Large-Scale Multiobjective Static Test Generation for Web-Based Testing with Integer Programming

Minh Luan Nguyen, *Member*, *IEEE*, Siu Cheung Hui, and Alvis C.M. Fong, *Senior Member*, *IEEE*

Abstract—Web-based testing has become a ubiquitous self-assessment method for online learning. One useful feature that is missing from today's web-based testing systems is the reliable capability to fulfill different assessment requirements of students based on a large-scale question data set. A promising approach for supporting large-scale web-based testing is static test generation (STG), which generates a test paper automatically according to user specification based on multiple assessment criteria. And the generated test paper can then be attempted over the web by users for assessment purpose. Generating high-quality test papers under multiobjective constraints is a challenging task. It is a 0-1 integer linear programming (ILP) that is not only NP-hard but also need to be solved efficiently. Current popular optimization software and heuristic-based intelligent techniques are ineffective for STG, as they generally do not have guarantee for high-quality solutions of solving the large-scale 0-1 ILP of STG. To that end, we propose an efficient ILP approach for STG, called branch-and-cut for static test generation (BAC-STG). Our experimental study on various data sets and a user evaluation on generated test paper quality have shown that the BAC-STG approach is more effective and efficient than the current STG techniques.

Index Terms—Web-based testing, static test generation, multiobjective optimization, integer programming

1 Introduction

With the rapid evolution of the web, web-based education has advanced significantly over the last 20 years and become a ubiquitous learning platform in many institutions to provide students with online learning courses and materials. Currently, we are also seeing more freely accessible educational websites together with learning technologies [1] being developed to support web-based education. Such websites aim to bring free education to the world by providing online contents, exercises, and quizzes such as Khan Academy,[1] or online classes such as Coursera,[2] and Udacity.[3] The large data sets of online materials have been created and evolved over time. Different from passive course archives like MIT OpenCourseWare,[4] the online classes are interactive and can assess learners automatically on what they have learned. The main benefit is that learners can take classes at their own pace and get immediate feedback on their proficiency, unlike traditional classes.

1. http://www.khanacademy.org/.
2. https://www.coursera.org/.
3. http://www.udacity.com/.
4. http://ocw.mit.edu/index.htm.

- *M.L. Nguyen and S.C. Hui are with the School of Computer Engineering, Nanyang Technological University, Block N4, B3c, DISCO Lab, 50 Nanyang Avenue, Singapore 639798. E-mail: {NGUY0093, asschui}@ntu.edu.sg.*
- *A.C.M. Fong is with the School of Computing and Math Sciences, Auckland University of Technology, New Zealand. E-mail: acmfong@gmail.com.*

Manuscript received 24 May 2012; revised 13 Sept. 2012; accepted 24 Nov. 2012; published online 29 Nov. 2012.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2012-05-0075.
Digital Object Identifier no. 10.1109/TLT.2012.22.

Web-based testing has been popularly used for automatic self-assessment especially in a distance educational learning environment [2], [3]. However, there is a problem on conducting self-assessment in an online class. As there may have many students[5] with different proficiency levels in an online class [4], it is difficult to fulfill different assessment requirements of students if using tests composed from a small question pool [5]. To overcome this problem, pedagogical practitioners have suggested composing tests from a large question pool with different question properties [6]. This in turn requires the availability of a large question data set and huge human effort on composing the tests to assess students' proficiency.

One promising approach to support large-scale web-based testing is static test generation (STG), which generates a test paper automatically according to user specification based on multiple assessment criteria. Here, the term "static test" refers to traditional test paper in psychometry [7]. Fig. 1 shows a typical workflow of a web-based testing environment with automatic assessment. In this environment, STG is the core component, which aims to find an optimal subset of questions from a question database to form a test paper automatically based on multiple assessment criteria such as total time, topic distribution, difficulty degree, discrimination degree, and so on. And the generated test paper can then be attempted over the web by students for assessment purpose as in traditional pen-and-pencil test. Finally, the students' answers will be checked automatically for proficiency evaluation.

Generating high-quality test papers that satisfy the constraints and maximize the assessment objective is critical

5. http://wp.sigmod.org/?p=165.

1939-1382/13/$31.00 © 2013 IEEE Published by the IEEE CS & ES

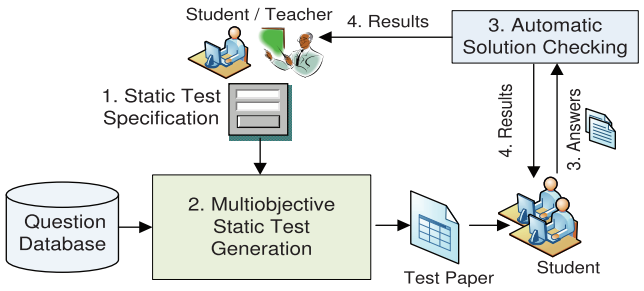


Fig. 1. Web-based testing workflow with automatic assessment.

for formal tests and examinations [8]. However, it is a challenging problem especially with a large number of questions [9]. Manually browsing and composing test papers by users is ineffective because of the exponential number of feasible combinations of questions. In essence, STG is an optimal subset selection problem, called a multidimensional knapsack problem (KP) [10], [11], which is also NP-hard [9]. Formally, it is a 0-1 integer linear programming (ILP), which optimizes multiobjective constraints. Moreover, STG should also be solved efficiently for online requirement. Currently, the quality of generated test papers are often unsatisfactory [12], [13], [14] according to users' test paper specifications.

One of the main issues of STG is the very large search space of possible candidates with multiobjective constraints. In the early 1980s, linear programming-based ILP [15], [16] was proposed to solve STG on very small question data sets. Popular up-to-date commercial optimization software packages such as CPLEX [17] and GUROBI [18] are inefficient for 0-1 ILP of STG because of its large number of variables in the 0-1 ILP formulation [19]. Recently, many heuristic-based intelligent techniques such as tabu search (TS) [13], biologically inspired algorithms [14], [20], swarm optimization [12], [21], [22] and divide and conquer (DAC) [23] have been proposed in the research community for automatic test paper generation. Although these heuristic-based techniques are straightforward to implement, they suffer from some drawbacks. These techniques are mainly based on traditional weighting parameters for multiobjective constraint optimization. They tend to get stuck in a local optimal solution especially in a huge search space of large-scale question data sets. As a result, these techniques generally do not have performance guarantee on both test paper quality and runtime efficiency.

In this paper, we propose an efficient 0-1 ILP approach for high-quality STG, called branch-and-cut for STG (BAC-STG). Generally, there exists many topics (e.g., differentiation, integration, etc.) in a subject (e.g., mathematics). When the STG problem is formulated in 0-1 ILP for a large question data set, it has the sparse matrix property. The proposed BAC-STG approach is based on the branch-and-bound method with the lifted cover cutting method for solving the 0-1 ILP by exploiting the sparse matrix property. As branch and bound is a global and parameter-free method to deal with multiple constraints of the STG, the proposed approach avoids getting stuck in local optimal solutions to achieve high-quality test papers as well as eliminates the need of using weighting parameters as in heuristic-based techniques. Our approach can be considered as an extension of previous work [15], [16] by taking advantages of the recent advancement in optimization techniques. Specifically, we have made the following two contributions in this paper:

- We propose an effective and efficient ILP approach for STG, which generates high-quality test papers in a huge search space of large question data sets efficiently. This was not possible in the past. Our proposed BAC-STG approach is able to support web-based testing on large question data sets for online learning environments. Our performance results on various data sets have shown that the proposed BAC-STG approach has outperformed the current STG techniques in terms of paper quality and runtime efficiency.
- We propose a novel framework for web-based testing with automatic assessment, in particular for mathematics testing. The proposed framework integrates the proposed BAC-STG approach for automatic test paper generation, automatic mathematics solution checking, and automatic question calibration. It is able to generate test papers automatically and provide students with immediate feedback on their performance.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 describes the problem specification of STG. Section 4 presents the proposed BAC-STG approach. Section 5 shows the performance results of the BAC-STG approach and its comparison with other STG techniques. Section 6 gives the proposed web-based testing framework. Finally, Section 7 concludes the paper.

2 Related Work

2.1 Automatic STG

There are two major paradigms for web-based testing: STG [7] and computerized adaptive testing (CAT) [24]. STG generates full test papers automatically based on multiple assessment criteria, whereas CAT generates question-by-question tests in a dynamic and sequential manner according to student's ability and item response theory (IRT). STG is basically a multiobjective combinatorial optimization problem, whereas CAT is a sequential optimization problem [25]. In this section, we focus only on reviewing related work on STG, which can be categorized into two main groups: linear programming-based integer programming and heuristic-based methods.

LP-based IP, which was proposed in 1986 by Adema et al. [15], [16], used the LANDO program to solve the 0-1 ILP of STG. It is similar to our proposed approach because of the use of linear programming (LP) and branch and bound. In [26], [27], Boekkooi-Timminga attempted to combine ILP with heuristics to improve runtime performance for multiple test paper generation. Although these approaches have rigorous mathematical foundations on optimization, they can only solve STG for very small data sets of about 300-600 questions due to the limitations of the state-of-the-art optimization methods at that time. An in-depth review of the LP-based IP for STG can be found in [28].

For heuristic-based methods, Theunissen [29] used a heuristic based on the characteristics of question item information function to optimize the objective function. Later, Luecht [30] proposed an efficient heuristic to solve STG on a data set with 3,000 questions. However, these heuristic-based methods were proposed to solve STG for small data sets and are ineffective for larger data sets.

Since 2003, there has been a revived interest for STG on larger data sets of about 3,000-20,000 questions by using modern heuristic methods. In [9], TS was proposed to construct test papers by defining an objective function based on multicriteria constraints and weighting parameters for test paper quality. TS optimizes test paper quality by evaluating the objective function. In [13], a genetic algorithm (GA) was proposed to generate quality test papers by optimizing a fitness ranking function based on the principle of population evolution. In [14], differential evolution (DE) was proposed for test paper generation. DE is similar to the spirit of GA with some modifications on solution representation, fitness ranking function, and the crossover and mutation operations to improve the performance. In [20], an artificial immune system was proposed to use the clonal selection principle to deal with the highly similar antibodies for elitist selection to maintain the best test papers for different generations. In [21], particle swarm optimization (PSO) was proposed to generate multiple test papers by optimizing a fitness function which is defined based on multicriteria constraints. In [12], ant colony optimization (ACO) was proposed to generate quality test papers by optimizing an objective function that is based on the simulation of the foraging behavior of real ants. Apart from these techniques for STG, an efficient DAC approach [23] was proposed for online STG, which is based on the principle of dimensionality reduction for multiobjective constraint optimization.

