Report Nr 01

Item selection using Multiobjective Programming

A.J.R.M. Gademann

Cito

ITEM SELECTION USING MULTIOBJECTIVE PROGRAMMING

by

A.J.R.M. Gademann

Report Nr 01

Project 'Optimal Item Selection'

CITO

Arnhem, the Netherlands (1987)

General Introduction

The purpose of project 'Optimal Item Selection' is to solve a number of issues in automated test design, making extensive use of optimization techniques. To this end, there has been close cooperation between the project and, among others, the department of Operations Research at Twente University. In each report, one or several theoretical issues are raised and an attempt is made to solve them. Furthermore, each report is accompanied by one or more computer programs, which are the implementations of the methods that have been investigated. The texts of these programs were included in the original thesis report, but will not be included in this version. In due time, requests for these programs can be sent to the program director.

T.J.J.M. Theunissen

project director

Summary

In this thesis two methods are described to solve the item selection problem when several criteria are involved. These criteria are represented by a composite objective function that has to be minimized or maximized over the feasible region.

The first method can be applied if the composite objective function is linear. A tractable implementation of this method is presented as well as some simulation results.

The second method can be applied if we are dealing with a quadratic composite objective function. A tractable implementation of this method is still in progress. Some simulation results for this method are presented. However, there are still some theoretical problems that have to be managed.

Some advices for users of these methods as well as some recommendations for a supplementary study can be found in this thesis.

Preface

In this report a study is made of methods to solve the item selection problem when several criteria are involved.

This study is part of a research program for developing automized testing service systems, a cooperative project of several institutes among which the National Institute for Educational Measurement (CITO) and the University of Twente (UT).

I'd like to thank the CITO for giving me the opportunity to perform this study. Furthermore I'd like to thank everyone that supported me.

Special thanks I'd like to give to my supervisors Phiel Theunissen (CITO) and Sjoerd Baas (UT) for their advices, the pleasant cooperation and the cups of coffee they offered me.

Noud Gademann
Faculty of Applied Mathematics
University of Twente

Contents

1 Test design

1.0 Introduction

In today's practice testing plays a very important role. Tests are used for example to identify a student's ability on certain subjects, or as part of an application procedure, to learn about someone's qualities.

Test theory is the theory concerned with the problem of how to design a good test. One straightforward way to design a test is by choosing items until a specified number has been reached. Another method of test design consists of selecting a number of items that together satisfy certain specified conditions on the test to be designed. The items are selected from a bank filled up with a great number of items, a so-called *item bank*. This second method of test design is the subject of the study in this report. The process is called *item selection*.

1.1 Test theory

In this paragraph a brief introduction to test theory will be given. For more details, see for example [1] and [7].

To design a test by item selection we need information about the items that are stored in the item bank. Since most tests are used to identify someone's ability on a certain subject, we need a mathematical expression about the way a person's answer to an item is related to his ability. For that purpose we introduce the *item response function*, which is a measure of the probability that a person with known ability answers correctly to a *dichotomous* item (an item that can be answered in only two distinct ways). Several item response functions are known in test theory. The one we will use in our study was introduced by Rasch.

Denoting by θ the *ability* to be measured, the *probability* p of a correct answer to an item is in the *Rasch-model* given by:

$$(1.1) \qquad p = p(\theta) = \frac{1}{1 + e^{-(\theta - b)}}$$

Commonly θ has a value between -3 and +3. The greater θ, the higher the ability. The 'e' in (1.1) is the mathematical constant 2.71828, that is $\ln^{-1}(1)$. The parameter b is called the *item difficulty*. From (1.1) we see that for very difficult items ($b \to +\infty$) $p \to 0$, and for very easy items ($b \to -\infty$) $p \to 1$.

The Rasch-model is a special case of the Birnbaum-model and is one of the simpler item response models. It ignores, in stead of many other response models, the influence on the probability p of guessing and the discriminating power of an item. But for a lot of purposes the Rasch-model is appropriate, and therefore, as well as for its simplicity, it is often used.

As mentioned in the previous paragraph, the items that are selected from the item bank have to satisfy certain conditions. An important type of condition is deduced from the so-called *information function*. The term information function will now be elucidated in brief using an introductory book by Lord [5]. Only the interpretation, not the mathematical backgrounds, will be elucidated.

Suppose we have a test consisting of n items, performed by a person with ability level θ. For every item we introduce a variable x_j. For a correct answer to item j we say that $x_j = 1$, otherwise $x_j = 0$. Now the total score x_0 of a person for this test is given by:

$$(1.2) \qquad x_0 = \sum_{j=1}^{n} x_j$$

What we want to achieve is a proper decision on someone's ability, based on his *test score* x_0. By using the item response function of all items in the test, it is possible to derive a 95 % confidence interval $[\ \underline{\theta}\ ,\ \overline{\theta}\ ]$ for the ability θ of the tested person from his test score.

Now, by definition the information function $I(\theta,x_0)$ for any test score x_0 is inversely proportional to the square of the length of the asymptotic confidence interval for estimating ability θ from x_0. (By asymptotic is meant, that the test contains a huge amount of items.) So, when the information function is relatively high at ability level θ_0, this means that at this level the estimation of a person's ability, based on his test score, is relatively good. This is an important characteristic, on which some of the most relevant conditions concerning a test are based.

When $p_j = p_j(\theta)$ is the item response function of item j, and y is the *fractional score* of correct answers, the information function is by definition:

$$(1.3) \qquad I(\theta,y) = \frac{\left(\frac{d}{d\theta}\,\mu_y|\theta\right)^2}{\mathrm{var}_y|\theta}$$

When we substitute for y the score z = (test score x_0)/n the information function becomes:

$$(1.4) \qquad I(\theta,z) = \frac{\left(\sum_{j=1}^{n}\frac{d}{d\theta}\,p_j\right)^2}{\sum_{j=1}^{n} p_j*(1-p_j)}$$

since $\mu_z|\theta = \frac{1}{n} * \sum_{j=1}^{n} p_j$ and $\mathrm{var}_z|\theta = \frac{1}{n^2} * \sum_{j=1}^{n} p_j*(1-p_j)$

Taking $y = x_j$ we get the so-called *item information function:*

$$(1.5) \qquad I(\theta,x_j) = \frac{\left(\frac{d}{d\theta}\,p_j\right)^2}{p_j*(1-p_j)}$$

Another important concept that requires further explanation is known as the *test information function*. When items in a test are independent - that means that an answer to an item is not influenced by another item in the test - the test information function is specified by:

$$(1.6) \qquad I(\theta) = \sum_{j=1}^{n} I(\theta, x_j)$$

The test information function forms an upperbound to the information that can be obtained from a test. An important observation is, that for independent items the information is additive. So when we have for example 5 items in a test, we can easily sum up the information at every point to get the 5-item test information function.

Using the Rasch-model we can derive the following expression for the item information function:

$$I(\theta, x_j) = \frac{(\frac{d}{d\theta} p_j)^2}{p_j * (1-p_j)} = \frac{\frac{(e^{-(\theta-b_j)})^2}{(1 + e^{-(\theta-b_j)})^4}}{\frac{e^{-(\theta-b_j)}}{(1 + e^{-(\theta-b_j)})^2}} = \frac{e^{-(\theta-b_j)}}{(1 + e^{-(\theta-b_j)})^2}$$

The maximum item information is obtained for $\theta^* = b_j$, that is where the ability level equals the item difficulty. It has a value of $I(\theta^*, x_j) = 0.25$.

2 Specification of the item selection problem

2.0 Introduction

In this chapter we will formally specify the item selection problem that is the subject of our study. This problem may be regarded as an extension of the problem specified by Boomsma ([1]).

2.1 The specification

Before we can specify the item selection problem we need to introduce the *target information function*. The target information function is a discrete function that specifies for a number of ability levels θ_i the amount of information a user wants to achieve in the test that is to be designed. In general the number of ability levels , that is values for θ, for which a 'target' is specified, is not greater than seven, and the ability levels are in the range [-3,+3].

The conditions that we specify are derived from the given target information function and from the test information function. We use the property of the latter that values at any point can be obtained by summing up the item information of all selected items, that is all items in the test. When a test designer uses the values of the target information function at m points on the θ-axis, say θ_i, i=1,..,m, this results in the following m conditions:

$$(2.1) \qquad \sum_{j=1}^{n} I(\theta_i,j) * x_j \geq T(\theta_i) \qquad \begin{array}{l} i=1,..,m \\ x_j=0 \text{ or } 1,\ j=1,..,n \end{array}$$

where:

- n is the number of items in the item bank,
- $T(\theta_i)$ is the specified target information at ability level θ_i,
- $I(\theta_i,j)$ is the item information of item j at ability level θ_i,
- x_j is the decision variable associated with item j,
- $x_j = \begin{cases} 1 \text{ if item j is selected} \\ 0 \text{ otherwise} \end{cases}$

We will regard the item selection problem as a deterministic problem, in spite of the stochastic nature of $I(\theta_i,j)$ and $I(\theta)$.

Now we have characterized the main constraints, but the problem is not complete without the formulation of an objective.

2.2 Justification of the multiobjective approach

Boomsma ([1]) used only constraints (2.1) and his objective was to minimize the number of items selected while satisfying these constraints. He studied several solution methods for his version of the item selection problem and compared those methods on accuracy and computation time. Boomsma's study can be continued in two ways:

1) The problem specification is not changed, but more accurate and faster (eventually heuristic) methods to solve the item selection problem are designed. This was done by Razoux Schultz ([6]). He developed some very fast heuristics that produced about the same level of accuracy as Boomsma's best methods. However, a disadvantage of these heuristics is, that they are very much depending on the structure of the problem. This means that when a slight change in the problem specification is made, often also the heuristic has to be adjusted. And although in theory this is not a very difficult task, one has to take into account that in practice a user is not always able to make an appropriate change in a heuristic, since he/she is not familiar with all the theoretical backgrounds.
2) The problem is extended in the following sense:
   A user could have more wishes than just satisfying the target information function, and more or other objectives than just minimizing the number of items selected. In that case the item selection problem should be extended with one or more objective functions and/or some other kinds of conditions.

In this report a study is made of methods to solve the item selection problem in the sense of point 2). We will elucidate why there is a need for such an extension of the problem specification.

When we examine some tests that result from using Boomsma's methods to solve the item selection problem, we observe that the items are often selected in one or more groups with about the same item difficulty within each group. This results from specifying the minimization of the number of selected items as objective. Razoux Schultz showed this and he used this feature to construct his heuristics. However, sometimes a user does not want the items to be selected in groups.

