Interfaces
Vol. 35, No. 2, March–April 2005, pp. 140–151
ISSN 0092-2102 | EISSN 1526-551X | 05 | 3502 | 0140

DOI 10.1287/inte.1040.0123
© 2005 INFORMS

Developing and Assembling the Law School Admission Test

Ronald Armstrong
Rutgers Business School, Rutgers University, 180 University Avenue, Newark, New Jersey 07102-1895, r.d.armstrong@att.net

Dmitry Belov, Alexander Weissman
Psychometric Research Group, Law School Admission Council, 662 Penn Street, Box 40, Newtown, Pennsylvania 18940-0040 {dbelov@lsac.org, aweissman@lsac.org}

Standardized tests are useful for assessing an individual's potential to succeed in various endeavors. In addition, institutions use them to measure student achievement and to measure the efficacy of pedagogical approaches. Operations research tools can help those developing rigorous standardized tests. Our mixed-integer program (MIP) provides a model for assembling forms for the Law School Admission Test (LSAT). Since 2002, our LSAT assembler—software we developed using a Monte Carlo approach—has produced test forms meeting all specifications. This software has saved thousands of hours of personnel time.

Key words: programming: integer, applications; education systems.
History: This paper was refereed.

Over 148,000 potential law school students took the Law School Admission Test (LSAT) between June 2002 and February 2003. Most law schools in the United States and Canada require applicants to take the LSAT and consider the scores important in admission decisions. The LSAT is designed to predict the academic success of a student in the first year of law school. The current LSAT has a linear paper-and-pencil (P&P) format. The test consists of four scored sections: two in logical reasoning (LR), one in analytical reasoning (AR), and one in reading comprehension (RC), and an unscored 30-minute writing sample. The LSAT is administered four times a year, with all test takers receiving the same four scored sections on any given administration. Across different administrations, these sections differ; however, all tests are intended to be parallel. Parallel test forms are those that have the same measurement characteristics, within some tolerance level, even though specific test forms differ in content. Those creating test forms must meet these measurement characteristics, and operations research tools are extremely useful for these purposes.

All the test questions (items) on the LSAT are multiple choice. For some items, test takers must read a passage before they can answer the items. A database of items, passages, and their associated characteristics is an item pool. If the passage and the item correspond one to one, the item is called a discrete item. Multiple items associated with a passage are called set based. The AR and RC sections are set based with four passages in each section, and the LR sections are composed of discrete items. More information on the LSAT can be found at http://www.lsac.org.

While analysts cannot solve most test-assembly problems with polynomial algorithms, this does not mean that assembling a single linear test form is difficult. Assembling most tests does not require optimizing objective functions. Any combination of items that meets the defined test specifications yields an acceptable test form. From a typical item pool, one can combine items in many ways to make a form. Theunissen (1985) and van der Linden (1998, 2000) propose using a general MIP software package (Nemhauser and Wolsey 1988) to assemble test forms. However, few testing agencies are using this approach.

In 1997, the Law School Admission Council initiated a project to automate the assembly of the LSAT. We detailed all specifications and wrote software that implemented a branch-and-bound algorithm during the first year of the project. Since 1998, the MIP from the LSAT specifications has been solved with our software. In 2002, we implemented a new LSAT assembler utilizing an adaptive stochastic search approach. Analysts have used adaptive stochastic search, including simulated annealing (Spall 2003), genetic algorithms (Spall 2003), tabu search (Glover et al. 1993), and Monte Carlo methods (Spall 2003), such as random search, and Markov chain simulation, to solve various practical global optimization problems. They succeeded partly because the methods are easy to implement and adapt to complex problems. We use a Monte Carlo approach along with tabu search to assemble LSATs. The Monte Carlo approach can handle nonlinear extensions to the basic model and analyze the properties of the item pool. The automated assembly has provided test forms meeting all specifications and has saved thousands of hours of personnel time.

Item Response Theory

Testing agencies commonly employ item response theory (IRT) to measure the examinee's abilities (trait levels) and the items' response properties. Of the various IRT models (Lord 1980, Hambleton et al. 1991), we used the one-dimensional three-parameter logistic (3PL) model for the LSAT items.

For a given item, the goal of IRT is to predict the probability that an examinee with a specific ability level will correctly answer that item. In the 3PL model, only two responses to an item are possible—correct or incorrect. Thus, if we consider an arbitrary item i, we can define a Bernoulli random variable $\mathbf{U}_i$ such that $\mathbf{U}_i = 0$ when an examinee responds incorrectly, and $\mathbf{U}_i = 1$ when he or she responds correctly. Further, a continuous random variable $\boldsymbol{\theta}$ represents the ability levels of the examinee population; for convenience, $\boldsymbol{\theta}$ is usually standardized so that most examinees' abilities will fall between a low of $\theta = -3$ and a high of $\theta = +3$, with an ability of $\theta = 0$ being average for the group of examinees. Although IRT does not require the ability distribution to be normal, we can assume that $\boldsymbol{\theta}$ is distributed standard normal.

For any item i, based on IRT we model the probability of an examinee responding correctly to that item as a function of that examinee's ability and a set of item parameters. We denote IRT parameters for item i by a_i, b_i, and c_i. The value of the pseudo-guessing parameter, c_i, gives the probability that a low-ability examinee will answer the item correctly. The quality of incorrect choices affects this parameter; for example, a low-ability examinee may eliminate incorrect choices, which will increase this value, or be drawn to attractive incorrect choices, which can lower the value. The value of b_i is a measure of the difficulty of the item and is on the same scale as $\boldsymbol{\theta}$; thus, easier items have a lower b_i and more difficult items have a higher b_i. The probability of a correct response to item i at ability b_i is $(1 + c_i)/2$. The value of a_i measures the discrimination of the item, or how well the item can distinguish between lower- and higher- ability examinees. The larger a_i, the steeper the curve about $\theta = b_i$.