To optimize the multiobjective criteria of test paper quality, the current STG techniques (except DAC) require weighting parameters and some other parameters such as population size, tabu length, and so on, for each test paper generation that are not only difficult but also computational expensive to determine. Hence, these techniques generally take long runtime for generating good quality test papers especially for large data sets of questions.

2.2 0-1 Integer Programming

The 0-1 ILP [10], [11] has been extensively studied for solving various real-world problems such as the traveling salesman problem, quadratic assignment problem, maximum satisfiability problem (MAX-SAT), KP, and so on. Specifically, the 0-1 ILP is a mathematical optimization program in which all of the variables are restricted to be binary:

$$\begin{aligned} &\text{maximize } \mathbf{c}x \\ &\text{subject to } \mathbf{A}x \leq \mathbf{b}, x \in \{0,1\}^n, \end{aligned}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A = [a_{ij}]_{m \times n}$ is a $m \times n$ matrix with $a_{ij} \in \mathbb{R}$, m is the number of constraints, and n is the number of variables or dimensions.

Solving a general 0-1 ILP problem is NP-hard. Despite this fact, there are fast solvers available today providing practical solutions for many 0-1 ILP problems. The performance depends on the dimensions n and degree of sparsity of the constraint matrix A. According to [11], there are four main methods for solving 0-1 ILP including heuristic algorithms, cutting planes method, branch-and-bound, and branch-and-cut (BAC). As mentioned earlier, although heuristic algorithms can be applied quite straightforwardly to solve many 0-1 ILP problems, they do not have any performance guarantee. The remaining three methods are global methods, which can find the exact optimal solution based on LP for 0-1 ILP problems.

The performance of these global methods depends on the algorithms used for LP, preprocessing techniques, and computational processing power of the computer hardware. In the early 1990s, there is not much improvement on the simplex algorithm for LP. Since the early 2000s, the development of the dual simplex algorithm and other techniques such as lifted cover cutting planes [31] have remarkably improved integer programming techniques [32]. The runtime performance has been improved significantly. Currently, a large ILP of about 18,000 variables can be solved in less than 3 minutes. However, the LP-based ILP is still not efficient in runtime performance especially for large-scale 0-1 ILP problems. In particular, the methods implemented in popular commercial optimization software such as CPLEX [17] and GUROBI [18] are ineffective to handle 0-1 ILP with more than twenty thousand of variables [19].

Among the three methods, BAC [19] is most efficient as it is able to solve and prove optimality for larger set of instances than the others. BAC is a global optimization method, which is based on the branch-and-bound method and cutting planes method such as the Gomory or Fenchel cutting planes [11]. The main idea of *cutting planes* method is to add extra constraints to reduce the feasible region and find the integral optimal solution. For 0-1 ILP problems with the sparse matrix property, lifted cover cutting is an effective method for enhancing runtime performance. However, the BAC method suffers from several drawbacks when solving large-sized 0-1 ILP problems. It is difficult to approximate the integral optimal solution from the fractional optimal solution of the 0-1 ILP problem. In addition, the simplex algorithm used to solve LP relaxation is also not very efficient on large-sized ILP problems. As BAC is an exact algorithm, the size of the branch-and-bound search tree may combinatorially explode with the number of variables. Hence, BAC generally suffers from poor runtime performance on large-sized ILP problems. Moreover, finding lifted cover cutting planes efficiently is challenging as it is NP-hard [31].

2.3 Discussion

From the above discussion, we have the following observations. First, we note that the objective functions of the STG formulation in the related studies may be different. It maximizes either the test information function [7], [28] or the average discrimination degree [12], [13]. Although they are different, the discrimination degree is easier to calibrate and thus preferred in practice by researchers than the information function. However, it is not important because the STG problems can be solved in either way using our proposed approach. Second, heuristic techniques are ineffective for large-scale STG, as they generally do not have guarantee for high-quality solutions. Third, although the current LP-based

TABLE 1
An Example of Math Data Set

(a) Question Table

$\mathcal{Q}$_id	o	a	e	t	d	c	y
q_1	...	...	5	5	5	c_1	y_1
q_2	...	...	6	10	7	c_2	y_2
q_3	...	...	7	6	7	c_1	y_1
q_4	...	...	7	12	6	c_2	y_2
q_5	...	...	7	14	9	c_1	y_2
q_6	...	...	7	10	9	c_3	y_2
q_7	...	...	6	8	4	c_3	y_3
q_8	...	...	6	8	4	c_3	y_3

(b) Topic Table

$\mathcal{C}$_id	$name$
c_1	Integration
c_2	Differentiation
c_3	Limits

(c) Question Type Table

$\mathcal{Y}$_id	$name$
y_1	Fill-in-the-blank
y_2	Long question
y_3	Multiple choice

ILP approach [15], [16] has quality guarantee for STG, the popular optimization software such as CPLEX and GUROBI are unable to solve large-scale 0-1 ILP problems efficiently [33]. In this paper, we propose an efficient integer programming approach for solving large-scale 0-1 ILP of the STG problem by exploiting the sparse matrix property.

3 Problem Specification

3.1 Question Data Set

Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ be a data set consisting of n questions, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ be a set of m different topics, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ be a set of k different question types. Each question $q_i \in \mathcal{Q}$, where $i \in \{1, 2, \ldots, n\}$, has eight attributes $\mathcal{A} = \{q_{id}, o, a, e, t, d, c, y\}$ defined as follows:

- *Question* q_{id}. It is used to store the question identity.
- *Content* o. It is used to store the content of a question.
- *Answer* a. It is used to store the answer of a question.
- *Discrimination degree* e. It is used to indicate how good the question is in order to distinguish user proficiency. It is an integer ranging from 1 to 7.
- *Question time* t. It is used to indicate the average time needed to answer a question. It is measured in minutes.
- *Difficulty degree* d. It is used to indicate how difficult the question is to be answered correctly. It is an integer ranging from 1 to 10.
- *Related topic* c. It is used to store a set of related topics of a question.
- *Question type* y. It is used to indicate the type of a question. There are mainly three question types, namely fill-in-the-blank, multiple choice, and long question.

Note that the discrimination degree and difficulty degree attributes here refer to the classical IRT definitions. Table 1 shows a sample Math question data set.

There are two possible ways to construct large-scale question data sets for web-based testing. It can be constructed by gathering questions from past tests and examinations on subjects such as TOEFL and GRE[6] accumulatively. Moreover, it can also be constructed by gathering freely available questions from online educational websites such as Khan Academy or Question Answering (Q&A) websites such as The Art of Problem Solving Portal.[7]

The large pool of questions has posed a great challenge on labeling all question attributes accurately and automatically. In this paper, we assume that question attributes are correctly calibrated. However, with the advancement in educational data mining techniques [33], it might be feasible to automatically label all the attributes of each question with little human effort in the future. Automatic text categorization techniques such as support vector machine can be used for automatic topic classification of questions [34]. However, human labeling on topics for training questions is still needed in the training phase. To calibrate the other attributes, we can use the historical correct/incorrect response information from students. These response information as well as other important information such as question time can be gathered automatically through the students' question answering activities [35] over a period of time. However, it is more difficult to calibrate the discrimination degree and difficulty degree attributes due to missing user responses on certain questions. To overcome this, it is possible to apply the collaborative filtering technique to predict missing user responses and use the IRT model to calibrate the two attributes automatically [36]. Moreover, in [36], it has also proposed an effective method to calibrate new questions, which do not have any student response information. As such, automatic labeling of question attributes for large-scale question data sets can be achieved.

3.2 Static Test Specification

A *static test specification* $\mathcal{S} = \langle N, T, D, C, Y \rangle$ is a tuple of five attributes which are defined based on the attributes of the selected questions as follows:

- *Number of questions* N. It is an input representing the number of questions specified for the paper.
- *Total time* T. It is the total time specified for the paper.
- *Average difficulty degree* D. It specifies the average difficulty degree of all questions in the paper.
- *Topic distribution* C. It specifies the proportion of topics. The user can enter either the proportion or the number of questions for each topic. If the number of questions is entered, then the number will be converted into the corresponding proportion.
- *Question type distribution* Y. It specifies the proportion of question types. The user can enter either the proportion or the number of questions for each question type. Similarly, if the number of questions is entered, then the number will be converted into the corresponding proportion.

3.3 Optimal STG

Given a static test specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$, where N is the number of questions, T is the total time, D is the average difficulty degree, $C = \{(c_1, pc_1), (c_2, pc_2), \ldots, (c_M, pc_M)\}$ is the topic distribution, and $Y = \{(y_1, py_1), (y_2, py_2), \ldots, ((y_K, py_K)\}$ is the question type distribution. The

6. http://www.ets.org.

7. http://www.artofproblemsolving.com/Forum/portal.php?ml=1.

$$
\begin{array}{lll}
maximize & \sum_{i=1}^{n} e_i x_i / \sum_{i=1}^{n} x_i & \text{(Average Discrimination Degree)} \\
subject\ to & \sum_{i=1}^{n} x_i = N & (1)\ \text{(Number of Questions)} \\
 & \sum_{i=1}^{n} t_i x_i = T & (2)\ \text{(Total Time)} \\
 & \sum_{i=1}^{n} d_i x_i / \sum_{i=1}^{n} x_i = D & (3)\ \text{(Average Difficulty Degree)} \\
 & \sum_{i=1}^{n} r_{il} x_i / \sum_{i=1}^{n} x_i = pc_l \quad \forall l = 1..M & (4)\ \text{(Topic Distribution)} \\
 & \sum_{i=1}^{n} s_{ij} x_i / \sum_{i=1}^{n} x_i = py_j \quad \forall j = 1..K & (5)\ \text{(Question Type Distribution)} \\
 & x \in \{0,1\}^n & (6)\ \text{(Binary Value)}
\end{array}
$$

Fig. 2. The 0-1 fractional ILP formulation of STG.