Furthermore a user often wants to construct several tests using the same item bank, the same conditions and the same solution method, but resulting in a different test (so-called *parallel tests*). When we apply one of Boomsma's methods we are not able to construct such parallel tests.

One of the extra wishes that a user often specifies is, that the selected items form a representative test for the subject the test is about. Suppose for example that a biology test is constructed and a user wishes that about 20 % of the selected items refers to the animal world, about 30 % refers to the vegetable kingdom and about 15 % refers to the human being.

Also there can be a number of so-called *cost factors* we have to reckon with when we design a test, for example: the total amount of text of the selected items may not exceed 2 pages for layout reasons.

From this we may conclude that the item selection problem as specified by Boomsma often needs to be extended to satisfy other conditions or to optimize other or more objectives. In this report so-called *multiobjective programming* methods are developed and/or examined. A multiobjective approach of the item selection problem is made for the following reasons:

- as mentioned above there is sometimes more than one objective involved when a test has to be designed;
- in general, there are two possibilities to translate a user's wishes into mathematical expressions. One way is to specify additional conditions that have to be satisfied, and thereby reduce, thus change, the feasible region. Another way is to leave the feasible region unchanged, but specify an objective function such that the user's wishes will be fulfilled as optimally as possible.

We should take into account that we are dealing with a problem of integer programming since the decision variables x_j, j=1,..,n, can only take the integer values 0 or 1 (see (2.1)). In general it is very difficult to find exact solutions to integer programming problems. In this report *quasi-exact* methods will be studied. This will be done in the following manner.

As in Boomsma ([1]) we will derive our solution to the item selection problem by solving the continuous version of the problem exactly and then determine the integer solution by rounding off. This rounding off is done in such a way that non-integer values - that are values in the range <0,1> - are rounded up to one. The continuous version of the item selection problem differs only from the discrete version in the sense that we demand for the decision variables x_j, that $0 \leq x_j \leq 1$, j=1,..,n, instead of demanding that x_j=0 or 1, j=1,..,n.

It is easy to see that, in general, when we derive the integer solution to the item selection problem from the solution to the continuous problem by rounding up the non-integer values, the fewer the number of variables that have non-integer values, the more exact the derived integer solution will be. Furthermore it is known that, when we are dealing with a problem that is specified by a linear objective function, linear functions that describe the feasible region and only integer-valued lower- and upperbounds to the decision variables, the number of non-integer values in a solution to that problem is not greater than the number of conditions that have to be fulfilled (see [2]).

For this reason we decided to translate a user's wishes into linear objective functions, if possible. So in general our item selection problem is specified by a few (eight or less) linear conditions that have to be fulfilled and one or more linear objective functions that have to be optimized. Hence we are confronted with a linear multiobjective programming problem.

In the next chapter we will describe a method to solve the item selection problem by this linear multiobjective approach.

However, it isn't always possible to translate a user's wishes into (a) linear objective function(s). In that case one can decide to formulate some additional linear conditions that have to be fulfilled, but there may also be rather tractable *quadratic objective functions* involved. Hence, in chapter 4 a general method to solve multiobjective programming problems with quadratic objective functions involved, will be described and examined for its usefulness to solve the item selection problem.

3 Linear multiobjective programming

3.0 Introduction

In this chapter we will describe a method to solve the item selection problem by a multiobjective approach when only linear objective functions are involved.

In paragraph 1 the formal describtion of the multiobjective programming approach will be given, and some theory about how to solve the resulting multiobjective programming problem will be elucidated. In paragraph 2 an interactive method to find satisfactory solutions to the item selection problem by multiobjective programming will be given. In paragraph 3 an implementation of this method in the form of a computer program for IBM-compatible PC's will be elucidated. In paragraph 4 some tests that resulted from applying this computer program, using several simulated item banks, will be examined. Finally in paragraph 5 some remarks about the discussed method and algorithm are made that could be of use for any user or for further investigations on this subject.

3.1 The multiobjective programming approach to solve the item selection problem

Suppose a user has ℓ objectives ($\ell \geq 1$) which he wants to optimize, and these objectives can be translated into ℓ linear objective functions f_i, $i=1,..,\ell$, that have to be minimized. Besides m constraints, derived from the target information function, have to be fulfilled. Then our linear multiobjective programming problem is given by:

$$\text{(3.1)} \qquad \text{Min } F(x) = (f_1(x),..,f_\ell(x)) \text{ , such that}$$

$$\text{(3.2.a)} \qquad \sum_{j=1}^{n} I(\theta_i,j)*x_j \geq T(\theta_i), \qquad i = 1,..,m ,$$

$$\text{(3.2.b)} \qquad x_j = 0 \text{ or } 1, \qquad j = 1,..,n ,$$

where:

$$f_i(x) = \sum_{j=1}^{n} c_j^i * x_j, \qquad i = 1,..,\ell ,$$

$x = (x_1,..,x_n)$,

n = number of items in the item bank involved.

Thus (3.1) gives the ℓ objective functions that have to be minimized, and (3.2) specifies the feasible region. To get the continuous version of this problem we have to replace (3.2.b) by:

(3.2.c) $\quad 0 \leq x_j \leq 1, \qquad\qquad j=1,..,n.$

As mentioned in the previous chapter, we will derive our solution to the item selection problem from the solution to the continuous version of the multiobjective programming problem. Therefore we will describe a method to solve the problem given by (3.1), (3.2.a) and (3.2.c), which we will refer to with the term CMOPP (Continuous MultiObjective Programming Problem). We have to take into account that in general there is not a unique solution to the CMOPP, since more than one objective function is involved which in general can not be minimized all at the same time by one test. Therefore we will look for so-called *nondominated solutions*.

(3.3) Definition: the *feasible region* of the CMOPP is given by:
$X = \{x = (x_1,..,x_n) \mid x$ fulfils (3.2.a) and (3.2.c)$\}$.

(3.4) Definition: a point $\bar{x} \in X$ is a nondominated solution of the CMOPP if there exists no $x \in X$ such that $F(x) \leq F(\bar{x})$ and $F(x) \neq F(\bar{x})$.
($F(x) \leq F(y)$ means that $f_i(x) \leq f_i(y)$, $i=1,..,\ell$)

In practice this means that $\bar{x} \in X$ is a nondominated solution if it is not possible to find an $x \in X$ that decreases the value of one of the objective functions without increasing the value of another function, compared with the objective function values $f_i(\bar{x})$, $i=1,..,n$.

(3.5) Definition: the set of nondominated solutions will be denoted by N, and the set of dominated solutions by D. Thus D is given by X - N.

From (3.4) it is seen that: $x,\bar{x} \in N \Leftrightarrow (F(x) \leq F(\bar{x}) \Rightarrow F(x)=F(\bar{x}))$. Now we have to give some definitions of tools that we use to identify nondominated solutions to the CMOPP.

(3.6) Definition: the *parametric space* Λ is given by:

$$\Lambda = \{\lambda \mid \lambda \in R^{\ell},\ \lambda_i \geq 0,\ i=1,..,\ell,\ \sum_{i=1}^{\ell} \lambda_i = 1\}.$$

(3.7) Definition: given $\lambda \in \Lambda$, let $P(\lambda)$ denote the problem:

$$\min_{x \in X} \sum_{i=1}^{\ell} \lambda_i * f_i(x) = \min_{x \in X} \lambda * F(x),$$

that is: find a point $\bar{x} \in X$ such that $\lambda * F(\bar{x}) \leq \lambda * F(x),\ \forall x \in X$.

(3.8) Definition: $L = \{x \mid x \in X, x \text{ solves } P(\lambda) \text{ for some } \lambda \in \Lambda\}$,

$(L) = \{x \mid x \in X, x \text{ solves } P(\lambda) \text{ for some } \lambda \in \text{int}(\Lambda)\}$.

It has been proved repeatedly that $(L) \subseteq N \subseteq L$ (e.g. see Yu [9]). This conclusion allows us to solve the CMOPP by means of $P(\lambda)$. We must take into account that N may not be equal to L if there are alternative solutions to $P(\lambda)$ for some $\lambda \in \Lambda$, but:

(3.9) Lemma: if for some $\lambda \in \Lambda$, $\bar{x}$ is the unique solution of $P(\lambda)$, then $\bar{x} \in N$.

Proof: suppose $\bar{x} \in D$. Then there exists $x \in X$ such that $F(x) \leq F(\bar{x})$. Since $\lambda \geq 0$, $\lambda * F(x) \leq \lambda * F(\bar{x})$. Thus $\bar{x}$ can not uniquely solve $P(\lambda)$, a contradiction. □

It has to be noted that this lemma is generally applicable since it does not explicitly require $F(x)$ (thus $f_1(x)...f_{\ell}(x)$) to be linear.

Zeleny [10] gives a method to find a finite covering $\{\Lambda_1,..,\Lambda_k\}$ of the paramatric space Λ that has the following qualities:

(3.10)
- $\bigcup_{i=1}^{k} \Lambda_i = \Lambda$, $\text{int}(\Lambda_i) \cap \text{int}(\Lambda_j) = \emptyset$ for $i \neq j$,
- for $i=1,..,k$ there exists a point $x_{\Lambda_i} \in N$ such that x_{Λ_i} solves $P(\lambda)$ for all $\lambda \in \Lambda_i$.

With the help of this method we could find the finite covering of Λ, and as a byproduct due to lemma (3.9) all nondominated solutions to the CMOPP, since $N \subseteq L$. However, this can be a rather time-consuming problem, because the number of subsets that cover the parametric space Λ can turn out to be very large, although it remains finite. But we should take into account that in practice often just a small part of the parametric space is of interest to find a suitable nondominated solution to the CMOPP and that we are not interested in the other solutions.

Suppose for example that a user wants to design a test and his objective is the following: the number of items that are selected to fulfil the target information has to be small but the test should not match recent tests to much. We suppose further that $f_1(x)$ expresses the objective of minimizing the number of items selected and $f_2(x)$ expresses the objective of matching recent tests as little as possible. Then the parametric space is given by (3.6) with $\ell=2$. It is clear that in general a user is not interested in the solution for which either $\lambda_1 = 0$ or $\lambda_2 = 0$ since this solution would eliminate the interest of the corresponding objective function. So a satisfactory nondominated solution will probably be found in the range $0.3 \leq \lambda_1 \leq 0.7$.

