Using SAS/OR® for Automated Test Assembly from IRT-Based Item Banks
Yung-chen Hsu, GED Testing Service, LLC, Washington, DC
Tsung-hsun Tsai, Research League, LLC, Matawan, NJ

ABSTRACT
In recent years, advanced development in psychometric theory and computer technology has led to dramatic changes in test construction practices. In testing organizations, the task of assembling test forms can be considered as an assignment to solve a mathematical programming problem, in which a certain objective function needs to be met subject to specific practical constraints, such as test length and content coverage. The SAS/OR® is a powerful modeling environment for solving mathematical optimization problems. Large or mid-sized testing organizations can greatly benefit from its available solvers to handle real test assembly problems and needs. We used procedure OPTMODEL to present the programming formulation of a test assembly problem. The merit of the PROC OPTMODEL is that the models in the SAS® statements are declared in a form which mimics the symbolic formulation of an optimization model. A generic test assembly model with specific statistical constraints based on item response theory and content related test specifications was shown as an example. The model can be easily modified to include additional practical constraints to obtain an optimal solution. Testing organizations can use this approach to automatically create test forms if relatively well developed item banks are available.

INTRODUCTION
A test assembly problem is to select a set of items from a large pool of pre-calibrated items, known as an item bank, based on the test specifications. An item bank is a repository of test items, essentially a database, which stores all information pertaining to the items such as item format, item characteristics and content domains. Optimization techniques seek optimal solutions under specific constraints of psychometric and test composition specifications within a given item bank. The techniques have been a major advancement for testing organizations to achieve automated test assembly. The early automated test assembly method using a mathematical programming approach was not based on the psychometric test theory (Feuerman and Weiss, 1973). Due to the advanced innovations in psychometric theory and computer technology, current test construction practices use mathematical programming techniques are generally based on item response theory (IRT). IRT is a modern psychometric test theory that describes the relationship between item characteristics and test taker abilities. The three-parameter logistic model (3PL) is one of the widely used unidimensional IRT models for dichotomous responses in various large-scale testing programs. The model can be expressed as

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

representing the probability of answering a particular dichotomously scored item correctly given the proficiency level \( \theta \). The parameters \( a_i, b_i, \) and \( c_i \) are the characteristics of item \( i \) and the common choice of the scaling constant \( D \) is 1.7. Generally, the item parameters can be estimated by using PROC NLMIXED (Sheu, Chen, Su, & Wang, 2005). The item information function, which is derived from Fisher information (Load, 1980, Suen 1990, van der Linden & Boekkooi-Timminga, 1989), is defined as

\[ I_i(\theta) = \frac{(P_i(\theta))^2}{P_i(\theta)Q_i(\theta)} \]

where \( Q_i = 1 - P_i \) and \( P_i'(\theta) = \frac{\partial P_i(\theta)}{\partial \theta} \). The test information of a test including \( n \) items is defined as

\[ I(\theta) = \sum_{i=1}^{n} I_i(\theta) \]
MATHEMATICAL APPROACHES OF CONSTRUCTING TEST FORMS

Thenissen (1985) first presented a binary integer programming approach to construct a test with a target information function. The objective of the model is to minimize the number of items in the test subject to the constraints that the information in the test is above the pre-specified target at a number of ability levels. Several practical constraints were considered to incorporate into the modeling approach (Thenissen, 1986; Baker, Cohen, & Barmish, 1988; and de Gruijter, 1990). A different perspective, the so-called Maximin model, that considers the specification of a relative target information function, was formulated by van der Linden & Boekkooi-Timminga (1989) in selecting items from an item bank. The model can be interpreted as specifying the relative shape of the target information function at certain ability points. The automated test assembly problem can be treated as an optimization of matching a target test information function subject to content coverage. Specifying an absolute target test information function may not be easy in practice if there is no available reference. The implementation of Maximin model for automated test generation targets a given shape of the IRT test information function is especially useful for new testing programs. The nature of the test can be more easily specified than assigning exact target information values. Actually, the formulation of Maximin model can be treated as a special case of the goal programming model approach (Hsu, 1993), a branch of multi-objective optimization. Most variations (e.g., Boekkooi-Timminga, 1990; Hsu, 1993; van der Linden & Adema, 1998) developed later for solving practical test assembly issues are generally based on Maximin model for optimal test construction.

Let $r_k$ be the relative information values at the ability point $\theta_k$ and assume that the items in the item bank are represented by decision variable $x_i$, $i = 1, \ldots, N$, denoting whether the items are to be included into the test form ($x_i = 1$) or not ($x_i = 0$). The model is formulated as

$$\text{maximize} \quad y$$

subject to

$$\sum_{i=1}^{N} I_i(\theta_k) x_i \geq r_k y, k = 1, \ldots, K$$

$$\sum_{i=1}^{N} x_i = n$$

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

$$y \geq 0$$

$I_i(\theta_k)$ denotes the information function of item $I_i$ at the ability point $\theta_k$ and $n$ is the test length. As an example, the second constraint represents the number of items in the test. The model allows additional practical constraints, such as test composition (e.g., cognitive levels, mutually exclusive items) and administration time, to be taken into account and specified into the model.

