2009

Three Applications of Automated Test Assembly within a User-Friendly Modeling Environment

Ken Cor

Cecilia Alves

Mark. J. Gierl

Follow this and additional works at: https://scholarworks.umass.edu/pare

Practical Assessment, Research & Evaluation

Three Applications of Automated Test Assembly within a User-Friendly Modeling Environment

Ken Cor, *Stanford University*
Cecilia Alves, *University of Alberta*
Mark Gierl, University *of Alberta*

While linear programming is a common tool in business and industry, there have not been many applications in educational assessment and only a handful of individuals have been actively involved in conducting psychometric research in this area. Perhaps this is due, at least in part, to the complexity of existing software packages. This article presents three applications of linear programming to automate test assembly using an add-in to Microsoft Excel 2007. These increasingly complex examples permit the reader to readily see and manipulate the programming objectives and constraints within a familiar modeling environment. A spreadsheet used in this demonstration is available for downloading.

Advances in educational measurement and technology continue to afford psychometricians new ways to deal with complex measurement problems. For example, measurement researchers (e.g. Armstrong, Jones, Kunce, 1998; Luecht, 1998; van der Linden, 1998) have recently outlined mathematical procedures[1] that, when used with generic optimization software such as ILOG OPL-CPLEX and LINGO, can automate the process of selecting questions from large item banks to construct parallel tests. Parallel tests are important in large-scale testing programs because they are equivalent in terms of both content and statistical properties and are considered interchangeable which leads to many practical benefits. Automated test assembly (ATA) is the most efficient and effective way to construct parallel tests. Unfortunately, however, existing software packages remain largely inaccessible to the wider psychometric community because of their complicated and unfamiliar modeling platforms. That is, in order to model ATA problems using available optimization software, psychometricians often require additional training in unfamiliar programming languages which might be acting as a deterrent to their widespread use.

In a recent review of the utility of conducting ATA in Microsoft Excel 2007 with a premium solver upgrade, Cor, Alves, and Gierl (2008) conclude that Excel offers a user-friendly modeling context that, combined with its popularity, could be used to bring ATA to a wider psychometric audience. The following paper builds on the Cor et al (2008) review by providing an in-depth description of how to model and solve three increasingly complex test assembly problems using Microsoft Excel 2007 and the Premium Solver Platform upgrade.

AUTOMATIC TEST ASSEMBLY: AN OVERVIEW

In Item Response Theory, the measurement precision of a test is characterized by its test information function. Test information functions indicate the strength of a test at each point on an ability scale (Davey & Pitoniak, 2006). ATA is possible is because of the property of conditional item independence (see Lord, 1980). Due to conditional item independence, the total information provided by a test at any one point on the ability scale can be defined as the sum of the information at that point provided by the individual items included in the test. This property allows assembly problems to be modeled as systems of linear equations. Once modeled, optimization algorithms can be employed to efficiently search the solution space of possible combinations of test items that serve to optimally satisfy both psychometric and content specifications.

Test assembly models are specified using a system of equations that define decision variables, an objective

function, and constraints associated with the problem. Decision variables are values that optimization algorithms change in order to find the solution to a given problem. In test assembly, the decision variables are defined as 1's and 0's to indicate whether an item is included (1) or excluded (0) in the final form. Mathematically, decision variables in test assembly problems are expressed as follows:

$$x_{it} = \begin{cases} 1 \text{ if item } i \text{ is included in test form t} \\ 0 \text{ if item } i \text{ is not included in test form t} \end{cases} \quad (1)$$

The objective function is the equation that fluctuates when decision variables change. Different test assembly problems have different objective functions. For example, in some testing situations the goal is to generate parallel tests that maximize the information at certain points on the ability scale. In these instances, the objective function can be mathematically represented as follows:

$$\text{Maximize: } \sum_{t}\sum_{k}\sum_{i} I_i(\theta_{kt})x_{it} \quad (2)$$

In words, Equation 2 states that the sum of the information, I, across all items, i, across each specified level of ability, θ_k, across each test, t, multiplied by the decision variable x_{it} should be maximized.

The next types of equations in the model are the test assembly constraints. In general, constraints define the boundaries that restrict how high, low, or close to a specified value an objective function can get. In test assembly problems, there are two types of constraints - item level and test level. Item level constraints are used to restrict or force the use of items that are not wanted or required on the final form. For example, the test developer, using a test bank made up of items that have been calibrated using a 3 parameter item response model, may want to restrict the selection of items to those with guessing parameters, c_i, less than 0.30. This item level constraint is represented mathematically as in Equation 3:

$$x_{it}\left(c_{it}\right) \le 0.30 \text{ for all items } i \text{ on all test forms } t \quad (3)$$

Other item level constraints include item life/overlap, item difficulty, reading level, or any item attribute that is discretely associated with items in the test bank.

Test level constraints are used to specify the attributes of a test. These types of constraints

can be psychometric or content related. For example, a test developer may wish to include exactly 9 items that belong to content category C_1 on each test form. This content related constraint is represented mathematically in the following way:

$$\sum_{i \in V_{C_1}} x_{it} = 9 \quad (4)$$

In words, Equation 4 states that the sum of the items that are a subset of content category C_1 must equal nine.

This concludes our brief overview of how to define systems of equations required to set up mathematical models to solve ATA problems. For a detailed description of the various types of objective functions and constraints that can be included in test assembly models, readers are referred to van der Linden (2005). We now turn to demonstrating how to represent and solve test assembly problems using Excel. In order to provide readers with a sense for how these problems can be set up in Excel, demonstrations with three increasingly large and diverse test assembly problems are described.

THREE DEMONSTRATIONS

Example 1: Small Scale Simultaneous Test Assembly

This first problem demonstrates how Excel can be used to model simple test assembly problems that, although small, were once considered not solvable in Excel (see van der Linden, 2005). In this scenario, the goal of the test assembly problem is to simultaneously produce three parallel tests containing ten items that are maximally informative and parallel at three points on the ability scale. For this problem, a set of 55 previously simulated items from an introductory course in item response theory were used[2]. The means of the a, b, and c parameters for these simulated items were 0.91, -0.14, and 0.20 respectively. Figure 1 shows the item information curves for each of the 55 items.

Figure 1 shows that, even in this small test assembly problem, trying to select three sets of ten items that result in statistically parallel tests (tests with overlapping information functions) is a difficult prospect.
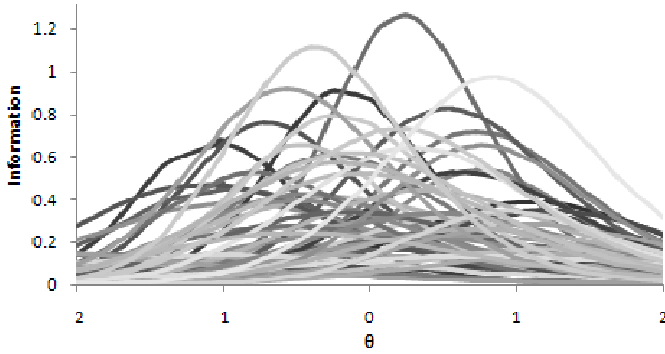
Figure 1. Item Information Curves for the 55 Items in Example 1

Figure 2 shows how the item bank was represented in Excel. The bank contains the a, b, and c parameters, the content categorization of each item, and the 3

set up. For full mathematical definitions of each problem, readers are referred to Appendix A.

Figure 3 shows how the unsolved model is represented in Excel. Each part of the model is described in turn. First, readers are referred to the table titled "Item level Decision Variables" in the middle left portion of the spreadsheet. This part of the model is used to specify the decision variables and the item level constraints for the problem. The blank cells in this table represent the item decision variable matrix (*item* x *test form*). These cells are the values that the solver algorithm change in order to search for an optimal solution. In the model, these cells are constrained to be binary so that a '1' indicates an item has been chosen for the associated test and a '0' means it was not.

The problem states that items are to be used only once across the three tests. In order to meet this overlap

	D	E	F	G	H	I	J	K	CS	CZ	DG
						Content Categories				Information 3PL; $I_i(\theta)$	
14	Item	a_{pram}	b_{pram}	c_{pram}	A	B	C	D	-1	0	1
15	1	1.3570	-1.2360	0.3480	1	0	0	0	0.67079	0.17562	0.01969
16	2	1.1140	0.6230	0.1150	0	1	0	0	0.03847	0.41533	0.66306
17	3	1.2650	-0.8380	0.2200	0	0	1	0	0.66707	0.42309	0.06641
18	4	0.8800	0.9080	0.1940	0	0	0	1	0.02126	0.16725	0.38393
19	5	0.9500	-1.1200	0.1920	1	0	0	0	0.45066	0.24715	0.06401
20	6	0.7970	-1.0420	0.0930	0	1	0	0	0.38233	0.25652	0.09212
21	7	0.9110	0.8200	0.3070	0	0	1	0	0.01438	0.13594	0.33073
22	8	1.1960	0.5530	0.3350	0	0	0	1	0.01212	0.25049	0.49587
23	9	0.5710	-0.1350	0.1870	1	0	0	0	0.1126	0.16382	0.13502
24	10	1.4220	-0.3080	0.2450	0	1	0	0	0.25469	0.86284	0.17018
25	11	1.4950	0.1470	0.1290	0	0	1	0	0.08117	1.1441	0.51078
26	12	0.6980	-0.2360	0.2960	0	0	0	1	0.11723	0.19859	0.14133
27	13	0.4490	-0.5300	0.2700	1	0	0	0	0.07427	0.08651	0.071
28	14	0.8270	-0.6530	0.2960	0	1	0	0	0.22128	0.25377	0.10982
29	15	0.6410	0.2090	0.1620	0	0	1	0	0.10331	0.20406	0.19443
30	16	1.0230	0.4810	0.1880	0	0	0	1	0.04651	0.3612	0.46842
31	17	1.0830	-0.3820	0.1720	1	0	0	0	0.33573	0.57281	0.18693
32	18	0.8420	-0.3540	0.1130	0	1	0	0	0.28762	0.39921	0.19674
33	19	0.9080	-1.2530	0.1170	0	0	1	0	0.46927	0.22829	0.06091
34	20	0.7980	0.0770	0.1870	0	0	0	1	0.12662	0.30891	0.24557
35	21	0.7760	-0.7990	0.2360	1	0	0	0	0.24977	0.23545	0.10144

