Education, November, 1989. - 39 pages

Abstract

Item banks, large sets of test items, can be used for the construction of achievement tests. Mathematical programming models have been proposed for the selection of items from an item bank for a test. These models make automated test construction possible. However, to find an optimal or even an approximate optimal solution to a test construction model can be time consuming. Here, it is shown how test construction models and in particular the Maximin Model are solvable by the program MPSX/370 V2. This program offers the user several implementations of the branch-and-bound method, which can be used for solving test construction models. Several implementations are compared. The results show that test construction models are solvable in a practical amount of time if the user applies the options offered by the program in an intelligent way.

Keywords: Test Construction; Item Banking; Branch-And-Bound.

Implementations of the Branch-and-Bound method for Test Construction Problems

In this paper it is assumed that an item bank is available. The items of the bank are supposed to be calibrated under an item response model and some characteristics of the items such as format and content are supposed to be known. One of the applications of such an item bank is the selection of items for an in some sense optimal test, for instance, optimal with respect to its information function.

An example of an item response model for binary scored responses is the 3-parameter model (Birnbaum, 1968). In this model the probability, $P_i(\theta)$, that an examinee with ability θ answers item i correctly is given by the formula

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

where a_i, b_i and c_i are the discrimination, difficulty, and guessing parameter of item i. The information function of item i is represented by

$$I_i(\theta) = \frac{a_i^2(1-c_i)}{(c_i+\exp(a_i(\theta-b_i)))(1+\exp(-a_i(\theta-b_i)))^2}.$$

Another popular item response model is the Rasch model (Rasch, 1960). In the Rasch model there is no guessing parameter ($c_i = 0$) and the item discrimination parameters are

equal. For a test with n items the test information function is found by addition of the item information functions:

$$I(\theta) = \sum_{i=1}^{n} I_i(\theta).$$

The test information for an unbiased estimator of ability is the reciprocal of the (asymptotic) sampling variance of the estimator. Therefore, the higher the information function value, the better the test measures at a given ability level. This feature will be used in the construction of tests. A more detailed description of item response models and information functions is found in the psychometric literature (e.g. Lord, 1980; Hambleton & Swaminathan, 1985).

The manual construction of tests from an item bank is almost practically impossible, because the number of possible tests is very large. Therefore, linear programming models (e.g. Theunissen, 1985; van der Linden & Boekkooi-Timminga, 1989; Adema & van der Linden, 1989) are proposed which provide the test constructor with a computerized method for designing optimal tests.

Linear programming (LP) is an optimization method applicable for the solution of problems in which the objective function and the constraints appear as linear functions of the decision variables (Rao, 1985). In the LP models for test construction all or almost all decision variables are of the 0-1 type. It is well known that in general models with integer variables are hard to solve. This

paper is addressed to the problem of solving test construction models, in particular the Maximin Model as proposed by van der Linden and Boekkooi-Timminga (1989).

In the following a short outline of the Maximin Model is given. Define the decision variables x_i as:

$$x_i = \begin{cases} 0 & \text{item } i \text{ not in the test} \\ & \qquad i = 1, 2, \ldots, I, \\ 1 & \text{item } i \text{ in the test,} \end{cases}$$

where I is the number of items in the item bank. The information function of the test to be constructed is only considered at the ability levels θ_k, k = 1, ..., K. The test constructor can specify the required precision by choosing the number and spacing of the ability levels θ_k. Let $I_i(\theta_k)$, i = 1, ..., I be the information of item i at ability level θ_k. The relative amount of information required at ability level θ_k is specified by r_k. The vector (r_k) constitutes a target for the test information function. Let y be a decision variable such that $(r_1 y, \ldots, r_K y)$ is a series of lower bounds to the information function of the test to be constructed. The idea of the Maximin Model is to maximize y, which implies that the lower bounds are maximized (maximin). If n is the number of items to be selected for the test, then the Maximin Model can be written as follows:

(1) Maximize y,

subject to

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

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

$$(4) \quad Ax = b,$$

$$(5) \quad x_i \in \{0,1\}, \qquad i = 1, 2, \ldots, I,$$

where x is $(x_1, x_2, \ldots, x_I)^T$, A is a mxI matrix, and b is a mx1 vector. Constraints (2) considers the lower bounds to the test information function. Constraint (3) assures that the number of items in the test is equal to n. The constraints in (4) are added as a general provision to deal with practical constraints, for instance, on test composition, administration time, and the like; for examples, see Adema and van der Linden (1989).

The purpose of this paper is to show how the Maximin Model can be solved by the computer program MPSX/370 V2. MPSX/370 V2 is an IBM licensed program for the handling of linear and mixed integer linear programming problems (LP problems with integer and continuous variables). The program offers the user a number of options for controlling the optimization process. The user is also provided with algoritmic tools which enable the experienced user to build his/her own heuristics and algorithms. A popular method for

solving mixed integer LP problems is the branch-and-bound (BAB) method. Several implementations of the BAB method are available in MPSX/370 V2, which gives the user the opportunity to compare these implementations for test construction models. Several studies describing comparisons of implementations (strategies) in a more general perspective can be found in the mathematical programming literature (e.g. Breu & Burdet, 1974; Benichou et al., 1971; Benichou, Gauthier, Hentges & Ribiere, 1977; Crowder, Johnson & Padberg, 1983; Ibaraki, 1987). The importance of studies like these is the large amount of CPU-time that can be saved by an effective implementation of the BAB method. In our case, fast solution methods are needed, for instance, to enable the test constructor to construct a test interactively.