Equation (1) gives the probability that an examinee with ability θ responds correctly (Figures 1 and 2):

$$P(\mathbf{U}_i = 1 \mid \theta) = p_i(\theta) = c_i + \frac{1 - c_i}{1 + \exp(-1.7a_i(\theta - b_i))}. \quad (1)$$

Related to the probability of correct response to an item is an item's information function, a measure of the precision with which the item can measure an

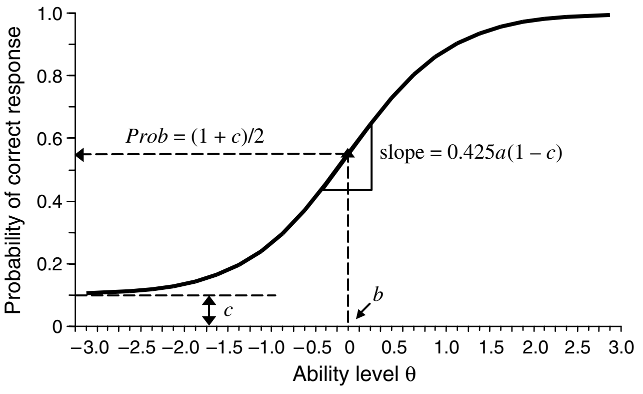

Figure 1: The goal of item response theory (IRT) is to predict the probability that an examinee with a specific ability level will correctly answer that item. We model the probability of an examinee responding correctly to an item as an item response function, which is a function of that examinee's ability level (θ) and a set of item parameters, shown in the figure as *a*, *b*, and *c*. The *b* parameter is a measure of the difficulty of the item and is on the same scale as θ, with less difficult items having lower *b* values and more difficult items having higher *b* values. The *a* parameter measures the discrimination of the item, or how well the item can distinguish between lower- and higher-ability examinees; the larger the value of *a*, the steeper the slope of the item response function as evaluated at ability level $\theta = b$. The *c* parameter, or pseudo-guessing parameter, gives the probability that a low-ability examinee will answer the item correctly. The probability of a correct response to an item for an examinee with $\theta = b$ is equal to $(1 + c)/2$.

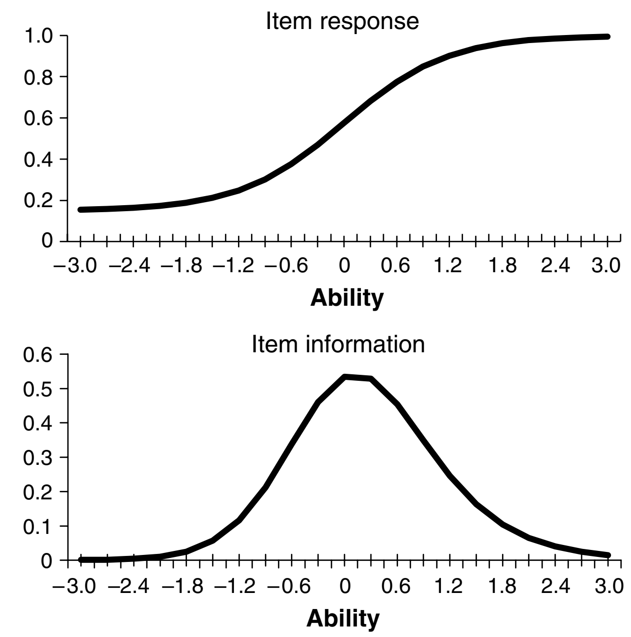

Figure 2: Related to the item response function (top) is the item information function (bottom). The item information function indicates the precision with which an item can measure an examinee's ability; thus, higher information is associated with smaller measurement error. By comparing the two figures, we see that an item provides greater measurement precision at those ability levels where the item response function is more sloped. This item has IRT parameters $a_i = 1$, $b_i = 0$, and $c_i = 0.15$, with examinee ability levels $\theta \in [-3, 3]$.

examinee's ability (Lord 1980, p. 73). Equation (2) gives this function (Figure 2):

$$I_i(\theta) = (1.7a_i)^2 \left[\frac{1.0 - p_i(\theta)}{p_i(\theta)}\right]\left[\frac{p_i(\theta) - c_i}{1 - c_i}\right]^2. \qquad (2)$$

The Life Cycle of an Item

We can best describe an item at its inception by its qualitative attributes rather than its quantitative attributes. Typically, items concern particular content areas, such as trigonometry for a mathematics test or analogies for a test of verbal ability. Test specialists develop taxonomies to categorize item attributes as precisely as possible. For example, a trigonometry item may be classified as (a) using sine, cosine, or tangent functions; (b) including no more than one unknown quantity; (c) including or not including an accompanying diagram. The specialists assemble test forms according to test blueprints, which insure that completed tests conform to desired specifications in terms of number and types of items.

Before specialists include items on a test form, these items must pass a series of quality-control checks. For example, an item must have exactly one correct answer. For multiple-choice items, the distractors, or incorrect choices, must be both attractive to examinees and incorrect. Occasionally a first draft of an item includes a distractor that is attractive but correct under certain unintended interpretations. Conversely, the correct answer should be correct under all reasonable interpretations. Specialists follow an item-review process designed to detect and rectify errors.

Once an item has been reviewed, it is ready for pretesting. In the pretesting stage, we assign items to a nonscored section of the test. When they take the test, examinees are not aware of which sections are not scored and which are scored. Thus, we can collect reliable response data on the new items without affecting examinees' scores. We then calibrate the items to estimate the item parameters a_i, b_i, and c_i (Equation (1)) for each of the items i being pretested. In solving the calibration problem, a nonlinear optimization problem, we obtain maximum likelihood estimates for the parameters. If the item behaves as intended, we move it to the next stage, called preoperational assembly.

In preoperational assembly, we create a complete section of a test that conforms to both qualitative and quantitative test specifications. Furthermore, we assemble the preoperational section as part of an entire operational test form. After we assemble a preoperational section, we administer it again to examinees in an unscored section. We conduct another IRT calibration to estimate the item parameters a_i, b_i, and c_i, and compare these estimates to those obtained when the items were administered at the pretesting stage. If at this point the test specialists accept the preoperational section, it is ready for operational use. To assemble an operational test, we combine the successful preoperational test sections into one operational form.