Item Bank / Model

Figure 2. Item Banking in Excel

parameter logistic information at the specified levels of ability[3]. In this example, items were simulated to be mutually exclusive in terms of content[4]. For example an item that measured content category A, has a 1 in the content column for A and 0's in the remaining columns. We now turn to describing how the model for this problem is represented in Excel (shown in Figure 3). For the purposes of the paper we limit our explanations to verbal descriptions of the spreadsheets and how they are

constraint the column directly to the left of the item decision variable matrix (starting at cell E15), which represents the sum of the decision variables across the three tests, is constrained to be less than or equal to one. For example, if item one was assigned to test one, it could not be assigned to tests two or three because it would violate the constraint that the sum across the three forms must be less than or equal to one. We now turn to the content specifications for this problem.
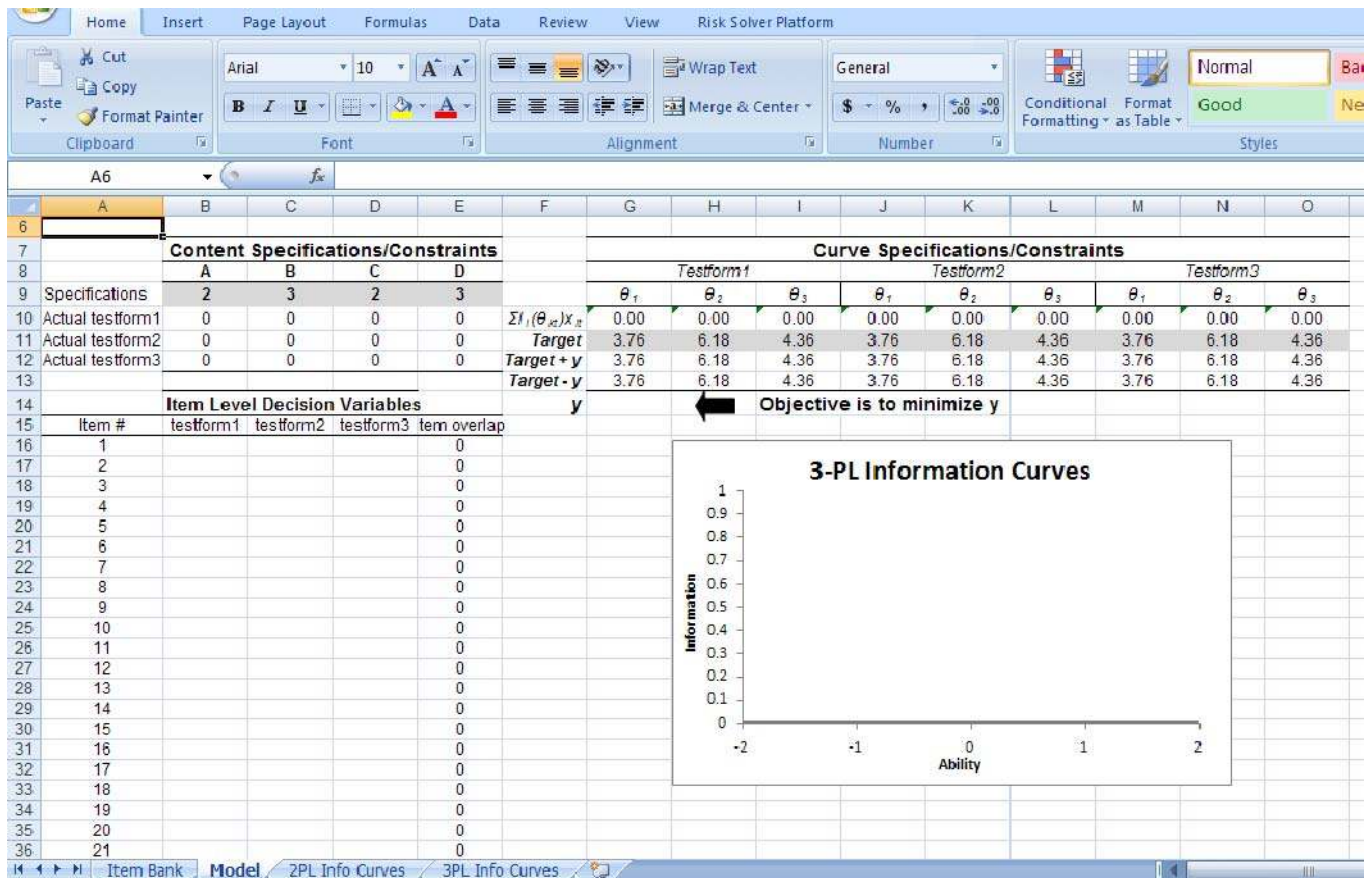
Figure 3. Defining a Simple Simultaneous Test Assembly Problem in Excel

The content specifications state that each test must contain two items from content category A, three items from content category B, two items from content category C, and three items from content category D. In order to build this specification into the model, the required values are directly input into the grey cells in the content specification table at the top left quadrant of the spreadsheet. The number of items measuring each content category on each test is then calculated in the rows below the entered specifications. These cells are defined to equal the sum of the product of the decision variables assigned to each item for each test and its category classification from the item bank (also a binary variable). For example, if in the solution to the problem, items three and seven are indicated as being included in test one and these items are the only items on the test measuring content category A, the value in cell B10 would equal two. In the model, the values in the cells in these rows (one row per test) must be less than or equal to the values in the grey cells entered above[5].

We now turn to the problem of specifying the objective function for this problem. In order to make the test maximally informative at the specified levels of ability, the objective function was formulated to minimize the difference between the resulting test information curves and an absolute target information curve (for a detailed description of this type of formulation see van der Linden, 2005, p. 109). The absolute targets for this problem are calculated by considering the best three tests that the item bank could hypothetically produce. When unconstrained, the most parallel three tests this item bank can produce will evenly share the total information available at each level of ability. For example, if the total information available at $\theta_1 = -1$ is 11.29, the hypothetical target for the three test forms is $11.29/3 = 3.76$. For this problem, three arbitrary levels of ability are considered, $\theta_1 = -1$, $\theta_2 = 0$, and $\theta_3 = 1$. The same procedure is used to determine the absolute targets for $\theta_2 = 0$ and $\theta_3 = 1$. The targets are entered in the grey cells of the Curve Specification/Constraint table in the upper right quadrant of the spreadsheet shown in Figure 2. Each test has the same absolute targets.

In order to force the solver algorithm to search for the solution that minimizes the difference between the absolute targets and the actual information included in
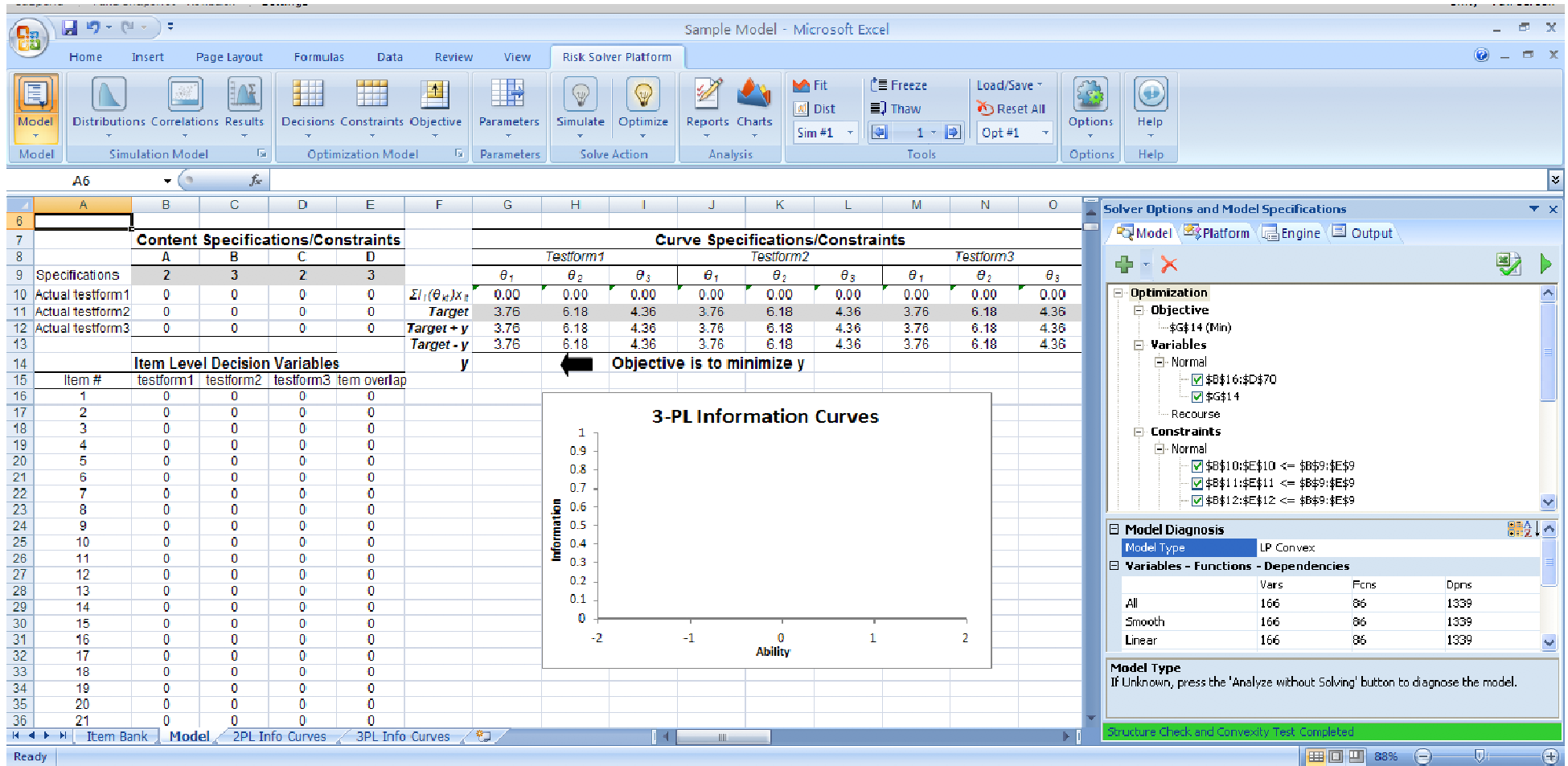
Cor, Alves & Gierl, Automated Test Assembly

