LSAC RESEARCH REPORT SERIES

- # Review of Modern Methods for Automated Test Assembly and Item Pool Analysis

  Dmitry I. Belov

- Law School Admission Council
  Research Report 16-01
  March 2016

A Publication of the Law School Admission Council

The Law School Admission Council (LSAC) is a nonprofit corporation that provides unique, state-of-the-art products and services to ease the admission process for law schools and their applicants worldwide. Currently, 222 law schools in the United States, Canada, and Australia are members of the Council and benefit from LSAC's services. All law schools approved by the American Bar Association are LSAC members. Canadian law schools recognized by a provincial or territorial law society or government agency are also members. Accredited law schools outside of the United States and Canada are eligible for membership at the discretion of the LSAC Board of Trustees; Melbourne Law School, the University of Melbourne is the first LSAC-member law school outside of North America. Many nonmember schools also take advantage of LSAC's services. For all users, LSAC strives to provide the highest quality of products, services, and customer service.

Founded in 1947, the Council is best known for administering the Law School Admission Test (LSAT®), with about 100,000 tests administered annually at testing centers worldwide. LSAC also processes academic credentials for an average of 60,000 law school applicants annually, provides essential software and information for admission offices and applicants, conducts educational conferences for law school professionals and prelaw advisors, sponsors and publishes research, funds diversity and other outreach grant programs, and publishes LSAT preparation books and law school guides, among many other services. LSAC electronic applications account for nearly all applications to ABA-approved law schools.

Table of Contents

Executive Summary

Automated methods have been developed for assembling test forms, evaluating a pool of test questions (i.e., items) to determine the number of test form assemblies it can support, and designing an item pool that can most efficiently support the test form assembly process. Automated methods have greatly maintained and improved such activities, all of which are essential to the support of every testing program. This report reviews the major approaches that have been applied in the development of these methods. Their potential application to the computerized adaptive or multistage delivery of the Law School Admission Test is also discussed.

Introduction

Testing organizations produce, on a strict periodic basis, test forms for assessments in various formats: paper-and-pencil (P&P), computer-based testing (CBT), multistage testing (MST), or computerized adaptive testing (CAT). Each test form includes items selected from an item pool to optimize a given objective function and/or to satisfy given test specifications in terms of both statistical and content constraints. Assembling such forms can be formulated as a combinatorial optimization problem, referred to here as a test assembly (TA) problem.

Combinatorial optimization (CO) is concerned with searching for an element from a finite set (called a *feasible set*) that would optimize (minimize or maximize) a given objective function. Numerous practical problems can be formulated as CO problems, where a feasible set is not given explicitly but is represented implicitly by a list of inequalities and inclusions.

Early in the 1980s, researchers in psychometrics started to apply CO to TA. Theunissen (1985) reduced a special case of a TA problem to a knapsack problem (Papadimitriou & Steiglitz, 1982). Van der Linden and Boekkooi-Timminga (1989) formulated a TA problem as a maximin problem. Later, Boekkooi-Timminga (1990) extended this approach to the assembly of multiple nonoverlapping test forms.[1] Soon after that, the TA problem attracted many researchers, whose major results are discussed in van der Linden (2005). The latest ATA developments from CO standpoint will be discussed in this report.

Currently, the importance of CO in psychometrics is growing due to its recent applications that go beyond TA, such as identification of cognitive models (Cen,

[1] Two tests are called *nonoverlapping* if they do not have items in common; otherwise, they are called *overlapping*.

Koedinger, & Junker, 2006), resource management (van der Linden & Diao, 2011), optimal learning (van der Linden, 2012), test security (Belov, 2014), and many other applications.

This report is structured as follows. First, general types of TA problems are introduced. Second, major solvers of TA problems applied in psychometrics are outlined. Third, various practical situations in which TA problems arise are described. Finally, a summary is provided.

Throughout this report, the following notation is used:

Small letters $a, b, c, \ldots$ denote scalars.

Bold small letters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$ denote vectors.

Capital letters $A, B, C, \ldots$ denote sets. The number of elements in a set S is denoted by $|S|$; $\varnothing$ denotes an empty set.

Bold capital letters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ denote functions.

General Types of Test Assembly Problems

Van der Linden (2005) described specific types of TA problems for different types of assessments (i.e., P&P, CBT, MST, CAT) in his seminal textbook on optimal test design. Real instances of TA problems have been presented by Ariel, Veldkamp, and Breithaupt (2006); Armstrong, Belov, and Weissman (2005); Belov, Armstrong, and Weissman (2008); Breithaupt, Ariel, and Veldkamp (2005); De Jong, Steenkamp, and Veldkamp (2009); Veldkamp (2002); and Veldkamp and van der Linden (2002).