Examiners score all items on an operational test form and report these scores to examinees. Although test specialists try to insure that test forms do not vary substantially in difficulty from one administration to the next (the LSAT is administered four times per year), small fluctuations in test-form difficulty are inevitable. The examiners use a procedure called equating to correct for these fluctuations and to insure that score scales remain stable (Kolen and Brennan 1995). By equating after each test administration, they ensure that one can compare scores across various administrations.

Typically, an item's life cycle ends with its operational use (Figure 3). Once they have been used on an operational test, items may be disclosed to the public (to appear in test-preparation booklets, for example).

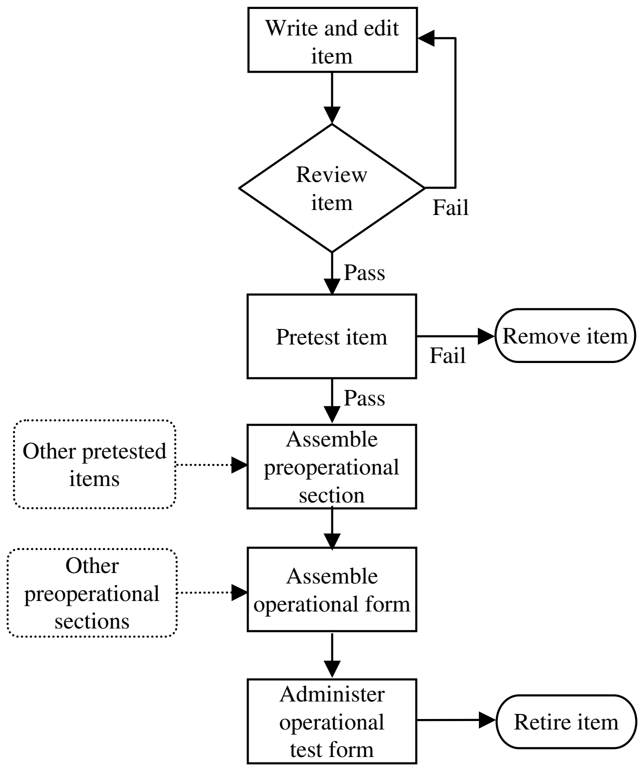
Write and edit item
Review item
Fail
Pass
Pretest item
Fail
Remove item
Pass
Other pretested items
Assemble preoperational section
Other preoperational sections
Assemble operational form
Administer operational test form
Retire item

Figure 3: We illustrate the life cycle of an item in this flowchart. Test specialists first write an item and then review and edit it until it passes a series of quality-control checks. Once an item has been reviewed, it is ready for pretesting, where we assign it to a nonscored section of the test. (Examinees are not aware of which sections are not scored and which are scored.) After pretesting, we estimate item response theory (IRT) parameters for the item. If the item's IRT parameter estimates and other statistical characteristics behave as intended, we move to the next stage where we assemble it with other successfully pretested items into a preoperational section, conforming to both qualitative and quantitative test specifications. We then administer this preoperational section in an unscored section of the test, re-estimate IRT parameters, and compare these new estimates to those obtained at the pretesting stage. If test specialists accept the preoperational section, we assemble this section with other successful preoperational sections to create an operational form. After administration, examiners score all items on an operational test form and report these scores to examinees. We then retire the item once it has been administered operationally.

LSAT Assembler

The LSAT assembler is a part of an integrated system, supporting the full life cycle of an item (Figure 4).

The test assembler reads the properties of passages and items, and the test-assembly specifications from the database. It assembles LSAT forms that satisfy these specifications and saves the forms in the database. The test specialists can use the form reviewer to check forms against additional constraints that were not coded or could not be coded in the test-assembly specifications. Based on this review, the specialists may flag items as inappropriate for a form, calling for reassembly to replace the flagged items.

The pool analyzer reads properties of items and passages, and test-assembly specifications from the database and extracts multiple LSAT forms. It then identifies properties of items that should be developed to create additional LSAT forms.

The Test-Assembly Problem

An item pool consists of pretested items, and the test-assembly process creates test sections for the preoperational stage. After a successful preoperational administration, the sections are combined to form an operational test form. The following constraints are considered for the assembly.

— No duplication: Test forms include items and passages at most once.

— Pretest positioning: Items appearing in an operational section are constrained by the position of the items in pretesting.

— Cognitive skill content: The test form must satisfy a specific distribution of the cognitive skills being tested.

— Item-set specifications: The number of items associated with a passage must fall between a lower bound and an upper bound. Some items must always appear with a passage.

— Word count: The number of words in each section must fall between lower and upper limits.

— Answer-key-count distribution: The distribution of the multiple-choice answer keys is constrained for each section.

— Topic: AR and RC sections must have specified numbers of passages on each topic.

— Diversity: Every RC section must have a specified diversity representation.

— Targets: Each section has specified ranges for information functions and response curves based on IRT.

— Enemies: Some item pairs and passage pairs must not be included on the same test.

— Predicted mean test score and standard deviation: The mean and the standard deviation of the predicted score distribution for the test should be within a specified range.

— Number of items: The number of items on a test form must be within a specified range.

The mixed-integer programming (MIP) model for the assembly of a single LSAT test form is provided in the appendix. Item responses conditional on ability are assumed to be independent.

Monte Carlo Test Assembler

The Monte Carlo test assembler is straightforward in concept:

Step 1. Generate a random sequence of the questions (items).

Step 2. Make sure this sequence satisfies all the constraints, and if it does, save it as a new test; otherwise, return to Step 1.

The challenge with the Monte Carlo technique is to avoid generating many "useless" sequences in Step 1. We have developed several strategies to shrink the search region. We exploit the properties of the constraints, using a divide-and-conquer technique and tabu search, and we prioritize constraint checking based on computational complexity (Appendix). We implemented the search in C/C++ using STL (Musser et al. 2001) as part of the test assembler and item-pool analyzer (Figure 4).