Figure 4. Specifying a Simultaneous Test Assembly Problem in Solver

the test, a new variable, 'y', that represents the absolute maximum difference between the targets and the actual information is defined in cell G14. This cell becomes both the objective and a variable that is free to change when solving the problem. In order to find a solution to the problem that minimizes the value of y, three types of cells are defined. First, the actual amount of information for each test at each level of ability based on a given solution is calculated in cells G9:O9. These cells calculate the sum of the product of the decision variables for each item on each test and the information that each item provides at the specified level of ability. For example, if a given solution indicated that items one through ten are to be included in test one, the actual amount of information at $\theta_1 = -1$ for test one is simply the sum of the information provided by each of these items at this level of ability (recall that the information at various levels of ability is calculated in the item bank).

Next, the upper bounds on the difference between the target and the actual information for each level of ability on each test are defined. These values are calculated as the specified target plus 'y' (shown in cells G11:O11). Finally, the lower bounds on the difference between the target and the actual information for each level of ability on each test are defined. These values are calculated as the specified target minus 'y'(shown in cells G12:O12). In order to ensure the solution produces a value of 'y' that is within the calculated upper and lower bounds, constraints indicating that the actual information from the calculated total information (cells G9:O9) is less than the upper bounds and greater than the lower bounds of each individual target are added. A description of how each of the constraints and variables described above are specified in the Solver interface is provided next.

Figure 4 shows the solver interface as it appears in the latest version of Excel 2007. The solver interface described in this paper is the most up to date version of solver that can be installed and used with Excel 2007[6]. This platform differs from the factory installed version of solver and the version of Solver evaluated by Cor et al. (2008) but requires the same basic specification strategy.

For the purposes of this demonstration we focus on the Solver Options and Model Specifications menu shown on the far right of the spreadsheet in Figure 4. Specifying a model in solver involves filling in the different branches of the modelling tree under the Model tab in the Solver Options and Model Specifications menu on the right hand side of the sheet.

For example, by selecting the branch titled Objective and then clicking the add button at the top left of the menu, users can select the cells they want to be specified as the objective. Once selected users are prompted to specify whether the objective is to be maximized, minimized, set to a value of zero, or set to a specific value. In this problem, the objective was specified to minimize the value in cell G14.

The same basic procedure is used to specify all of the components of the model. To specify the decision variables, the Variables branch is selected and the appropriate cells are chosen from the spreadsheet. In this example, the variables are located in the *item* x *test form* matrix as well as in cell G14. Next, users begin to define the constraints. For example, in order to add content constraints to the model, cells B10:E10 must be constrained to be less than or equal to cells B9:E9. After defining all the constraints described in the model (item overlap constraints and target bounds) users turn to specifying variable bounds. That is, recall that the item decision variables must be bound to be integers. In order to facilitate this constraint, the Model tab includes an Integer branch where users add the cells in the *item* x *test form* matrix. The last thing that must be constrained for this problem is the range for 'y'. That is, 'y' represents the maximum absolute difference between the target and the actual information calculated for each test. As a result, 'y' cannot be less than zero. Users specify this constraint in the Bound branch under the Model tab[7]. This concludes the description of all information that must be entered into the Solver interface in order to solve the problem.

After fully defining the assembly problem in the Solver Options and Model Specifications interface, users select an engine to solve the optimization problem. Without getting too technical, most ATA problems can be solved using the Large-scale LP engine which is suitable for Mixed Integer Programming problems with an unlimited number of variables and unlimited constraints. In cases where the Large-scale LP engine is taking a substantial amount of time to solve the problem, the Gurobi engine can be used to solve all test assembly problems. However, there is a substantial cost difference between the two engines so.

In order to start the solution of the problem, users click on the play button at the top right of the Solver Options and Model Specifications interface. The display automatically switches to the Output tab and begins to show solution progress. Once a solution has been found or the iteration or time limits have been reached, users
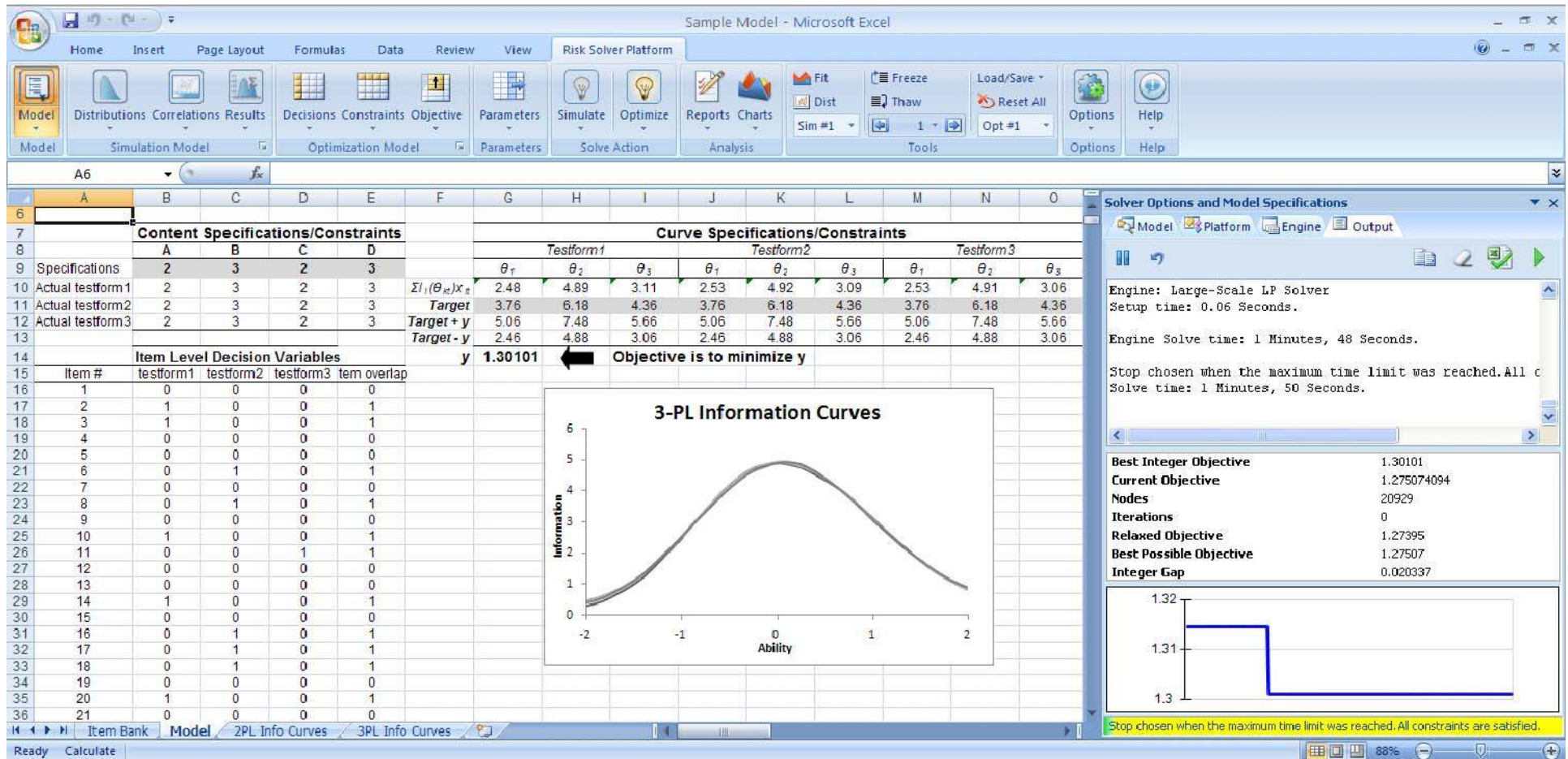
Cor, Alves & Gierl, Automated Test Assembly

Figure 4. Solution to Simple Simultaneous Test Assembly Problem

are presented with a window stating the status of the solution. Given a successful solution with all constraints being met, users select 'ok' and are directed back to the sheet. Figure 4 shows the solution to this problem.