The method that will be given in the next paragraph to solve the CMOPP is based on a part of the theory to produce a finite covering of the parametric space. For more details about this theory see [10].

3.2 An interactive method to solve the CMOPP

Suppose we have ℓ linear objective functions $f_i(x)$, $i=1,..,l$. Furthermore we have a parametric space Λ, defined by (3.6) and a feasible region X defined by (3.3). We now define the *composite objective function* λF as follows:

(3.11) Definition: $\lambda F = \lambda * F(x) = \sum_{i=1}^{\ell} \lambda_i * f_i(x)$ for some $\lambda \in \Lambda$.

Problem (3.7) now can be given by:

(3.12) $\min_{x \in X} \lambda F$

To distinguish from the composite objective function λF we will call the objective functions $f_i(x)$ from now on *single objective functions*.

We can interprete the parameter λ as a vector of weights λ_i, where each λ_i is a measure of the importance to the user of the corresponding single objective function $f_i(x)$. The method that we will describe uses the simplex method to solve the linear programming problem (3.12). Some definitions will be given to make clear how this method works. For symbols that are used in the definitions and that are not explained we refer to the general simplex tableau given in figure (3.16). We assume that the reader is familiar with the simplex method (otherwise see [8]).

When we denote the current basic solution as $\bar{x}_C = (\bar{x}_B, \bar{x}_N)$, where $\bar{x}_B$ is the m-vector of basic variables and $\bar{x}_N$ is the (n-m)-vector of nonbasic variables, we can give the following definitions:

(3.13) Definition: J is the set of indices of the basic variables, and $\bar{J}$ of the nonbasic variables.

(3.14) Definition: the reduced costs of variable x_j associated with the single objective function $f_i(x)$ are given by:

$$\delta_j^i = \sum_{r \in J} c_r^i * y_{rj} - c_j^i, \qquad j=1,..n, \quad i=1,..,\ell.$$

(3.15) Definition: the reduced costs of variable x_j associated with the composite objective function λF are given by:

$$z_j = \sum_{i=1}^{\ell} \lambda_i * \delta_j^i, \qquad j=1,..,n.$$

(3.16) Figure: a general simplex tableau.

r	Basis	c^*	b^0	c_1^*		c_m^*	c_{m+1}^*		c_j^*		c_n^*
				x_1	...	x_m	x_{m+1}	...	x_j	...	x_n
1	x_1	c_1^*	y_1^0	1	...	0	y_{1m+1}	...	y_{1j}	...	y_{1n}
:	:	:	:	:	:	:	:	:	:	:	:
m	x_m	c_m^*	y_m^0	0	...	1	y_{mm+1}	...	y_{mj}	...	y_{mn}
			z_0	0	...	0	z_{m+1}	...	z_j	...	z_n

For simplicity reasons it is assumed that variables x_1 to x_m are in the basis (thus form x_B) and variables x_{m+1} to x_n are nonbasic variables (thus form x_N). c_i^* stands for the vector $(c_i^1,..,c_i^\ell)$.

From the theory of the simplex method we know that an optimal solution x^* to (3.12) has been found if x^* satisfies (3.17):

$$(3.17) \qquad z_j = \sum_{i=1}^{\ell} \lambda_i * \delta_j^i \geq 0, \qquad j=1,..,n.$$

Besides we know that $z_j = 0$ for all $j \in J$. Now the following theorem characterizes the domain in the Λ-space associated with x^*:

(3.18) Theorem: Let $\lambda \in \Lambda$ and let x^* solve (3.12) with reduced costs δ_j^i associated with the single objective functions.
Let $\Lambda(x^*) = \{\lambda | \lambda \in R^\ell; \sum_{i=1}^{\ell} \lambda_i * \delta_j^i \geq 0 \text{ for } j \in \bar{J}\}$.
Then $\lambda \in \Lambda(x^*)$ and x^* solves (3.12) for all $\lambda \in \Lambda(x^*)$.

The proof of this theorem is straightforward since the optimality conditions of the simplex method given by (3.17) reduce to $z_j \geq 0, j \in \bar{J}$, when we consider that $z_j=0$ for $j \in J$, and because of the fact that the reduced costs δ_j^i are independent of λ according to definition (3.14). So the following holds:

(3.19) Proof of (3.18): x^* solves (3.12) $\Leftrightarrow z_j \geq 0, j \in \bar{J} \Leftrightarrow$

$$\sum_{i=1}^{l} \lambda_i * \delta_j^i \geq 0, j \in \bar{J} \Leftrightarrow \lambda \in \Lambda(x^*)$$

and for all $\lambda \in \Lambda(x^*)$ the same holds. □

$\Lambda(x^*)$ is a member of the finite set $\{\Lambda_1, .., \Lambda_k\}$ that covers Λ. A useful property of the sets $\Lambda_i, i=1, .., k$ is that they are convex.

Since the number n of decision variables x_j is in general quite large (an item bank contains usually at least 300 items) and the number of basic variables is in general small (usually not greater than 8), the number n-m of nonbasic variables is also quite large. Therefore it still isn't easy to characterize $\Lambda(x^*)$ exactly.

But given a $\tilde{\lambda} \in \Lambda$ it is very easy to check whether or not $\tilde{\lambda} \in \Lambda(x^*)$. We can simply compute $\tilde{z}_j$ from (3.15) using $\tilde{\lambda}$, and if $\tilde{z}_j \geq 0, j \in \bar{J}$, then $\lambda \in \Lambda(x^*)$ because of theorem (3.18). In that case we don't have to solve (3.12) for $\tilde{\lambda}$ to see whether we'll find a better nondominated solution.

Hence there is a method available by which it is quite easy to examine the feasible region X for suitable nondominated solutions. We can simply start with assigning weights to the single objective functions to compose λF. Then we solve (3.12) and from the continuous solution we derive an integer solution by rounding the non-integer values up to 1. This solution then represents a test that consists of selected items from the item bank involved, and that satisfies the conditions that were deduced from the target information function. If this test is not satisfactory, we vary the parameter λ and if the new λ is not a member of $\Lambda(x^*)$ - and therefore will not lead to the same solution - we solve (3.12) again for this new λ. This procedure can be continued until a satisfactory test has been designed.

To finish the theoretical part of this method some remarks about its usefulness must be made. An advantage of this method is that it prohibits a user from solving (3.12) for several assigned weights λ that will all lead to the same test.

Another advantage is, that the number of single objective functions that are involved in the item selection process hardly influences the computation time to solve the CMOPP. This is clear to see since the simplex method is applied to the composite objective function λF and deleting or adding single objective functions or varying the parameter λ only changes the objective function coefficients of λF. However the computation time for the simplex method depends mainly on the number of variables n and on the number of constraints m.

Although in this way a large number of objectives may be used, it must be taken into account that the larger the number of objective functions used, the harder it is for a user to keep insight in which part of the parametric space Λ is useful.

This leads us to what one could see as a disadvantage of the method. Assigning useful weights to the single objective functions might cause problems for some users. Insight in the theoretical backgrounds as proposed in this report, as well as experience in applying this method, might make it more easy to assign useful weights.

Some results from applying this method to solve the item selection problem will be given in paragraph 4, but before doing so an implementation will be described in the next paragraph.

3.3 The computer program

The purpose of this study was to develop a useful method by which various versions of the item selection problem can be solved in combination with a tractable implementation of this method. This resulted in a computer program that will be elucidated in this paragraph.

The computer program is written in Turbo-Pascal and can be implemented on any IBM-compatible PC under MS-DOS. The program was developed on an Olivetti M24 PC. It makes no sense to compare the computation time of this program to solve the item selection problem with the computation times given in Boomsma [1], since Boomsma implemented his programs on a DEC 10 mainframe.

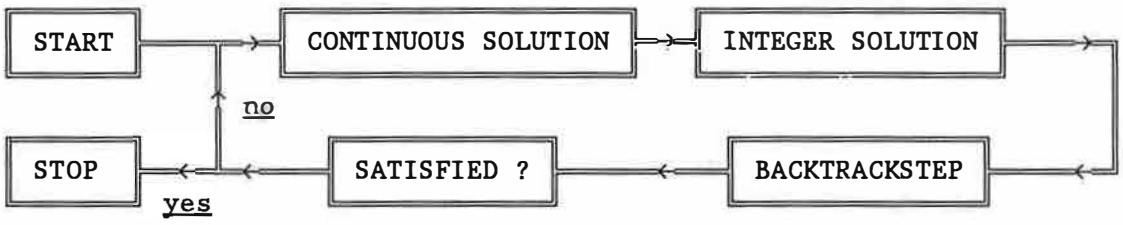
Besides we have to take into account that it was not the main purpose of this study to develop a very fast algorithm, as for instance Razoux Schultz [6] did, but an algorithm that could be applied to solve various versions of the item selection problem without changing the algorithm.

The main part of the algorithm uses the revised simplex method. An extension has been made to construct the composite objective function λF out of several single objective functions by assigning weights λ_i. The simplex method computes the reduced costs z_j associated with the composite objective function λF but an extension has been made to compute the reduced costs δ_j^i associated with the single objective functions $f_i(x)$ also. Some other extensions have been made that are of special use for solving the item selection problem. That way we've got an algorithm that works as follows:

STEP 1 start: assign weights to the single objective functions.
STEP 2 solve the continuous version of the item selection problem.
STEP 3 derive an integer-solution to the item selection problem.
STEP 4 perform a backtrackstep on the integer-solution.
STEP 5 if a satisfactory test has been designed: save it and go to step 6, else change the parameter λ so that not the same solution will be found and return to step 2.
STEP 6: stop.

A block scheme of the algorithm is given in figure (3.20)

(3.20) Figure: block scheme of the algorithm.



Ad 1) Before the problem can be solved the composite objective function λF has to be determined. This is done by assigning weights to the single objective functions $f_i(x)$.

Ad 2) The revised simplex method is applied to the continuous version of the item selection problems (the CMOPP) and the reduced costs associated with the single objective functions are computed.

Ad 3) An integer-solution to the item selection problem is derived by rounding off the solution of 2) as follows: variables that have already an integer value (0 or 1) remain unchanged, the other values are rounded up to one. This means in practice that those items which associated variables had non-zero values in the solution of 2) are selected. This way it is guaranteed that the conditions associated with the target information function are fulfilled.

Ad 4) Performing a backtrackstep means, that it is checked whether or not it is possible to eliminate one of the selected items without violating any of the constraints. The items are checked in order of ascending item number and the first item that suits is eliminated.