The problem is to assemble a single form from an item pool that has, for example, 1,350 discrete LR items, 110 AR passages with 950 items, and 110 RC passages with 1,030 items. The MIP problem has about 5,050 variables and 2,000 constraints. To solve this problem, we used a desktop personal computer with a Pentium 4 CPU, 2.00 GHz, 1.00 GB of RAM, and Windows XP operating system. The solution time to assemble one test is generally less than two minutes, which matches the performance of a lead-

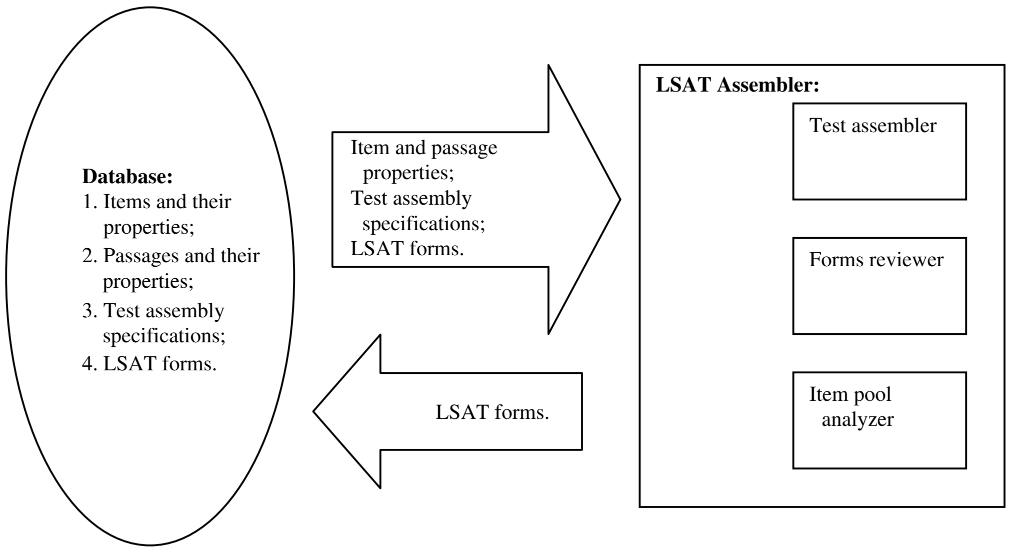
Database:
1. Items and their properties;
2. Passages and their properties;
3. Test assembly specifications;
4. LSAT forms.
Item and passage properties;
Test assembly specifications;
LSAT forms.
LSAT forms.
LSAT Assembler:
Test assembler
Forms reviewer
Item pool analyzer

Figure 4: Here we show the general architecture of the LSAT assembler. Our software consists of two major components, the database and the LSAT assembler. The database manages information about items (IRT parameters, cognitive skills, etc.), passages (number of words, topic, etc.), test-assembly specifications, and assembled LSAT forms. The LSAT assembler includes three applications: (1) the test assembler to assemble LSAT forms, (2) the form reviewer to analyze assembled forms, and (3) the item pool analyzer to generate multiple nonoverlapping forms and to identify properties of new items. The two arrows show the direction and type of information flow.

ing commercial MIP package, CPLEX (ILOG 2002). However, the Monte Carlo approach can easily handle nonlinear constraints and various problems of pool analysis and design that are intractable for an MIP package.

Assembly of Multiple Test Forms

Test specialists often want to assemble many test forms from a single pool of items. Tests that have items or passages in common are called overlapping. In practice, we want to find multiple (ideally, the maximum number of) nonoverlapping tests from the item pool. One approach is to assemble test forms sequentially, with the Monte Carlo approach, while removing any previously used items and passages from the problem. When we used this approach, we obtained, after repeated attempts, at most nine nonoverlapping tests from our item pool. We developed a better approach.

A viable approach for assembling multiple tests is to create many nonoverlapping sections first and then combine them into complete test forms, ensuring that each test satisfies the constraints. Suppose that we can determine the maximum number of nonoverlapping sections for each item type (AR, RC, or LR). Let $\max_{AR}$, $\max_{RC}$, and $\max_{LR}$ represent these numbers. If the number of nonoverlapping test forms equals $\min(\max_{AR}, \max_{RC}, \max_{LR}/2)$, we will have assembled the maximum number of nonoverlapping forms from the item pool. We can then assemble multiple nonoverlapping versions of the LSAT and in some cases verify that we have obtained the maximum number.

We can assemble a large number of feasible overlapping sections from the item pool. However, if we obtain all the feasible sections with unique passage combinations, we can extract the maximum number of nonoverlapping sections from this set of overlapping sections as follows (Figure 5):

(1) Assemble all or many overlapping sections that satisfy the constraints on each of the three item types and save the sections in the database.

(2) Extract the maximum number of nonoverlapping sections from the sections assembled in Step 1. Because several section groupings may yield the maximum number of nonoverlapping sections, we may extract more than one maximum group. Perform this grouping for all three item types and save the extracted sections in a separate database table.

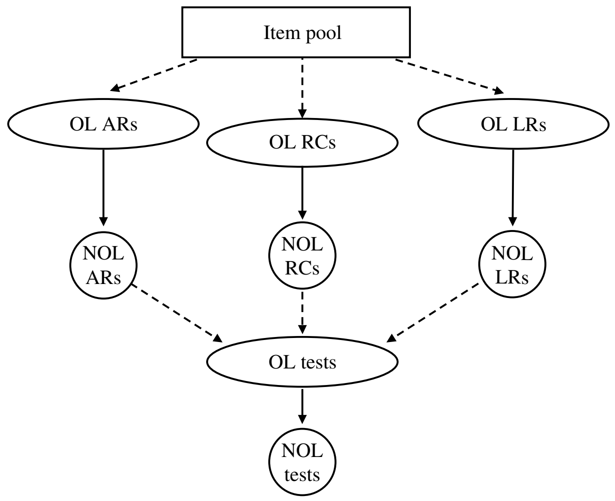

Figure 5: In this diagram for assembling multiple nonoverlapping LSAT forms, the ellipses show overlapping (OL) sections or tests, and the circles show nonoverlapping (NOL) sections or tests. The arrows show the direction of data flow. Each solid arrow indicates solving a corresponding maximum-set-packing problem; each broken arrow indicates assembling sections or composing overlapping tests.

(3) Combine the nonoverlapping sections to create the complete set of feasible overlapping test forms using these sections.

(4) Extract the maximum number of feasible nonoverlapping test forms from the set of forms created in Step 3.