The Output tab (shown in the far right of Figure 4) displays the solution time and other solution characteristics for the problem. Using the Large-Scale LP engine on a computer with a 2.0 GHz processor and 1.0 GB of RAM, a satisfactory solution[8] to the test assembly problem was found in 1 minute and 48 seconds that satisfied all the problem constraints. The items selected for each test are indicated in the *item* x *test form* matrix. Using these values and the data in the item bank, the item information curves for each form are automatically generated in the graph shown in the middle of the spreadsheet. Based on the overlap of these curves, the tests generated appear statistically parallel. Finally, we see that the maximum difference between the target and the actual total information for each test at each ability level was 1.301. We now turn to an example that is based on real item data. This concludes the first demonstration.

Example 2 - Observed-Score Pre-Equating using ATA

The first test assembly demonstration, although sophisticated, was rather small by test assembly

meaning from year to year. One way to limit the amount of equating that is required is to try to construct successive tests so that they have overlapping test characteristic curves. Test characteristic curves are calculated as the linear combination of the item characteristic curves for any given test. In a similar way that we were able to use ATA to model linear combinations of item information, the linear combinations of item characteristic curves can be used as a basis for the model.

In this example, the goal of the test assembly problem is to create a new version of a criterion referenced achievement test that has the same test characteristic curve as the original form. In order to set this problem up, a bank consisting of 168 science nine achievement test items administered to approximately 40,000 Canadian students from 1995 to 1999 was assembled. Using BILOG, the 3-PL parameters for each item were estimated so that all parameters were placed on the 1995 scale[9]. Figure 5 shows the item bank data for this problem. The only difference between the bank shown in example one and the current bank is that instead of showing the calculations for item information, this bank shows the calculation of the 3-PL probabilities. The model defining this problem is defined next.

	Year	Item #	a	b	c	C1	C2	C3	C4	C5	C6	-3.4	-3.2	-3	-2.8	-2.6	-2.4
			IRT Parameters			Content Categories						3-PL Probabilities					
5	1995	SC95_01	0.724	-0.042	0.174	1	0	0	0	0	0	0.1870	0.1906	0.1951	0.2008	0.2080	0.2170
6	1995	SC95_02	0.377	-1.094	0.041	1	0	0	0	0	0	0.2191	0.2385	0.2593	0.2817	0.3055	0.3308
7	1995	SC95_03	0.443	-0.063	0.064	1	0	0	0	0	0	0.1341	0.1446	0.1564	0.1697	0.1847	0.2014
8	1995	SC95_04	0.458	-2.236	0.09	1	0	0	0	0	0	0.3519	0.3818	0.4135	0.4467	0.4810	0.5160
9	1995	SC95_05	0.583	-0.821	0.266	1	0	0	0	0	0	0.3189	0.3295	0.3419	0.3565	0.3735	0.3929
10	1995	SC95_06	0.342	0.686	0.284	1	0	0	0	0	0	0.3449	0.3517	0.3592	0.3674	0.3763	0.3861
11	1995	SC95_09	0.57	0.235	0.142	1	0	0	0	0	0	0.1666	0.1717	0.1778	0.1850	0.1937	0.2040
12	1995	SC95_11	0.463	0.168	0.123	1	0	0	0	0	0	0.1729	0.1808	0.1899	0.2003	0.2122	0.2256
13	1995	SC95_54	0.528	-1.435	0.029	1	0	0	0	0	0	0.1711	0.1943	0.2203	0.2494	0.2815	0.3165
14	1995	SC95_58	0.437	0.916	0.124	1	0	0	0	0	0	0.1581	0.1633	0.1693	0.1761	0.1839	0.1927
15	1996	SC96_05	0.347	-1.622	0.059	1	0	0	0	0	0	0.3031	0.3251	0.3481	0.3723	0.3974	0.4234
16	1996	SC96_06	0.729	-0.926	0.083	1	0	0	0	0	0	0.1238	0.1347	0.1482	0.1649	0.1853	0.2101
17	1997	SC97_18	0.729	-2.113	0.081	1	0	0	0	0	0	0.2360	0.2706	0.3106	0.3559	0.4059	0.4596
18	1997	SC97_19	0.635	-2.398	0.046	1	0	0	0	0	0	0.2875	0.3285	0.3732	0.4211	0.4712	0.5225
19	1997	SC97_20	0.645	-1.395	0.03	1	0	0	0	0	0	0.1269	0.1478	0.1724	0.2012	0.2343	0.2719
20	1997	SC97_24	0.442	-1.122	0.065	1	0	0	0	0	0	0.2080	0.2272	0.2483	0.2715	0.2967	0.3238
21	1998	SC98_05	0.595	0.41	0.146	1	0	0	0	0	0	0.1637	0.1676	0.1723	0.1780	0.1848	0.1930
22	1998	SC98_06	0.809	-0.109	0.305	1	0	0	0	0	0	0.3124	0.3148	0.3178	0.3218	0.3269	0.3335

Figure 5. Pre-equating Item Bank

standards. That is, the model had only 166 variables and 86 constraints. In this second example, a larger more real world example is demonstrated.

In many large-scale testing programs, equating procedures are used to place successive year's test scores on the same scale so that reported scores have the same

In the present problem, a test is to be constructed that contains the same number of items from the six content categories as was included in a pre-specified reference test. Further, in order to facilitate future linking, 23 of the items on the new form must be common to both the reference test and the new test.

Finally, the new test must minimize the difference between the observed cut scores of the reference test (θ_{accep} = -0.80 and θ_{excel} = 0.80) and the new test. Figure 6 shows the unsolved model for this problem. For the sake of brevity, only new aspects of the model will be described in detail for this test assembly problem. Once again, readers are referred to Appendix A for a full mathematical representation of the model.

Similarities between this model and the model described in example one include the way in which the item decision variables, the content specifications, and the objective function have been formulated. That is, the same formulas that were used to specify these components in the previous example, with one exception, are also used in this model. The exception is the values calculated in cells J3:K3. That is, rather than calculating the total information for each test at each ability level as was done in example one, the observed cut scores at the indicated levels of ability are calculated instead (cells J3:K3). Observed cut scores are calculated as the sum of the product of the calculated 3-PL probabilities and the decision variables describing whether each item is included in the test for each of the two cut scores (θ_{accep} = -0.80 and θ_{excel} = 0.80). The cut score targets are obtained directly from the test characteristic curve of the reference form. The graph in the lower right quadrant of the spreadsheet shows the target test characteristic curve based on the items included in the reference test. The target cut-score values based on this curve are T_{accep} = 28 and T_{excel} = 44, respectively. The new features of the present model are described next.

First, the model includes the specification of the reference form data starting in cell B16. These values function in a similar way as item decision variables do in that they represent whether or not an item is included in the reference test. They differ in that, for this problem, these values are fixed and are only used as a basis for calculating item overlap constraints. Next, the total item constraint is specified in cell B9. This value is calculated as the sum of the item decision variables for the new form. It is included in the model so that the total items in the new form can be constrained to the specified value of 55. Finally, sets of cells constraining the problem to have 23 items that overlap between the new and old form are included in the model.

Specifically, a new set of overlap decision variables, z_i, are defined (starting at cell D16) to represent whether each item is included in one, $z_i = 0$, or both of the tests, z_i

= 1. In other words, the Solver algorithm is free to change these values in order to find the best combination of overlapping items to minimize 'y'. To ensure that exactly 23 items overlap, the sum of the overlap variables (z_i) is calculated in cell B12 and is constrained to equal the specified value in cell B11. Next, to facilitate the proper selection of overlap decision variables, these variables are constrained to be greater than or equal to sum of the item level decision variables across the two forms minus one (starting at cell F16). Finally, the sum of the item variables (starting at cell E16) are constrained to be less than or equal to two times the overlap variable for each item (starting at cell G16). Readers are referred to van der Linden (2005, p. 144) for a complete explanation of how these sets of constraints function to control for item overlap.

The last step to solve this problem is to specify the model in the Solver Options and Model Specification tab on the far right of Figure 6. The same procedures are used to define the decision variables, constraints, bounds, and integer constraints for this problem as were used in example 1. Figure 7 shows the solution to the observed score pre-equating problem. Using the Large Scale LP engine on a computer with a 2.0 GHz processor and 1 GB of RAM, the solution time for this problem was 2 minutes and 55 seconds. The problem involved 337 variables and 348 constraints. The graph of the test characteristic curves in the bottom right quadrant of the spreadsheet of Figure 7 shows how the new test characteristic curve almost perfectly parallels the reference form. Further, there is almost no difference between the new test cut scores and the specified targets. Finally, all constraints were satisfied. We now turn to example 3.