STG process aims to find a subset of questions from a question data set $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ to form a test paper P with specification $\mathcal{S}_P$ that maximizes the average discrimination degree and satisfies the static test specification such that $\mathcal{S}_P = \mathcal{S}$.

Based on the user test specification $\mathcal{S}$ and the question attributes, the STG problem can be formulated as a 0-1 fractional ILP problem [11] as shown in Fig. 2. In Fig. 2, constraint (1) is the constraint on the number of questions, where $x_i \in \{0,1\}$ is a binary variable associated with question $q_i, i = 1..n$, in the data set. Constraint (2) is the total time constraint. Constraint (3) is the average difficulty degree constraint. Constraint (4) is the topic distribution constraint. The relationship of a question $q_i, i = 1..n$, and a topic $c_l, l = 1..M$, is represented as r_{il} such that $r_{il} = 1$ if question q_i relates to topic c_l and $r_{il} = 0$ otherwise. Constraint (5) is the question type distribution constraint. The relationship of a question $q_i, i = 1..n$, and a question type $y_j, j = 1..K$, is represented as s_{ij} such that $s_{ij} = 1$ if question q_i is related to question type y_j and $s_{ij} = 0$ if otherwise.

4 Proposed BAC-STG Approach

In this section, we propose an efficient 0-1 ILP approach for high-quality STG, called BAC-STG. When the STG problem is formulated in 0-1 ILP, we observe that it has the sparse matrix property. By exploiting the sparse matrix property and domain-specific property of the STG problem, the proposed approach combines the branch-and-bound method with the lifted cover cutting method to efficiently solve the large-sized 0-1 ILP of the STG problem. The proposed BAC-STG approach has the following important characteristics:

- When the 0-1 ILP problem has the sparse matrix property, the proposed approach is able to approximate the binary optimal solution of the 0-1 ILP problem with the fractional optimal solution.
- The proposed approach uses the primal-dual interior point (PDIP) [37] which is the most efficient algorithm for solving the LP relaxation problem. In addition, the simplex method [11] is also used for solving the LP relaxation problem efficiently in subsequent steps of the approach when new cutting planes are added.
- An effective branching strategy is proposed for reducing the size of the branch-and-bound search tree.
- An efficient approach is proposed for finding effective lifted cover cutting planes.

4.1 0-1 ILP Formulation

In the proposed BAC-STG approach, we first reformulate the 0-1 fractional ILP of the STG problem into a standard 0-1 ILP, which is given in Fig. 3. Note that as the number of questions N is a constant, the denominator of the maximizing cost function can be eliminated from the fractional ILP during re-formulation. In addition, as each question has only a few related topics and a question type, most of the coefficients in the topic constraint and question type constraint are zeros. Thus, for large-sized STG problems, the matrix A of the 0-1 ILP is very sparse.

4.2 Branch and Bound

The branch-and-bound method is based on the DAC strategy, which iteratively partitions the original ILP problem into a series of subproblems. Each subproblem is then solved by LP relaxation to obtain an upper bound on its objective value. The key idea of the branch-and-bound method is that if the upper bound for the objective value of a given subproblem is less than the objective value of a known integer feasible solution, then the given subproblem does not contain the optimal solution of the original ILP problem. Hence, the upper bounds of subproblems are used to construct a proof of optimality without exhaustive search. Fig. 4 shows the main steps of the branch-and-bound method in the BAC-STG approach.

In BAC-STG, the subproblems are organized as an *enumeration tree* that is constructed iteratively in a top-down manner with new nodes created by branching on an existing node in which the optimal solution of the LP relaxation is fractional. The problem at the root node of the tree is the original 0-1 ILP. When a new node N^i is created, it contains the corresponding 0-1 ILP subproblem and is stored in the list $\mathcal{L} = \{N^1, \ldots, N^n\}$, $i \in \{1..n\}$, of all unevaluated or leaf nodes. Let F^i be the formulation of the feasible region of the problem at node N^i. Let z^i_{ub} be the

$$
\begin{array}{lll}
maximize & \sum_{i=1}^{n} e_i x_i & \\
subject\ to & \sum_{i=1}^{n} x_i = N & (7) \\
 & \sum_{i=1}^{n} t_i x_i = T & (8) \\
 & \sum_{i=1}^{n} d_i x_i = \lfloor DN \rfloor & (9) \\
 & \sum_{i=1}^{n} r_{il} x_i = \lfloor pc_l N \rfloor \quad \forall l = 1..M & (10) \\
 & \sum_{i=1}^{n} s_{ij} x_i = \lfloor py_j N \rfloor \quad \forall j = 1..K & (11) \\
 & x \in \{0,1\}^n & (12)
\end{array}
$$

Fig. 3. The 0-1 ILP formulation of STG.

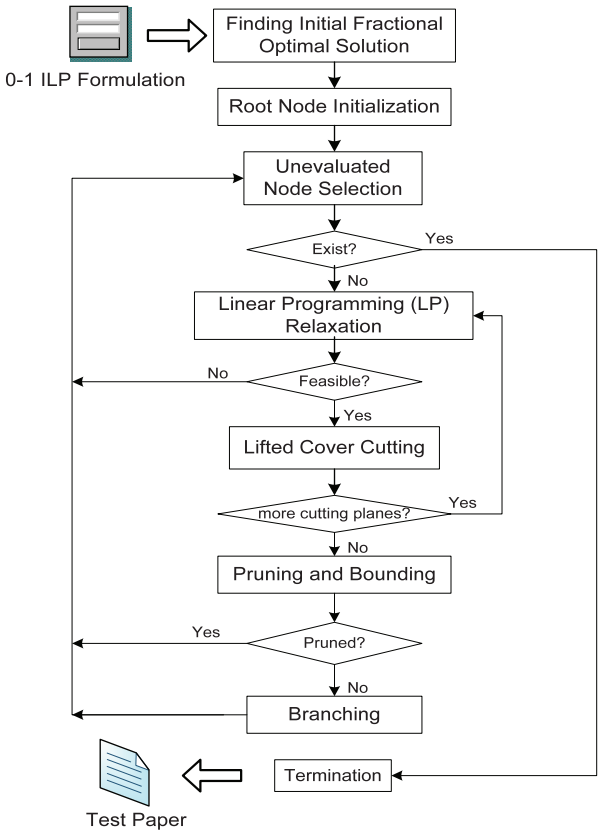


Fig. 4. Branch-and-bound flowchart of the BAC-STG.

local upper bound at each node N^i and z_{lb} be the current global lower bound of the 0-1 ILP solution.

4.2.1 Finding Initial Fractional Optimal Solution

In this step, we find the fractional optimal solution x^* of the original 0-1 ILP problem. This is done by relaxing the constraints on binary value of variables. The 0-1 ILP formulation of the STG problem shown in Fig. 3 is transformed into a standard LP as follows:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax = b, 0 \le x \le 1 \end{array}$$

or equivalently: $\text{maximize}\ \{cx|x \in F\}$, where $F = \{x \in \{0,1\} : Ax \le b\}$ denotes the constraint set of feasible regions of the original ILP problem.

The LP problem can then be solved by using the most efficient PDIP algorithm [37]. PDIP solves the LP problem by resolving the following logarithmic barrier optimization problem:

$$\begin{aligned} &\text{maximize}\ cx - \frac{1}{t}\sum_{i=1}^{n}(\log x_i + \log(1 - x_i)) \\ &\text{subject to}\ Ax = b, \end{aligned}$$

where t is a barrier parameter.

The optimal solution x^* of PDIP will be used to construct the corresponding tableau of the simplex method [11]. It consists of two steps: initial tableau construction and simplex tableau construction.

In the *initial tableau construction*, the simplex algorithm works on inequalities of the form $\sum a_{ij}x_j \le b_i$ and the 0-1 ILP of the STG problem needs to satisfy the equality constraints given in (7)-(12) of the form $\sum a_{ij}x_j = b_i$. Thus, we replace each constraint of the form $\sum a_{i,j}x_j = b_i$ by the following two constraints: $\sum a_{ij}x_j \le b_i$ and $-\sum a_{ij}x_j \le -b_i$. So far, all the replaced constraints given in (7)-(12) are now in the form $\sum a_{ij}x_j \le b_i$. By introducing new slack variables, we have the following *initial tableau*: $\{\text{maximize}\ cx \mid Ax + s = b\}$, where s is the vector of slack variables.

In the *simplex tableau construction*, we perform pivoting operations on the initial tableau such that all variables x_j with $x_j^* > 0$ are basic variables, whereas others are nonbasic variables. As a result, the optimal solution x^* and its corresponding simplex tableau $\mathcal{T}$ of the form $x_i + \sum a_{i,j}x_j = b_i$ are obtained.

4.2.2 Root Node Initialization

It first creates the root node N^1 of the enumeration tree that contains the original 0-1 ILP problem with its fractional optimal solution x^* and simplex tableau $\mathcal{T}$. Next, it initializes the local upper bound at N^1 as $z_{ub}^1 = \text{maximize}\{cx^* : x^* \in F^1\}$, the global lower bound $z_{lb} = -\infty$ and the current best 0-1 solution $x_{best} = \emptyset$. Then, the root node is stored in the list $\mathcal{L}$ for further processing.