Test Assembly as a Problem of Combinatorial Optimization

Without loss of generality, a TA problem can be formulated as the following CO problem:

$$\begin{array}{l} \text{maximize } \mathbf{F}(\mathbf{x}) \\ \text{subject to } \mathbf{x} \in X \end{array} . \qquad (1)$$

$\mathbf{x} = (x_1, x_2, ..., x_n)^T$ is a binary decision vector defining a test, such that if $x_i = 1$ then item i is included in the test; otherwise (i.e., $x_i = 0$), item i is not included in the test.

n is the number of items in the item pool.

Set X contains all binary vectors, each defining a feasible test. Therefore, this set is called a *feasible set*. In practice, a feasible set is not given explicitly but is represented implicitly by a list of inequalities and inclusions constraining the decision vector $\mathbf{x}$. This list is constructed directly from test specifications. For example, the following represents a feasible set with all tests containing 5–10 items:

$$\begin{array}{l} 5 \le \sum_{i=1}^{n} x_i \le 10 \\ x_i \in \{0,1\} \end{array},$$

where the second constraint is included in any CO problem (i.e., each feasible solution $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ has to be a binary vector).

$\mathbf{F}(\mathbf{x})$ is an objective function (possibly a vector function; for a multiobjective TA problem, see Veldkamp, 1999). For example, in CAT, the following linear objective maximizes the Fisher information of a test at ability estimate $\hat{\theta}$:

$$\text{maximize} \sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i , \tag{2}$$

where $\mathbf{I}_i(\hat{\theta})$ is the Fisher information of item i at ability level $\hat{\theta}$ (Lord, 1980).

Test Assembly as a Problem of Constraint Satisfaction

A TA problem can also be formulated as the following constraint satisfaction problem:

$$\mathbf{x} \in X. \tag{3}$$

Many practical problems can be reduced to the analysis of the feasible set X. For example, in P&P and CBT modes, each item can be administered only once. Therefore, it is crucial for item pool maintenance to have an estimate of the maximum number of nonoverlapping tests available from an item pool given the test specifications.

An approximate solution can be found by sampling from the feasible set and then solving the maximum set packing problem (given a collection of subsets, find a maximum subcollection with mutually disjoint subsets [for more details, see Garey & Johnson, 1979]) for the resulting sample. For the sampling, Problem (3) can be solved multiple times such that each vector from X has an equal probability of being a solution. In other words, every test from the feasible set X has $1/|X|$ probability of being assembled. This process is therefore called *uniform test assembly* (UTA). For more details on UTA and its applications, see Belov (2008), Belov and Armstrong (2005, 2008, 2009), and Belov et al. (2008).

Often a good lower bound for the objective function is known or can be easily computed (Belov & Armstrong, 2009). Subsequently, Problem (1) can be approximated by Problem (3). For example, the following represents a feasible set with all possible tests containing 5–10 items and having Fisher information at ability estimate $\hat{\theta}$ above the lower bound 3:

$$\begin{aligned} & 5 \le \sum_{i=1}^{n} x_i \le 10 \\ & \sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i \ge 3\,. \\ & x_i \in \{0,1\} \end{aligned} \tag{4}$$

Interestingly, Problem (3) can be approximated by Problem (1) as well:

$$\begin{aligned} & \text{maximize } \sum_{i=1}^{n} \alpha_i x_i \\ & \text{subject to } \mathbf{x} \in X, \end{aligned} \tag{5}$$

where $\alpha_1, \alpha_2, ..., \alpha_n$ are independent and uniformly distributed on [0, 1). Vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)^T$ is resampled each time before Problem (5) is solved, thus allowing the assembly of different tests. Intuitively, one would expect that since vector $\boldsymbol{\alpha}$ is uniformly distributed, then the resultant sample of assembled tests will be uniform.

However, Belov (2008) proved that a sequence of optimal solutions to Problem (5), in general, will not provide a uniform sample from the feasible set. Only if a feasible set contains pairwise nonoverlapping tests (which hardly ever happens in practice) will a sequence of optimal solutions to Problem (5) provide a uniform sample. In other words, UTA cannot be formulated as Problem (5). The question as to whether UTA can be achieved via a sequence of optimal solutions to a certain instance of Problem (1) is still open.

Test Assembly Problem Under Uncertainty

Usually, inequalities defining the feasible set X can be grouped into content constraints [i.e., the first inequality in (4)] and statistical constraints [i.e., the second inequality in (4)]. Content constraints are known precisely. Statistical constraints usually include parameters of item response theory (IRT) models (Lord, 1980), which are calibrated from the response data and therefore subject to error. Thus, the assembled test may not actually satisfy the statistical constraints, and/or the objective function can be over- or underestimated. Note that all real-life instances of TA problems are under uncertainty due to estimation errors in the statistical parameters of items.

Optimization under uncertainty is a well-studied topic of optimization. There are two major approaches: stochastic optimization (Birge & Louveaux, 1997) and robust optimization (Bertsimas, Brown, & Caramanis, 2011). Let us consider a common TA problem:

$$\begin{aligned}&\text{maximize } \sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i \\ &\text{subject to } \mathbf{x} \in X\end{aligned}, \tag{6}$$

where feasible set X is defined by content constraints (known precisely), but each coefficient $\mathbf{I}_i(\hat{\theta})$ has an error associated with it as a result of item parameter estimation using a given procedure such as expectation maximization (EM) or Markov chain Monte Carlo (MCMC).

Assume that distributions of each parameter estimated under an IRT model are known, where these distributions are outcomes of a specific MCMC procedure that was used to estimate the item parameters. Then a stochastic counterpart of Problem (6) is formulated as follows:

$$\begin{aligned}&\text{maximize} \sum_{i=1}^{n} \mathbf{E}[\mathbf{I}_i(\hat{\theta}, h_{i1}, h_{i2}, ...)]x_i, \\ &\text{subject to } \mathbf{x} \in X\end{aligned} \tag{7}$$

where expectation is taken over parameters $h_{i1}, h_{i2}, ...$ of item i. In this instance, Problem (7) can be solved directly.

Assume that $\mathbf{I}_i(\hat{\theta})$ has an error such that $\mathbf{I}_i(\hat{\theta}) \in [u_i - d_i,\ u_i]$ with high probability, where u_i and d_i are estimated by EM or MCMC procedures. From Bertsimas and Sim (2003), it follows that a robust counterpart of Problem (6) can be formulated:

$$\begin{aligned}&\text{maximize}\left[\sum_{i=1}^{n} u_i x_i - \max_{\{S|S\subseteq N,\ |S|\leq g\}} \sum_{j\in S} d_j x_j\right], \\ &\text{subject to } \mathbf{x} \in X\end{aligned} \tag{8}$$

where $N = \{1, 2, ..., n\}$ and g is a parameter chosen beforehand. An optimal solution to Problem (8) defines a test with Fisher information at ability level $\hat{\theta}$ above a certain threshold. This inequality holds under uncertainty in at most g items. Clearly, Problem (8) cannot be directly solved. However, Bertsimas and Sim (2003) developed a method to solve Problem (8) by solving $n+1$ problems directly:

$$\begin{aligned}&\max_{l=1}^{n+1}\left[-gd_l + \max\left[\sum_{i=1}^{n} u_i x_i - \sum_{j=1}^{l}(d_j - d_l)x_j\right]\right], \\ &\text{subject to } \mathbf{x} \in X\end{aligned} \tag{9}$$

where, without loss of generality, $d_1 \geq d_2 \geq ... \geq d_n \geq d_{n+1} = 0$ is assumed. More details on the application of robust optimization for automated test assembly (ATA) can be found in Veldkamp (2012).

An alternative approach to accommodate the uncertainty in item parameters is to state the TA problem as (3) but use narrower bounds for statistical constraints. New bounds should be computed (e.g., by a Monte Carlo method) such that the probability of a feasible test violating original bounds is below a given significance level. This approach can be implemented within existing ATA methods.

Automated Test Assembly Methods

From a geometrical standpoint, the TA problem is solved by searching through the vertices of the hypercube $\{\mathbf{x}=(x_1,x_2,...,x_n)^T \mid 0 \le x_i \le 1,\ i=1,2,...,n\}$ until a vertex $\mathbf{x}_0 \in X$ that optimizes the objective function $\mathbf{F}(\mathbf{x})$ is found [see Problem (1)] or until a vertex $\mathbf{x}_0 \in X$ is found [see Problem (3)]. The number of vertices of the hypercube is 2^n. Therefore, the search can run for some time, exponentially dependent on the number of items in the pool. However, in practice, this problem is often solvable in a reasonable amount of time on a personal computer. This section will briefly review the major ATA methods.

Branch-and-Bound

The Branch-and-Bound (B&B) method solves Problem (1) by performing an intelligent search through vertices of the hypercube. It starts by finding an optimal solution to the relaxation of Problem (1) without the key constraint $x_i \in \{0,1\}$, $i=1,2,...,n$. The relaxation can be solved in polytime, which means that the running time of the solver is bounded by a polynomial in size of the problem (see Garey & Johnson, 1979), resulting in a fast convergence. An optimal solution to the relaxation provides a choice of branching decisions and an upper bound for Problem (1). More precisely, one selects a coordinate $1 \le j \le n,$ where an optimal solution to the relaxation has a fractional value. Then one adds two new subproblems to a list of subproblems (initially empty): relaxation with additional constraint $x_j = 0$ and relaxation with additional constraint $x_j = 1$. Each subproblem on the list is solved, where one of the following cases is possible:

1. The subproblem is infeasible; that is, the corresponding feasible set $X = \varnothing$.
2. An optimal solution to the subproblem is binary, which provides a feasible solution to Problem (1). This solution is used to update the global solution.
3. An optimal solution to the subproblem is not binary, and its objective function is less than or equal to the global objective found so far.
4. An optimal solution to the subproblem is not binary, and its objective function is higher than the global objective found so far.

In cases 1–3, one removes the subproblem from the list and analyzes the next subproblem on the list. In case 4, one applies branching of the subproblem (see above) and then removes the subproblem from the list. When the list is empty, one can claim that an optimal solution to Problem (1) has been found. For more details, see Papadimitriou and Steiglitz (1982) and Nemhauser and Wolsey (1988).

With the B&B method, one can prove optimality of a feasible solution to Problem (1). The success of applying B&B depends on how well a solver adapts to each instance of Problem (1)—more precisely, how well the structure of an instance is taken into account to organize effective branching and bounding.

When Problem (1) is linear and its matrix of the system of inequalities is totally unimodular (Nemhauser & Wolsey, 1988), the relaxation of Problem (1) will have a binary optimal solution. Even more, several fast polytime algorithms are available to solve the relaxation (Ahuja, Magnanti, & Orlin, 1993). If a large submatrix of the matrix of the system of inequalities is totally unimodular, then the assembly of linear tests can be performed efficiently (Armstrong, Jones, & Kunce, 1998; Armstrong, Jones, & Wu, 1992) by a combination of the following methods: network flow programming, Lagrangian relaxation, and B&B.

The B&B method is a core of the mixed-integer programming (MIP) approach to large practical problems of item pool analysis and design. Typically, a real-life problem is formulated as an instance of Problem (1) and then solved directly with the B&B method. The next section describes multiple applications of the MIP approach.

Heuristics

Heuristic methods (heuristics) provide a relatively fast search through vertices of the hypercube that are likely to discover a near solution: In the case of Problem (1), it is a suboptimal solution; in the case of Problem (3), it is a subfeasible solution. A comprehensive review of ATA heuristics is given by van der Linden (2005).

Some heuristics (Swanson & Stocking, 1993) move the constraints to the objective function, which essentially is a Lagrangian relaxation (Nemhauser & Wolsey, 1988). Then set X is no longer a feasible set, because some vectors from X may violate constraints that were incorporated into objective function $\mathbf{F}(\mathbf{x})$. For example, consider the following TA problem:

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i \\ &\text{subject to} && 5 \le \sum_{i=1}^{n} x_i \le 10\,, \\ & && x_i \in \{0,1\} \end{aligned} \qquad (10)$$

where the feasible set contains only vertices of the hypercube with 5–10 positive coordinates (corresponding to tests with 5–10 items). By applying Lagrangian relaxation, the TA Problem (10) is transformed into the following:

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i + \lambda_1\left(5 - \sum_{i=1}^{n} x_i\right) + \lambda_2\left(10 - \sum_{i=1}^{n} x_i\right) \\ &\text{subject to} && x_i \in \{0,1\} \qquad , \\ & && \lambda_1 \le 0 \\ & && \lambda_2 \ge 0 \end{aligned} \qquad (11)$$

where the feasible set contains all vertices of the hypercube.

Most heuristics in ATA literature are based on sequential item selection: One item is selected at a time until the required number of items is reached, where each selection minimizes the current value of a residual. There are numerous types of residuals (Ackerman, 1989; Leucht, 1998; Swanson & Stocking, 1993) driven by various TA constraints and/or TA objectives. These heuristics minimize the current value of the residual in the hope that when the required number of items are selected, they will satisfy the constraints and/or optimize the objective. Such heuristics belong to a class known in the optimization literature as *greedy heuristics* (Papadimitriou & Steiglitz, 1982). For example, consider the following TA problem:

$$\begin{aligned} &\sum_{i=1}^{n} \mathbf{I}_i(\hat{\theta}) x_i = t \\ &\sum_{i=1}^{n} x_i = 10 \qquad , \\ &x_i \in \{0,1\} \end{aligned} \qquad (12)$$

where t is a target value of the Fisher information at the current ability estimate. Assume that three items $S = \{i_1, i_2, i_3\}$ were already selected. Then, according to Leucht (1998), the fourth item i_4 should minimize the following residual:

$$\left| \mathbf{I}_{i_4}(\hat{\theta}) - \left(t - \sum_{i \in S} \mathbf{I}_i(\hat{\theta}) x_i \right) / 7 \right|. \qquad (13)$$

While greedy heuristics are fast, their solutions are only locally optimal and, therefore, may violate some of the constraints [e.g., see TA Problem (11)]. At the same time, in high-stakes testing, violation of certain or all constraints is not acceptable (Ariel, Veldkamp, & Breithaupt, 2006; Armstrong et al., 2005; Breithaupt et al., 2005; De Jong et al., 2009; Veldkamp, 2002; Veldkamp & van der Linden, 2002). Several CO approaches have been applied to avoid getting stuck in local optimum while solving a TA problem, such as simulated annealing (van der Linden et al., 2004) and genetic algorithms (Verschoor, 2004).

Monte Carlo Test Assembler

The Monte Carlo test assembler (MCTA) was introduced by Belov and Armstrong (2004, 2005) to solve TA Problem (3). It is straightforward in concept and consists of two steps:

Step 1: Generate a random vector of items.

Step 2: If this vector satisfies test specifications, save it as a new test and stop; otherwise, return to Step 1.

The biggest challenge with the Monte Carlo technique is to avoid generating many “useless” vectors at Step 1. Belov and Armstrong (2004, 2005) have developed several strategies to reduce the search space such that it still has a nonempty intersection with the feasible set. They exploited properties of the constraints, using a divide-and-conquer principle and tabu search, and prioritized constraint checking based on their computational complexity. MCTA has been applied for P&P (Belov & Armstrong, 2004, 2005), MST (Belov & Armstrong, 2008), and constrained CAT (Belov et al., 2008). The performance of MCTA is surprisingly fast. For example, Belov et al. (2008) reported that the Monte Carlo CAT performed 20 times faster than the shadow CAT (van der Linden & Reese, 1998).

The major advantage of MCTA is its ability to perform uniform sampling from the feasible set X. This advantage is crucial in practice (see next section). For example, due to its random nature, the convergence rate of MCTA determines how large the feasible set is: the higher the rate, the large the feasible set. The size of the feasible set directly indicates how given test specifications match a given item pool. Other potential approaches to produce a uniform sampling from the feasible set are analyzed by Belov (2008).

MCTA is a core of the UTA approach to large practical problems of item pool analysis and design, where properties of a feasible set of a given instance of Problem (3) are explored and exploited via uniform sampling from the feasible set. The next section describes multiple applications of the UTA approach.

Item Pool Analysis and Design

The major purpose of ATA is to assemble one test at a time. The specifics of a particular assessment, however, may influence the methods described in the previous section. In CAT, the shadow CAT (MIP approach by van der Linden & Reese, 1998) selects the next item maximizing Fisher information at the current ability estimate, such that the administered sequence of items satisfies content constraints. Monte Carlo CAT (Belov et al., 2008) allows balancing between the maximization of Fisher information and the robustness of the ability estimate to possible mistakes made by the test taker during a test. In MST, each path in an MST form has to be assembled taking into account common testlets between paths (for more details, see Belov & Armstrong, 2008).

In each assessment, there are multiple other tasks in which ATA plays a crucial role (van der Linden, 2005). From a mathematical standpoint, most of these tasks can be reduced to the analysis of properties of the feasible set X.

How Do We Explore Properties of the Feasible Set?

For real-life item pool and test specifications, computing the whole feasible set is intractable. The analysis of the matrix of the system of inequalities is very limited and possible only for linear systems. Therefore, in general, the only way to study the properties of a feasible set is to construct and analyze a uniform sample from the feasible set.

Let us assume that there is a way to assemble tests such that each test from the feasible set has an equal probability of being assembled (UTA). Then assemble multiple tests without withdrawing their items from the pool. Since the resulting sample is drawn uniformly, it can be considered representative of the feasible set. Therefore, the statistical inference about properties of the feasible set can be acquired from this sample. For example, an item usage frequency can be calculated. Given a set of tests, the *usage frequency* of an item is the number of tests that include this item, where an item with the highest usage frequency is called the *most usable item* and an item with the lowest usage frequency is called the *least usable item.* The computation of item usage frequency is straightforward:

Step 1: Assemble multiple tests uniformly without withdrawing their items from the pool.

Step 2: For each item in the pool, count how many assembled tests include this item.

Is the Test Assembly Problem Feasible? If Not, Why Not?

Belov and Armstrong (2005) used uniform sampling from embedded feasible sets (where each embedding corresponds to additional subset of constraints) to identify the most difficult constraints. Difficult constraints dramatically reduce the size of the feasible set and may cause the feasible set X to be empty, which makes the corresponding TA problem infeasible. An alternative approach based on MIP is presented by Huitzing, Veldkamp, and Verschoor (2005).

How Many Nonoverlapping Tests Are There?

Given test specifications and an item pool, the number of available nonoverlapping tests is a critical indicator for testing organizations producing P&P, CBT, and MST, because each corresponding test form can be administered only once.

A simple heuristic (Boekkooi-Timminga, 1990) is to assemble a test, then withdraw its items from the pool, then assemble another test, and so on, until the TA problem becomes infeasible (or a TA solver cannot assemble a test within given period of time). Such an approach is known as a *sequential assembly*. However, Belov (2008) demonstrated that this approach often cannot assemble many nonoverlapping tests. Subsequently, alternative methods were developed that greatly outperform sequential assembly by utilizing properties of the feasible set.

Belov and Armstrong (2006) suggested a set packing approach (SPA). They assembled multiple nonoverlapping tests in two stages:

Stage 1: Sample from the feasible set.

Stage 2: Solve the maximum set packing (or, equivalently, the maximum clique) problem (Garey & Johnson, 1979) for the resulting sample.

They applied this approach for P&P, in particular for the LSAT (Belov & Armstrong, 2005) and MST (Belov & Armstrong, 2008). Belov (2008) developed a modified sequential assembly (MSA) by using item usage frequency in order to keep more usable items for further assemblies. The SPA and MSA both demonstrated twice the speed and resulted in the same number of nonoverlapping tests when the least usable items were withdrawn from the pool before running SPA and MSA (Belov, Williams, & Kary, 2015). In addition, Belov et al. (2015) studied a mixed method where the use of item usage frequency is combined with the SPA such that its two stages are repeated as follows:

Step 1: Apply SPA to the pool without the most usable items.

Step 2: Withdraw items of the assembled tests from the pool.

Step 3: Apply SPA to the pool with the most usable items added.

An alternative approach based on MIP modeling to improve the sequential assembly, called *shadow test assembly*, was developed by van der Linden and Adema (1998). More specifically, to assemble m nonoverlaping tests, at step $i,\ i=1,2,...m$ the following two problems are solved simultaneously:

Problem 1: Assemble the test i.

Problem 2: Assemble a shadow test that satisfies relaxed inequalities with original bounds multiplied by $m-i$. A major issue with this approach though is that some constraints cannot be relaxed this way.

Obviously, all of the above methods are also applicable when a partial overlap between tests is allowed.

How Can an Item Pool Be Maintained Efficiently?

In all assessments, test developers need to identify the properties of future items that would help maintain their item pool efficiently. In particular, in P&P, CBT, and MST, test developers need to minimize the number of new items needed in order to maximize the number of nonoverlapping test forms available from an existing pool. This minimax problem can be solved by exploiting item usage frequency (Belov & Armstrong, 2005, 2008), which is computed from a uniform sample of tests. In experiments with an LSAT item pool and constraints, Belov and Armstrong (2005) demonstrated that adding just a few new items that have properties similar to those of the most usable items dramatically increases the number of nonoverlapping tests that can be assembled.

Alternative approaches for designing and maintaining item pools are based on MIP modeling (Ariel, van der Linden, & Veldkamp, 2006; Ariel, Veldkamp, & Breithaupt, 2006; Ariel, Veldkamp, & van der Linden, 2004).

How Is an Item Pool Assembled for CAT?

Usually, in CAT there is a large master pool from which one has to assemble a smaller CAT pool for the next administration. Any realistic method of CAT pool assembly should guarantee the following two CAT design objectives: the existence of at least one feasible form (i.e., a form satisfying all content constraints); and bounded values of mean squared error and bias for estimated ability. Exploiting the MIP approach, a CAT pool was assembled by van der Linden, Ariel, and Veldkamp (2006) as a set of nonoverlapping feasible forms, where each form maximized information at a certain point and points were distributed according to the expected population. Via computer simulations, van der Linden et al. (2006) demonstrated a satisfaction of the two CAT design objectives. However, their heuristic is information greedy, causing each subsequent CAT pool assembled from the master pool to be less and less informative. A modification of this method (based on the UTA approach) by Belov and Armstrong (2009) enables the assembly of multiple (information-parallel) CAT pools that guarantee the two CAT design objectives.

What Test-Taker Population Will an Item Pool Serve Best?

Exploiting the UTA approach, Belov and Armstrong (2009) computed a distribution of test takers most suitable for a given item pool and test specifications in two stages:

Stage 1: Sample from the feasible set.

Stage 2: Compute the distribution based on test information functions from tests found in the previous stage.

How Can IRT Targets Be Computed?

When a testing organization migrates from P&P to MST format, content constraints for each path in an MST form are the same as in a P&P form. However, IRT targets for each path (targets for the test characteristic curve and the test information function of each path in an MST form) should differ in order for the assembled MST form to adapt to test-taker ability. Belov and Armstrong (2008) address this issue as follows:

Step 1: Build a uniform sample from the feasible set of linear forms, where each form is a vector of items satisfying the content constraints of the MST path.

Step 2: Administer the resultant sample to simulated test takers drawn from a given distribution.

Step 3: Use the resultant scores to partition the sample such that the target for each MST path is constructed from items most informative at the corresponding ability range.

This UTA-based method allows balancing between the measurement precision of assembled MST forms and the utilization of an item pool.

Summary

The development of ATA methods reduces the workload of test developers and ensures the quality of tests by utilizing the computational power of modern computers. This report reviews recent developments in general types of TA problems, major ATA methods, and various practical situations where a TA problem arises. Due to recent achievements in CO methods, multiple new practical problems in test development and design that were infeasible before can now be solved. Therefore, one can conclude that the TA problem is no longer a central issue for test development but is rather a subproblem embedded in larger practical problems. This review distinguishes two major approaches to these larger problems:

MIP approach: Treating a TA problem as Problem (1) and solving it with the B&B method

UTA approach: Treating a TA problem as Problem (3) and solving it with the Monte Carlo method, resulting in a uniform sampling from the feasible set

Both approaches are successfully applied in practice (see multiple references above). The MIP approach is a natural one for testing programs in which the test is defined by constraints and an objective function to be optimized. On the other hand, the UTA approach is a natural choice for testing programs in which the test is defined by constraints only.

References

Ackerman, T. (1989). *An alternative methodology for creating parallel test forms using the IRT information function*. Paper presented at the annual meeting of the National Council on Measurement in Education, San Francisco, CA.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Ariel, A., van der Linden, W. J., & Veldkamp, B. P. (2006). A strategy for optimizing item-pool management. *Journal of Educational Measurement*, *43*(2), 85–92.

Ariel, A., Veldkamp, B. P., & Breithaupt, K. (2006). Optimal testlet pool assembly for multi-stage testing designs. *Applied Psychological Measurement*, *30*, 204–215.

Ariel, A., Veldkamp, B. P., & van der Linden, W. J. (2004). Constructing rotating item pools for constrained adaptive testing. *Journal of Educational Measurement*, *41*, 345–359.

Armstrong, R. D., Belov, D. I., & Weissman, A. (2005). Developing and assembling the Law School Admission Test. *Interfaces*, *35*, 140–151.

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

Armstrong, R. D., Jones, D. H., & Wu, I. L. (1992). An automated test development of parallel tests from a seed test. *Psychometrika*, *57*(2), 271–288.

Belov, D. I. (2008). Uniform test assembly. *Psychometrika*, *73*, 21–38.

Belov, D. I. (2014). Detecting item preknowledge in computerized adaptive testing using information theory and combinatorial optimization. *Journal of Computerized Adaptive Testing*, *2(3)*, 37–58.

Belov, D. I., & Armstrong, R. D. (2004). *A Monte Carlo approach for item pool analysis and design*. Presented at the annual meeting of the National Council on Measurement in Education, San Diego, CA, April 2004.

Belov, D. I., & Armstrong, R. D. (2005). Monte Carlo test assembly for item pool analysis and extension. *Applied Psychological Measurement*, *29*, 239–261.

Belov, D. I., & Armstrong, R. D. (2006). A constraint programming approach to extract the maximum number of nonoverlapping test forms. *Computational Optimization and Applications*, *33(2/3)*, 319–332.

Belov, D. I., & Armstrong, R. D. (2008). A Monte Carlo approach to the design, assembly and evaluation of multi-stage adaptive tests. *Applied Psychological Measurement*, *32*, 119–137.

Belov, D. I., & Armstrong, R. D. (2009). Direct and inverse problems of item pool design for computerized adaptive testing. *Educational and Psychological Measurement*, *69*, 533–547.

Belov, D. I., Armstrong, R. D., & Weissman, A. (2008). A Monte Carlo approach for adaptive testing with content constraints. *Applied Psychological Measurement*, *32*, 431–446.

Belov, D. I., Williams, M., & Kary, D. (2015). *Exploiting properties of a feasible set to improve item pool utilization*. Presented at the international meeting of the Psychometric Society, Beijing, China.

Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, *53*(3), 464–501.

Bertsimas, D., & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, *98*, 49–71.

Birge, J. R., & Louveaux, F. (1997). *Introduction to stochastic programming*. New York: Springer-Verlag.

Boekkooi-Timminga, E. (1990). The construction of parallel tests from IRT-based item banks. *Journal of Educational Statistics*, *15*(2), 129–145.

Breithaupt, K., Ariel, A., & Veldkamp, B. P. (2005). Automated simultaneous assembly for multi-stage testing. *International Journal of Testing*, *5*, 319–330.

Cen, H., Koedinger, K., & Junker, B. (2006). *Learning factors analysis–a general method for cognitive model evaluation and improvement*. Proceedings of the 8th international conference on Intelligent Tutoring Systems, June 26–30, Jhongli, Taiwan.

De Jong, M. G., Steenkamp, J. B. E. M., & Veldkamp, B. P. (2009). A model for the construction of country-specific, yet internationally comparable short-form marketing scales. *Marketing Science*, *28*, 674–689.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Company.

Huitzing, H. A., Veldkamp, B. P., & Verschoor, A. J. (2005). Infeasibility in automatic test assembly models: A comparison study of different methods. *Journal of Educational Measurement*, *42*, 223–243.

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

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

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

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

Swanson, L., & Stocking, M. L. (1993). A model and heuristic for solving very large item selection problems. *Applied Psychological Measurement*, *17*(2), 151–166.

Theunissen, T. (1985). Binary programming and test design. *Psychometrika*, *50*(4), 411–420. doi: 10.1007/bf02296260

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

van der Linden, W. J. (2012). *Key methodological concepts in the optimization of learning and educational resource availability*. Paper presented at the annual meeting of the National Council on Measurement in Education, Vancouver, Canada.

van der Linden, W. J., & Adema, J. J. (1998). Simultaneous assembly of multiple test forms. *Journal of Educational Measurement*, *35*, 185–198.

van der Linden, W. J., Ariel, A., & Veldkamp, B. P. (2006). Assembling a CAT item pool as a set of linear tests. *Journal of Educational and Behavioral Statistics*, *31*, 81–99.

van der Linden, W. J., & Boekkooi-Timminga, E. (1989). A maximin model for IRT-based test design with practical constraints. *Psychometrika*, *54*(2), 237–247. doi: 10.1007/bf02294518

van der Linden, W. J., & Diao, Q. (2011). Automated test form generation. *Journal of Educational Measurement*, *48*, 206–222.

van der Linden, W. J., & Reese, L. M. (1998). A model for optimal constrained adaptive testing. *Applied Psychological Measurement*, *22*, 259–270.

van der Linden, W. J., Veldkamp, B. P., & Carlson, J. E. (2004). Optimizing balanced incomplete block designs for educational assessments. *Applied Psychological Measurement*, *28*, 317–331.

Veldkamp, B. P. (1999). Multiple objective test assembly problems. *Journal of Educational Measurement*, *36*, 253–266.

Veldkamp, B. P. (2002). Multidimensional constrained test assembly. *Applied Psychological Measurement*, *26*(2), 133–146.

Veldkamp, B. P. (2012). Application of robust optimization to automated test assembly. *Annals of Operations Research*. doi: 10.1007/s10479-012-1218-y

Veldkamp, B. P., & van der Linden, W. J. (2002). Multidimensional constrained adaptive testing. *Psychometrika*, *67*, 575–588.

Verschoor, A. (2004). *IRT test assembly using genetic algorithms*. Arnhem: Cito.