Ad 5) The test that resulted from step 4) is examined by the user. If he/she is satisfied the test can be saved. Else a user can assign new weights to the single objective functions and if these weights will not lead to the same solution the algorithm returns to step 2) to produce a new test.

Ad 6) Steps 2) until 5) can be continued until the user has designed (a) satisfactory test(s). Then the algorithm stops.

A listing of the procedures of the computer program that represent the extensions to the revised simplex method that have been made, is presented in appendix B.

One of the options of the main menu of the algorithm is to build up a single objective function with penalty coefficients for matching tests that have already been designed once. Among others this option will be elucidated in the next paragraph, where some of the test results from applying the described algorithm are discussed.

3.4 Some test results.

In this paragraph we will generate several tests using the algorithm of figure (3.20) to illustrate some of its possibilities. The items will be selected from a *simulated item bank* containing 200 items with difficulty parameters in the range [-3,+3]. The items in the bank are sorted in order of increasing item difficulty, thus: $i>j \Rightarrow b_i \geq b_j$. In our test problem we specify 3 target information points. The discrete target information function is given by:

(3.21)
$$T(-1) = 3.50$$
$$T(0) = 4.00$$
$$T(+1) = 3.50$$

The specification of these target information points leads to 3 constraints that have to be fulfilled by selecting the items from the item bank. As we know from chapter 1, to solve the item selection problem we have to know the item information of the items in the bank for the ability levels that targets are specified for. These item information values are stored in the item bank. Figure (3.22) shows a part of the simulated item bank that we use in this test example. The entire bank is presented in appendix A.

(3.22) Figure: a simulated item bank.

Number of items in the bank: 200
Number of target information points: 3

Item-number	Item information for θ =			Item difficulty
	-1.000	0.000	1.000	
1	0.130784	0.059110	0.023589	-2.698028
2	0.143329	0.066508	0.026843	-2.561758
3	0.159519	0.076809	0.031508	-2.391304
⋮	⋮	⋮	⋮	⋮
43	0.249867	0.192382	0.101344	-1.046163
44	0.249969	0.198629	0.106786	-0.977706
⋮	⋮	⋮	⋮	⋮
95	0.201239	0.249834	0.191891	-0.051473
96	0.193828	0.249942	0.199363	0.030465
⋮	⋮	⋮	⋮	⋮
151	0.106442	0.198245	0.249980	0.981961
152	0.097793	0.188106	0.249470	1.092139
⋮	⋮	⋮	⋮	⋮
198	0.016850	0.043219	0.101136	3.048831
199	0.015603	0.040194	0.095059	3.128319
200	0.005954	0.015856	0.040810	4.111701
	Target information:			
	3.500	4.000	3.500	

The first objective that we specify is to minimize the number of items:

$$\min f_1(x) = \min \sum_{j=1}^{200} x_j \qquad (3.23)$$

Our first item selection problem now consists of optimizing (3.23) while satisfying the conditions associated with the target information function. This results in the following test:

(3.24) Test 1

Number of items in the test: 18

The following items are selected:

86 87 88 89 90 91 92 93 94 95

96 97 98 99 100 101 102 103

The test information that is obtained by this test is given by:

(3.25) $I(-1) = 3.54880$

$I(0) = 4.48141$

$I(+1) = 3.51864$

So from (3.21) and (3.25) we see that the target information conditions are fulfilled. To find this solution the algorithm required 18 iterations. One item was eliminated in the backtrackstep. It has to be noted that the optimal objective function value of the CMOPP was about 17.7 so we can be sure that we do definitely need at least 18 items to fulfil the constraints.

When we examine test 1 we notice that only items are selected with difficulty parameters close to 0. This is a consequence of the problem specification (see [6]). A user might not be satisfied by this test since he wishes for example that items of various levels of difficulty are selected. This aim can be translated into the objective that also items have to be selected with difficulty parameters that differ considerably from 0. Therefore we can for instance formulate the following objective function:

$$\text{(3.26)} \qquad \min f_2(x) = \min \sum_{j=1}^{200} (|b_j| - 2.5)^2 * x_j$$

Optimizing this objective function would lead to selecting items with difficulty levels close to -2.5 or close to +2.5. So one could expect that when we combine (3.23) and (3.26) a more satisfactory test would result. We can use the information we obtained from test 1 to adjust our problem specification for further investigations.

Since we know that it is possible to fulfil the target information constraints by selecting 18 items, we can investigate what we can obtain when we optimize (3.26) but don't want to select significantly more than 18 items. Therefore we add the following constraint to our problem specification:

$$(3.27) \qquad \sum_{j=1}^{200} x_j \leq M$$

The M in (3.27) stands for the maximum number of items we want to be selected in the test. We have to consider that if we solve the CMOPP while fulfilling (3.27) it is very well possible that the resulting test contains more than M items since non-integer values of the solution of the CMOPP are rounded up to one. So if we allow for instance no more than 25 items in our test we should give M a value about 24 or 23, depending on the number of constraints in the problem specification. If we solve our item selection problem for M = 18.5, $\lambda_1 = 0$ and $\lambda_2 = 1$ - so only objective (3.27) is involved - this results in the following test:

(3.28) Test 2

Number of items: 19

The following items are selected:

57 58 59 60 61 62 63 64 65 126
127 128 129 130 131 132 133 134 135

I(-1) = 3.55456, I(0) = 4.30199, I(+1) = 3.63416.

We see that 19 items are selected and the items are selected in two groups with difficulty parameters close to -0.60 or +0.60. Thus the variation in the item difficulty parameters has indeed increased and therefore a user might prefer test 2 although it contains one extra item in comparison with test 1.

So when we are looking for a test that satisfies several wishes, we will often not select a minimum number of items. This is an important observation which we will use in our further investigations. From now on we will use a value of 20 for M, which means in practice that we allow at the most 21 items to be selected.

In the next step in our investigation of the item selection problem we combine (3.23) and (3.26) to a composite objective function. We solve the item selection problem for $\lambda_1 = \lambda_2 = 0.5$ (with M = 20 !). This results in the following test:

(3.29) Test 3

Number of items: 20

The following items are selected:

42	43	44	45	46	47	48	49	50	51
144	145	146	147	148	149	150	151	152	153

I(-1) = 3.57420, I(0) = 4.00267, I(+1) = 3.58378.

We see that test 3 also consists of two groups of items with difficulty parameters near -1 and near +1. Furthermore we notice that in this case the influence of (3.23) is in fact nil since both objective functions have the same weight 0.5 but the objective function coefficients in (3.26) are only less than 1 - thus less relevant than the coefficients of (3.23) - for items with difficulty parameter greater than 1.5 or less than -1.5. Thus as long as such items are not selected the solution is not influenced by (3.23).

An observation we can make when we examine the three tests designed so far is, that no items are selected twice. This is a coincidence since we did not specify an objective function that would minimize the number of matching items between consecutive tests. If we had increased M for example to 18.8 instead of 20, test 3 would probably contain several items that are also selected in test 2. However, for practical purposes it is interesting to design a test that matches previous tests as little as possible. Therefore we specify the following objective :

$$(3.30) \qquad \min f_3(x) = \min \sum_{j=1}^{200} c_j^3 * x_j$$

The objective function coefficient c_j^3 is a penalty that can be put to a positive value if item j has been selected in a previous test. A user can select these coefficients by assigning penalty values to tests he/she does not want to match too much.Then c_j^3 is the sum of these penalty values of the tests that contain item j. The following test is a result of assigning a penalty value of 2 to tests 2 and 3. So $c_j^3 = 2$ for the items selected in test 2 or test 3 since they are all selected once, and $c_j^3 = 0$ for the other items. We choose $\lambda_1 = 0$, $\lambda_2 = \lambda_3 = 0.5$.

(3.31) Test 4

Number of items: 21

The following items are selected:

38 39 40 41 52 53 54 55 56 136
137 138 139 140 141 142 143 154 155 156
157

I(-1) = 3.55358, I(0) = 4.20973, I(+1) = 3.95047.

We observe that again one extra item is selected. However, if we look at the spread of the selected items over the item bank, this is the best result obtained so far. We will derive one more test involving $f_2(x)$ and $f_3(x)$. We choose the same λ's and we assign penalty coefficients of value 2 to tests 2, 3 and 4. This results in the following test:

(3.32) Test 5

Number of items: 20

The following items are selected:

33 34 35 36 37 66 67 68 69 70
122 123 124 125 158 159 161 162 163 164

I(-1) = 3.51886, I(0) = 4.00011, I(+1) = 3.57333.

In table (3.33) all results are presented. Tests 6, 7, 8 and 9 were derived by assigning penalty values 1 to all previous tests. Test 10 was derived by assigning penalty values of respectively 8,6,2,7,1,4,3,5 to tests 1..8, and test 11 was derived by assigning penalty values of respectively 1,2,3,4,5,6,7,8 to tests 1..8. The weights for the single objective functions that were used for tests 6..11 are: $\lambda_1 = \lambda_2 = 0$ and $\lambda_3 = 1$.

(3.33) Table: some resulting tests.

Test number	Number of items	Selected items	I(-1)	I(0)	I(+1)
1	18	86..103	3.55	4.48	3.51
2	19	57..65 126..135	3.55	4.30	3.63
3	20	42..51 144..153	3.57	4.00	3.58
4	21	38..41 52..56 136..143 154..157	3.55	4.21	3.95
5	20	33..37 66..70 122..125 158 159 161..164	3.52	4.00	3.57
6	21	27..32 82..85 104..108 160 165..169	3.67	4.18	3.71
7	21	21..26 80 81 109..116 170..174	3.61	4.16	3.73
8	21	17..20 71 72 75..79 117..121 175..179	3.65	4.20	3.70
9	21	14..16 72..79 117..121 180..184	3.57	4.16	3.69
10	20	14..16 66..70 73 74 109 110 122..125 159 180..182	3.53	4.07	3.53
11	21	13..16 73 74 93..103 180..183	3.68	4.27	3.67

The mean computation time to solve these problems was about 90 seconds. We recognize that test 9 is the first test that contains items that were also selected in previous tests. It contains 9 items of test 8 and none of the other lower numbered tests. If tests 1..8 were used in practice in order of ascending test number, this would mean that test 9 matches only the most recent test, which is not very preferable.