AN EXAMPLE TEST ASSEMBLY PROBLEM

We simulated a set of item parameters to create an item bank that has four content domains and each domain contains 50 items. For simplicity, we set the parameter $c$ to zero. In practice, the $c$ parameter has very small variation because items with large value of $c$ will not be included in the item bank. Generally, parameter $c$ has little impact in test assembly.

ITEM INFORMATION

Maximin model is used as an illustrative example. We computed the item information at 13 ability points ($\theta = (-3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3)$).

```r
libname nesugdat 'c:\nesug2012\dat';
%let D=1.7;
%let D2=%sysevalf(&D*&D);
data inf;
set nesugdat.itembank;
array ri(13) r1-r13;
a2=a*a;
```
do i=1 to 13;
   p=1/(1+exp(-&D*a*((i-7)/2.0)-b));
   q=1-p;
   ri(i)=&D2*a2*p*q;
end;
keep r1-r13;
run;
proc transpose data=inf out=infc prefix=px; run;
data infc; set infc (drop=_name_); run;

PROBLEM FORMULATION IN PROC OPTMODEL

The task is to compose a classification test of 40 items and the test is divided into four equal sections, with items sequenced in position 1-10, 11-20, 21-30, and 31-40 for the four domains, respectively. We assumed that the classification test has multiple cut points, which means the test information curve would have two peaks. The test assembly problem is formulated as

\[
\text{maximize } y
\]

subject to

\[
\sum_{i=1}^{200} I_i(\theta_k) x_i \geq r_k y, k = 1, \ldots, 13 \\
\sum_{i=1}^{50} x_i = 10 \\
\sum_{i=51}^{100} x_i = 10 \\
\sum_{i=101}^{150} x_i = 10 \\
\sum_{i=151}^{200} x_i = 10 \\
x_i \in \{0,1\}, i = 1, \ldots, 200 \\
y \geq 0
\]

The following code shows the use of the procedure OPTMODEL for the test assembly tasks: (1) a selective test with a single cut-off point; (2) a classification test multiple with cut-off points; and (3) a diagnostic test with no cut-off point. The relative information values at different ability points are in Table 1. We specified two cut-off points for the classification test.

<table>
<thead>
<tr>
<th>(\theta)</th>
<th>-3</th>
<th>-2.5</th>
<th>-2</th>
<th>-1.5</th>
<th>-1</th>
<th>-0.5</th>
<th>0</th>
<th>0.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selective</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.7</td>
<td>1.0</td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Classification</td>
<td>0.1</td>
<td>0.1</td>
<td>0.7</td>
<td>1.0</td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.7</td>
<td>1.0</td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

SAS/OR® provides a convenient modeling language within PROC OPTMODEL for formulating, solving, and maintaining optimization models. We start the problem statement with PROC OPTMODEL. Because we are dealing a huge amount of variables, we use set statement to group the numbers for indexing the variables. Then we declare the decision variables in the model. The code uses the selective test as an example. Once the problem is solved, the value represents whether an item is selected or not. The objective function is simple. The constraints relate the decision variables with the four domains. The solve statement invokes an appropriate solver to solve this mixed integer linear programming problem. The results are stored for further processing.
proc optmodel;
  set theta=1..13;
  set niBank=1..200;
  num iInf{theta,niBank};
read data infc into [j=_N_] {i in niBank} <iInf[j,i]=col("px"||i)>;
  num r1(theta)=[0.1 0.1 0.1 0.1 0.1 0.7 1.0 0.7 0.1 0.1 0.1 0.1];

var x{niBank} BINARY, y;
max obj=y;
  con tInf{j in theta}: sum{i in niBank}iInf[j,i]*x[i]>=r1[j]*y;
  con ca1: sum{i in 1..50}x[i]=10;
  con ca2: sum{i in 51..100}x[i]=10;
  con ca3: sum{i in 101..150}x[i]=10;
  con ca4: sum{i in 151..200}x[i]=10;

solve with milp;
  create data nesugdat.testset3 from [id]={niBank} sel=x;
quit;

Figure 1 shows the distribution of the values for the $\log(a_i)$ and $b_i$ of the items in the item bank. Figures 2, 3, and 4 are the test information for the three tests, respectively. The test information has one peak for the selective test and two peaks for the classification. The diagnostic test has flat test information.

FIGURE 1. Scatter plot of $\log(a_i)$ and $b_i$.

FIGURE 2. Selective test.

FIGURE 3. Classification test.

FIGURE 4. Diagnostic test.
CONCLUSION

For testing organizations, the test assembly task is to solve a constrained combinatorial optimization problem. If relatively well developed items banks have been developed, the problem involves a large amount of variables. PROC OPTMODEL has a succinct way to read and create data sets. It provides a powerful modeling language to formulate and solve the optimization model. This procedure can interface to various optimization solvers to compute solutions to the formulated problems. We showed how to formulate a simple test assembly problem. The model can be easily inspected and modified to address a wide variety of test specifications.

REFERENCES


Hsu, Y.-C. (1993). The goal programming approach for test construction (Master thesis). The University of Arizona, Tucson, AZ.


ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yung-chchen Hsu
GED Testing Service, LLC
1155 Connecticut Ave., NW 4th Floor
Washington, DC 20036
Work Phone: 202-471-2214
Email: yung-chchen.hsu@GEDtestingservice.com
Web: www.gedtestingservice.com