4.2.3 Unevaluated Node Selection

This step selects an unevaluated node in the list $\mathcal{L}$ for processing and solving. If there is no unevaluated node, the algorithm will terminate. Otherwise, a node in the list $\mathcal{L}$ will be selected. Here, we use a greedy strategy, namely *best bound*, to choose the most promising node N^i in $\mathcal{L}$ with the largest local upper bound value z_{ub}^i:

$$N^i = \underset{N^i \in \mathcal{L}}{\text{argmax}}\ z_{ub}^i.$$

4.2.4 LP Relaxation

It iteratively solves the subproblem of the selected unevaluated node N^i based on the optimal solution and simplex tableau of its parent node's problem (except the root node). At the kth iteration of processing a node N^i, it solves the following LP problem: $z_{ub}^{i,k} = \text{maximize}\{cx : x \in F^{i,k}\}$. If the returned result is infeasible (i.e., when $F^{i,k} = \emptyset$), it ignores this node and continues processing another node in the list $\mathcal{L}$. Otherwise, it goes to the next step on lifted cover cutting for adding cutting planes. Note that for efficiency, it adds the new constraints into the simplex tableau of its parent node and continue reoptimizing this tableau. After solving the LP Relaxation at node N^i, the fractional optimal solution x^* and its corresponding simplex tableau are obtained.

4.2.5 Lifted Cover Cutting

The main purpose of the lifted cover cutting is to add extra constraints, called *cutting planes*, to reduce the feasible region and approximate the binary optimal solution, which is nearest to x^*. Based on the current fractional optimal solution x^* and its corresponding simplex tableau, this step helps LP relaxation to gradually approximate more closely to the binary optimal solution x_{best} of the subproblem. It adds extra constraints or cutting planes into the current subproblem. To achieve this, it adds some *lifted cover inequalities* to the current formulation $F^{i,k}$ such that a new

$$
\begin{aligned}
\text{maximize} \quad & \frac{5x_1+6x_2+7x_3+7x_4+7x_5}{x_1+x_2+x_3+x_4+x_5} && \text{(Average Discrimination Degree)}\\
\text{subject to} \quad & x_1+x_2+x_3+x_4+x_5=2 && \text{(Number of Questions)}\\
& 5x_1+10x_2+6x_3+12x_4+14x_5=15 && \text{(Total Time)}\\
& \frac{5x_1+7x_2+7x_3+6x_4+9x_5}{x_1+x_2+x_3+x_4+x_5}=6 && \text{(Average Difficulty Degree)}\\
& \frac{x_1+x_3+x_5}{x_1+x_2+x_3+x_4+x_5}=0.5 && \text{(Topic Distribution)}\\
& \frac{x_1+x_3}{x_1+x_2+x_3+x_4+x_5}=0.5 && \text{(Question Type Distribution)}\\
& x_i\in\{0,1\}, i=1,2,3,4,5 && \text{(Binary Value)}
\end{aligned}
$$

(a) The 0-1 Fractional ILP Formulation Example

$$
\begin{aligned}
\text{maximize} \quad & 5x_1+6x_2+7x_3+7x_4+7x_5\\
\text{subject to} \quad & x_1+x_2+x_3+x_4+x_5=2\\
& 5x_1+10x_2+6x_3+12x_4+14x_5=15\\
& 5x_1+7x_2+7x_3+6x_4+9x_5=12\\
& x_1+x_3+x_5=1\\
& x_1+x_3=1\\
& x_i\in\{0,1\}, i=1,2,3,4,5
\end{aligned}
$$

(b) The 0-1 ILP Formulation Example

Fig. 5. The ILP formulation example.

formulation $F^{i,k+1}$ is formed. Then, this new formulation will go back to the LP Relaxation step for optimization. For efficiency, at most three cuts are added at each iteration according to an empirical study in [31]. It repeats until no more cutting plane is found. The lifted cover cutting will be discussed later in Section 4.3.

4.2.6 Pruning and Bounding

After processing a node N^i, it will consider whether this node should be pruned. To determine this, it checks the obtained local upper bound of the LP Relaxation at node N^i (after the kth iteration) and the global lower bound of the 0-1 ILP solution of the original 0-1 ILP:

- If $z_{ub}^{i,k} \leq z_{lb}$, it prunes the node N^i.
- If $z_{ub}^{i,k} \geq z_{lb}$ and the current fractional optimal solution $x^{i,k}$ is a binary solution, it updates the new global lower bound $z_{lb} = z_{ub}^{i,k}$ and the current best 0-1 ILP solution $x_{best} = x^{i,k}$. Then, it prunes this node and all unevaluated nodes in the list $\mathcal{L}$ whose upper bound z_{ub}^{i} is less than the new global lower bound z_{lb}, and processes another node in $\mathcal{L}$. If $z_{ub}^{i,k} \geq z_{lb}$ and $x^{i,k}$ is fractional, it goes to the branching step.

4.2.7 Branching

If the solution of the LP relaxation in node N^i is fractional, the branching step creates two child nodes of N^i. First, it chooses the fractional variable x_j^* in the current fractional optimal solution $x^{i,k}$ of the LP relaxation and performs the branching. We use a common choice, namely *most fractional variable* [11], to select the variable x_j^*: $j = \text{argmax}_{j\in C} \min[f_j, 1-f_j]$, where $f_j = x_j^* - \lfloor x_j^* \rfloor$. Then, the two child nodes are placed into the list $\mathcal{L}$ for further processing:

- $N^{i+1} = \max\{cx : x \in F^{k+1} = \{F^k \bigcap (x_j = 0)\}\}$.
- $N^{i+2} = \max\{cx : x \in F^{k+2} = \{F^k \bigcap (x_j = 1)\}\}$.

The size of the search tree may grow exponentially if branching is not controlled properly. To effectively reduce the size of the tree, we use a heuristic based on the number of specified questions N in the generated test paper. Consider a path from the root to a given unevaluated node of the tree, if the number of branching variables x_j with value 1 along the path is larger than or equal to N, we stop branching at that node. The reason is that we only need N questions in the generated test paper.

4.2.8 Termination

It returns the current best solution x_{best} of the 0-1 ILP.

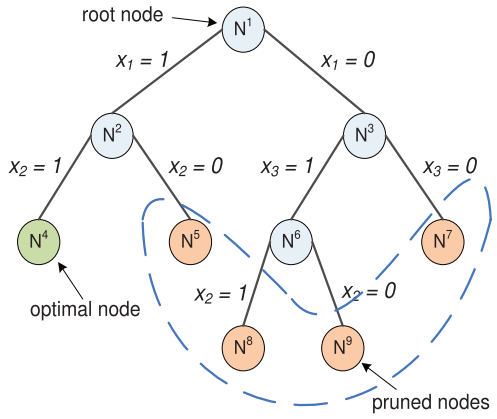
4.2.9 An Example

Suppose that we need to generate a test paper P from the Math data set given in Table 1 based on the specification $\mathcal{S} = \langle 2, 15, 6, \{(c_1, 0.5), (c_2, 0.5)\}, \{(y_1, 0.5), (y_2, 0.5)\}\rangle$. We associate each question with a binary variable x_i, $i = 1..8$. However, we eliminate inappropriate variables x_6, x_7, and x_8 because they cannot satisfy the specification $\mathcal{S}$. Here, we formulate the problem as a 0-1 fractional ILP with five binary variables, which is shown in Fig. 5a. The 0-1 fractional ILP problem is then transformed into a standard 0-1 ILP, which is shown in Fig. 5b.

Fig. 6 shows an example on the construction of the enumeration tree of subproblems during the BAC-STG process. Initially, the LP relaxation of this problem is solved by the PDIP algorithm to obtain the fractional optimal solution $x^* = (x_1, x_2, x_3, x_4, x_5) = (0.50, 0.70, 0.79, 0.00, 0.00)$. Next, the fractional optimal solution is updated when new lifted cover cutting planes are added to obtain the new fractional optimal solution x^*. Then, the local upper bound value at N^1 is set as $z_{ub}^1 = 12.23$; the global lower bound is set as $z_{lb} = -\infty$; and the current best 0-1 solution is set as $x_{best} = \emptyset$.

After that, the root node N^1 is branched on the variable x_1 to create two child nodes N^2 and N^3 during branching. Then, the unevaluated node N^2 is selected for processing, as it has the largest local upper bound. After processing and branching at N^2, two child nodes N^4 and N^5 are created in which $N^4 = (1.00, 1.00, 0.00, 0.00, 0.00)$ is a feasible binary solution with its objective value $z_{ub}^4 = 11$. Then, the global local bound and best solution are updated as $z_{lb} = 11$ and $x_{best} = (1.00, 1.00, 0.00, 0.00, 0.00)$, respectively. The node N^5 is then pruned because its local upper bound is less than the current global lower bound. Subsequently, branching at node N^3 will create N^6 and N^7. Similar to N^5, N^7 will then be pruned. Branching at node N^6 will create N^8 and N^9, which are then pruned similarly to N^5.

Finally, the best solution $x_{best} = (1.0, 1.0, 0.0, 0.0, 0.0)$ is obtained. It corresponds to the test paper $P = \{q_1, q_2\}$ for the specification $\mathcal{S}$. The generated test paper has the average discrimination degree $E = 5.5$ and specification $\mathcal{S}_P = \langle 2, 15, 6, \{(c_1, 0.5), (c_2, 0.5)\}, \{(y_1, 0.5), (y_2, 0.5)\}\rangle$.



(a) Enumeration Tree

	$x^* = (x_1, x_2, x_3, x_4, x_5)$	z^i_{ub}	z_{lb}
N^1	0.50, 0.70, 0.79, 0.00, 0.00	12.23	$-\infty$
N^2	1.00, 0.85, 0.00, 0.15, 0.00	11.55	$-\infty$
N^3	0.00, 0.90, 0.50, 0.60, 0.00	12.10	$-\infty$
N^4	1.00, 1.00, 0.00, 0.00, 0.00	11.00	11.00
N^5	1.00, 0.00, 0.00, 0.50, 0.00	8.50	*pruned*
N^6	0.00, 0.50, 1.00, 0.20, 0.00	11.40	11.00
N^7	*infeasible*	$-\infty$	*pruned*
N^8	*infeasible*	$-\infty$	*pruned*
N^9	0.00, 0.00, 1.00, 0.30, 0.00	9.1	*pruned*

(b) Node List $\mathcal{L}$

Fig. 6. The enumeration tree example.

4.3 Lifted Cover Cutting

This step aims to generate the lifted cover cutting planes from the fractional optimal solution x^* and simplex tableau $\mathcal{T}$ for the LP relaxation step. Before discussing this step in detail, we need to define some basic terminologies. Consider the set $X = \{x \in \{0,1\}^n : \sum_{j=1}^n a_j x_j \leq b\}$, which represents a row of the simplex tableau $\mathcal{T}$.