In the next section a general outline of the branch-and-bound method is given. More details about practical implementations of the BAB method are given further on in the paper.

Branch-and-Bound for 0-1 Linear Programming

In this section the branch-and-bound method as described by Papadimitriou and Steiglitz (1982, pp.433-438) is briefly reviewed for the case that the integer variables are of the 0-1 type. All the ideas to be presented carry over unchanged to the mixed 0-1 linear programming problem. Generally, a 0-1 linear programming (0-1 LP) problem can be written as:

Problem 0: Max. $z = c^T x = c(x)$,

subject to

$Ax \leq b$,

$x_i \in (0,1)$, $i = 1, \ldots, I$,

where $c^T = (c_1, \ldots, C_I)$ is the vector of cost coefficients. The branch-and-bound method starts with the computation of the optimal solution x^0 to the LP relaxation, i.e., Problem 0 with $0 \leq x_i \leq 1$, $i = 1, \ldots, I$ instead of $x_i \in (0,1)$. This computation is done by the simplex method, a well known method for solving LP problems. The objective function value $z_0 = c(x^0)$ is an upper bound on the optimal objective function value $z_* = c(x^*)$, where x^* is the optimal solution to the 0-1 LP problem. If x^0 is a 0-1 solution, then $x^* = x^0$ and the 0-1 LP problem is solved. If x^0 is not a 0-1 solution, then two subproblems are created. Suppose that, for instance, decision variable x_j in x^0 is not equal to 0 or 1. Then the two subproblems are as follows:

Problem 1: Max. $z = c^T x = c(x)$,

subject to

$Ax \leq b$,

$x_i \in (0,1)$, $i = 1, \ldots, I$,

$x_j = 0$,

Problem 2: Max. $z = c^T x = c(x)$,

subject to

$Ax \leq b$,

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

$x_j = 1$.

The variable x_j is called the branching variable.

Both subproblems are solved by the simplex method. Then, one of the subproblems, say Problem 1, is chosen for branching. Several choice criteria are given in one of the forthcoming sections. Again a fractional decision variable, say x_k, in x^1 is chosen and the Problems 3 and 4 are created by adding the constraint $x_k = 0$ and $x_k = 1$ to Problem 1. This process can be continued such that a tree like in Figure 1 is created.

Insert Figure 1 about here

The root of the tree is formed by the LP relaxation and the children of a node are formed by setting a fractional variable at its lower or upper bound (0 or 1).

The branching process cannot continue indefinitely, because the number of variables is bounded. This implies that an optimal solution to the original 0-1 LP problem will always be found. Branching from a node is not needed in the following cases:

1) The LP solution corresponding to the node is a 0-1 solution. Branching from this node is not needed, because adding constraints will give less optimal solutions. If the objective function value of this 0-1 solution is better than the objective function value (z_m) of the best 0-1 solution obtained so far (incumbent), then the solution becomes the incumbent and z_m is now the objective function value of the new incumbent. If no 0-1 solution is known when the standard BAB method is started then the objective function value of the incumbent is initialized by $z_m = -\infty$.

2) There is no feasible solution for the LP problem corresponding to the node.

3) The objective function value, say $z_k = c(x^k)$ of the node is smaller than or equal to z_m. This implies that a solution x that would be obtained as a descendant of x^k would have objective function value c(x):

$$c(x) \leq z_k \leq z_m.$$

If there is no leaf left to branch on, the method stops and the optimal solution to the 0-1 LP problem is the incumbent if one exists.

The next two sections address a number of implementations of the BAB method.

Limiting the search

The BAB method as given in the preceding section can be implemented in several ways. The following three ways of limiting the search are considered in this paper (see Adema, Boekkooi-Timminga & van der Linden, submitted):

LIM1: After the LP relaxation (Problem 0) is solved, variables are fixed using the reduced costs (see, e.g., Murtagh, 1981, p.25). Suppose d_i is the reduced cost corresponding to item i. The next two rules are used for fixing variables ($H_1<1$ is prespecified):

- Fix x_i to 0, if in the relaxed solution $x_i = 0$ and $z_0 - H_1 z_0 < d_i$;
- Fix x_i to 1, if in the relaxed solution $x_i = 1$ and $z_0 - H_1 z_0 < -d_i$.

LIM2: Initialize z_m by $z_m = H_2 z_0$ for some prespecified $H_2<1$ instead of $z_m = -\infty$. This will decrease the number of branchings through the search tree.

LIM3: Stop the search after the first 0-1 solution is found. This strategy is most useful in combination with LIM2, where H_2 is close to 1, because this guarantees that the first 0-1 solution found is a good solution. It is senseless to search for an exact solution, because the information function values in the test construction models are stimates and not

known exactly.