In each step, we solve multiple discrete-optimization problems. We developed an integer-programming (IP) model for extracting the maximum number of nonoverlapping sections (or tests) from a set of overlapping sections (or tests) (appendix). This IP problem is a maximum-set-packing problem (MSP) (Nemhauser and Wolsey 1988). We transform the MSP to a maximum clique problem and solve it with a graph branch-and-bound algorithm (Wood 1997). We implemented this approach as a part of the item-pool analyzer (Figure 4). By solving this problem, we obtained 16 test forms from our pool as opposed to the nine obtained using a sequential-assembly approach.

Model Extensions

Although we could use a commercial MIP solver to assemble tests, it has disadvantages. Problems often become intractable when one enforces nonlinear constraints or we require certain types of analyses. We find Monte Carlo methods beneficial in two selected situations:

(1) A common method for appraising the quality of a test is to measure its reliability, that is, the correlation between the examinee's true ability and his or her observed test score. We can measure the score in various ways, but in practice, we must use a nonlinear expression to derive the correlation. For example, suppose that the score is simply the number of correct responses. To obtain the covariance of the score and true ability, we need the score distribution, which requires a nonlinear function.

(2) The responses to items are not always independent. In particular, items relating to the same passage may be interdependent. The correlation matrix for the item response probabilities creates nonlinearity if we must calculate the expected score or test-information function.

The Monte Carlo test assembler we developed easily supports nonlinear constraints, and we can use it for various analyses of a given item pool and test-form constraints, for example, to extract multiple test forms, to calculate the selection frequency of each item and passage in the overlapping sections, to find the most difficult constraint(s) for test assembly, and to count the violations of the most difficult constraint(s). Test developers rely on all of these analyses to make more robust and productive item pools. A side benefit of the random-search technique is that convergence times are directly related to the match between characteristics of the item pool and specifications for test forms. Because faster assembly times are associated with the presence of greater numbers of potential tests, a comparison of assembly times can be used to appraise different pools as well as test specifications.

Computerized Adaptive Testing

During the 1990s, paper-and-pencil (P&P) tests were supplemented or entirely replaced by computerized adaptive tests (CATs) within many large-scale standardized testing programs, such as the Graduate Record Examination (GRE). CATs have advantages over conventional P&P tests. CAT assemblers determine the items to administer in real time, tailoring each test form to an examinee's ability level. That is, an examinee's responses to items are input to a CAT assembler, and this assembler updates an estimate of the examinee's ability regularly. It can then choose subsequent items that closely match the examinee's ability. By adapting the test to an individual, CAT uses fewer items to acquire more information than a traditional test. Other potential advantages of CATs are immediate scoring, frequent or flexible administrations, monitoring item-response latencies, and the means to implement innovative item formats and types, for example, items employing video and audio technologies. Wainer et al. (1990) provide a general review of CATs.

Although each examinee faces fewer items on a CAT than on an equivalently reliable P&P test, examiners have more serious concerns about item exposure with CATs than with P&P assessments. A typical P&P test form is exposed to many examinees at one administration, then disclosed and not used again. CAT administrations are typically spread over longer periods of time, and items, although not disclosed, are continuously exposed. Therefore, high-stakes CAT administrations may be more vulnerable to compromise than their P&P counterparts.

A conventional CAT form is highly personalized because an adaptive testing algorithm chooses items from an item pool during the test. Because a CAT given to an individual examinee might never be administered again, test scores from different examinees are difficult to equate. Testing agencies construct multiple P&P forms for a standardized test to be parallel and can easily make adjustments for small differences between forms after each administration because of the large number of examinees responding to each form. In addition, the number of correct responses to a conventional CAT has little meaning, and examiners may have difficulty explaining ability estimates to examinees. Before each P&P test administration, test specialists have the opportunity to review the P&P test form. Reviewing a conventional CAT would be infeasible because the number of possible exams increases exponentially with the number of items in the pool. For example, even a 10-item exam from a 20-item bank would give rise to almost 200,000 possible forms without considering item sequencing. In a CAT environment, a testing agency must rely

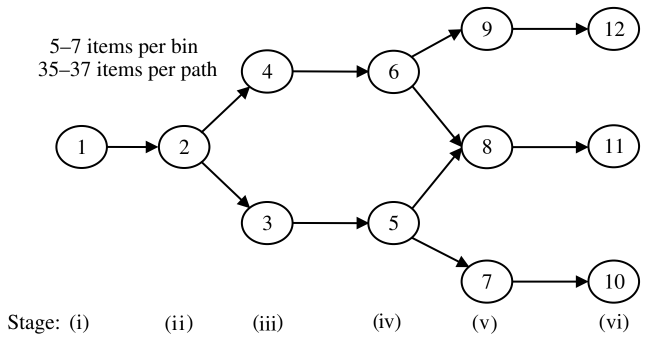

Figure 6: A multiple-stage test (MST) is an ordered collection of testlets, or sets of items bundled together, that allows for adaptation based on an examinee's ability while exposing a predefined number of items and providing a reasonable number of possible forms. In this example of an MST, we show six stages, 12 bins, and four paths. An algorithm makes routing decisions after bins 2, 5, and 6. We assign a testlet of five, six, or seven items to each bin. We constrain the total number of items along any path to be between 35 and 37 items.

on a computer item-selection algorithm to construct exams, which as a result may have subtle similarities or clueing across items.

A multiple-stage adaptive test (MST) (Figure 6) alleviates some of the difficulties described above. An MST is an ordered collection of testlets (sets of items bundled together; see Wainer and Kiely 1987) that allows for adaptation based on an examinee's ability while exposing a predefined number of items and providing a reasonable number of possible forms. This test structure is a hybrid between the conventional P&P and CAT formats. The Law School Admission Council has evaluated the MST approach to testing as a part of its research effort to explore computerized testing for LSAT.

Within a stage, we index the bins using the convention that the lower index number corresponds to a lower ability group. For discussion purposes, we assume that every testlet for our example MST design (Figure 6) contains five to seven items, and each path contains 35 to 37 items. The first two stages (Bin 1 and Bin 2) contain two testlets (of five to seven items each) designed for the complete ability range of the test-taking population. The examinee advances to one of the two bins in the third stage based on the number of correct responses in the first two testlets. A possible method for routing out of Stage (ii) is the following. Proceed to Bin 3 if the total number of correct responses to the items in Bins 1 and 2 is less than 7, and proceed to Bin 4 otherwise. The routing can be based on an estimate of ability instead of the number of correct responses.