Example 3: Large-Scale Balanced Incomplete Block Design

Example three constitutes the largest and most advanced problem of the three demonstrated in this set. In this example, the goal of the test assembly problem is to generate 26 parallel forms in a balanced incomplete block design (BIB). An item bank consisting of 613 grade 4 mathematics items from the Brazilian National Basic Education Assessment System (Sistema Nacional de Avaliação da Educação Básica - SAEB) is used to solve this problem. The bank includes the 3-PL item parameters along with content categorization based on 4 mutually exclusive themes and 28 mutually exclusive descriptor categories. Figure 8 shows how the bank is represented in Excel.
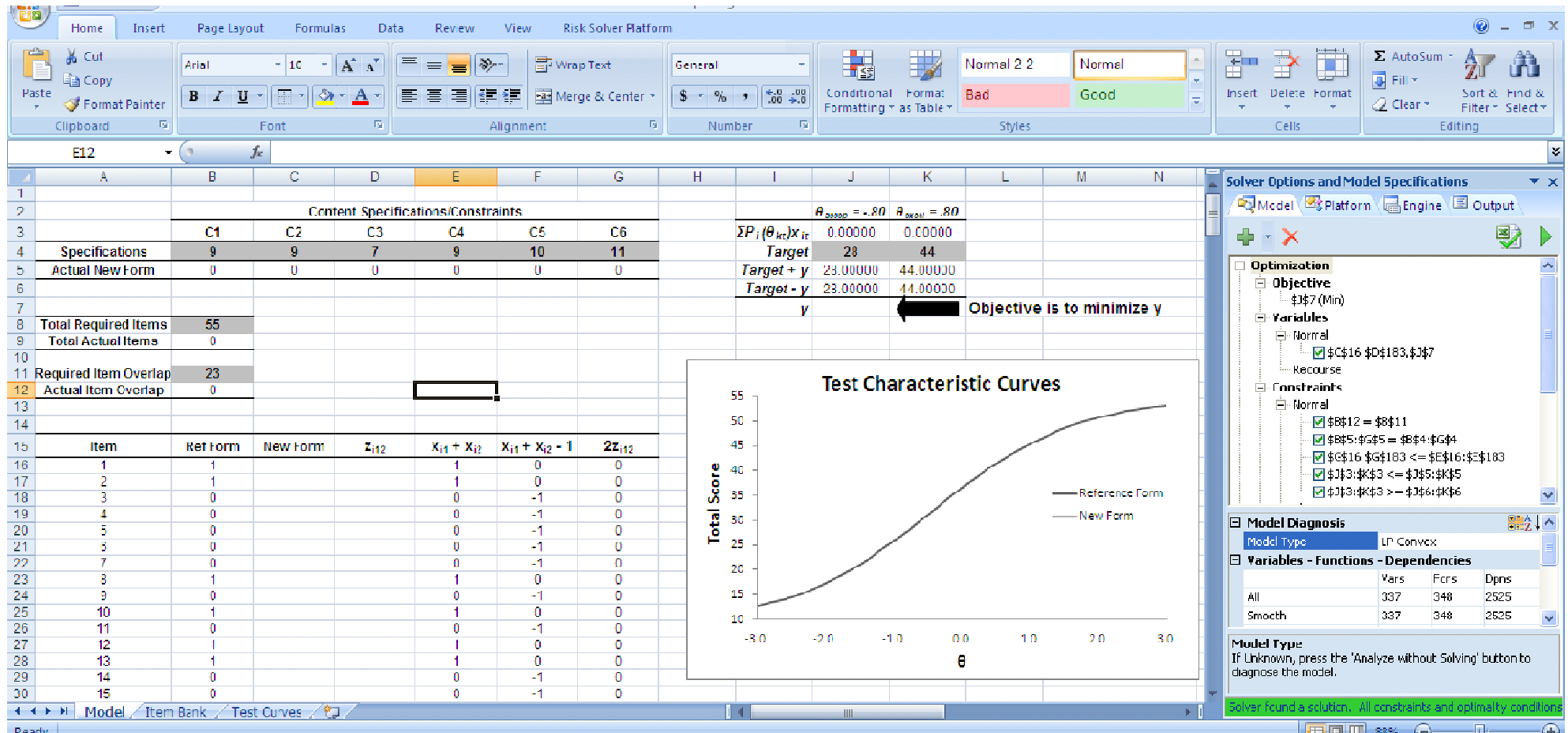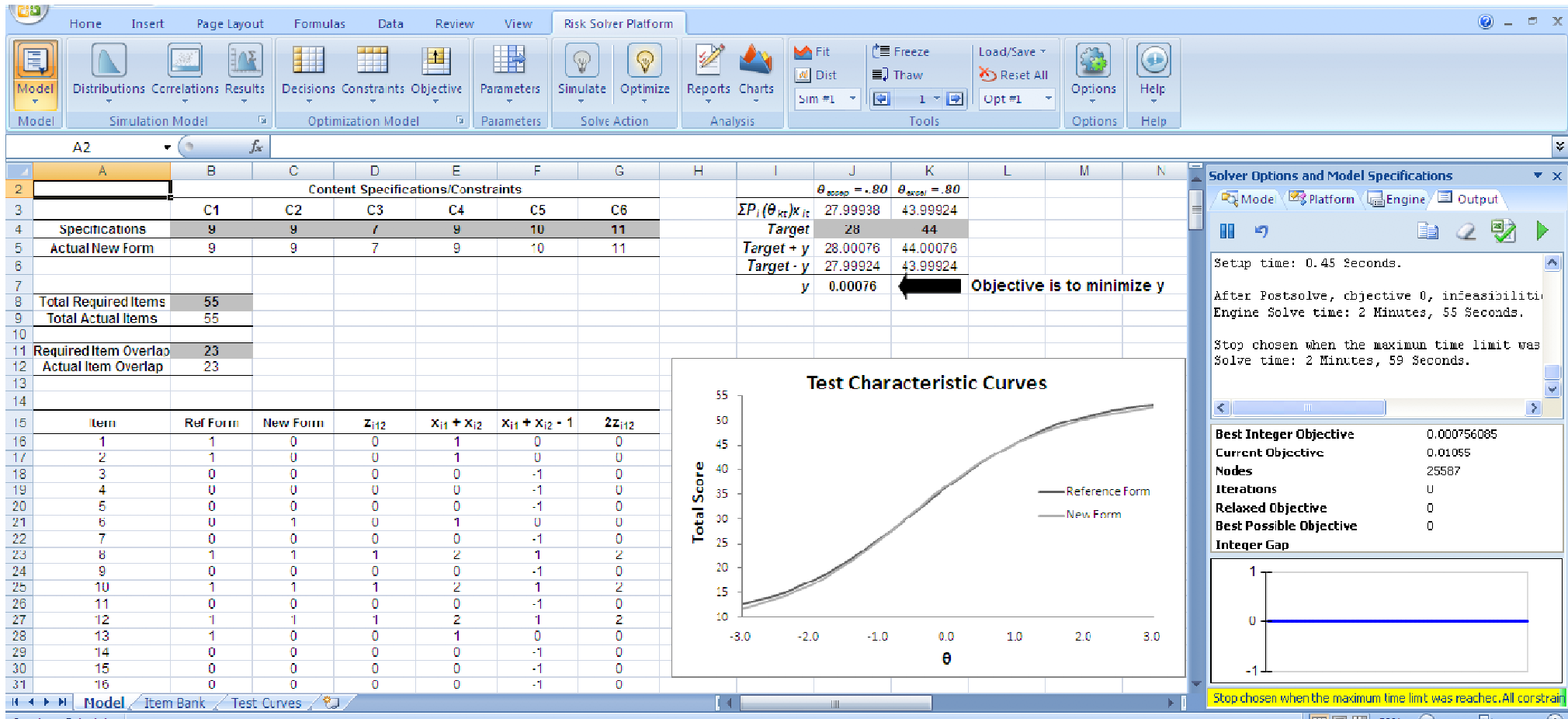
Cor, Alves & Gierl, Automated Test Assembly

Figure 6. Observed Score Pre-equating Model

Cor, Alves & Gierl, Automated Test Assembly

Figure 7. Solution to Observed Score Pre-Equating Model

Figure 8. Large-Scale BIB Item Bank

The bank includes the item number, the 3PL IRT parameters, four content themes, and 28 content categories (Figure 8 displays only 4 of the 28 descriptors). Also included in the bank is the item information function specified at multiple points on the theta scale (the figure displays only four of these points: −2.0, −1.84, 1.84, 2.0).

The BIB requirement necessitates two stages of optimization in order to solve this problem. In the first stage, 13 parallel blocks consisting of 13 unique items are assembled. In stage two the 13 blocks are organized into 26 parallel forms in a BIB design. The model for the first stage of the problem is presented in Figure 9 with the complete mathematical representation presented in Appendix A.

Once again, this model contains the usual suspects, the decision variables (represented in an *item×test block* matrix in cells B27 to N637), constraints, and an objective function (cell O23). The unique features of this problem include more complex content constraints and a new way of formulating the objective function. Each of these new features are discussed in turn. First, the content constraints in the problem are modelled at two levels. At the broadest level, each block must contain a specific number of items that measure each curriculum theme. These constraints are formulated in the same way as was described in the two previous examples (cells B5:F18). At a more specific level, there

is a requirement that when taken together, the test blocks must have sampled the specified number of items for each descriptor (descriptors are nested within themes). In order to accommodate this constraint more relaxed content constraints were employed. That is, in order to facilitate a solution the total number of items measuring each descriptor (cells F20:AG20) was allowed to vary within plus or minus one of the specified requirements (cells F5:AG5).