So when items have to be selected that were also selected in previous tests, a user might prefer items from certain tests to be selected twice if necessary. Test 10 was derived by assigning various penalty values to tests 1..8 which are a measure of the importance to a user of disjunct tests. Test 11 is a special case of this strategy: the penalty values are assigned in such a way that matching a previous test is less relevant if that test is less recent.

As mentioned before, test 9 is the first test that contains items that were selected once before. This is a very nice and perhaps surprising result that is especially interesting for a user who wants to generate parallel tests. For it appeared to be possible in this case to design 8 strictly distinct tests that used 161 of the available 200 items in the simulated item bank. And also the number of items selected does not vary too much.

In the next paragraph some overall conclusions about the solution method presented in this chapter, will be given.

(3.5) Conclusions

In this chapter a method is described to solve the item selection problem by a linear multiobjective approach. This resulted in a tractable computer implementation. The results, presented in paragraph 4, show that it is quite easy to examine the feasible region for satisfactory tests. The upperbound on the number of items that are allowed to be selected, as well as the penalty function $f_3(x)$ turned out to be very helpful tools to solve the item selection problem.

The results are of special importance to users that want to design parallel tests. But also tests consisting of items that were nicely distributed over the simulated item bank were designed (f.e. tests 5, 6, 8 and 10).

The overall conclusion can be that a useful method has been developed to design tests that fulfil more objectives than just selecting a small number of items. The interactive working of the algorithm is of great importance to find convenient results. Designed tests have to be examined repeatedly and based on that, other objectives have to be included or other weights or penalty values have to be assigned. From real-life applications it should become clear whether a user is able to make proper decisions that actually lead to satisfactory results.

Further investigations to develop other useful objective functions have to be performed and other examples using various target information functions have to be examined for possible problems.

4 Multiobjective programming with quadratic objective functions

4.0 Introduction

In this chapter we will elucidate a general method to solve a multiobjective programming problem with linear constraints and a quadratic composite objective function. This means that the single objective functions can be either linear or quadratic, and it is supposed that at least one objective function is quadratic.

In paragraph 1 necessary and sufficient conditions for a solution to be optimal are presented. In paragraph 2 a method to find a solution that satisfies these conditions will be elucidated. In paragraph 3 a discussion about problems that might occur when this method is applied, will be presented. In paragraph 4 some test results will be presented and discussed and finally in paragraph 5 some conclusions and recommendations will be presented.

4.1 The Kuhn-Tucker conditions to characterize an optimal solution of a quadratic programming problem

Consider the following multiobjective programming problem:

(4.1) $\min F(x) = (f_1(x),..,f_\ell(x))$, such that:

(4.2.a) $\sum_{j=1}^{n} I(\theta_i,j)*x_j - x_{n+i} = T(\theta_i), \quad i=1,..m,$

(4.2.b) $x_j \geq 0,\ j=1,..,n+m, \quad x_j \leq 1, \quad j=1,..,n,$

where:

- $f_i(x)$ is a linear or quadratic objective function for $i=1,..,\ell$ and at least one $f_i(x)$ is quadratic,
- $x = (x_1,..,x_n)$, the vector of decision variables associated with the n items in the item bank.
- x_{n+i} is a slack variable for condition i, $i=1,..,m$, where condition i is associated with target $T(\theta_i)$.

Thus in fact (4.2) specifies the same feasible region as (3.2.a) and (3.2.c). The only difference in the specification is the introduction of the *slack variables* x_{n+i}. So the problem specified by (4.1) and (4.2) differs only from the CMOPP in chapter 3 because quadratic single objective functions are involved. We will refer to the problem specified by (4.1) and (4.2) with the term QMOPP (Quadratic MultiObjective Programming Problem). As in chapter 3 we will define a composite objective function as follows:

(4.3) Definition: given parametric space Λ defined by (3.6) and some $\lambda \in \Lambda$, the composite objective function λF is given by:
$\lambda F = \lambda * F(x) = \sum_{i=1}^{\ell} \lambda_i * f_i(x)$, with F(x) as in (4.1).

The feasible region X of the QMOPP is given by (4.2). For $\lambda \in \Lambda$ we can formulate – as in chapter 3 – the following subproblem of the QMOPP:

(4.4) $\min_{x \in X} \lambda F.$ (X as in (4.2))

As we know from chapter 3, a unique solution of (4.4) is a nondominated solution of the QMOPP, and since we are looking for nondominated solutions we can solve the QMOPP by means of (4.4).

Wolfe developed a general method to solve problems with a quadratic objective function and linear constraints, such as (4.4) using the Kuhn-Tucker necessary conditions for constrained non-linear programming. The search for such a solution is performed in the following manner. Given problem (4.4) we first introduce the associated *Lagrangian function* L that is given by:

$$(4.5) \quad L = L(x,\kappa,\mu,\nu) = \lambda F + \sum_{i=1}^{m} \kappa_i * \left[T(\theta_i) - \left(\sum_{j=1}^{n} I(\theta_i, j) * x_j - x_{n+i} \right) \right] + \sum_{j=1}^{n} (x_j - 1) * \mu_j - \sum_{j=1}^{n+m} x_j * \nu_j$$

where the $\kappa_i, i=1,..,m$, $\mu_j, j=1,..n$, and $\nu_j, j=1,..,n+m$ are the so-called *Lagrange multipliers*.

Now the *Kuhn-Tucker necessary and sufficient conditions* for a vector $x \in R^{n+m}$ to be an optimal solution of (4.4) in case F(x) is convex and the feasible region X is non-empty and closed in R^n are:

(4.6) Theorem: a vector $x \in R^{n+m}$ is an optimal solution of (4.4) if and only if there exist vectors $\kappa \in R^m$, $\mu \in R^n$ and $\nu \in R^{n+m}$, $\kappa,\mu,\nu \geq 0$, such that the following conditions are satisfied:

1) $\frac{\delta L}{\delta x_j} = 0$, $\quad j=1,..,n+m,$

2) $\frac{\delta L}{\delta \kappa_i} = 0$, $\quad i=1,..,m,$

3) $\frac{\delta L}{\delta \mu_j} \leq 0$, $\quad j=1,..,n,$

4) $\frac{\delta L}{\delta \nu_j} \leq 0$, $\quad j=1,..,n+m,$

5) $\mu_j * \frac{\delta L}{\delta \mu_j} = 0$, $\quad j=1,..,n,$

6) $\nu_j * \frac{\delta L}{\delta \nu_j} = 0$, $\quad j=1,..,n+m.$

For the proof of this theorem we refer to [3]. Conditions 2), 3) and 4) guarantee that the solution x is an element of the feasible region X. Conditions 5) and 6) express that a dual variable can only be non-zero if the associated constraint is active. Finally condition 1) expresses that a solution x must not only be feasible but also optimal for $L(x,\kappa,\mu,\nu)$, that is besides satisfying 2) to 6) also all partial derivatives of L with respect to $x_1,.. x_{n+m}$ must be 0.

Since problem (4.4) has linear constraints and a quadratic objective function, the associated Kuhn-Tucker conditions form a system of linear equations and inequalities, apart from the so-called *complementarity conditions* 5) and 6). So when we apply the Kuhn-Tucker conditions, finding a solution to (4.4) is translated into finding a solution to an almost linear system. This suggests the use of some adapted LP method (e.g. simplex). In the next paragraph such a method will be presented.

4.2 The algorithm of Wolfe to find a solution that satisfies the Kuhn-Tucker conditions

Wolfe [3] has introduced a method to find a solution x that satisfies the Kuhn-Tucker conditions as presented in theorem (4.6). Before we will describe this method we will give a somewhat different representation of the Kuhn-Tucker conditions. This is done by making the partial derivatives of the Lagrangian function L as they occur in (4.6) explicit. The system specified by the Kuhn-Tucker conditions for problem (4.4) then becomes:

$$(4.7.1)\quad \frac{\delta(\lambda F)}{\delta x_j}(x) - \sum_{i=1}^{m} \kappa_i * I(\theta_i,j) - \nu_j + \mu_j = 0, \quad j=1,..,n,$$

$$\kappa_i - \nu_{n+i} = 0 \Leftrightarrow \kappa_i = \nu_{n+i}, \qquad i=1,..,m,$$

$$(4.7.2)\quad \sum_{j=1}^{n} I(\theta_i,j)*x_j - x_{n+i} = T(\theta_i), \qquad i=1,..,m,$$

$$(4.7.3)\quad x_j \leq 1, \qquad j=1,..,n,$$

$$(4.7.4)\quad x_j \geq 0, \qquad j=1,..,n+m,$$

$$(4.7.5)\quad \mu_j*(1-x_j) = 0 \Leftrightarrow \mu_j=0 \text{ or } x_j = 1, \qquad j=1,..,n,$$

$$(4.7.6)\quad \nu_j*x_j = 0 \Leftrightarrow \nu_j=0 \text{ or } x_j=0, \qquad j=1,..,n+m.$$

When we combine (4.7.1) and (4.7.6) for j=n+1,..,n+m we can delete (4.7.1) for j=n+1,..,n+m and replace (4.7.6) by:

$\nu_j=0$ or $x_j=0$, $\qquad j=1,..,n,$

$\kappa_i=0$ or $x_{n+i}=0$, $\qquad i=1,..,m.$

From (4.7) we see that the Lagrange multipliers ν_{n+i}, i=1,..,m are redundant and therefore they are eliminated. Wolfe suggested a method to find a solution that satisfies (4.7) by using simplex. This method is now presented in the form of an algorithm.

(4.8) *Wolfe's algorithm* to find a solution to system (4.7)

STEP 1 Apply phase 1 of the simplex method to find a point $\hat{x}$ in the feasible region X of (4.4), that is find an $\hat{x} \in X$ that satisfies conditions (4.7.2), (4.7.3) and (4.7.4).

STEP 2 Add n non-negative artificial variables u_j to the conditions (4.7.1), such that we get the following conditions:

$$(4.7.1^*)\quad \frac{\delta(\lambda F)}{\delta x_j}(x) - \sum_{i=1}^{m} \kappa_i * I(\theta_i, j) - \nu_j + \mu_j + \delta_j * u_j = 0,\ j=1,..,n,$$

$$\text{with } \delta_j = \begin{cases} +1 \text{ if } \dfrac{\delta(\lambda F)}{\delta x_j}(\hat{x}) \leq 0 \\ -1 \text{ if } \dfrac{\delta(\lambda F)}{\delta x_j}(\hat{x}) > 0 \end{cases}$$

STEP 3 Assign values to the artificial variables and the Lagrange multipliers in the following manner:

$$u_j := \left| \frac{\delta(\lambda F)}{\delta x_j}(\hat{x}) \right| \geq 0,\ j=1,..,n, \qquad \kappa:=0,\ \mu:=0,\ \nu:=0.$$

Then the vector $(\hat{x},\kappa,\mu,\nu,u)$ satisfies $(4.7.1^*)$ and (4.7.2),..,(4.7.6).