The first two strategies speed up the BAB method considerably if H_1 and H_2 are close to 1, because these strategies will reduce the number of variables and the search tree. An important characteristic of test construction problems is that in general the difference between z_0 and z_* is small, this characteristic makes it possible to choose H_1 and H_2 close to 1. However, if H_1 and/or H_2 are chosen to be close to 1, two kinds of problems can arise. Firstly, it is possible that too many variables are fixed at 0 and 1. This can imply that no feasible solution will be found. The second kind of problem occurs if no 0-1 solution exists with an objective function value between z_0 and H_2z_0. In both cases the branch-and-bound tree will be small and it does not take much time before it is clear that no feasible solution with objective function value higher than H_2z_0 can be found for the chosen values of H_1 and H_2. The values of H_1 and/or H_2 can be adjusted and the solution procedure can be applied for the new values. A procedure for choosing H_1 and H_2 will be given in the discussion section.

Branching

MPSX/370 V2 offers a variety of built-in strategies for the BAB method. The effectiveness of a number of these strategies for test construction problems is regarded in the next section. In this section these strategies are described, but

first the notion of node estimation is introduced.

Node estimation

A node estimation (IBM MPSX/370 V2 Program Reference Manual, 1988, p. 381) is a number that is attached to each node waiting for processing (waiting node). It estimates the objective function value of the best 0-1 solution that can be expected at a descendent node and can be considered as a measure for the interest to continue the search by branching from the node. Here, two types of node estimations are considered. The first type of node estimation, say for node k, is found by computing

$$(6) \quad E_k = z_k - \sum_{i=1}^{I} \min [PCL_i * f_i, PCU_i * (1-f_i)],$$

where

E_k = estimation for node k,

z_K = objective function value of node k,

PCL_i = lower pseudo-cost of 0-1 variable x_i,

PCU_i = upper pseudo-cost of 0-1 variable x_i,

f_i = fractional part of the current value of 0-1 variable x_i.

Pseudo-costs were introduced by Benichou, Gauthier, Girodet, Hentges, Ribiere and Vincent (1971) and are well known in the literature of mathematical programming. They are used for predicting the detoriation of the objective function value when an integer variable with a fractional value is forced to take an integral value. During a search it is possible that

not all the pseudo-costs are known. Here, these missing pseudo-costs are assumed to be zero. For the computation of the pseudo-costs in MPSX/370 V2 the reader is referred to the IBM MPSX/370 V2 Program Reference Manual (1988, pp. 378-380).

The second type of estimation for node k is found by computing:

$$(7) \qquad E_k = - \sum_{i=1}^{I} \min(f_i, 1-f_i).$$

It should be noticed that the second type of estimation does not depend on the objective function value of node k.

Branching Strategies

The first branching strategy used in the forthcoming numerical experiments is:

PURE: - Among the 0-1 variables with a fractional value the variable with the lowest subindex i is selected as branching variable.

- Choice of the branching node: Let k be the last branching node. The next branching node is chosen according to the processing results of the two branches (k,n+1) and (k,n+2) (See Figure 2).

Insert Figure 2 about here

Three possible cases should be considered:

1) Waiting nodes n+1 and n+2 satisfy the candidature conditions given in section "Branch-and-Bound for 0-1 Linear Programming", i.e., there is no rule indicating that further branching from these nodes is senseless. In this case the node with the best estimation is chosen as branching node, where the first type of estimation (6) is used.
2) Exactly one of the waiting nodes n+1 and n+2 satisfies the candidature conditions. Then this node is chosen as branching node.
3) Node n+1 and n+2 do not satisfy the candidature conditions. From among all the other candidate nodes the one with the best estimation (type 1) is chosen as the next branching node.

The strategies PURE1 through PURE4 differ from the PURE strategy only by the choice of the branching variable:

PURE1: As branching variable the one with its value furthest from 0 and 1 is chosen.

PURE2: As branching variable the one with its value closest to 0 or 1 is chosen.

PURE3: As branching variable the one with its value closest to 0 is chosen.

PURE4: As branching variable the one with its value closest to 1 is chosen.

In addition, the strategies INT through INT4 correspond to PURE through PURE4, the difference being that the choice of the branching node is based on the estimation of type 2. The last set of strategies are USER through USER4. These strategies correspond to PURE through PURE4. Now, however, the choice of the branching node is not based on node estimation but on the objective function value: The node with the highest value is chosen.

The strategies PURE, INT1, INT2, INT3, and INT4 are equal to the strategies with the same name in the IBM MPSX/370 V2 Program Reference Manual (1988). In this manual the reader can also find a more detailed description of the strategies.

Computational Experience

The strategies described in the previous sections are compared in this section. The experiments were conducted with two simulated item banks containing 450 grammar items. The items in the banks fitted the Rasch ($b_i \sim N(0,1)$) and 3-parameter model ($a_i \sim U(0.5,1.5)$, $b_i \sim N(0,1)$, $c_i = 0.2$), respectively.

In the Maximin Model as used in the experiments the ability levels were chosen to be $\theta_1 = -1$, $\theta_2 = 0$, and $\theta_3 = 1$. The relative information at these ability levels was set equal to 1 ($r_k = 1$, $k = 1, 2, 3$). The Maximin Model was formulated as follows:

(8) Maximize y,

subject to

$$(9) \quad \sum_{i=1}^{450} I_i(\theta_k) x_i - y \geq 0, \qquad k = 1, 2, 3,$$

$$(10) \quad \sum_{i=1}^{450} x_i = 20,$$

$$(11) \quad x_i \in \{0,1\}, \qquad i = 1, \ldots, 450.$$

Constraints (8) through (11) give the basic Maximin Model. As follows from constraint (10), the number of items in the test was required to be equal to 20.