The other unique aspect of this model is how the objective function was formulated. In this example the goal has no specific target. That is, the goal is to maximize the information provided at the hypothetical cut score of $\theta = 1$. This formulation is much simpler than the previous examples and involves pushing the sum of the information provided across all test blocks at $\theta = 1$ (calculated in cell O23) to its maximum. Although this formulation is simpler, it can be problematic because it does not necessitate a parallel solution. That is, the original formulation can result in an uneven spread of the information across the test blocks. In order to spread the information out evenly across the blocks, a second round of optimization is required. In the second round of optimization, the total information for each test block (cells B23:N23) is constrained to be less than or equal to the average information produced for each block in the first round of optimization. For this problem, the first optimization run resulted in

Figure 9. Stage 1 of Large-Scale BIB Model
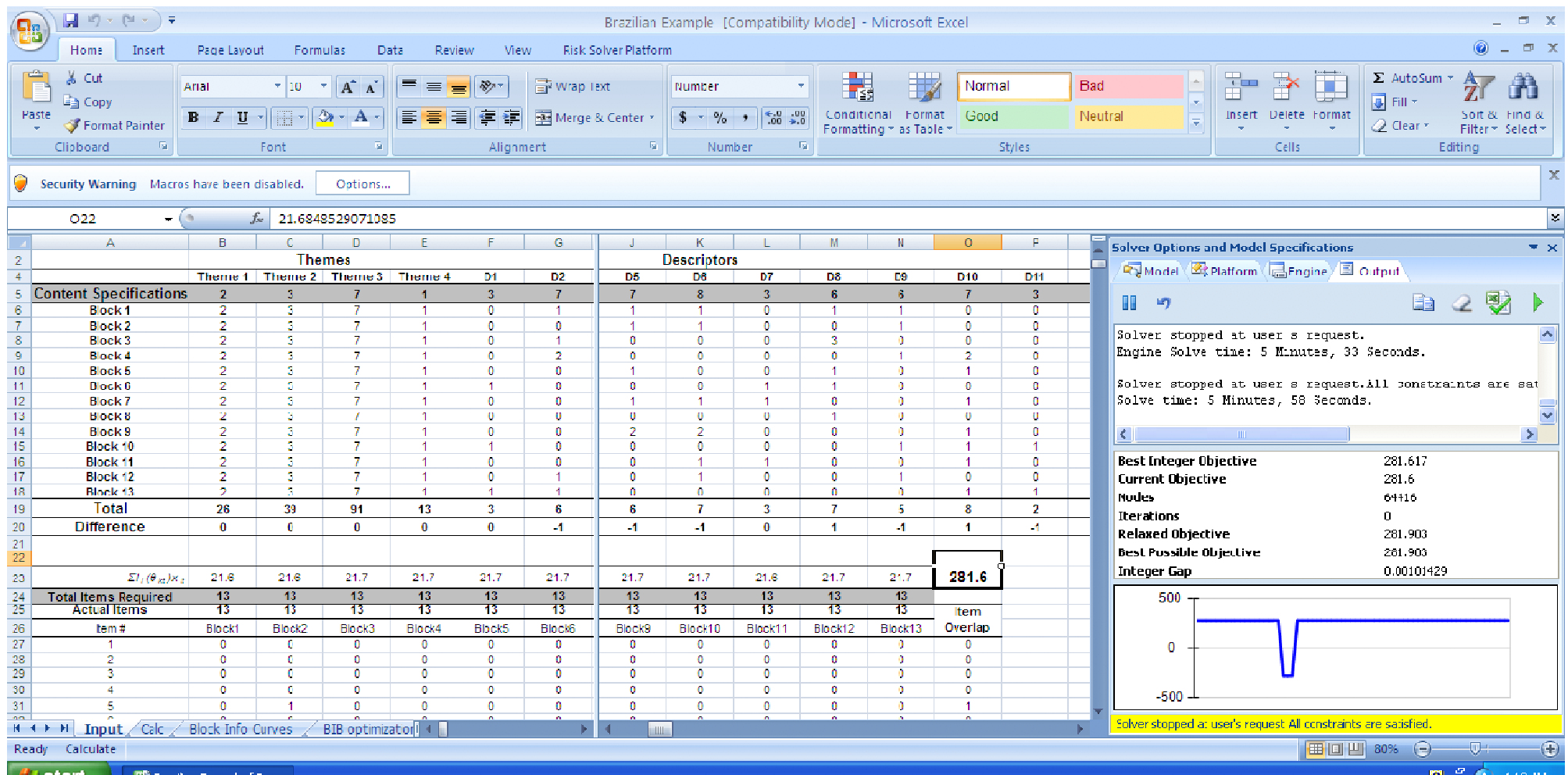
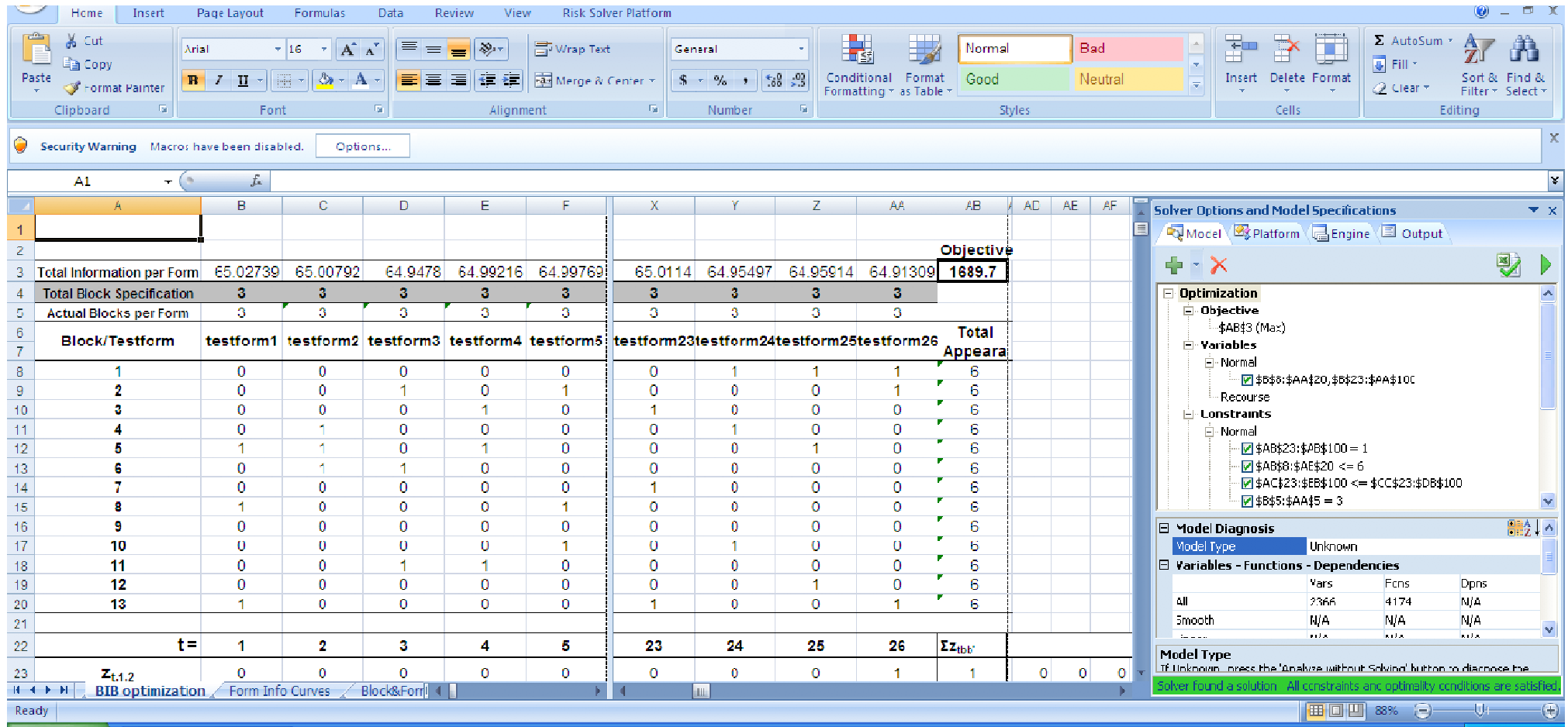Cor, Alves & Gierl, Automated Test Assembly

Figure 10. Stage I of the Large-Scale BIB solution

Page 15

Cor, Alves & Gierl, Automated Test Assembly

Figure 11. Model and Solution for Stage II of the Large-Scale BIB Problem

Cor, Alves & Gierl, Automated Test Assembly

average information of 21.7 for each block of items. After solving the problem a second time with the new constraint that no one block could have more than 21.7 units of information, a more parallel solution was found.

Figure 10 shows the solution to the problem. Due to the large size of this model (7969 variables and 2108 constraints), the Gurobi Solver engine was used to solve the problem. Using a computer with a 2.0 GHz processor and 1.0 GB of RAM, the first round of optimization took 24.69 seconds. The second round of optimization (with the added constraint that test blocks could not exceed 21.7 units of information) was stopped after 5 minutes and 58 seconds because a strong enough solution had been reached. All constraints were satisfied and the solution produced 13 blocks that provided total information ranging between 21.6 and 21.7 units of information.