Stages (i) and (ii) contain testlets intended for 100 percent $[0, 100]$ of the test-taking population. Stage (iii) has two bins. The testlet in Bin 3 is intended for examinees with ability levels in the bottom 50th percentile $[0, 50]$ of this population and the testlet in Bin 4 is intended for examinees with ability levels in the top 50th percentile $[50, 100]$. Stage (v) has three bins, 7, 8, and 9, intended for testlets that target people of progressively increasing ability. In other words, the testlets assigned to Bin 7 could be created for examinees in the lowest 33rd percentile $[0, 33]$, Bin 8 for the middle 34th percentile $[33, 67]$, and Bin 9 for the top 33rd percentile $[67, 100]$ of the test-taking population.

While we will not describe the MIP model for MST assembly, it follows along the lines of the P&P model, but with constraints imposed along the paths of the MST. The advantages of an MST over a P&P test are similar to those of a CAT over a P&P test, but they are not as pronounced because adaptation occurs less often.

Conclusions

Testing agencies are increasing their use of operations research tools. The Law School Admission Council (LSAC) has used our Monte Carlo assembler since 2002. In addition to saving thousands of personnel hours, all of the test forms assembled meet LSAT specifications. LSAC is a nonprofit organization and is concerned with benefits beyond monetary savings. It has improved the quality of the LSAT and uses the resources previously devoted to manually assembling the LSAT for other services. In addition, the software has greatly improved the usability of the item pool.

The LSAC is using Monte Carlo methods to assemble tests and to analyze item pools. Monte Carlo optimization provides advantages over standard MIP optimization in terms of flexibility for handling nonlinear constraints and monitoring the strengths and weaknesses of an item pool. Testing agency administrators can then direct item writers to focus on creating more items that will allow the assembler to generate more test forms. In this manner, a testing agency may use its item pool more efficiently.

The wide availability of computers and the ability to adapt tests to examinees make computer-based testing the method of the future. We are studying various designs for multiple-stage adaptive tests. Our analysis of these designs uses mathematical programming to assemble tests, and stochastic techniques to develop routing rules and evaluate the designs. We are using simulations to define the path IRT targets.

Many problems in testing and educational measurement can be addressed by using operations research techniques. We used operations research techniques to assemble test forms. However, analysts can also use operations research in designing tests, calibrating items, and sampling examinees from populations of interest.

Appendix

The MIP Model for Assembling a Single LSAT

x_i is a binary variable; $x_i = 1$ if item i is assigned to the test, and $x_i = 0$ if item i is not assigned to the test. There are two logical reasoning sections; thus, each logical reasoning item is conceptually duplicated to allow a distinct index for assigning an item to a section.

y_j is a binary variable; $y_j = 1$ if passage j is assigned to the test, and $y_j = 0$ if it is not assigned to the test. The logical reasoning items are discrete items, and $y_i \equiv x_i$ for these items.

SET is the index set of passages and $ITEM$ is the index set of items. $ITEM(s)$ is the index set of items eligible to be assigned to section s.

LS_j and US_j are the lower and upper limits on the number of items related to passage j that must be used on a test if passage j is used. The set $I(j)$ gives the indices of items related to passage j.

$I^*(j)$ is a subset of $I(j)$; at least one item from $I^*(j)$ must be used if passage j is chosen for the test.

$LCOG_k$ and $UCOG_k$ are the lower and upper limits on cognitive skill k. $CITEM(k)$ is the index set of items testing cognitive skill k, and $NCOG$ gives the number of cognitive skills.

$LPRE_k$ and $UPRE_k$ are the lower and upper limits on pretest position k. $PSET(k)$ is the index set of passages pretested at a position sufficiently close to k, and $NPRE$ gives the number of pretest positions restricted.

$LTOP_k$ and $UTOP_k$ are the lower and upper limits on topic k. $TSET(k)$ is the index set of passages with topic k, and $NTOP$ gives the number of topics restricted.

$LDIV_k$ and $UDIV_k$ are the lower and upper limits on diversity type k. $DSET(k)$ is the index set of passages with diversity type k, and $NDIV$ gives the number of diversity types.

$LWOR_s$ and $UWOR_s$ are the lower and upper word limits on section s. $WSET(s)$ is the index set of passages eligible for section s.

$LKEY_s$ and $UKEY_s$ are the lower and upper limits on the number of correct responses for any answer key in section s. $KITEM(s, \ell)$ is the index set of items having answer key ℓ and eligible for section s.

$E1A(k)$ and $E1B(k)$ are enemy item pairs, and $E2A(k)$ and $E2B(k)$ are enemy passage pairs. The duplicated LR item pairs automatically become enemies of each other. $NEN1$ and $NEN2$ are the number of item and passage enemy pairs, respectively.

$UI_s(\theta_k)$ and $LI_s(\theta_k)$ are, respectively, the upper and lower information function limits for the section s at θ_k. $UR_s(\theta_k)$ and $LR_s(\theta_k)$ are, respectively, the upper and lower response function limits for the section s at θ_k. The value of $I_i(\theta_k)$ and $p_i(\theta_k)$ are, respectively, the information and probability of a correct response for the item associated with item i conditioned on θ_k for the IRT model.

The total number of items on the test is a fixed value given by $NTEST$. Also, the mean score on the test should be in the interval $[LSCORE, USCORE]$. The probability of a correct response to item i, p_i, is calculated numerically assuming a normal distribution of ability.