The experiments were conducted on an IBM9370. If an user wants to apply the options offered by MPSX/370 V2 in an advanced way, he/she has to write an ECL program for controlling the optimization process. The appendix shows the ECL program used in the experiments. ECL is a computer language based on PL\1. The CPU-times in the forthcoming tables are the times needed for the execution of the ECL program.

In all the experiments strategies LIM1, LIM2, and LIM3 were applied for limiting the search together with one of the branching strategies. The values of H_2 in LIM2 were set to 0.995 in all cases, implying that the difference between the objective function of the optimal 0-1 solution ($z_* = c(x^*)$)

and the 0-1 solution found was at most 0.5% of z_*, because z_0 is an upper bound on z_*.

Table 1 and 2 display the CPU-times (in minutes) for the branch-and-bound part of the solution method and the objective function values for the computed 0-1 solutions. To be more specific the CPU-time is the time needed for the procedures MIXFLOW and SOLUTION in the ECL program (see Appendix), because the CPU-times of those procedure depend on the branching strategy. In Table 1 the results for the bank with items fitting the Rasch model are shown. In this case two values for H_1 were used namely 0.995 and 0.9999. In Table 2 the items fitted the 3-parameter model and H_1 was 0.995 and 0.9975. Two values for H_1 were chosen to detect the sensitivity of the results to the H_1 value. In the 3-parameter case the H_1 value can not be chosen as close to 1 as in the Rasch case, because in the former case too many variables would be fixed, which would increase the chance of not finding a feasible 0-1 solution.

Insert Table 1 and 2 here

Looking at the two tables, it is seen that the differences between the CPU-times comparing the two values for H_1 are larger for the Rasch model than for the 3-parameter model. A possible explanation of this is the number of items fixed, which is much larger for the 3-parameter model for both

values of H_1.

The basic Maximin Model can be solved relatively easy, because the number of fractional values in the optimal solution for a relaxed (sub)problem is always smaller than or equal to the number of constraints. If a number of constraints is added to the model, a more interesting and realistic model is created. Therefore, the basic Maximin Model (8)-(11) was extended with the following constraints:

$$(12) \quad \sum_{i=1}^{450} t_i x_i \leq 60,$$

$$(13) \quad \sum_{i=1}^{150} x_i = 7,$$

$$(14) \quad \sum_{i=151}^{300} x_i = 8,$$

$$(15) \quad \sum_{i=301}^{450} x_i = 5,$$

$$(16) \quad \sum_{i=1}^{75} x_i + \sum_{i=151}^{225} x_i + \sum_{i=301}^{375} x_i = 12,$$

$$(17) \quad \sum_{i=76}^{150} x_i + \sum_{i=226}^{300} x_i + \sum_{i=376}^{450} x_i = 8.$$

Constraint (12) implied that the administration time did not

exceed 60 minutes. The coefficients t_i are estimates of the time a student from the population needs for answering item i. Here, they were drawn from the distribution U(1,6). The item banks were supposed to be partitioned in the following subsets with respect to content:

Items 1-150: noun items;

Items 151-300: verb items;

Items 301-450: adjective items.

Constraints (13)-(15) implied that the test contained 7 noun, 8 verb, and 5 adjective items. Within the specified subsets the first 75 items were multiple choice items. e other items were of the matching type. According to constraints (16) and (17) the examinees had to answer 12 multiple choice and 8 matching items. In the experiments constraint (10) was left out of the LP model, because it was implied by the constraints (13)-(15) and (16)-(17).

The same experiments as with the basic Maximin Model were conducted with the extended version. Table 3 and 4 show the CPU-times and the objective function values of the computed 0-1 solutions for the Rasch model and the 3-parameter model, respectively.

Insert Table 3 and 4 here

As in Table 1 and 2 the choice of the branching variable is important. In most of the experiments the best results in

terms of CPU-time were found when the branching variable was the variable furthest from an integer (PURE1, INT1, and USER1) and closest to 1 (PURE4, INT4, and USER4). Remarkably, the strategies INT through INT4 gave good results for all choices of branching variables when the items fitted the 3-parameter model but not when they fitted the Rasch model (see Table 1 and 3).

An interesting question is how hard is it to find an exact optimal 0-1 solution. Therefore, the basic and extended Maximin Model were also solved with the BAB method, where LIM3 was not applied, i.e. the algorithm was not stopped after the first 0-1 solution was found. LIM1 and LIM2 were applied with $H_1 = H_2$, which guaranteed that an exact optimal 0-1 solution was found if it had an objective function value higher than $H_2 z_0$ (see Discussion). For the 3-parameter model H_1 and H_2 were set equal to 0.995. For the Rasch model the gap between z_0 and z_* is smaller, and therefore the values of H_1 and H_2 were set equal to 0.9999. Only branching strategy PURE1 was applied. The CPU-times and optimal objective function values are displayed in Table 5.

Insert Table 5 here

The difference between z_0 and z_* is indeed much smaller for the Rasch model. The reason is the larger similarity among the Rasch items, which makes it also harder to solve the

Maximin Model even if H_1 and H_2 are very close to 1.