STEP 4 Apply phase 2 of the simplex method to the following problem:

(*) $\min \sum_{j=1}^{n} u_j,\ u_j \geq 0,\ j=1,..,n,$

such that $(4.7.1^*)$ and (4.7.2),..,(4.7.6) are satified.

Start with the feasible vector $(\hat{x},\kappa,\mu,\nu,u)$ of step 2.

A slight modification of the simplex method has to be made to apply it in step 4, since complementarity conditions (4.7.5) and (4.7.6) are involved. This variation on the standard simplex method means that it is not allowed to introduce a variable into the basis if that causes a violation of conditions (4.7.5) or (4.7.6).

It is clear to see that, if the minimum derived in step 4 of the algorithm is equal to zero, a feasible solution to system (4.7) has been found, since in that case all artificial variables u_j are equal to zero and therefore $(4.7.1^*)$ reduces to (4.7.1). And because of theorem (4.6) we know that this feasible solution contains also a vector x^* that solves problem (4.4), if a solution exists.

From this we might conclude that Wolfe's algorithm is a convenient tool to find nondominated solutions to the QMOPP and therefore to find solutions to the item selection problem, since they can be deduced from the continuous solutions by rounding off (c.f. chapter 3.). But there are still some theoretical and practical problems that might occur when we apply Wolfe's algorithm to (4.4). These problems will be discussed in the next paragraph.

4.3 Problems that might occur when Wolfe's algorithm is applied

In the previous paragraph we described Wolfe's algorithm as a tool to solve (4.4). But will the algorithm always end with a solution for which all artificial variables are equal to zero, and if not, are there certain conditions that have to be satisfied to guarantee such a solution in step 4 of the algorithm? This is of course an important question since the solution derived by the algorithm is optimal only if it satisfies the Kuhn-Tucker conditions (4.7), thus if all artificial variables are equal to zero. The objective function λF plays an important role in whether or not Wolfe's algorithm will be successful in deriving a solution to (4.4). Since λF is a quadratic function it can be given in the following general form:

$$\lambda F(x) = cx + x'Dx, \tag{4.9}$$

that is as the sum of a linear and a quadratic function. It is known that λF is convex if and only if $x'Dx$ is a positive definite or positive semidefinite form, that is if D is a positive definite or positive semidefinite matrix. If D is positive definite λF is strictly convex. In that case there exists a unique solution to (4.4).

It has also been proved ([3])that, if λF is strictly convex, Wolfe's algorithm always terminates with a solution for which all artificial variables are equal to zero, provided that the feasible region X is non-empty and continuous, which is certainly the case for the problems considered here. So there are no problems if λF is strictly convex.

There might occur some problems if D is not a positive definite matrix since in that case we can't be sure that there exists a unique optimum. If λF is not convex, local optima might cause problems, since in that case it depends on the start vector whether or not Wolfe's algorithm will identify a global optimum if one exists. If λF is not strictly convex there might exist alternative solutions or even an unbounded solution to (4.4), especially if the feasible region is unbounded. But if a user is familiar with the item selection problem he will not specify conditions that represent an unbounded feasible region. There might be some alternative solutions but this does not cause a serious problem since Wolfe's algorithm will identify one of these alternatives when local optima do not interfere. Another point that is beneficial for the algorithm to identify the global optimum, is that in general the feasible region of (4.4) is convex and continuous, so that if a bounded optimum exists, it will not be isolated (that is, it can be reached by the algorithm since the feasible region is continuous).

So we don't expect many problems in practice when Wolfe's algorithm is applied, in the sense that in general a solution to (4.4) will be found. It has to be noted that usually it is very difficult to establish whether or not D is positive definite or positive semidefinite since D is a (n x n)-matrix and n can be very large (300 or more).

Another aspect that we have to discuss is the problem size. Wolfe's algorithm might in theory be a nice tool to solve (4.4) since from the above it leads to a solution in most cases. But we have to take into account that in step 4 of the algorithm the simplex method has to be applied to problem (*) with (4*n + 2*m) variables and (n + m) constraints, where n is the number of items in the item bank involved (that is the number of decision variables in (4.4)) and m is the number of constraints in (4.4). This means that, compared with the linear problem (3.12) where n variables and m constraints where involved, the computation time will be huge.

Another problem is that the size of arrays in an implementation of the algorithm for most computers is limited. So for large problems we have to find an alternative way to store the coefficient-matrix of (*). However, all the problems mentioned so far can be managed.

An aspect that might cause a problem that is most difficult to solve, is: is the solution of (4.4) that we will find, suitable to derive a solution to our item selection problem by rounding off? In the linear case we knew that the solution to the CMOPP contains at most m non-integer values for the decision variables and therefore is suited to derive an integer solution by rounding off. But when a quadratic objective function is to be optimized it is very well possible that in the optimal solution a great number of decision variables have non-integer values. This depends very much on the problem specification. We can illustrate this by the following example. Look at the 2 simple quadratic programming problems below.

$$(4.10) \qquad \max \sum_{j=1}^{n} x_j*(1-x_j)$$

$$(4.11) \qquad \min \sum_{j=1}^{n} x_j*(1-x_j)$$

Suppose the feasible region for both problems is given by:

$$(4.12) \qquad x_j = 0 \text{ or } 1, \; j=1,..,n.$$

It is clear to see that the optimum for both problems is equal to zero (and is independent of the values of the x_j, provided that they fulfil (4.12)). But now suppose that the feasible region is given by:

$$(4.12^*) \qquad 0 \leq x_j \leq 1, \; j=1,..,n.$$

The optimum of (4.11) is again zero and all decision variables have integer values (either 0 or 1). But (4.10) reaches its maximum of n*0.25 for x_j=0.5, j=1,..n, so all variables have non-integer values in the optimum.

This simple example illustrates that the problem specification is of great influence whether or not the continuous solution of a problem differs considerably from the integer solution and therefore is less or better suited to derive a satisfactory solution to the item selection problem. In paragraph 5 some examples are presented and discussed to show the possibilities and problems when Wolfe's algorithm is applied to solve the item selection problem.

4.4 The computer program

The algorithm, described in the previous paragraph, has not yet been implemented in a tractable computer program like the program in chapter 3. However, all the essential parts to construct such a program have already been developed. There are seperate programs available that solve the steps 1 and 4 of Wolfe's algorithm. The program that solves step 1 is just phase one of the revised simplex method. The program that solves step 4 is a somewhat changed version of the revised simplex method to provide that conditions 4.7.5 and 4.7.6. will be satisfied. Some essential procedures of the latter program are presented in appendix D. The development of a tractable program is still in progress.

Hence we are already able to derive simulation results. Some results will be presented in the next paragraph.

4.5 Some simulation results

Due to a lack of time the algorithm of paragraph 4.3 has not yet been applied extensively to simulated item selection problems. However, some theoretical ideas have been developed and some results are available.

Applying the algorithm only makes sense if any quadratic function exists that represents a user's wish concerning a test that has to be designed. We will pay attention in this paragraph to the wish of a user to build up a representative test in the sense that the selected items have to represent certain subjects (remember the example of a biology test in chapter 2).

Suppose there are s subjects involved (like human being, vegetable kingdom etc.) and that a user has specified for some of these subjects a desirable fraction of the total number of items in the test. We define:

(4.13) DF_i = desired fraction of the selected items that refers to subject i, $0 \leq DF_i \leq 1$, $i \in \{1,..,s\}$.

S_i = set of items in the itembank that refer to subject i, $i = 1,..,s$.

If for some subject i no desired fraction has been specified, then the user has no special wishes concerning that subject. Of course the sum of the specified desired fractions has to be less than or equal to 1. Now the objective of the designer to build up a representative test can be given by:

(4.14) $$\min g_1(x) = \min \sum_i (DF_i * \sum_j x_j - \sum_{j \in S_i} x_j)^2,$$

where the sum has to be taken over those subjects i for which a fraction DF_i has been specified.

In (4.14) $DF_i * \sum_j x_j$ refers to the desired fraction of selected items that refers to subject i and $\sum_{j \in S_i} x_j$ refers to the selected number of items that refer to subject i. The minimum has to be derived of course over the feasible region. We see that the theoretical minimum 0 of $g_1(x)$ is obtained if all desirable fractions are exactly met by the test. Minimizing $g_1(x)$ over the feasible region thus means fulfilling the user's wish as good as possible.

The matrix of partial derivatives of $g_1(x)$ is given by:

$$(4.15) \qquad D = \begin{bmatrix} d_{11} & \dots & d_{1n} \\ \vdots & & \\ d_{n1} & \dots & d_{nn} \end{bmatrix}$$

$$\text{with: } d_{jk} = d_{kj} = \begin{cases} (DF_{s_j} - 1)^2 + \sum_{i \neq s_j} DF_i^2 & \text{if } s_j = s_k \\ \sum_{i=j,k} (DF_{s_i} - 1) * DF_{s_i} + \sum_{i \neq j,k} DF_{s_i}^2 & \text{otherwise} \end{cases}$$

and s_j is the subject that item j refers to.

It is clear to see that the rows of matrix D are identical for all j that belong to the same set S_i. We can solve this by adding a matrix $\epsilon * I_n$ (ϵ times the n-identity matrix) to D. Then the rank of D is n.

If we minimize $g_1(x)$ subject to conditions (2.1) and (3.27) our problem specification is complete. Some simulation results will be presented now, using the (not sorted) item bank of appendix C.

Our simulation problem is specified as follows:
Two objectives are involved. The first one is to minimize the number of items in the test (c.f. (3.23)) and the second objective is to build up a representative test (c.f. (4.14)). The item bank contains 50 items. There are three subjects for which a desirable fraction is specfied. The associated sets of items are given by: $S_1 = \{1,..,15\}$, $S_2 = \{21,..,30\}$ and $S_3 = \{36,..,45\}$. For several desired fractions DF_i, i=1..3, and assigned weigths λ_k, k=1,2 results are presented in table (4.16).