In stage 2 of the problem, blocks are combined into 26 parallel forms subject to the following two constraints. First, each final form must contain exactly three blocks of items. Second, each block must appear a total of 6 times across the forms. To model these constraints, the same principles used to set up the overlap constraints in example two are applied. That is, test blocks are treated as items that are constrained in terms of how many times they can overlap across the final forms. Figure 11 shows how stage two of the problem is modeled in Excel as well as the associated solution. Because this model does not introduce any new modeling components, the modeling sheet is not explained. Once again, readers are referred to Appendix A for a complete mathematical definition of the problem. Also, a detailed description of how to define overlap constraints for the blocks can be found in van der Linden (2005, p. 152)

Once again, due to the complexity of this problem (2366 variables and 4174 constraints), the Gurobi Solver engine was used. A solution time of 18.02 seconds was indicated on a computer with a 2.0 GHz processor and 1GB of RAM. All constraints were satisfied and the best possible solution is reported. As a point of comparison, Figure 13 shows the information curves that resulted from the actual manual assembly process used to construct forms along with the information curves produced based on the automated test assembly model.

Figure 12 shows that the tests constructed based on the automated model are more parallel and more uniform across the ability scale that the tests created manually. It should be noted, however, that the

automated test assembly solution shown in Figure 13 does not include all constraints that would have been required to produce the manual test assembly solution on the left. That being said, more constraints would not be expected to change the nature of the comparison dramatically. The test information curves would be expected to provide less overall information but would still be highly statistically parallel (the curves would still overlap) in comparison to the manually generated forms.
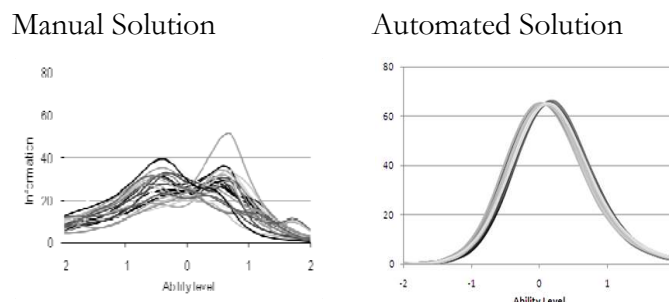

Figure 12. Comparing the Manual and Automated Test Assembly Solutions.

SUMMARY

The above demonstration served as an introduction to how to model and solve test assembly problems using Microsoft Excel 2007 and the newly updated Solver platform. We have taken readers through increasingly large and complex test assembly problems and have shown that ATA in Excel is feasible. In our experience, learning to model test assembly problems in Excel is not a problem for psychometricians already familiar with the speadsheet program. Further, the actual program interface for the Solver add-in is user friendly and self explanatory. It is concievable that after some intitial instruction, users would be able to set up basic test assembly spreadsheets in a mater of a couple hours. Once basic ATA sheets are created, users can continually modify sheets to become increasingly more sophisticated in terms of the number of and types of constraints they contain.

Having solved the three increasingly large test assembly problems, it is now possible to comment on the factors that affect solution times. Solution times for the three examples varied from 18.2 seconds to 5 minutes and 58 seconds. The primary factor affecting solution time was the nature of the constraints and objective function modeled in each test assembly problem. Specifically, absolute constraints and targets lead to longer solution times than more open targets. For example, in the BIB example, when the objective

Cor, Alves & Gierl, Automated Test Assembly

was to merely maximize the information at a single point on the ability scale (part one of stage one and stage two), solution times were fast (24 and 18 seconds respectively).   Alternatively, with multiple absolute targets (example 1, 2, and step two of stage one of example 3), solution times were much longer (1 min 48 sec, 2 min 55 sec, and 5 min 58 sec respectively). Even though there is substantial variation in solution time. In comparison to manual test assembly procedures, which, depending on the program, require days or even weeks to generate multiple forms, ATA in Excel is extremely less time intensive.[10]

In general, the above discussion paints modeling in Excel as a straightforward and easy to learn process. There are, however, some limitations. For example, with more sophisticated constraints, modelling in a spreadsheet becomes unwieldy and inefficient. In these situations a more flexible modeling language becomes more appealing. Also, cost might be a concern for some users[11]. For example, small business users may find the cost too high to justify. Alternatively, for users involved in large testing programs interested in implementing ATA, cost reductions that will result from increased efficiency in test construction are expected to more than offset any initial investment. At the end of the day, we hope that because Excel is so widley used, demonstrating its utility in solving all types of test assembly problems will, at the very least, help to bring ATA and all of its benefits to the wider psychometric community.

REFERENCES

Armstrong, R. D., & Jones, D. H. (1998). IRT test assembly using network-flow programming. *Applied Psychological Measurement, 22*(3), 237-247.

Cor, K., Alves, C., & Gierl, M. J. (2008). Computer Software Review: Conducting Automated Test Assembly Using the Premium Solver Platform Version 7.0 With Microsoft Excel and the Large-Scale LP/QP Solver Engine Add-In. *Applied Psychological Measurement, 32*(8), 652-663.

Davey, T., Pitoniak, M.J., (2006). Designing Computerized Adaptive Tests. In S. M. Downing and T. M. Haladyna (Eds.), *Handbook of Test Development* (pp. 543-574). Routledge.

Luecht, R. M. (1998). Computer-assisted test assembly using optimization heuristics. *Applied Psychological Measurement, 22*(3), 224-236.

van der Linden, W. J. (1998). Optimal assembly of psychological and educational tests. *Applied Psychological Measurement, 22*(3), 195-211.

van der Linden, W. J. (2005). Linear models for optimal test design. New York: Springer-Verlag

Notes

1. For a complete overview and step-by-step guide on how to model various test assembly problems, readers are referred to Wim van der Linden's (2005) book entitled *"Linear models for optimal test design"*.

2. The simulated items were not generated by the authors of this paper and as a result were not in any way designed to facilitate a tractable solution to the present test assembly problem.

3. Due to a lack of readability of the screenshot that would have shown the full 55-item bank, only a portion of it can be shown in the figure.

4. Item content can be categorized in more complicated ways. For example, in the third demonstration, content categorization is done at two hierarchical levels. This leads to dependencies that do not cause problems for ATA models.

5. An exact constraint is not used for each category because this constrains the problem more than it needs to be. That is, the solution will always push the less than or equal to constraint to its maximum in order to produce the best possible solution. For a complete explanation, readers are referred to van der Linden, 2005 p. 56-57.

6. Readers interested in trying out the problem described in Example 1 can download the Excel file used to model the problem from the PARE site http://pareonline.net/sup/v14n14/ata.xlsx. A 30-day trial version of the Solver Add-in components required to run the sheet can be downloaded from http://solver.com/dwnxls.php.

7. The Bound branch is not viewable in Figure 3. In the actual file, this branch appears in sequence following the Constraints branch in the Model tree.

8. The solution provided to this problem is not the optimal solution. That is, the solver had reached a maximum time constraint set for the engine and automatically prompted us to choose if we would like to stop. Given that the best possible solution (shown in the bottom of the output tab) was 1.28, and the current solution of 1.301, a decision was made to stop the search because the solution provided was deemed strong enough.

9. Parameter equating is made possible because of the linking structure employed in the original administration.

10. Solution times will also vary depending on which Solver engine is used. For the most part, more expensive engines can be expected to result in faster solution times.

11. The total onetime cost of the configuration used to solve each of the examples in this demonstration is $9325. Academic discounts are available for professors and students. Further, the configuration used in this paper is more sophisticated than is required for most test assembly problems. Users are encouraged to contact Solver support to determine the configuration that best suits their specific optimization needs. For example, the problems in this paper could have been solved with a configuration costing $6490. This configuration would have lead to slightly slower solution times. For a complete price list visit http://www.solver.com/pricexls.php

Citation

Cor, Ken, Alves, Cecilia., & Gierl, Mark. J. (2009). Three Applications of Automated Test Assembly within a User-Friendly Modeling Environment. *Practical Assessment, Research & Evaluation*, 14(14). Available online: http://pareonline.net/getvn.asp?v=14&n=14.

Corresponding Author

Ken Cor
Stanford University
Palo Alto, CA

Email: kcor [at] Stanford.edu

Cor, Alves & Gierl, Automated Test Assembly

APPENDIX A

Mathematical Model for Example 1

The decision variable for this problem (and for the remaining problems in this investigation) is represented by equation A1.

$$x_{it} = \left\{ \begin{matrix} 1 \\ 0 \end{matrix} \right. \tag{A1}$$

The decision variable is constrained so that it can only take the form of a one or a zero. Within the excel framework this constraint can be accomplished by using equation A2.

$$x_{it} = \text{binary}; \text{i.e.} \, x_{it} \left\{ \begin{matrix} 1 \text{ if item } i \text{ is included in test form t} \\ 0 \text{ if item } i \text{ is not included in test form t} \end{matrix} \right. \tag{A2}$$

The following *multiple test level constraints* (as generalized by equation A3) ensure that each item, i, can only be used for one test, t, i.e. no item overlap. This general expression results in 55 constraints.

$$\sum_{t=1}^{3} x_{it} \leq 1 \quad \text{for all } i \tag{A3}$$

There are 12 *content constraints* (4 categories x 3 forms) that are used to restrict the total number of items in each content category, V_c, to the corresponding specification, C_{Spec}, for each test (equation A4).

$$\sum_{i \in V_c} x_{it} \leq C_{spec} \quad \text{for all } t \tag{A4}$$

It should be noted that although it appears that this general expression would admit solutions with fewer than the required number of items from each content category; this will never be the case. The solution to this problem will necessarily reach the upper bound set by each of the content specifications because each test form can only gain more information as result of the inclusion of more items. Inequalities are used because they tend to be less restrictive on linear programming algorithms and results in faster solution times as well as a lower likelihood of infeasibility.