Discussion

In this paper a comparison has been made among several search strategies for the branch-and-bound method. From the results it can be concluded that the strategies for limiting the search are very effective. Fixing a large number of variables by choosing H_1 very close to 1 and stopping the search after the first 0-1 solution has been found, reduces the CPU-time considerably. A good 0-1 solution is also guaranteed by choosing H_2 close to 1. Based on the experiments the following procedure for choosing the values H_1 and H_2 is proposed:

Step 1: If the items fit the Rasch model then $H_1 = 0.9999$ and $H_2 = 0.995$. If the items fit the 3-parameter model then $H_1 = 0.995$ and $H_2 = 0.995$.

Step 2: Solve the Maximin Model with the BAB method. If a feasible 0-1 solution is found then go to Step 4 else go to Step 3.

Step 3: If $H_1 > H_2$ then $H_1 := H_2$, else $H_2 := H_2 * H_2$. Go to Step 2.

Step 4: Stop.

The starting values of H_1 and H_2 are based on the results in Table 1 through 4. If $H_1 = H_2$ and no feasible 0-1 solution is found in Step 2, then no feasible solution exists with an objective function value higher than $z_m = H_2 z_0$ (see, e.g.,

Nemhauser & Wolsey, 1988, p.389; Crowder, Padberg & Johnson, 1983). Hence, the adjustment scheme in Step 3 guarantees that the difference between the objective function value of the optimal 0-1 solution z_* and the 0-1 solution found z_F is at most 0.5%. If a smaller difference ϵ is wanted, it can be realized by initializing H_2 by $H_2 = 1 - \epsilon$ with $\epsilon \leq 0.005$ and $H_1 \geq H_2$. Then, by adjusting the H_1 and H_2 values according to Step 3 the algorithm in this paper is an ϵ-approximation algorithm, i.e. $(z_* - z_F) / z_* \leq \epsilon$.

Experiments with several branching strategies were conducted. The tables showed that especially the choice of the branching variable was important. Good choices for the branching variable were the variable furthest from an integer and the one closest to 1. The strategies INT1 through INT4 are recommended in the IBM MPSX/370 V2 Program Reference Manual (1988) for models having a large number of 0-1 variables as compared to the number of continuous variable and for which the user also tries to find a good solution but not necessarily the optimal. According to the tables these strategies were not much better, and in some cases even worse (see Table 1 and 3), if the Maximin Model is considered and the items fit the Rasch model. Table 5 shows that exact optimal 0-1 solutions can be found in a reasonable amount of CPU-time if LIM1 and LIM2 are used. For interactive test construction, however, searching for an optimal 0-1 solution can take too much time especially if the Rasch model is involved.

The above conclusions are based on experiments with

models specified by the author. Nevertheless, they probably hold for Maximin Models in general, because the results showed the same tendencies for the branching strategies under varied conditions (with and without practical constraints, Rasch- and 3-parameter model). The comparisons were also not influenced by the choice of computer implementation, computer characteristics etc, because all the experiments were conducted on the same computer and with the same program.

In this paper it was shown how the computer program MPSX/370 V2 can be used for solving the Maximin Model. Other programs (e.g., HS/LP, Haverly Systems Inc.; SCICONIC, Sciconic Computer Services Ltd.; APEX4, Control Data Corporation) also provide the user with a number of options for controlling the search. Hence, the results can also be important for users of other optimization programs. The reader interested in computer codes for solving mixed integer linear programming problems is referred to Land and Powell (1979) and Powell (1985).

Appendix

The program MPSX/370 V2 enables the user to control the search by an Extended Control Language (ECL) program. ECL is based on the computer programming language PL\1. The ECL program used for the experiments can be divided in three parts. In the first part the initialization takes place and the relaxed LP problem (Problem 0) is solved. Then, the

problem is reduced by fixing 0-1 variables at their bounds. Finally, the reduced problem is solved.

```
 CONTROL:PROCEDURE OPTIONS(MAIN);
/*******************************************************/
/* PART 1. INITIALIZATION AND OPTIMIZATION OF THE     */
/* RELAXED PROBLEM                                     */
/*******************************************************/
  %INCLUDE DPLINIT;
  DCL DUMMY $REAL;
  XBPNAME='BANK450';      /*  NAME OF THE PROBLEM      */
  XDATA='BANK450';
  XOLDNAME='BANK450';
  CALL CONVERT;
  XMINMAX='MAX';
  CALL SETUP;
  XOBJ='OBJE';
  XRHS='RIHA';
  CALL OPTIMIZE;          /*  OPTIMIZATION OF THE      */
                          /*  RELAXED PROBLEM          */
  CALL SOLJTION;          /*  PRINT THE RESULTS        */
```

```
/*****************************************************/
/* PART 2. FIXING OF 0-1 VARIABLES                   */
/*****************************************************/
  XMXDROP=0.9975*XFUNCT; /*  THE VALUE OF H1 IS      */
                         /*  SPECIFIED               */
  DUMMY=XFUNCT;
  CALL SETVAR('FILE','REVIFILE','NOHEUR','NOPRINT');
                         /*  THE VARIABLES TO BE     */
                         /*  FIXED ARE FLAGGED       */
  XPBNAME='RBANK450';    /*  NAME OF THE REDUCED     */
                         /*  PROBLEM                 */
  CALL REVISE('FILE','REVIFILE');
                         /*  THE PROBLEM IS REVISED  */
                         /*  BY LEAVING OUT FIXED    */
                         /*  VARIABLES               */

/*****************************************************/
/* PART 3. OPTIMIZATION OF THE REDUCED PROBLEM       */
/*****************************************************/
  CALL SETUP('BOUND','BOUND');
  CALL OPTIMIZE;         /*  OPTIMIZATION OF THE     */
                         /*  RELAXED REDUCED PROBLEM*/
  CALL MIXSTART;
  XMXMAXNO=1;            /*  THE SEARCH STOPS AFTER */
                         /*  XMXNAXNO 0-1 SOLUTIONS */
                         /*  ARE OBTAINED            */
  XMXSTRAT='USER';       /*  THE SEARCH STRATEGY IS */
                         /*  SPECIFIED BY THE USER  */
```

```
XMXSWT='0000004';      /*  TO SPECIFY THE STRATEGY*/
                       /*  7 SWITCHES ARE CHOSEN: */
                       /*  0000004=PURE1          */
XMXDROP=0.995*DUMMY;   /*  THE VALUE OF H2 IS     */
                       /*  SPECIFIED              */
CALL MIXFLOW;          /*  SEARCH FOR 0-1 SOLUTION*/
                       /*  USING THE BAB METHOD   */
CALL SOLUTION;         /*  PRINT THE RESULTS      */
END CONTROL;
```