(4.16) Table: some simulation results

Test number	1	2	3	4
DF_1 DF_2 DF_3		0.30 0.25 0.25	0.30 0.25 0.25	0.40 0.30 0.30
λ_1 λ_2	1 0	0.5 0.5	0.33 0.67	0.33 0.67
M		10	10	10
Number of items	9	9	10	10
Items in S_1	10	3 10	3 10	3 10 14
Items in S_2	24 25	25	25	25
Items in S_3	39 40 41 44	40	40 41	40
Other items	31 34	17 31 34 47 48	17 31 34 47 48	17 31 34 47 48

The mean computation time to solve these problems was about 360 seconds. These results show that we indeed can design more representative tests when we involve the quadratic objective function $g_1(x)$. However, there are some theoretical problems we have to reckon with when we specify the desired fractions and assign the weights to the single objective functions.

The first problem is that the specified fractions have to be realistic. If a test consists for example of about 20 items and there are only 3 items in the item bank that refer to subject i, it makes no sense to specify $DF_i = 0.5$. So we have to be familiar with the composition of the item bank to be able to specify realistic desired fractions.

The second problem is that, when λ_2 is relatively great compared with λ_1, there might be a considerable number of fractional values in the solution of the QMOPP, and therefore this solution would not be useful.

Finally we have to take into account that matrix D might not be (semi-) positive definite and therefore might cause problems in finding a solution to the QMOPP.

4.6 Conclusions

The only quadratic objective function that has been used in a simulated item selection problem so far, is $g_1(x)$ of (4.14). From the previous paragraph we know that including this function can lead to representative tests. However, there are still some theoretical problems that have to be managed. Further investigations have to prove whether or not we can solve these problems in a proper way.

Apart of this function there are some other quadratic functions that are of interest for further investigations. They have not yet been applied to simulated item selection problems.

The first one we will specify is associated with the objective to improve the *calibration* of the item bank that is involved in the item selection problem. An item bank is said to be well calibrated if the estimation error for the difficulty parameter b_j is about the same for all items in the bank. The value of b_j is estimated, based on previously examined tests. The following simple example will illustrate why differences in the estimation error do occur.

Suppose there have been 20 tests examined so far. In all those tests both item i and item j were selected. There have been 7 correct answers to item i and to item j. Based on these results, item i is estimated to be twice as difficult as item j (we do not concern about how this is reflected by the item difficulty parameter). Items k and m however have been selected only three times together in the same test.

In those three tests there have been 2 correct answers to item k and one to item m. Based on this, item m is estimated to be twice as difficult as item k. (Of course the estimation is not as easy as proposed here, since all items are involved in the estimating process.) It is obvious that the first estimation, concerning items i and j, is more confident than the second estimation since the first estimate involves many more tests. This example illustrates why the estimation errors can differ considerably.

Suppose there exists a matrix D that expresses how proper pairs of items are calibrated, in such a way that the smaller d_{ij}, the worse pair i,j is calibrated. We now specify the following function:

$$(4.17) \qquad g_2(x) = x'Dx = \sum_i \sum_j d_{ij} * x_i * x_j$$

Now our objective to improve the calibration of the item bank can be expressed by minimizing $g_2(x)$ over the feasible region.

The same idea can be used when *logical conditions* are involved that concern only pairs of items. Suppose a test has to be designed by item selection, but certain pairs of items are not allowed to be selected, that is: the two items that form the pair are not allowed to be selected both. If we then specify the elements d_{ij} of matrix D as follows:

$$(4.18) \qquad d_{ij} = \begin{cases} +H & \text{if pair i,j is not allowed to be selected} \\ & \text{(H stands for a huge integer)} \\ +1 & \text{otherwise} \end{cases}$$

we can again express our objective by minimizing x'Dx over the feasible region.

These two examples of using quadratic functions to express a user's objective are correctly formulated in case only 0-1 variables are involved. Further investigations will have to prove whether the solutions to the associated QMOPP's are also useful.

Besides studying several quadratic objective functions, also further investigations have to be performed to find tractable methods for identifying more than one nondominated solutions, like the method presented in chapter 4.

5 Review

In this thesis two methods to solve item selection problems involving several criteria are discussed. The first method - described in chapter 3 - can be applied when criteria can be translated into linear objective functions. A tractable implementation of this method has been developed. Further investigations to identify useful objective functions have to be performed.

The second method - described in chapter 4 - has to be applied when quadratic objective functions are involved. A tractable implementation of this method is still in progress. Some simulation results have been presented. From the simulations we know that there are still some theoretical problems that have to be managed. Further investigations on this subject as well as on the possibilities to combine both methods have to be performed. We could think for instance of deriving conditions that specify whether or not a solution to a QMOPP will remain unchanged if the parameter λ is changed (c.f. chapter 3).

We will finish this thesis with a survey of subjects concerning test design that have been paid attention to. The numbers in parentheses refer to the methods that were involved when these subjects were included in simulated item selection problems. (Number 3 refers to the method described in chapter 3, and number 4 refers to the method described in chapter 4.)

1) Specification of target information points
   This is expressed by linear constraints (3,4)

2) Minimize the number of items in a test
   This is expressed by a linear objective function (3,4)

3) Specification of the (minimum or maximum) number of items
   This is expressed by a linear constraint (3,4)

4) Design of a representative test
   This is expressed by a quadratic objective function (4)

5) Minimize the matching of consecutive tests

This is expressed by a linear objective function (3)

6) Specification of logical conditions concerning pairs of items

A quadratic objective function has been suggested. The method of chapter 4 has to be applied when this function is included.

7) Improve the calibration of the item bank

A quadratic objective function has been suggested. The method of chapter 4 has to be applied when this function is included.

References

[1] Y. Boomsma, *Item selection by mathematical programming*, Arnhem, 1986, Cito special bulletin nr. 47.

[2] Y.M.I. Dirickx, S.M. Baas, B. Dorhout, *Operationele Research,* Enschede, 1986.

[3] G. Hadley, *Nonlinear and dynamic programming*, Reading etc., 1964.

[4] H.W.Kuhn and A.W. Tucker, *Nonlinear programming*, *Proceedings on the second Berkeley symposium on mathematical statistics and probability*, Berkeley, 1951.

[5] F.M. Lord, *Applications of item response thery to practical testing problems*, Hillsdale, 1980.

[6] A.F Razoux Schultz jr., *Item selection using heuristics*, Enschede, 1987.

[7] T.J.J.M. Theunissen, "Binary programming and test design", Psychometrika, 50 (1985), 4, p. 411-420.

[8] H.M. Wagner, *Principles of Operations Research*, *With applications to managerial decisions*, second edition, Englewood cliffs, 1975.

[9] P.L. Yu, "Cone convexity, cone extreme points and nondominated solutions in decision problems with multiobjectives", Center for system science nr. 72-2, Rochester, 1972.

[10] M. Zeleny, *Linear multiobjective programming*, Berlin etc., 1974, Lecture notes in economics and mathematical systems nr. 95.

Appendix A Simulated itembank of chapter 3

Number of items in the bank: 200

Number of target information points: 3

Item number	Item information for θ =			Item difficulty parameter b_j
	-1.000	0.000	1.000	
1	0.130784	0.059110	0.023589	-2.698028
2	0.143329	0.066508	0.026843	-2.561758
3	0.159519	0.076809	0.031508	-2.391304
4	0.160441	0.077425	0.031792	-2.381704
5	0.168996	0.083312	0.034538	-2.292748
6	0.186350	0.096372	0.040841	-2.110865
7	0.189541	0.098970	0.042133	-2.076775
8	0.194231	0.102922	0.044122	-2.026075
9	0.206585	0.114234	0.049992	-1.887444
10	0.211112	0.118784	0.052432	-1.833999
11	0.212039	0.119748	0.052956	-1.822821
12	0.217559	0.125748	0.056263	-1.754293
13	0.226124	0.136165	0.062227	-1.638966
14	0.227346	0.137793	0.063186	-1.621294
15	0.227452	0.137936	0.063271	-1.619744
16	0.230011	0.141497	0.065399	-1.581362
17	0.231932	0.144317	0.067111	-1.551216
18	0.232022	0.144452	0.067194	-1.549774
19	0.232327	0.144915	0.067478	-1.544845
20	0.232642	0.145395	0.067773	-1.539737
21	0.235639	0.150203	0.070769	-1.488860
22	0.236328	0.151374	0.071511	-1.476525
23	0.236591	0.151829	0.071800	-1.471746
24	0.237874	0.154111	0.073263	-1.447800
25	0.239164	0.156524	0.074831	-1.422558
26	0.240344	0.158853	0.076366	-1.398247
27	0.240610	0.159398	0.076729	-1.392563
28	0.241185	0.160599	0.077531	-1.380054
29	0.241934	0.162220	0.078623	-1.363187
30	0.242105	0.162601	0.078882	-1.359225
31	0.242172	0.162749	0.078983	-1.357678
32	0.242803	0.164193	0.079968	-1.342663
33	0.243234	0.165216	0.080671	-1.332034
34	0.244366	0.168056	0.082649	-1.302516
35	0.245054	0.169916	0.083964	-1.283187
36	0.245148	0.170179	0.084152	-1.280451
37	0.245523	0.171256	0.084922	-1.269250
38	0.245534	0.171286	0.084944	-1.268931
39	0.247782	0.178952	0.090599	-1.188925
40	0.248253	0.180984	0.092152	-1.167589
41	0.248280	0.181110	0.092249	-1.166267
42	0.249084	0.185374	0.095591	-1.121225
43	0.249867	0.192382	0.101344	-1.046163
44	0.249969	0.198629	0.106786	-0.977706
45	0.249953	0.199091	0.107202	-0.972564