In order to facilitate a solution to this problem, the constraints described above must be applied to an objective function. Equations A5 through A7b define the objective function and system of constraints required to achieve an absolute target test information curve required for this problem.

$$\text{objective: minimize } y \tag{A5}$$

Subject to:

$$y \geq 0 \tag{A6}$$

$$\sum_i I_i(\theta_{kt}) x_{it} \geq T_{kt} - y$$
$$\quad \text{for all levels of ability, } k, \text{ and all tests, } t$$
$$\sum_i I_i(\theta_{kt}) x_{it} \leq T_{kt} + y$$

$$\text{(A7a \& A7b)}$$

For this problem, equations 6a and 6b result in 9 curve constraints that restrict the sum of the information for the items included in each test, t, at each ability level, k, to be between the upper and lower bounds of the absolute targets, T_{kt}.

Cor, Alves & Gierl, Automated Test Assembly

Mathematical Model for Example 2

Stage I: Creating the Reference Form

The goal of this problem is to create a new test, t, with specific cut scores at the acceptable (θ_{accep}) and excellent (θ_{excel}) levels of achievement. The two decision variables required to formulate this problem are x_i, which allows items to be selected or not selected for the new test, and y, which represents the maximum absolute difference between the target cut scores and the actual cut scores on the reference form. Both variables, shown in equations 8 and 9, are free to change so that an optimal solution to the problem can be found. The item level decision variable, x_i, is constrained to be either a one or a zero using equation 10.

$$x_{it} \tag{A8}$$

$$y_t \tag{A9}$$

$$x_{it} = \text{binary; i.e. } x_{it} \begin{cases} 1 \text{ if item } i \text{ is included in test form t} \\ 0 \text{ if item } i \text{ is not included in test form t} \end{cases} \tag{A10}$$

The objective function for this problem is to minimize the absolute maximum difference, y, between the target cut scores and the actual cut scores. In order to initiate this objective, a set of psychometric test level constraints must be implemented. Equations A11 through A16 produce five psychometric constraints that constrain *y*.

$$\text{minimize} \quad y \tag{A11}$$

$$\sum_{i=1}^{168} P_i\left(\theta_{accep}\right) \leq T_{accep} + y \tag{A12}$$

$$\sum_{i=1}^{168} P_i\left(\theta_{accep(t)}\right) \geq T_{accep(t)} - y_t \tag{A13}$$

$$\sum_{i=1}^{168} P_i\left(\theta_{excel(t)}\right) \leq T_{excel(t)} + y_t \tag{A14}$$

$$\sum_{i=1}^{168} P_i\left(\theta_{excel(t)}\right) \geq T_{excel(t)} - y_t \tag{A15}$$

$$y \geq 0 \tag{A16}$$

Equation A17 produces a test level constraint that ensures the reference form contains exactly 55 items. Equation A18, results in six content constraints ensuring that the reference form meets the test blue print specifications, C_{spec}, for each content category, C.

$$\sum_{i=1}^{168} x_{it} = 55 \tag{A17}$$

$$\sum_{i \in C} x_{it} = C_{spec} \text{ for all content categories } C \tag{A18}$$

Cor, Alves & Gierl, Automated Test Assembly

The new test form must contain 23 items in common with the reference form. This common item specification requires an additional decision variable that describes whether item i has been assigned to forms t and t'. Equation A19 is used to specify the overlap decision variables for this stage of the problem. Like x_i, $z_{itt'}$ must be constrained to be either a one or a zero as shown in equation A20.

$$(A19)$$

$$z_{itt'} = \text{binary; i.e. } z_{itt'} \begin{cases} 1 \text{ if item } i \text{ is included in test form t} \\ 0 \text{ if item } i \text{ is not included in test form t} \end{cases} \tag{A20}$$

Equations A21 through A23 generate the item overlap constraints to ensure that exactly 23 items are common between the two forms.

$$\sum_{i=1}^{168} z_{itt'} = 23 \text{ for all } t < t' \tag{A21}$$

$$2z_{itt'} \leq x_{it} + x_{it'} \text{ for all } i \text{ and } t < t' \tag{A22}$$

$$z_{itt'} \geq x_{it} + x_{it'} - 1 \text{ for all } i \text{ and } t < t' \tag{A23}$$

Mathematical Model for Example 3

The BIB requirement necessitates two stages of optimization in order to solve this problem. *Stage I* requires the creation of 13 parallel blocks consisting of 13 unique items, while *Stage II* organizes the 13 blocks into 26 parallel forms in a BIB design. The following is a description of the models and specifications used to facilitate the solution of each stage of the automated test assembly problem.

Stage I: Creating 13 Parallel Blocks

The decision variable for this problem, x_i, is represented and constrained in the same way as is shown in the previous two scenarios (see equations A1 and A2). In order to ensure no item overlap, 613 *overlap constraints* (equation A24) are imposed on the items.

$$\sum_{t=1}^{13} x_{it} \leq 1 \text{ for all } t \tag{A24}$$

This test assembly problem requires 429 *test level constraints*. Of these 429 test level constraints, 13 are used to ensure the number of items contained in each form equals 13 (equation A25), 52 are used to ensure that each theme specification is exactly met (equations A26 to A29), and the remaining 364 are used to ensure that sum of items used to assess each descriptor across all forms is within one deviation of the descriptor specifications (as generalized by equations A30 and A31).

$$\sum_{i=1}^{613} x_{it} = 13 \text{ for all } t \tag{A25}$$

$$\sum_{i \in T1} x_{it} = 2 \text{ for all } t \tag{A26}$$

$$\sum_{i \in T2} x_{it} = 3 \text{ for all } t \tag{A27}$$

$$\sum_{i \in T3} x_{it} = 7 \text{ for all } t \tag{A28}$$

$$\sum_{i \in T4} x_{it} = 1 \quad \text{for all } t \tag{A29}$$

$$\sum_{t=1}^{13} x_{it} - D_{spec} \leq 1 \quad \text{for all } i \in D_{spec} \text{ and all } D_{spec} \tag{A30}$$

$$\sum_{t=1}^{13} x_{it} - D_{spec} \geq 1 \quad \text{for all } i \in D_{spec} \text{ and all } D_{spec} \tag{A31}$$

In order to facilitate a reasonable solution for this problem, the objective function was formulated to maximize the information at $\theta = 0$ on the ability scale. The problem is akin to a criterion referenced testing situation in which the most important point on the ability scale is $\theta = 0$. For the purposes of stage one, the objective function used to assemble the 13 blocks is shown in equation A32 where k = 1.

$$\text{maximize} \sum_t \sum_k \sum_i I_i(\theta_{kt}) x_{it} \tag{A32}$$

The mathematical representation described above concludes the definition of the model required to solve stage one of example 3.

Stage II: Implementing BIB

The objective for this BIB model is to combine the 13 parallel blocks created in *Stage I,* into 26 parallel test forms. In order to facilitate the assignment of blocks of items to each test form, a new decision variable is required (equation A33).

$$x_{jt} = \begin{cases} 1 \text{ if block } j \text{ is assigned to test } t \\ 0 \text{ if block } j \text{ is not assigned to test } t \end{cases} \tag{A33}$$

The BIB design also requires a decision variable to describe whether pairs of item blocks have been assigned to individual test forms. Equation A34 shows this new decision variable which allows the test developer to constrain the total amount of overlap that can exist between item blocks across all resulting test forms.

$$z_{jkt} = \begin{cases} 1 \text{ if the pair of blocks } (j,k) \text{ is assigned to test } t \\ 0 \text{ if the pair of blocks } (j,k) \text{ is assigned to test } t \end{cases} \tag{A34}$$

The BIB design for this problem requires the following set of constraints (equations A35 to A39):

$$\boxed{\times} \sum_{j=1}^{13} x_{jt} = 3 \quad \text{for all } t \tag{A35}$$

$$\sum_{t=1}^{26} x_{jt} \leq 6 \quad \text{for all } j \tag{A36}$$

$$\sum_{t=1}^{26} z_{jkt} = 1 \quad \text{for all } j < k \tag{A37}$$

$$2z_{jkt} \leq x_{jt} + x_{kt} \quad \text{for all } t \text{ and } j < k \tag{A38}$$

$$z_{jkt} \geq x_{jt} + x_{kt} - 1 \quad \text{for all } t \text{ and } j < k \tag{A39}$$

Equation A35 results in *26 individual constraints* that force the number of blocks in each test form to be equal to three. Equation A36 defines *13 constraints* that limit the total number of appearances for each block of items to six, and equation A37 creates *78 constraints* that force the number of times that each set of blocks can appear together across all

Cor, Alves & Gierl, Automated Test Assembly

of the final test forms to one. The last 78 constraints ensure that each of the final forms share a maximum of 13 items with only one of the remaining forms. The constraints that are generated as a result of equations A38 and A39 ensure that pairs of blocks are assigned to each test form only if the individual blocks have also been assigned. When taken together, the last two general expressions result in an additional *4056 constraints*.

Because the total amount of information that can be obtained when these individual forms is fixed, the objective function for this optimization problem merely serves as a tool to facilitate the organization of the blocks to meet the aforementioned combinatorial constraints. The objective function that is used to initiate the optimization is shown in equation A40.

$$\text{Maximize} \sum_t \sum_j \sum_i I_i(\theta)x_{jt} \tag{A40}$$

The objective function reported above maximizes the total information at $\theta = 0$ for each of the items included in each block across each block included in each final form.