The information values $I_i(\theta_k)$, k = 1, 2, 3; i = 1,...., were all multiplied by 1000 in the input file BANK450, because SETVAR did not work well with the non scaled values. The ECL program is given here to show how MPSX/370 V2 enables the user to control the optimization process and in particular how the optimization process can be controlled for the Maximin Model. For a description of the ECL language the reader is referred to the IBM MPSX/370 V2 Program Reference Manual (1988).

References

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

Adema, J.J., Boekkooi-Timminga, E., & van der Linden, W.J. (1989). Achievement test construction using 0-1 linear programming. Manuscript submitted for publication.

Benichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribiere. G., & Vincent, O. (1971). Experiments in mixed-integer linear programming. Mathematical Programming, 1, 76-94.

Benichou, M., Gauthier, J.M., Hentges, G., & Ribiere, G. (1977). The efficient solution of large-scale linear programming problems - some algorithmic techniques and computational results. Mathematical Programming, 13, 280-322.

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

Breu, R., & Burdet, C.A. (1974). Branch and bound experiments in zero-one programming. Mathematical Programming Study, 2, 1-50.

Crowder, H., Johnson, E.L., & Padberg, M. (1983). Solving large scale zero-one programming problems. Operations Research, 31, 803-834.

Hambleton, R.K., & Swaminathan, H. (1985). Item response theory: Principles and applications. Boston: Kluwer-Nijhoff.

Ibaraki, T. (1987). Enumerative approaches to combinatorial optimization...Part 1 and 2. In P.L. Hammer (Editor-in-chief), Annals of operations research: Vol. 11 (Complete Volume) No. 1-4. Basel, Switzerland: J.C. Baltzer AG.

IBM Mathematical Programming System Extended/370 Version 2 (MPSX/370 V2) Program Reference Manual (1988). Form number SH 19-6553-0, IBM corporation.

Land, A., & Powell, S. (1979). Computer codes for problems of integer programming. In P.L. Hammer, E.L. Johnson and B.H. Korte (Eds.), Annals of discrete mathematics 5: Discrete optimization 2. Amsterdam: North Holland Publishing Company.

Lord, F.M. (1980). Applications of item response theory to practical testing problems. Lawrence Erlbaum Associates, Hillsdale, New Yersey.

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

Nemhauser, G.L., & Wolsey, L.A. (1988). Integer and combinatorial optimization. New York: John Wiley & Sons, Inc.

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

Powell, S. (1985). Software. In M. O'hEigearteigh, J.K. Lenstra and A.G.H. Rinnooy Kan (Eds.), Combinatorial optimization: Annotated bibliographies. Chichester: John Wiley & Sons, Inc.

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

Rasch, G. (1960). Probabilistic models for some intelligence and attainment tests. Copenhagen: Nielsen and Lydicke.

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

van der Linden, W.J., & Boekkooi-Timminga, E. (1989). A maximin model for test design with practical constraints. Psychometrika, 53, 237-247.

Table 1

Results for the branching strategies applied to the basic Maximin Model where the items fit the Rasch model.

	H_1=0.995[a]		H_1=0.9999[b]	
Branching Strategy	CPU-time (mins.)	Objective Function Value	CPU-time (mins.)	Objective Function Value
PURE	0.26	3.9299	0.09	3.9299
PURE1	0.16	3.9313	0.07	3.9313
PURE2	0.32	3.9300	0.08	3.9300
PURE3	0.32	3.9300	0.07	3.9300
PURE4	0.16	3.9313	0.06	3.9313
INT	0.29	3.9312	0.16	3.9312
INT1	0.24	3.9287	0.09	3.9287
INT2	0.90	3.9307	0.15	3.9307
INT3	0.89	3.9307	0.15	3.9307
INT4	0.24	3.9287	0.09	3.9287
USER	0.25	3.9299	0.10	3.9299
USER1	0.16	3.9313	0.06	3.9313
USER2	0.33	3.9300	0.08	3.9300
USER3	0.33	3.9300	0.08	3.9300
USER4	0.17	3.9313	0.06	3.9313

[a] Number of fixed variables: 133. Elapsed CPU-time before MIXFLOW was called: 0.15 mins.