46	0.249953	0.199103	0.107212	−0.972436
47	0.249667	0.203141	0.110932	−0.927011
48	0.249560	0.204101	0.111840	−0.916066
49	0.249402	0.205317	0.113003	−0.902115
50	0.249240	0.206398	0.114051	−0.889619
51	0.248058	0.212004	0.119711	−0.823253
52	0.247959	0.212372	0.120097	−0.818789
53	0.247914	0.212536	0.120270	−0.816795
54	0.247473	0.214052	0.121882	−0.798246
55	0.246396	0.217212	0.125357	−0.758711
56	0.244852	0.220887	0.129613	−0.711016
57	0.244659	0.221296	0.130103	−0.705573
58	0.244464	0.221702	0.130593	−0.700143
59	0.244072	0.222489	0.131551	−0.689544
60	0.243399	0.223765	0.133135	−0.672099
61	0.243120	0.224269	0.133771	−0.665115
62	0.241582	0.226831	0.137102	−0.628786
63	0.240204	0.228863	0.139874	−0.598815
64	0.238924	0.230575	0.142310	−0.572652
65	0.237987	0.231736	0.144024	−0.554344
66	0.236971	0.232917	0.145820	−0.535225
67	0.236338	0.233616	0.146911	−0.523649
68	0.235415	0.234587	0.148466	−0.507192
69	0.234567	0.235435	0.149861	−0.492457
70	0.232480	0.237354	0.153173	−0.457634
71	0.228781	0.240263	0.158690	−0.399950
72	0.228721	0.240305	0.158775	−0.399055
73	0.227921	0.240861	0.159917	−0.387154
74	0.227807	0.240938	0.160079	−0.385471
75	0.226505	0.241789	0.161900	−0.366511
76	0.226238	0.241956	0.162270	−0.362670
77	0.226231	0.241961	0.162279	−0.362572
78	0.225458	0.242432	0.163338	−0.351562
79	0.224804	0.242816	0.164224	−0.342344
80	0.223500	0.243542	0.165964	−0.324256
81	0.216859	0.246528	0.174371	−0.236800
82	0.216312	0.246726	0.175034	−0.229892
83	0.215762	0.246918	0.175695	−0.222991
84	0.215701	0.246939	0.175768	−0.222225
85	0.215632	0.246962	0.175851	−0.221363
86	0.214602	0.247301	0.177078	−0.208542
87	0.213041	0.247772	0.178911	−0.189353
88	0.210837	0.248353	0.181449	−0.162699
89	0.208865	0.248792	0.183672	−0.139253
90	0.207373	0.249076	0.185327	−0.121727
91	0.203107	0.249671	0.189931	−0.072594
92	0.202708	0.249711	0.190352	−0.068066
93	0.202167	0.249760	0.190922	−0.061937
94	0.201448	0.249819	0.191673	−0.053829
95	0.201239	0.249834	0.191891	−0.051473
96	0.193828	0.249942	0.199363	0.030465
97	0.190063	0.249684	0.202982	0.071179
98	0.189007	0.249575	0.203976	0.082502
99	0.186813	0.249300	0.206014	0.105929
100	0.184746	0.248981	0.207899	0.127882

101	0.180146	0.248065	0.211975	0.176396
102	0.180055	0.248044	0.212053	0.177348
103	0.178756	0.247734	0.213174	0.190974
104	0.176932	0.247262	0.214725	0.210071
105	0.176208	0.247063	0.215333	0.217629
106	0.175548	0.246876	0.215884	0.224524
107	0.175530	0.246870	0.215900	0.224716
108	0.174592	0.246594	0.216676	0.234497
109	0.174519	0.246572	0.216737	0.235264
110	0.174445	0.246550	0.216798	0.236032
111	0.172738	0.246016	0.218192	0.253815
112	0.172285	0.245869	0.218559	0.258534
113	0.170950	0.245418	0.219629	0.272425
114	0.169993	0.245082	0.220388	0.282381
115	0.166996	0.243955	0.222716	0.313535
116	0.166565	0.243784	0.223044	0.318016
117	0.164290	0.242844	0.224755	0.341658
118	0.161730	0.241711	0.226628	0.368284
119	0.160288	0.241038	0.227659	0.383294
120	0.157612	0.239723	0.229525	0.411193
121	0.156771	0.239293	0.230098	0.419974
122	0.152402	0.236919	0.232975	0.465723
123	0.152181	0.236793	0.233116	0.468052
124	0.151553	0.236432	0.233514	0.474643
125	0.146288	0.233219	0.236701	0.530258
126	0.144102	0.231789	0.237943	0.553510
127	0.143848	0.231619	0.238084	0.556222
128	0.143074	0.231097	0.238510	0.564477
129	0.139056	0.228273	0.240619	0.607630
130	0.138662	0.227986	0.240817	0.611889
131	0.137487	0.227119	0.241395	0.624604
132	0.137250	0.226942	0.241510	0.627177
133	0.135439	0.225568	0.242366	0.646881
134	0.129863	0.221096	0.244754	0.708237
135	0.127368	0.218976	0.245697	0.736078
136	0.126484	0.218206	0.246011	0.746006
137	0.125931	0.217720	0.246202	0.752235
138	0.125186	0.217060	0.246453	0.760646
139	0.120666	0.212911	0.247809	0.812228
140	0.118153	0.210500	0.248433	0.841340
141	0.116908	0.209276	0.248707	0.855890
142	0.115456	0.207825	0.248994	0.872976
143	0.114872	0.207235	0.249100	0.879882
144	0.113651	0.205986	0.249304	0.894386
145	0.113558	0.205891	0.249319	0.895488
146	0.112746	0.205050	0.249439	0.905192
147	0.112242	0.204523	0.249508	0.911238
148	0.110271	0.202436	0.249736	0.935017
149	0.107784	0.199734	0.249925	0.965393
150	0.107181	0.199068	0.249954	0.972821

151	0.106442	0.198245	0.249980	0.981961
152	0.097793	0.188106	0.249470	1.092139
153	0.097783	0.188093	0.249469	1.092278
154	0.096848	0.186941	0.249318	1.104570
155	0.089907	0.178036	0.247553	1.198522
156	0.089625	0.177661	0.247456	1.202441
157	0.089104	0.176966	0.247271	1.209708
158	0.088952	0.176763	0.247216	1.211835
159	0.088507	0.176165	0.247051	1.218086
160	0.086150	0.172957	0.246087	1.251540
161	0.085548	0.172125	0.245816	1.260205
162	0.085186	0.171622	0.245648	1.265433
163	0.085021	0.171393	0.245570	1.267817
164	0.084129	0.170148	0.245137	1.280772
165	0.080749	0.165329	0.243281	1.330859
166	0.078342	0.161804	0.241745	1.367510
167	0.078083	0.161421	0.241569	1.371502
168	0.078072	0.161404	0.241561	1.371676
169	0.077441	0.160465	0.241122	1.381456
170	0.077362	0.160347	0.241066	1.382685
171	0.076344	0.158819	0.240327	1.398604
172	0.075799	0.157996	0.239917	1.407188
173	0.073344	0.154235	0.237943	1.446496
174	0.072135	0.152354	0.236892	1.466230
175	0.071950	0.152065	0.236726	1.469270
176	0.069732	0.148554	0.234641	1.506261
177	0.065959	0.142425	0.230654	1.571421
178	0.064428	0.139881	0.228868	1.598737
179	0.063356	0.138079	0.227557	1.618197
180	0.060919	0.133924	0.224389	1.663443
181	0.057665	0.128243	0.219728	1.726284
182	0.055218	0.123869	0.215877	1.775573
183	0.048587	0.111574	0.203821	1.919270
184	0.047712	0.109902	0.202040	1.939500
185	0.047478	0.109453	0.201556	1.944961
186	0.046703	0.107960	0.199929	1.963223
187	0.043185	0.101068	0.192055	2.049703
188	0.041897	0.098498	0.188968	2.082927
189	0.041423	0.097546	0.187802	2.095387
190	0.039003	0.092628	0.181601	2.161101
191	0.038259	0.091097	0.179607	2.182049
192	0.038119	0.090807	0.179226	2.186047
193	0.037758	0.090062	0.178241	2.196368
194	0.035201	0.084715	0.170967	2.272252
195	0.032494	0.078941	0.162689	2.358310
196	0.025335	0.063102	0.137650	2.622842
197	0.024279	0.060692	0.133534	2.667721
198	0.016850	0.043219	0.101136	3.048831
199	0.015603	0.040194	0.095059	3.128319
200	0.005954	0.015856	0.040810	4.111701

Target information:

3.500 4.000 3.500

Appendix B Simulated item bank of chapter 4

The items marked with a (*) are selected in the simulation results of chapter 4.

Number of items: 50
Number of target information points: 2

Item number	Item information for θ = -1.000	Item information for θ = 1.000	Item difficulty parameter b_j	
1	0.247959	0.120097	-0.818789	
2	0.113558	0.249319	0.895488	
3	0.226231	0.162279	-0.362572	(*)
4	0.248253	0.092152	-1.167589	
5	0.242105	0.078882	-1.359225	
6	0.078342	0.241745	1.367510	
7	0.080749	0.243281	1.330859	
8	0.088507	0.247051	1.218086	
9	0.047712	0.202040	1.939500	
10	0.184746	0.207899	0.127882	(*)
11	0.085548	0.245816	1.260205	
12	0.249867	0.101344	-1.046163	
13	0.089104	0.247271	1.209708	
14	0.114872	0.249100	0.879882	(*)
15	0.107181	0.249954	0.972821	
16	0.249560	0.111840	-0.916066	
17	0.138662	0.240817	0.611889	(*)
18	0.160441	0.031792	-2.381704	
19	0.113651	0.249304	0.894386	
20	0.038259	0.179607	2.182049	
21	0.236591	0.071800	-1.471746	
22	0.086150	0.246087	1.251540	
23	0.143329	0.026843	-2.561758	
24	0.201239	0.191891	-0.051473	(*)
25	0.157612	0.229525	0.411193	(*)
26	0.243399	0.133135	-0.672099	
27	0.156771	0.230098	0.419974	
28	0.137487	0.241395	0.624604	
29	0.244852	0.129613	-0.711016	
30	0.069732	0.234641	1.506261	
31	0.175530	0.215900	0.224716	(*)
32	0.212039	0.052956	-1.822821	
33	0.072135	0.236892	1.466230	
34	0.215701	0.175768	-0.222225	(*)
35	0.146288	0.236701	0.530258	
36	0.085186	0.245648	1.265433	
37	0.228781	0.158690	-0.399950	
38	0.216312	0.175034	-0.229892	
39	0.215632	0.175851	-0.221363	(*)
40	0.178756	0.213174	0.190974	(*)

41	0.180146	0.211975	0.176396	(*)
42	0.227921	0.159917	-0.387154	
43	0.152181	0.233116	0.468052	
44	0.215762	0.175695	-0.222991	(*)
45	0.245523	0.084922	-1.269250	
46	0.249969	0.106786	-0.977706	
47	0.224804	0.164224	-0.342344	(*)
48	0.228721	0.158775	-0.399055	(*)
49	0.238924	0.142310	-0.572652	
50	0.242803	0.079968	-1.342663	

Target information:

1.600 1.700