Definition 1 (Dominance). *If $a_1 x \leq b_1$ and $a_2 x \leq b_2$ are two valid inequalities for $X = \{x \in R^n_+ : Ax \leq b\}$, $a_1 x \leq b_1$* dominates *$a_2 x \leq b_2$ if $\{x \in R^n_+ : a_1 x \leq b_1\} \subseteq \{x \in R^n_+ : a_2 x \leq b_2\}$.*

If there exists any nonnegative coefficient a_j in X, the variable x_j can be replaced by its complementary variable $x'_j = 1 - x_j$, so that X contains only coefficients $a_j > 0$. As all coefficients in the LHS of X are now nonnegative, we may assume that the RHS $b > 0$. Let $N = \{1, 2 \ldots, n\}$ in the following definitions.

Definition 2 (Cover). *Let $C \subseteq N$ be a set such that $\sum_{j \in C} a_j > b$, then C is a* cover. *A* cover *is* minimal *if $C \setminus \{j\}$ is not a* cover *for any $j \in C$.*

The following two propositions are derived directly from the *cover* definition:

Proposition 1 (Cover inequality). *Let $C \subseteq N$ be a* cover *for X, the* cover inequality *$\sum_{j \in C} x_j \leq |C| - 1$ is valid for X, where $|C|$ is the cardinality of C.*

Proposition 2 (Extended cover inequality). *Let $C \subseteq N$ be a* cover *for X, the* extended cover inequality: *$\sum_{j \in \mathcal{E}(C)} x_j \leq |C| - 1$ is valid for X, where $\mathcal{E}(C) = C \cup \{j : a_j \geq a_i, \forall i \in C\}$.*

Definition 3 (Lifted cover inequality (LCI)). *LCI is an extended cover inequality which is not dominated by any other extended cover inequalities.*

In general, the problem of finding LCI is equivalent to the finding of the best possible values for α_j for $j \in N \setminus \{C\}$ such that the inequality:

$$\sum_{j \in N \setminus \{C\}} \alpha_j x_j + \sum_{j \in C} x_j \leq |C| - 1$$

is valid for X, where C is a *cover inequality*.

When the matrix A in the 0-1 ILP is sparse, the lifted cover cutting plane defined by LCI is an effective cutting plane for the pruning step in the BAC method. However, the problem on finding LCI has been shown to be NP-hard [32]. In this research, we propose to generate LCIs efficiently as follows:

First, we find a *minimal cover inequality* based on the N most significant basic variables in the fractional optimal solution x^* of the LP relaxation, where N is the number of questions given in the test paper specification. Specifically, consider a row of the form $\sum a_{ij} x_j = b_i$ in the tableau, the basic variables $x^*_j > 0$ of the fractional optimal solution are then sorted in nonincreasing order. Let U be a list of the first N largest coefficients, and V be the list of the remains of the sorted list. If k is the minimal number such that the sum of the coefficients a_{ij} w.r.t. the first basic variables of the list V exceeds b_i, i.e., $\sum_{j=1}^k a_{ij} > b_i$, then the set $C = \{j_1, \ldots, j_k\}$ is a *minimal cover*.

Next, we generate *extended cover inequalities* from the *minimal cover* C as follows:

$$\sum_{t=1}^{N} \alpha_{j_t} x_{j_t} + \sum_{j \in C} x_j \leq |C| - 1.$$

To generate LCIs from the *extended cover inequalities*, we need to calculate the largest lifting value $\alpha_{j_t} \in U, t = 1 \ldots N$, for each variable x_{j_t}. This can be done by using an incremental algorithm to calculate α_{j_1}, then α_{j_2} until α_{j_N} in step by step manner. Specifically, the algorithm starts from calculating α_{j_1}, the result of α_{j_1} will then be used to calculate α_{j_2} and so on. To obtain α_{j_t}, $t = 1..N$, we need to solve the following 0-1 KP (0-1 KP):

$$\text{maximize} \sum_{m=1}^{t-1} \alpha_{j_m} x_{j_m} + \sum_{j \in C} x_j$$

$$\text{subject to} \sum_{m=1}^{t-1} a_{j_m} x_{j_m} + \sum_{j \in C} a_j x_j \leq b_i - a_{j_t}$$

$$x \in \{0,1\}^{|C|+t-1}.$$

The largest lifting value α_{j_t} is then calculated as $\alpha_{j_t} = |C| - 1 - \gamma_t$, where γ_t, $t = 1..N$, is the objective function value according to the optimal solution obtained from the 0-1 KP.

It can be seen that γ_t computes the maximum weight corresponding to the set $\{j_1, j_2, \ldots, j_{t-1}\} \bigcup C$ in the *LCI* when $x_{j_t} = 1$.

Gu et al. [31] solved the 0-1 KP by using a dynamic algorithm that requires high computational complexity of $O(n^4)$. The experimental results have shown that the runtime performance is poor when handling test cases with a few thousand variables. This is not acceptable in STG in which the 0-1 ILP may have tens of thousands of variables that need to be solved efficiently. In this research, we apply an approximation algorithm from Martello and Toth [38] to efficiently solve the 0-1 KP that only requires $O(n \log n)$.

5 Performance Evaluation

In this section, we evaluate the performance of the proposed BAC-STG approach for STG. The experiments are conducted on a Windows XP environment, using an Intel Core 2 Quad 2.66 GHz CPU with 3.37 GB of memory. The BAC-STG approach is implemented in Java with the CPLEX API package, version 11.0 [17]. From the CPLEX package, we use the PDIP and simplex methods. The performance of BAC-STG is measured and compared with other techniques including GA [39], PSO [21], DE [14], ACO [12], TS [9], DAC [23], and the conventional BAC method [31]. These techniques are reimplemented based on the published articles. We compare the BAC-STG approach with the conventional BAC technique, which has been shown to be more effective for large-scale 0-1 ILP with the sparse matrix property than the commercial software [31].

5.1 Experiments

We have conducted two sets of experiments. In the first set of experiments, we analyze the quality and runtime efficiency of our BAC-STG approach based on four large-scale data sets by using different specifications. In the second set of experiments, we evaluate the effectiveness of our proposed approach by conducting a user evaluation for the quality of test papers generated from different specifications based on the G.C.E A-Level math and the undergraduate engineering math data sets.

In the experiments, the test paper generation process is repeated until one of the following two termination conditions is reached:

- *Quality satisfaction*. The algorithm will terminate if a high-quality test paper is generated.
- *Maximum number of evaluated nodes in which no better solution is found*. This parameter is experimentally set to 300 nodes for BAC and BAC-STG. Similarly, for other heuristic techniques, this parameter is set to the maximum number of iterations in which no better solution is found. It is generally set to 200 iterations.

5.2 Quality Measures

The performance of the proposed BAC-STG approach is evaluated based on paper quality and runtime. To evaluate the quality, we define mean discrimination degree and mean constraint violation (CV).

Definition 4 (Mean discrimination degree). *Let $P_1, P_2, \ldots, P_k$ be the generated test papers on a question data set $\mathcal{D}$ w.r.t. different test paper specifications $\mathcal{S}_i$, $i = 1..k$. The mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ is defined as*

$$\mathcal{M}_d^{\mathcal{D}} = \frac{\sum_{i=1}^{k} E_{P_i}}{k},$$

where E_{P_i} is the average discrimination degree of P_i.

CV indicates the differences between the test paper specification and generated test paper. Let $\mathcal{S} = \langle N, T, D, C, Y \rangle$ be a test paper specification and $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P \rangle$ be a generated test paper specification. CVs can be measured according to total time, average difficulty degree, topic distribution and question type distribution between the test paper specification ($\mathcal{S}$) and the generated test paper specification ($\mathcal{S}_P$) as follows:

- *Total time CV:* $\triangle T(\mathcal{S}_P, \mathcal{S}) = \frac{|T_P - T|}{T}$.
- *Average difficulty degree CV:* $\triangle D(\mathcal{S}_P, \mathcal{S}) = \frac{|D_P - D|}{D}$.
- *Topic distribution CV:*

$$\triangle C(\mathcal{S}_P, \mathcal{S}) = D_{KL}(pc_p \| pc) = \sum_{i=1}^{M} pc_p(i) \log \frac{pc_p(i)}{pc(i)}.$$

- *Question type distribution CV:*

$$\triangle Y(\mathcal{S}_P, \mathcal{S}) = D_{KL}(py_p \| py) = \sum_{j=1}^{K} py_p(j) \log \frac{py_p(j)}{py(j)},$$

where D_{KL} is the Kullback-Leibler divergence [40], which is used to measure the statistical differences of the topic and question type distributions between $\mathcal{S}_P$ and $\mathcal{S}$; and $\lambda = 100$ is a constant used to scale the value between 0 and 100.

The CV of a generated test paper P w.r.t. the test paper specification $\mathcal{S}$ can then be calculated as the average of the four violations:

$$CV(P, \mathcal{S}) = \frac{\lambda * \triangle T + \lambda * \triangle D + \log \triangle C + \log \triangle Y}{4}.$$

Definition 5 (Mean CV). *The mean CV $\mathcal{M}_c^{\mathcal{D}}$ of k generated test papers $P_1, P_2, \ldots, P_k$ on a question data set $\mathcal{D}$ w.r.t. k test paper specifications $\mathcal{S}_i$, $i = 1..k$, is defined as*

$$\mathcal{M}_c^{\mathcal{D}} = \frac{\sum_{i=1}^{k} CV(P_i, \mathcal{S}_i)}{k},$$

where $CV(P_i, \mathcal{S}_i)$ is the CV of P_i w.r.t. S_i.

A high-quality test paper P should maximize the average discrimination degree and minimize CVs. In other words, it should have a high value on E_P and a low value on the constraint violation $CV(P, \mathcal{S})$. Hence, the overall quality of a generated test paper depends on user's preference between these two aspects. According to a pedagogical perspective, users often pay more attention to constraint satisfaction of test papers. To determine the quality of a generated test paper, CVs could be defined in a certain range. Here, we set the following four thresholds for a high-quality test paper: $\triangle T(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\triangle D(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\log \triangle C(\mathcal{S}_P, \mathcal{S}) \leq 5$ and

TABLE 2
Test Data Sets

	#Questions	#Topics	#Question Types
D_1	20000	40	3
D_2	30000	50	3
D_3	40000	55	3
D_4	50000	60	3

$\log \triangle Y(\mathcal{S}_P, \mathcal{S}) \leq 5$. The threshold values are obtained experimentally (as shown in Section 6.4). Based on these thresholds, we set $\mathcal{M}_c^{\mathcal{D}} \leq 10$ for high-quality test papers, $10 < \mathcal{M}_c^{\mathcal{D}} \leq 30$ for medium-quality test papers and $\mathcal{M}_c^{\mathcal{D}} > 30$ for low-quality test papers.

5.3 Performance on Quality and Runtime

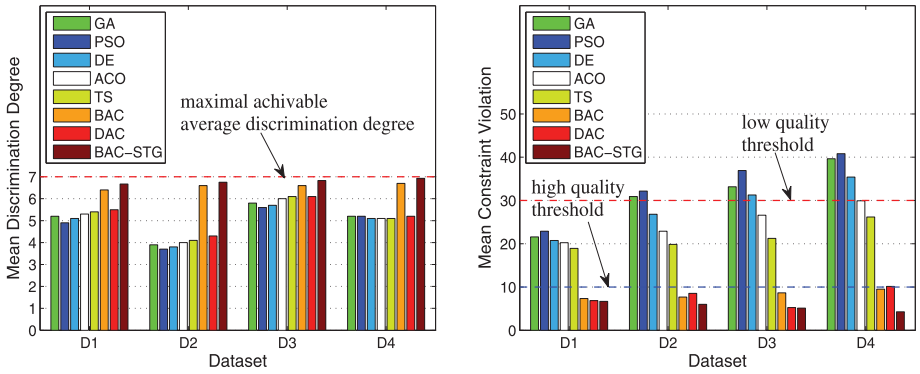
Data sets. As there is no benchmark data set available, we generate four large-sized synthetic data sets, namely D_1, D_2, D_3, and D_4, for performance evaluation. Specifically, these four data sets D_1, D_2, D_3, and D_4 have number of questions of 20,000, 30,000, 40,000, and 50,000, respectively. There are mainly three question types in each data set, namely fill-in-the-blank, multiple choice, and long question. In the two data sets, D_1 and D_3, the value of each attribute is generated according to a uniform distribution. However, in the other two data sets, D_2 and D_4, the value of each attribute is generated according to a normal distribution. Our purpose is to measure the effectiveness and efficacy of the test paper generation process of each algorithm for both balanced data sets D_1 and D_3, and imbalanced data sets D_2 and D_4. Intuitively, it is more difficult to generate good quality test papers for the data sets D_2 and D_4 than the data sets D_1 and D_3. Table 2 summaries the four data sets.

Experimental procedures. To evaluate the performance of the BAC-STG approach, we have designed 12 test specifications in the experiments. We vary the parameters in order to have different test criteria in the test specifications. The number of topics is specified between 2 and 40. The total time is set between 20 and 240 minutes, and it is set proportional to the number of selected topics for each specification. The average difficulty degree is specified randomly between 3 and 9. We perform the experiments according to the 12 test specifications for each of the following eight algorithms: GA, PSO, DE, ACO, TS, BAC, DAC, and BAC-STG. We measure the runtime and quality of the generated test papers for each experiment.

Performance on quality. Fig. 7 shows the quality performance results of the eight techniques based on the mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ and mean CV $\mathcal{M}_c^{\mathcal{D}}$. As can be seen from Fig. 7a, BAC-STG has consistently achieved higher mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ than the conventional BAC and other heuristic techniques for the generated test papers. Particularly, BAC-STG can generate test papers with quality close to the maximal achievable value of $\mathcal{M}_d^{\mathcal{D}} \approx 7$. In addition, we also observe that BAC-STG has consistently outperformed the other techniques on mean CV $\mathcal{M}_c^{\mathcal{D}}$ based on the four data sets. The average CVs of BAC-STG tend to decrease whereas the average CVs of the other techniques increase quite fast when the data set size or the number of specified constraints gets larger. In particular, BAC-STG can generate high-quality test papers with $\mathcal{M}_c^{\mathcal{D}} \leq 6$ for all data sets. Also, BAC-STG is able to generate higher quality test papers on larger data sets while the other techniques generally degrade the quality of the generated test papers when the data set size gets larger.

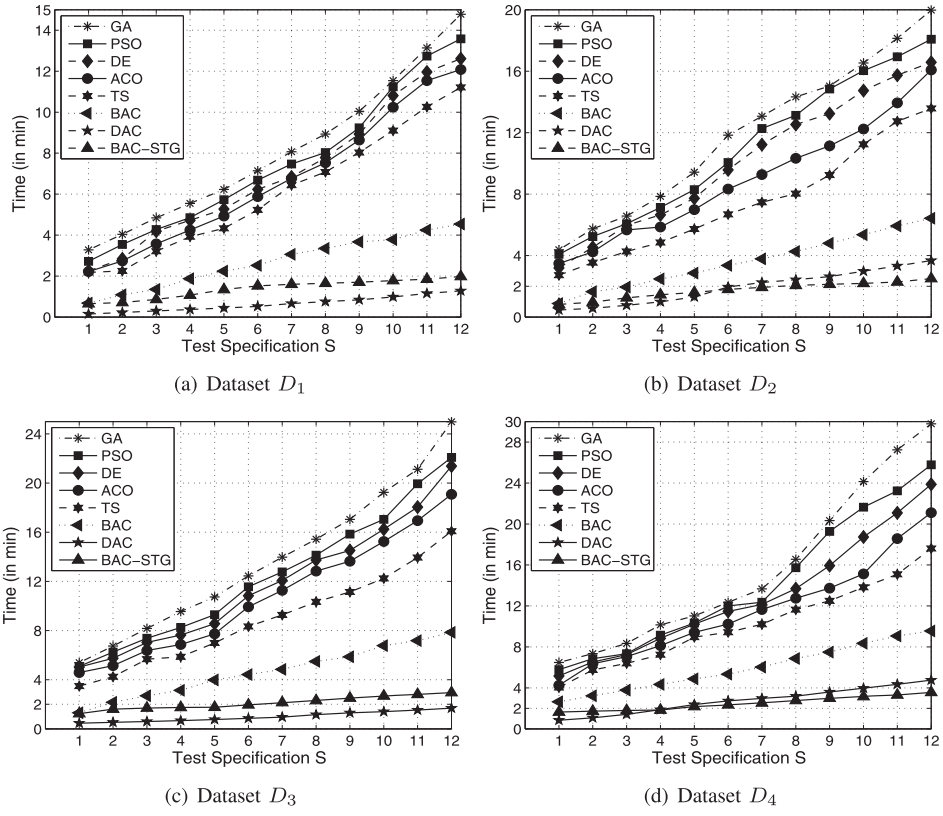
Performance on runtime. Fig. 8 compares the runtime performance of the eight techniques based on the four data sets. Here, the 12 specifications are sorted increasingly according to the number of topics in the constraints. The results have clearly shown that the proposed BAC-STG approach outperforms the conventional BAC and other heuristic techniques, except DAC, in runtime for the different data sets. BAC-STG generally requires less than 2 minutes to complete the paper generation process. Moreover, the proposed BAC-STG approach is quite scalable in runtime on different data set sizes and distributions. In contrast, the other techniques (except DAC) are not efficient to generate high-quality test papers. Particularly, the runtime performance of these techniques degrades quite badly as the data set size or the number of specified constraints gets larger, especially for imbalanced data sets D_2 and D_4.

Discussion. The good performance of BAC-STG is due to three main reasons. First, as BAC-STG is based on LP relaxation, it can maximize the average discrimination degree effectively and efficiently while satisfying the multiple constraints without using weighting parameters. As such, BAC-STG can achieve better paper quality and runtime efficiency as compared with other heuristic-based



(a) Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$ (b) Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$

Fig. 7. Performance results based on average quality of the 12 specifications $S_1, S_2, \ldots, S_{12}$.



(a) Dataset D_1 (b) Dataset D_2

(c) Dataset D_3 (d) Dataset D_4

Fig. 8. Performance results based on runtime of the 12 specifications $S_1, S_2, \ldots, S_{12}$.

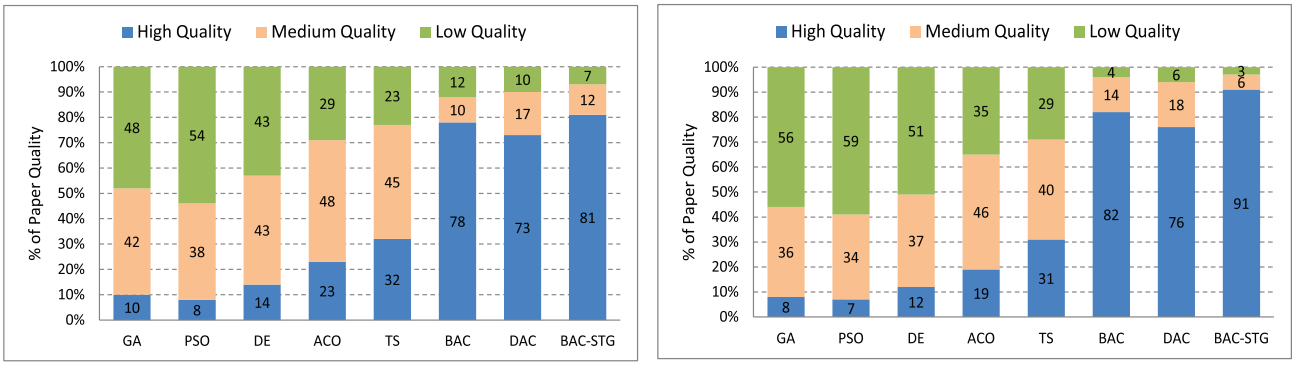
techniques. Second, BAC-STG has a more effective branching strategy than that of conventional BAC, thereby pruning the search space more effectively. This helps BAC-STG improve runtime and search on promising unvisited subproblem nodes of the search tree for paper quality enhancement. Third, BAC-STG uses an efficient algorithm to generate the *lifted cover inequalities*. Thus, BAC-STG can also improve its computational efficiency on large-scale data sets as compared with the conventional BAC technique. Moreover, as there are more questions with different attribute values on larger data sets and LP relaxation is effective for global optimization, BAC-STG is able to generate higher-quality test papers. Therefore, the BAC-STG approach is effective for STG in terms of paper quality and runtime efficiency.

Comparison between BAC-STG and DAC. DAC [23] is an efficient STG approach for online test paper generation. Table 3 gives the performance comparison between BAC-STG and DAC. The performance results are obtained based on the average results of the 12 test specifications for each data set. In Fig. 7, the quality performance of BAC-STG is consistently better than DAC for the four data sets. However, the runtime performance of BAC-STG and DAC depends on the user specifications and data set distributions, as shown in Table 3 and Fig. 8.

For the balanced uniform distribution data sets D_1 and D_3, DAC outperforms BAC-STG in runtime. DAC is in fact very fast, as it is designed for achieving online runtime requirement. As there may have enough relevant questions on D_1 and D_3 for DAC to optimize its solution without getting stuck on local optimal, the runtime performance of DAC outperforms BAC-STG in this situation because it is a heuristic-based technique, whereas BAC-STG is a global optimization method. For imbalanced normal distribution data sets D_2 and D_4, BAC-STG achieves better runtime performance if the specified test papers contain many topics or have high total time. Otherwise, DAC achieves better runtime performance. The main reason is that the sparse matrix property of 0-1 ILP formulation of BAC-STG is satisfied in this situation, which makes lifted cover cuttings

TABLE 3
Performance Comparison between BAC-STG and DAC

	Algorithm	D_1	D_2	D_3	D_4
Average Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$	DAC	5.50	4.30	6.25	5.20
	BAC-STG	6.67	6.75	6.83	6.92
Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$	DAC	6.85	8.53	5.25	10.15
	BAC-STG	5.35	4.68	4.24	3.05
Average Runtime (seconds)	DAC	38.4	116.8	59.4	165.8
	BAC-STG	83.6	104.9	126.7	149.2



(a) A-$Math$ (b) U-$Math$

Fig. 9. Expert calibration on generated test quality measures.

become more effective for 0-1 ILP optimization. Furthermore, as DAC focuses only on optimizing the constraint satisfaction of a unique initial solution, and it could easily lead to a local optimal. It is especially the case for imbalanced data sets, where there may not have enough relevant questions for DAC to optimize the unique initial solution. As shown in Fig. 7, the quality performance of DAC degrades quite badly on both D_2 and D_4.

In short, the quality performance of BAC-STG consistently outperforms DAC, while its runtime performance is comparable to that of DAC.

5.4 Expert Calibration of Test Quality Measures

Data sets. To gain further insight into the paper quality generated by the proposed BAC-STG approach, we have conducted a user evaluation. Here, we use two math data sets, namely A-$Math$ and U-$Math$, which are constructed from G.C.E. A-Level Math and Undergraduate Math, respectively. For experimental purposes, the question type and topic attributes of question are calibrated automatically, whereas the difficulty degree attribute is calibrated by 10 tutors who are tutors of the first year undergraduate mathematics subject CZ1800 Engineering Mathematics in the School of Computer Engineering, Nanyang Technological University. These tutors have good knowledge of the math contents contained in the two math data sets. In the tutor calibration, we adopt a rating method [41] for the tutors to rate the difficulty degree of each question from 1 to 7 corresponding to the following seven discretized difficulty levels of IRT: extremely easy, very easy, easy, medium, hard, very hard, and extremely hard. In addition, according to the IRT Normal Ogive model [24], an initial value for the discrimination degree of a question can be computed based on its relation formula with a fixed user's proficiency value and the difficulty degree. Table 5 summarizes the two data sets.

Experimental procedures. In the experiments, we have designed 12 new specifications for the A-Math data set and 12 new specifications for the U-Math data set with different parameter values. Then, we generate the test papers based on the test specifications using the eight techniques. As a result, a total of 96 papers are generated for each data set. To conduct the user evaluation, the same 10 tutors are participated. In the experiments, the tutors are asked to evaluate the test paper quality by comparing its attributes with its original specification. The tutors compare each corresponding attribute of the generated test with its specification by their own intuition without knowing our defined measures. Based on the similarities, they are asked to evaluate the overall quality of the generated test and classify it into one of the following three categories: high, medium, and low. As a result, a total of 960 test paper evaluations are conducted for each data set.

Expert calibration on quality results. Fig. 9 shows the user evaluation results on the generated test paper qualities from the eight techniques. As can be seen, BAC-STG has achieved better performance on quality than the other techniques. For the A-Math data set, BAC-STG has achieved promising results with 81 percent (97 papers), 12 percent (14 papers), and 7 percent (9 papers) for high-, medium-, and low-quality generated papers, respectively, from a total of 120 papers evaluated. Similarly, for the U-Math data set, BAC-STG has achieved promising results with 91 percent (109 papers), 6 percent (7 papers) and 3 percent (4 papers) for high-, medium-, and low-quality generated papers, respectively. On average, BAC-STG has achieved 86, 9, and 5 percent for high-, medium-, and low-quality generated papers, respectively. In addition, it can also be observed that the proposed BAC-STG approach is able to improve the quality of the generated papers with a larger data set. It has performed better with the U-Math data set than the A-Math data set.

Moreover, we have further analyzed the quality of the generated papers based on the user evaluation results. After all the generated papers are classified into high, medium, and low quality, we have analyzed the CVs on topic, question type, difficulty degree, and total time based on each of the two data sets. Table 4 gives the analysis results in terms of mean and standard deviation of the corresponding CVs. For the A-Math data set, we have obtained the mean CVs of 10 ± 2.2, 21 ± 2.8, and 30 ± 1.9 for high-, medium-, and low-quality generated papers, respectively. Similarly, for the U-Math data set, we have obtained the mean CVs of 9 ± 1.4, 19 ± 1.4, and 29 ± 1.2 for high-, medium-, and low-quality generated papers respectively. On average, we have obtained 9.5 ± 1.3, 20 ± 1.6, and 29.5 ± 1.1 CVs for high-, medium-, and low-quality generated papers. In fact, we have used these values as the thresholds when we determine high, medium, or low quality for the generated papers during performance evaluation on test paper quality.

TABLE 4
Quality Analysis of the Generated Test Papers According to the Expert Calibration Results

User Assessment Criteria	A-$Math$			U-$Math$		
	High	Medium	Low	High	Medium	Low
Topic Violation (log $\triangle C$)	5 ± 3	14 ± 4	14 ± 4	4 ± 2	13 ± 2	13 ± 2
Question Type Violation (log $\triangle Y$)	6 ± 3	15 ± 5	17 ± 3	10 ± 3	13 ± 4	15 ± 2
Difficulty Degree Violation ($\triangle D$)	13 ± 5	25 ± 6	43 ± 4	11 ± 3	23 ± 2	42 ± 3
Time Violation ($\triangle T$)	16 ± 6	29 ± 7	48 ± 4	12 ± 3	26 ± 3	46 ± 2
Constraint Violation CV	10 ± 2.2	21 ± 2.8	30 ± 1.9	9 ± 1.4	19 ± 1.4	29 ± 1.2

6 Web-Based Testing Framework

In this research, we have investigated a web-based testing system for e-learning in mathematics. Fig. 10 shows the proposed system framework, which consists of the following components: webserver, math question database server using MySQL, STG, automatic solution checking, and automatic question calibration. The STG component is implemented based on the proposed BAC-STG approach. The automatic solution checking component is implemented based on the mathematical equivalence checking algorithm [42]. The automatic question calibration component is currently under development based on the educational data mining techniques mentioned earlier in Section 3.1. These components are implemented in Java. The system can be accessed through a web browser.

As equivalent mathematical expressions can be expressed in different forms (e.g., $x + \frac{1}{x}$ and $\frac{x^2+1}{x}$), the automatic solution checking component automatically checks the equivalence of the students' answers with the standard solutions to evaluate the correctness of the students' answers. Currently, it focuses on automatic answer checking for mathematical expressions that are the most common form of required answers in most math questions. To check mathematical answers, a randomized algorithm is proposed based on the probabilistic numerical testing method [42]. The proposed algorithm has shown promising performance on different types of mathematical expressions such as multivariate polynomials, trigonometric functions, and so on. As compared with other web-based testing systems, which are only able to support multiple choice answer checking, our proposed system has an additional advantage on supporting advanced mathematical expression answer checking.

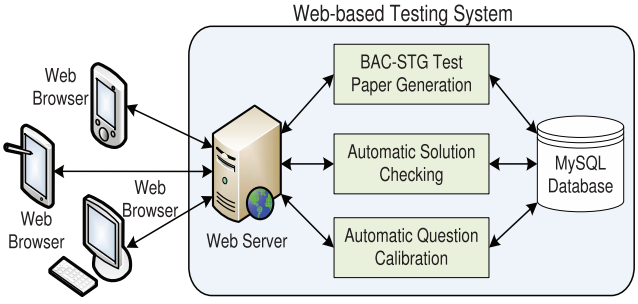


Fig. 10. Web-based testing framework.

TABLE 5
Math Data Sets

	#Questions	#Topics	#Question Types
$A - Math$	1500	12	3
$U - Math$	3000	36	3

7 Conclusion

In this paper, we have proposed an efficient ILP approach called BAC-STG for high-quality STG from large-scale question data sets. The proposed BAC-STG approach is based on the branch-and-bound and lifted cover cutting methods to find the near-optimal solution by exploiting the sparse matrix property of 0-1 ILP. The performance results on various data sets and a user evaluation on generated test paper quality have shown that the BAC-STG approach has achieved test paper generation with not only high quality, but also runtime efficiency when compared with other STG techniques. As such, the proposed BAC-STG approach is particularly useful for web-based testing and assessment for online learning environment.