[b] Number of fixed variables: 375. Elapsed CPU-time before MIXFLOW was called: 0.16 mins.

Table 2

Results for the branching strategies applied to the basic Maximin Model where the items fit the 3-parameter model.

	H_1=0.995[a]		H_1=0.9975[b]	
Branching Strategy	CPU-time (mins.)	Objective Function Value	CPU-time (mins.)	Objective Function Value
PURE	0.20	4.1024	0.16	4.1024
PURE1	0.07	4.1024	0.04	4.1024
PURE2	0.24	4.1043	0.05	4.1060
PURE3	0.25	4.1043	0.05	4.1060
PURE4	0.07	4.1024	0.04	4.1024
INT	0.08	4.1033	0.06	4.1033
INT1	0.07	4.1043	0.05	4.1043
INT2	0.09	4.1085	0.06	4.1085
INT3	0.09	4.1085	0.06	4.1085
INT4	0.07	4.1043	0.05	4.1043
USER	0.10	4.1058	0.09	4.1058
USER1	0.06	4.1024	0.04	4.1024
USER2	0.16	4.0995	0.05	4.1060
USER3	0.16	4.0995	0.05	4.1060
USER4	0.06	4.1024	0.04	4.1024

[a] Number of fixed variables: 393. Elapsed CPU-time before MIXFLOW was called: 0.14 mins.

[b] Number of fixed variables: 419. Elapsed CPU-time before MIXFLOW was called: 0.15 mins.

Table 3

Results for the branching strategies applied to the extended Maximin Model where the items fit the Rasch model.

	H_1=0.995[a]		H_1=0.9999[b]	
Branching Strategy	CPU-time (mins.)	Objective Function Value	CPU-time (mins.)	Objective Function Value
PURE	0.23	3.9308	0.09	3.9308
PURE1	0.25	3.9295	0.15	3.9257
PURE2	1.35	3.9308	0.21	3.9206
PURE3	0.60	3.9308	0.17	3.9308
PURE4	0.21	3.9308	0.09	3.9308
INT	0.43	3.9306	0.19	3.9306
INT1	0.99	3.9296	0.19	3.9296
INT2	2.59	3.9245	0.17	3.9299
INT3	1.91	3.9310	0.14	3.9310
INT4	0.38	3.9212	0.14	3.9212
USER	0.22	3.9308	0.08	3.9308
USER1	0.21	3.9295	0.14	3.9257
USER2	1.35	3.9308	0.21	3.9206
USER3	0.60	3.9308	0.17	3.9308
USER4	0.22	3.9308	0.09	3.9308

[a] Number of fixed variables: 133. Elapsed CPU-time before MIXFLOW was called: 0.19 mins.

[b] Number of fixed variables: 377. Elapsed CPU-time before MIXFLOW was called: 0.20 mins.

Table 4

Results for the branching strategies applied to the extended Maximin Model where the items fit the 3-parameter model.

	H_1=0.995[a]		H_1=0.9975[b]	
Branching Strategy	CPU-time (mins.)	Objective Function Value	CPU-time (mins.)	Objective Function Value
PURE	0.43	4.0531	0.17	4.0583
PURE1	0.13	4.0583	0.20	4.0607
PURE2	0.84	4.0580	0.37	4.0607
PURE3	0.36	4.0565	0.48	4.0580
PURE4	1.20	4.0580	0.14	4.0607
INT	0.12	4.0580	0.09	4.0580
INT1	0.09	4.0589	0.09	4.0580
INT2	0.41	4.0575	0.18	4.0575
INT3	0.41	4.0575	0.17	4.0575
INT4	0.12	4.0580	0.08	4.0580
USER	0.27	4.0583	0.17	4.0583
USER1	0.25	4.0583	0.17	4.0583
USER2	0.56	4.0607	0.78	4.0589
USER3	0.30	4.0565	0.39	4.0566
USER4	0.47	4.0607	0.26	4.0583

[a] Number of fixed items: 390. Elapsed CPU-time before MIXFLOW was called: 0.17 mins.

[b] Number of fixed items: 414. Elapsed CPU-time before MIXFLOW was called: 0.17 mins.

Table 5

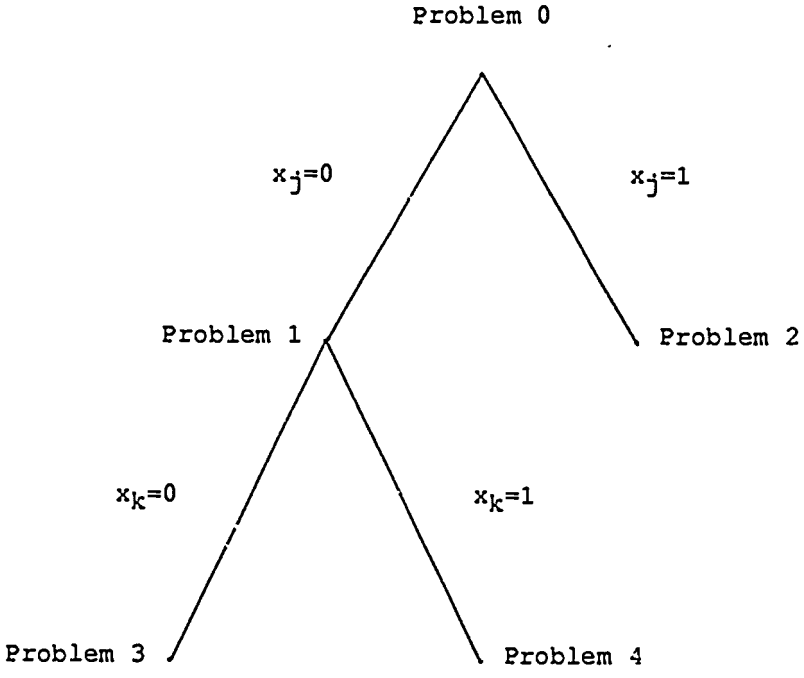
Exact optimal 0-1 solutions by branching strategy PURE1