The constraints for the problem of assembling a single LSAT follow:

$$y_j = 0 \text{ or } 1, \quad j \in SET, \quad x_i = 0 \text{ or } 1, \quad i \in ITEM, \tag{3}$$

$$LS_j y_j \leq \sum_{i \in I(j)} x_i \leq US_j y_j, \quad j \in SET, \tag{4}$$

$$y_j - \sum_{i \in I^*(j)} x_i \leq 0 \quad \forall j \quad \text{where } I^*(j) \neq \text{empty set}, \tag{5}$$

$$LCOG_k \leq \sum_{i \in CITEM(k)} x_i \leq UCOG_k, \quad k = 1, \ldots, NCOG, \tag{6}$$

$$LPRE_k \le \sum_{j \in PSET(k)} y_j \le UPRE_k, \quad k=1,\dots,NPRE, \tag{7}$$

$$LTOP_k \le \sum_{j \in TSET(k)} y_j \le UTOP_k, \quad k=1,\dots,NTOP, \tag{8}$$

$$LDIV_k \le \sum_{j \in DSET(k)} y_j \le UDIV_k, \quad k=1,\dots,NDIV, \tag{9}$$

$$LWOR_s \le \sum_{j \in WSET(s)} w_j y_j \le UWOR_s, \quad s=1,2,3,4, \tag{10}$$

$$LKEY_s \le \sum_{i \in KITEM(s,\ell)} x_i \le UKEY_s, \quad s=1,2,3,4, \quad \ell=1,2,3,4,5, \tag{11}$$

$$x_{E1A(k)} + x_{E1B(k)} \le 1, \quad k=1,\dots,NEN1, \tag{12}$$

$$y_{E2A(k)} + y_{E2B(k)} \le 1, \quad k=1,\dots,NEN2, \tag{13}$$

$$LI_s(\theta_k) \le \sum_{i \in ITEM(s)} I_i(\theta_k) x_i \le UI_s(\theta_k), \quad s=1,2,3,4, \quad k=1,2,\dots,KQ, \tag{14}$$

$$LR_s(\theta_k) \le \sum_{i \in ITEM(s)} p_i(\theta_k) x_i \le UR_s(\theta_k), \quad s=1,2,3,4, \quad k=1,2,\dots,KQ, \tag{15}$$

$$\sum_{i \in ITEM} x_i = NTEST, \tag{16}$$

$$LSCORE \le \sum_{i \in ITEM} p_i x_i \le USCORE. \tag{17}$$

Various objective functions could be considered. A commonly used objective for linear tests is to minimize the distance that the test-information and response functions are from the middle of the lower and upper acceptable limits. Minimizing the sum of the absolute deviations and minimizing the maximum absolute deviation both give rise to linear objective functions. CPLEX and our Monte Carlo approach both have the capability to solve MIPs with quadratic objective functions; therefore, the squared deviation could be considered. Weights could be allocated to the deviations to give less importance to the extreme ability values. However, any solution to the constraints yields an acceptable test in terms of the test specifications. The purpose of the assembly is to produce as many nonoverlapping tests as possible that meet the specifications. The objective function used in the following analysis assigns random costs to the items. The objective function is the following:

$$\text{Minimize} \sum_{i \in ITEM} u_i x_i, \tag{18}$$

where u_i is a uniform random number between 0 and 1.

The Maximum Set Packing (MSP) Model

An MSP model extracts the maximum number of nonoverlapping sections from n overlapping sections. The variable $\alpha_j = 1$ if section j is included in the set of nonoverlapping sections and $\alpha_j = 0$ otherwise. There are m passages in the pool, and the index set $JS(i)$ gives the overlapping sections containing passage i.

$$\text{Maximize} \sum_{j=1}^{n} \alpha_j \tag{19}$$

$$\text{subject to} \sum_{j \in JS(i)} \alpha_j \le 1, \quad i = 1,\dots,m, \tag{20}$$

$$\alpha_j = 0 \text{ or } 1, \quad j = 1,\dots,n. \tag{21}$$

When the problem is to extract the maximum number of nonoverlapping tests from a set of overlapping tests, n is the number of overlapping tests assembled from the nonoverlapping sections, m is the number of nonoverlapping sections, and $JS(i)$ gives the tests containing section i.

Methods to Improve Random Search

The simple idea of a pure random search hints at possible improvement strategies. Consider two sets A and $B \subset A$ (Figure 7), where set A (the search region) consists of all possible combinations of items that satisfy constraint (16), and its subset B consists of all combinations of items resulting in a test form. The pure random search is based on the uniform distribution and converges to a test form with probability $p = |B|/|A|$, where $|\ |$ denotes the size of a

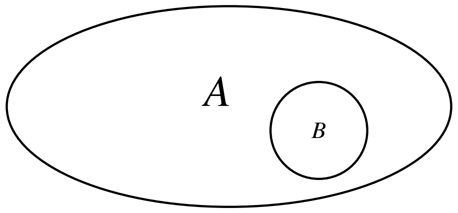

Figure 7: *A* **is the set of all possible combinations of items (search region), and** *B* **is the set of all combinations from** *A* **resulting in a test form.**

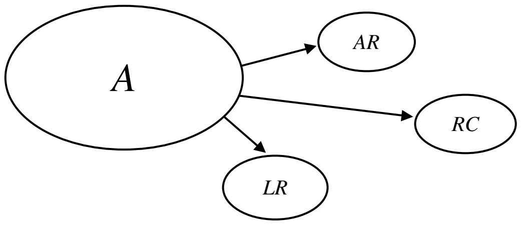

Figure 8: We can reduce A into search regions for the sections of each type (AR, RC, LR). Search region AR(RC, LR) consists of all combinations of analytical reasoning (reading comprehension, logical reasoning) items, where each combination must include a bounded number of items, $|A| \gg |AR| + |RC| + |LR|$.

set. Thus, if we shrink set A without losing combinations from B, then we will increase p and consequently increase the speed of the Monte Carlo test assembly.

We can use three methods to shrink set A:

(1) A test form consists of sections that are sequences of passages, and each passage introduces a group of items that can also be grouped. Taking into account this hierarchical structure of a test form and the divide and conquer principle, we can substantially reduce the size of the search region. This allows us to handle constraints (3–6), (8), and (16) (Figures 8 and 9).

(2) For computationally easy constraints (3–13) and (16), we developed a simple greedy heuristic based on tabu search (Glover et al. 1993). If a random combination of passages/items does not satisfy (3–13) and (16), we move this combination from the passage/item pool to the passage/item tabu region. After we find a combination obeying (3–13) and (16) or exhaust the pool, we move all passages/items from the tabu region back to the passage/item pool (Figure 10).

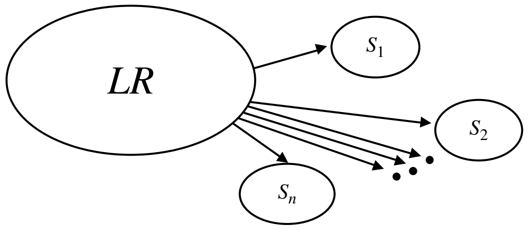

Figure 9: We can reduce search region LR into search regions for the groups of items corresponding to the cognitive skills constraint (6). Here subregion S_i has all combinations of logical reasoning items for the cognitive skill i, and each combination includes a bounded number of items, $|LR| \gg \sum_{i=1}^{n} |S_i|$.

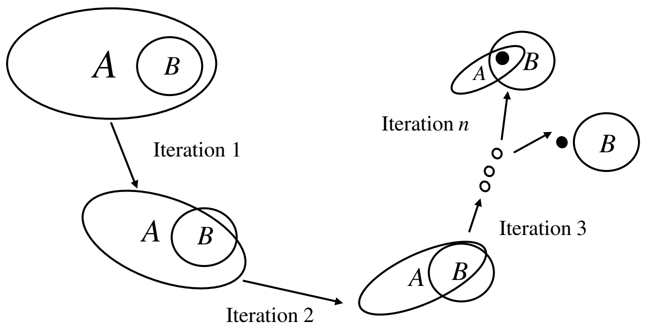

Figure 10: We employ a greedy algorithm to shrink the search region A until we find a combination from B or until A becomes empty.

(3) The third method is based on a constraint-enumeration principle. Let us assume that $R = \{r_1, r_2, \ldots, r_M\}$ and each $r_j \in R$ is a range $r_j = [r_j^L, r_j^U]$. We introduce a function $Enumeration(R, s)$ that generates a vector L of sequences $L_i = \{i_1, i_2, \ldots, i_M\}$ such that

$$\sum_{j=1}^{M} i_j = s \quad \text{and} \quad \forall\, i_j \in L_i \rightarrow i_j \in r_j.$$

In the application, all elements are integer and the resulting vector L has a reasonable size. We automatically satisfy constraints (3–6), (8), (11), and (16) by using $Enumeration(R, s)$. We will use a nonrealistic example to illustrate this principle:

$$NTEST = 10 - \text{number of items per test form.} \quad (22)$$

$$M = 2 - \text{number of sections in a test form.} \quad (23)$$

$$R_1 = [4, 8] - \text{allowed number of items in a first section.} \quad (24)$$

$$R_2 = [3, 7] - \text{allowed number of items in a second section.} \quad (25)$$

Calculate $C = Enumeration(\{R_1, R_2\}, NTEST) = (\{4, 6\}, \{5, 5\}, \{6, 4\}, \{7, 3\})$. By randomly selecting element $\{i_1, i_2\}$ from C, we satisfy constraints (22–25) and reduce the search region to a set of tests consisting of two sections with i_1 and i_2 numbers of items, respectively.

These three methods provide a combination that satisfies constraints (3–13) and (16); then the algorithm checks this combination versus constraints (14), (15), and (17).

Acknowledgments
This research was partially funded by a grant from the Law School Admission Council. Discussions with numerous Law School Admission Council employees led to the model presented here. All authors contributed equally.

References
Glover, F., M. Laguna, E. Taillard, D. de Werra. 1993. Tabu search. *Ann. Oper. Res.* **41**(1) 4–32.

Hambleton, R. K., H. Swaminathan, H. Rogers. 1991. *Fundamentals of Item Response Theory.* Sage Publications, Newbury Park, CA.

ILOG. 2002. *CPLEX 8.0 User's Manual*. Incline Village, NV.

Kolen, M. J., R. L. Brennan. 1995. *Test Equating Methods and Practices*. Springer, New York.

Lord, F. 1980. *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum, Hillsdale, NJ.

Musser, D. R., G. J. Derge, A. Saini. 2001. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*, 2nd ed. Addison-Wesley, Boston, MA.

Nemhauser, G., L. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley and Sons, Hoboken, NJ.

Theunissen, T. J. J. M. 1985. Binary programming and test design. *Psychometrika* **50**(4) 411–420.

van der Linden, W. J. 1998. Optimal assembly of psychological and educational tests. *Appl. Psych. Measurement* **22**(3) 195–211.

van der Linden, W. J. 2000. Optimal assembly of tests with item sets. *Appl. Psych. Measurement* **24**(3) 225–240.

Wainer, H., G. L. Kiely. 1987. Item clusters and computerized adaptive testing: A case for testlets. *J. Educational Measurement* **24**(3) 185–201.

Wainer, H., H. Wagner, N. J. Dorans, R. Flaugher, B. F. Green, R. J. Mislevy, L. Steinberg, D. Thissen. 1990. *Computerized Adaptive Testing: A Primer*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Wood, D. 1997. An algorithm for finding a maximum clique in a graph. *Oper. Res. Lett.* **21**(5) 211–217.

Peter J. Pashley, Principal Research Scientist and Director of Testing and Research, Law School Admission Council, Box 40, Newtown, Pennsylvania 18940-0040, writes: "This letter is an endorsement of the paper 'Developing and Assembling the Law School Admission Test' by Ronald Armstrong, Dmitry Belov, and Alexander Weissman. We have been using a mixed-integer programming technique to help assemble the Law School Admission Test (LSAT) for the past five years, saving many hours of personnel time. Although test specialists review the assembled tests before administration, typically only minor modifications are necessary. When assembled, these tests match the technical specifications set for the LSAT. The mixed-integer programming approach also assists in this review process by identifying acceptable changes. In addition, we are currently exploring the use of multiple stage adaptive tests as a potential computer-based testing approach. These investigations use operations research techniques such as mixed-integer programming, simulation, and stochastic processes.

"While it is difficult to place a monetary value on the increased efficiency obtained by using these operations research methods, it is substantial. These methods have expanded our ability not only to assemble tests, but also to evaluate them and explore possible future administration methods and testing designs."