From a pedagogical perspective on generating high-quality test papers, the current web-based testing approach and system can be extended in the following four ways. First, automatic calibration of question attributes should be supported to generate large-scale data sets. Currently, we are investigating different data mining techniques to achieve this purpose. Second, the current approach could be extended to support the generation of multiple test papers simultaneously based on the same test paper specification. The quality of these generated test papers should be comparable. Therefore, to evaluate students in an online class, different test papers of equivalent or similar properties and quality can be used. Third, in the expert calibration, it would also be very useful to consider a more general quality definition based on the capability of a test to make a valid cognitive assessment of a user over set of skills. Finally, the current system does not allow further changes on individual questions of a generated test paper after the test paper generation process. This could be enhanced by allowing users to edit, modify, and update individual questions of a generated test paper. As such, the web-based testing system will be more flexible in creating high-quality test papers.

References

[1] C. Ramos, C. Frasson, and S. Ramachandran, "Introduction to the Special Issue on Real World Applications of Intelligent Tutoring Systems," *IEEE Trans. Learning Technologies*, vol. 2, no. 2, pp. 62-63, Apr. 2009.

[2] P. Brusilovsky and P. Miller, "Web-Based Testing for Distance Education," *Proc. World Conf. WWW and Internet (WebNet)*, vol. 99, pp. 24-30, 1999.

[3] R. Conejo, E. Guzmán, E. Millán, M. Trella, J.L. Pérez-De-La-Cruz, and A. Ríos, "SIETTE: A Web-Based Tool for Adaptive Testing," *Int'l J. Artificial Intelligence in Education,* vol. 14, no. 1, pp. 29-61, 2004.

[4] F.G. Martin, "Will Massive Open Online Courses Change How We Teach?" *Comm. ACM,* vol. 55, no. 8, pp. 26-28, http://doi.acm.org/10.1145/2240236.2240246, Aug. 2012.

[5] K. Hopkins, *Educational and Psychological Measurement and Evaluation.* ERIC, 1998.

[6] D. Thissen, B. Reeve, J. Bjorner, and C. Chang, "Methodological Issues for Building Item Banks and Computerized Adaptive Scales," *Quality of Life Research,* vol. 16, pp. 109-119, 2007.

[7] T. Theunissen, "Binary Programming and Test Design," *Psychometrika,* vol. 50, no. 4, pp. 411-420, 1985.

[8] G. Hwang, B.M. Lin, and T. Lin, "An Effective Approach for Test-Sheet Composition with Large-Scale Item Banks," *Computers & Education,* vol. 46, no. 2, pp. 122-139, 2006.

[9] G.J. Hwang, P.Y. Yin, and S.H. Yeh, "A Tabu Search Approach to Generating Test Sheets for Multiple Assessment Criteria," *IEEE Trans. Education,* vol. 49, no. 1, pp. 88-97, Sept. 2006.

[10] A. Schrijver, *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

[11] D. Bertsimas and R. Weismantel, *Optimization over Integers.* Dynamic Ideas, 2005.

[12] X.M. Hu, J. Zhang, H.S.H. Chung, O. Liu, and J. Xiao, "An Intelligent Testing System Embedded with an Ant-Colony-Optimization-Based Test Composition Method," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 39, no. 6, pp. 659-669, Nov. 2009.

[13] G.J. Hwang, B. Lin, H.H. Tseng, and T.L. Lin, "On the Development of a Computer-Assisted Testing System with Genetic Test Sheet-Generating Approach," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 35, no. 4, pp. 590-594, Nov. 2005.

[14] W.F. Rui, W.W. Hong, P.Q. Ke, Z.F. Chao, and J.J. Liang, "A Novel Online Test-Sheet Composition Approach for Web-Based Testing," *Proc. Symp IT in Medicine Education,* pp. 700-705, 2009.

[15] J. Adema and W. Vander Linden, "Algorithms for Computerized Test Construction Using Classical Item Parameters," *J. Educational and Behavioral Statistics,* vol. 14, no. 3, pp. 279-290, 1989.

[16] J. Adema, E. Boekkooi-Timminga, and W. van der Linden, "Achievement Test Construction Using 0-1 Linear Programming," *European J. Operational Research,* vol. 55, no. 1, pp. 103-111, 1991.

[17] IBM, "CPLEX Optimizer (11.0)," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2012.

[18] GUROBI, "Gurobi Optimizer (Version 5.0)," http://www.gurobi.com/, 2011.

[19] J.E. Mitchell, "Integer Programming: Branch-and-Cut Algorithms," *Encyclopedia of Optimization,* vol. 2, pp. 519-525, Kluwer Press, 2001.

[20] C.L. Lee, C.H. Huang, and C.J. Li, "Test-Sheet Composition Using Immune Algorithm for E-Learning Application," *New Trends in Applied Artificial Intelligence,* vol. 4570, pp. 823-833, 2007.

[21] T.F. Ho, P.Y. Yin, G.J. Hwang, S.J. Shyu, and Y.N. Yean, "Multi-Objective Parallel Test-Sheet Composition Using Enhanced Particle Swarm Optimization," *J. Educational Technology Soc.,* vol. 12, no. 4, pp. 193-206, 2008.

[22] K.H. Tsai, T.I. Wang, T.C. Hsieh, T.K. Chiu, and M.C. Lee, "Dynamic Computerized Testlet-Based Test Generation System by Discrete PSO with Partial Course Ontology," *Expert Systems with Applications,* vol. 37, no. 1, pp. 774-786, 2009.

[23] M.L. Nguyen, S.C. Hui, and A.C.M. Fong, "An Efficient Multi-Objective Optimization Approach for Online Test Paper Generation," *Proc. IEEE Symp. Computational Intelligence in Multicriteria Decision-Making (MDCM),* pp. 182-189, 2011.

[24] F.B. Baker and S.H. Kim, *Item Response Theory.* Marcel Dekker, 1992.

[25] P. Songmuang and M. Ueno, "Bees Algorithm for Construction of Multiple Test Forms in E-Testing," *IEEE Trans. Learning Technologies,* vol. 4, no. 3, pp. 209-221, July-Sept. 2011.

[26] E. Boekkoi-Timminga, *Simultaneous Test Construction by Zero-One Programming.* Dept. of Education, Univ. of Twente, 1986.

[27] E. Boekkooi-Timminga, "The Construction of Parallel Tests from IRT-Based Item Banks," *J. Educational and Behavioral Statistics,* vol. 15, no. 2, pp. 129-145, 1990.

[28] R. Fletcher, "A Review of Linear Programming and Its Application to the Assessment Tools for Teaching and Learning (as TTle) Projects," technical report, Massey Univ., 2000.

[29] T. Theunissen, "Some Applications of Optimization Algorithms in Test Design and Adaptive Testing," *Applied Psychological Measurement,* vol. 10, no. 4, pp. 381-389, 1986.

[30] R. Luecht, "Computer-Assisted Test Assembly Using Optimization Heuristics," *Applied Psychological Measurement,* vol. 22, no. 3, pp. 224-236, 1998.

[31] G.J. Hwang, "A Test-Sheet-Generating Algorithm for Multiple Assessment Requirements," *IEEE Trans. Education,* vol. 46, no. 3, pp. 329-337, 2003.

[32] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh, "Lifted Cover Inequalities for 0-1 Integer Programs: Complexity," *INFORMS J. Computing,* vol. 11, pp. 117-123, 1999.

[33] E. Johnson, G. Nemhauser, and M. Savelsbergh, "Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition," *INFORMS J. Computing,* vol. 12, no. 1, pp. 2-23, 2000.

[34] R. Baker and K. Yacef, "The State of Educational Data Mining in 2009: A Review and Future Visions," *J. Educational Data Mining,* vol. 1, no. 1, pp. 3-17, 2009.

[35] S. Cetintas, L. Si, Y. Xin, D. Zhang, and J. Park, "Automatic Text Categorization of Mathematical Word Problems," *Proc. 22nd Int'l FLAIRS Conf.,* pp. 27-32, 2009.

[36] K. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper, *A Data Repository for the EDM Community: The PSLC DataShop.* CRC Press, 2010.

[37] M.L. Nguyen, S.C. Hui, and A.C.M. Fong, "Content-Based Collaborative Filtering for Question Difficulty Calibration," *Proc. Pacific Rim Int'l Conf. Trends in Artificial Intelligence,* pp. 359-371, 2012.

[38] M. Kojima, N. Megiddo, and S. Mizuno, "A Primal-Dual Infeasible-Interior-Point Algorithm for Linear Programming," *Math. Programming,* vol. 61, no. 1, pp. 263-280, 1993.

[39] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, 1990.

[40] S. Kullback, *Information Theory and Statistics.* Dover, 1997.

[41] K. Wauters, P. Desmet, and W. van den Noortgate, "Acquiring Item Difficulty Estimates: A Collaborative Effort of Data and Judgment," *Proc. Int'l Conf. Education Data Mining,* 2011.

[42] M.L. Nguyen, S.C. Hui, and A.C.M. Fong, "Web-Based Mathematics Testing with Automatic Assessment," *Proc. Pacific Rim Int'l Conf. Trends in Artificial Intelligence,* pp. 347-358, 2012.

Minh Luan Nguyen is currently working toward the PhD degree at the School of Computer Engineering, Nanyang Technological University, Singapore. He is a member of the IEEE.

Siu Cheung Hui received the BSc degree in mathematics in 1983 and the DPhil degree in computer science in 1987 from the University of Sussex, United Kingdom. He is an associate professor at the School of Computer Engineering, Nanyang Technological University, Singapore. He was with IBM China/Hong Kong Corporation as a system engineer from 1987 to 1990. His current research interests include data mining, web mining, Semantic Web, intelligent systems, information retrieval, intelligent tutoring systems, timetabling, and scheduling.

Alvis C.M. Fong is a professor at the Auckland University of Technology, New Zealand. Previously, he was an associate professor with Nanyang Technological University, Singapore. His research interests include information processing and management, multimedia, and communications. He is a senior member of the IEEE.