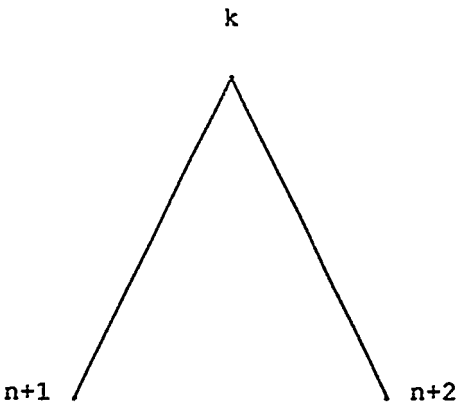
IRT Model	Extended Maximin	Objective Function Value: Relaxed	Objective Function Value: 0-1	CPU-time (mins.)
Rasch	no	3.9318	3.9317	3.24
	yes	3.9314	3.9313	4.23
3-param.	no	4.1188	4.1105	0.82
	yes	4.0731	4.0607	2.64

Figure Captions

Figure 1. Branch-and-bound tree.

Figure 2. Node k and its children.


Problem 0
$x_j=0$
$x_j=1$
Problem 1
Problem 2
$x_k=0$
$x_k=1$
Problem 3
Problem 4


k
n+1
n+2

Titles of recent Research Reports from the Division of Educational Measurement and Data Analysis, University of Twente, Enschede, The Netherlands.

RR-88-1 E. van der Burg & J. de Leeuw, *Nonlinear redundancy analysis*

RR-88-2 W.J. van der Linden & J.J. Adema, *Algorithmic test design using classical item parameters*

RR-88-3 E. Boekkooi-Timminga, *A cluster-based method for test construction*

RR-88-4 J.J. Adema, *A note on solving large-scale zero-one programming problems*

RR-88-5 W.J. van der Linden, *Optimizing incomplete sample designs for item response model parameters*

RR-88-6 H.J. Vos, *The use of decision theory in the Minnesota Adaptive Instructional System*

RR-88-7 J.H.A.N. Rikers, *Towards an authoring system for item construction*

RR-88-8 R.J.H. Engelen, W.J. van der Linden, & S.J. Oosterloo, *Item information in the Rasch model*

RR-88-9 W.J. van der Linden & T.J.H.M. Eggen, *The Rasch model as a model for paired comparisons with an individual tie parameter*

RR-88-10 H. Kelderman & G. Macready, *Loglinear-latent-class models for detecting item bias*

RR-88-11 D.L. Knol & M.P.F. Berger, *Empirical comparison between factor analysis and item response models*

RR-88-12 E. van der Burg & G. Dijksterhuis, *Nonlinear canonical correlation analysis of multiway data*

RR-88-13 J. Kogut, *Asymptotic distribution of an IRT person fit index*

RR-88-14 J.J. Adema, *The construction of two-stage tests*

RR-88-15 H.J. Vos, *Simultaneous optimization of decisions using a linear utility function*

RR-88-16 H. Kelderman, *An IRT model for item responses that are subject to omission and/or intrusion errors*

RR-88-17 H. Kelderman, *Loglinear multidimensional IRT models for polytomously scored items*

RR 88-18 H.J. Vos, *Applications of decision theory to computer based adaptive instructional systems*

RR-89-1 R.J.H. Engelen & R.J. Jannarone, *A connection between item/subtest regression and the Rasch model*

RR-89-2 E. Boekkooi-Timminga, *The construction of parallel tests from IRT-based item banks*

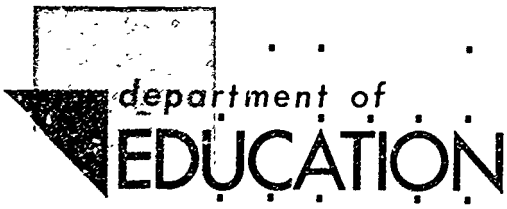
RR-89-3 D.L. Knol, *Stepwise item selection procedures for Rasch scales using quasi-loglinear models*

RR-89-4 M.P.F. Berger, *On the efficiency of IRT models when applied to different sampling designs*

RR-89-5 H.J. Vos, *A simultaneous approach to optimizing treatment assignments with mastery scores*

RR-89-6 J.J. Adema, *Implementations of the Branch-and-Bound method for test construction problems*

Research Reports can be obtained at costs from Bibliotheek, Department of Education, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.



A publication by
the Department of Education
of the University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands