Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Education 46 (2006) 122–139
COMPUTERS & EDUCATION
www.elsevier.com/locate/compedu

An effective approach for test-sheet composition with large-scale item banks

Gwo-Jen Hwang [a], Bertrand M.T. Lin [b,*], Tsung-Liang Lin [a]

[a] *Department of Information Management, National Chi Nan University, Pu-Li, Nan-Tou County, Taiwan 545, ROC*
[b] *Department of Information and Finance Management, National Chiao Tung University, Hsinchu, Taiwan 300, ROC*

Received 11 October 2003; accepted 29 November 2003

Abstract

A well-constructed test sheet not only helps the instructor evaluate the learning status of the students, but also facilitates the diagnosis of the problems embedded in the students' learning process. This paper addresses the problem of selecting proper test items to compose a test sheet that conforms to such assessment requirements as average difficulty degree, average discrimination degree, length of test time, number of test items, and specified distribution of concept weights. A mixed integer programming model is proposed to formulate the problem of selecting a set of test items that best fit the multiple assessment requirements. As the problem is a generalization of the knapsack problem, which is known to be $\mathcal{NP}$-hard in the literature, computational challenge hinders the development of efficient solution methods. Seeking approximate solutions in an acceptable time is a viable alternative. In this paper, we propose two heuristic algorithms, based upon iterative adjustment, for finding quality approximate solutions. Extensive experiments are also conducted to assess the performances of different solution methods. Statistics from a series of computational experiments indicate that our proposed algorithms can produce near-optimum combinations of the test items subject to the specified requirements in a reasonable time.

Keywords: Computer-aided instruction; Test sheet; Mixed integer programming; Heuristic algorithm; Approximate solution

* Corresponding author. Tel.: +886 3 5131472; fax: +886 3 5729915.
E-mail address: bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

0360-1315/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2003.11.004

1. Introduction

As computer techniques progress rapidly in recent years, researchers have been trying to enable computers to perform intelligent behaviors and consequently foster the advances of artificial intelligence (AI) techniques. The call for the development of Intelligent Computer-Assisted Instruction (ICAI) systems has therefore attracted considerable attention from AI as well as CAI researchers.

In 1989, Johnson et al. presented MITT (Microcomputer Intelligence for Technical Training) Writer, an authoring environment for building intelligent tutoring systems for computer courses. It represented a practical application of artificial intelligence (AI) in technical training (Johnson, Neste, & Duncan, 1989). Later, Gonzalez and Ingraham (1994) designed an intelligent tutoring system, which is capable of automatically determining exercise progression and remediation during a training session according to past student performance. Vasandani and Govindaraj (1995) developed an intelligent tutoring system that helps organize system knowledge and operational information to enhance the operator's performance. Meanwhile, Harp, Samad, and Villano (1995) employed neural networks to model the behaviors of students in the context of intelligent tutoring systems, using self-organizing feature maps to capture the possible states of student knowledge from an already existing test database. Furthermore, Rowe and Galvin (1998) employed planning methods, consistency enforcement, objects and structured menu tools to construct intelligent simulation-based tutors for procedural skills. Antao, Brodersen, Bourne, and Cantwell (2000) presented their experiences of developing intelligent tutorial systems for teaching simulation in engineering education.

While computer intelligence has inspired the development of intelligent tutoring systems, the advent of network technologies in the mean time has helped realize distance learning systems. For example, Sun and Chou (1996) presented the CORAL (Cooperative Remotely Accessible Learning) system, which is aimed to institute a collaborative learning environment on computer networks. One of the branches of CORAL is the Intelligent Tutoring and Evaluation System (ITES) project, which focuses on applying the techniques of artificial intelligence to enhance the tutoring process (Hwang, 1998). ITES erhployed a fuzzy expert system to support the tutoring strategies of a distance learning environment; in addition, a set of parameters were defined to examine the behaviors of the students on networks through their on-line operations.

It is also widely recognized that the growing popularity of computer-assisted instruction has led computer-based testing to receiving increasing attention. Most of earlier studies focused on whether computer-based tests were equivalent to paper-and-pencil tests, assuming identical tests were administered in the two formats. Olsen, Maynes, Slawson, and Ho (1986) compared the relative performances of paper-administered, computer-administered, and computer-adaptive tests in testing the mathematical abilities of the third- and sixth-grade students. They found that paper-administered and computer-administered tests did not significantly differ from each other with regard to the testing quality. That study also identified the three types of tests to be equivalent in terms of score rank order, means, dispersions, and distribution shapes.

In New Zealand, three researchers propounded a ''Knowledge Based Computer Assisted Instruction System'', which can change the numeric part of items when the test is in progress so as to prevent students from memorizing the answers (Fan, Tina, & Shue, 1996). Another branch of relevant research is the Computerized Adaptive Testing, which applies some prediction methodologies to shorten the length of the tests without sacrificing the precision (Wainer, 1990).

Besides the traditional multiple-choice, fill-in-the-blank, and short essay type questions. Rasmussen, Northrup, and Lee (1997) suggested that Web-based instruction could also allow students' progress to be evaluated by participation in group discussions and portfolio development. Furthermore, Khan (1997) indicated the possibility that designers of Web-based instruction systems could create facilities to facilitate the students to submit their comments on courseware design and delivery. Feldman and Jones (1997) attempted to perform semiautomatic testing of student software under Unix systems. Chou (2000) endeavored to build the GATES system, a collective and collaborative project intended to integrate an interactive testing system with theoretical and practical research on complex technology-dependent learning environments. Hwang (2003a) employed the conceptual map approach for diagnosing students learning problems after an on-line test was performed. Since then, such systems that change the form of tests from paper-and-pencil to on-line have been proliferating rapidly.

In a testing system, the aggregate quality of the test items usually significantly affects the accuracy of the test. Therefore, several measures, such as degree of difficulty and degree of discrimination, have been proposed to represent the quality of each test item. The measures can be incrementally derived and adjusted in accordance with the statistics of each test. Especially in a network-based testing system, the testing results are recorded and analyzed for updating the degrees of difficulty and discrimination to improve the quality of the item bank (Lira, Bronfman, & Eyzaguirre, 1990). However, the quality of a test not only depends on the quality of the item bank, but also relates to the way the test sheet was constructed. That is, it is important to select proper test items to construct a test sheet that meets multiple assessment requirements, such as average difficulty degree, average discrimination degree, length of test time, number of test items, and specified distribution of concept weights.

This paper addresses two issues about the composition of test sheets. We propose a mixed integer programming model to formulate the problem of selecting a set of test items that best fit the multiple assessment requirements. As the problem is $\mathcal{NP}$-hard (see Section 3 for details), computational challenge hinders the development of efficient methods that can produce optimal solutions in a reasonable time. As an alternative, we propose two heuristic algorithms to find quality approximate solutions. Statistics from a series of computational experiments indicate that our approach is able to efficiently generate near-optimum combinations of test items that satisfies the specified requirements.

2. Background and motivations

Instructional applications of computers have increased rapidly in the most recent years, so have the computerized testing systems. Taking the GRE (Graduate Record Examinations) as an example, people have taken the test through computers since 1992. The GRE has been offering a computerized form since 1993 and has abandoned the paper-and-pencil form since 1999. The IBM Co. and Arthur Andersen Co. have also begun to work on the development of a computerized testing system. Generally, the quality of a test depends on three factors:

- The accurate measure of psychometric features (such as difficulty degree and discrimination degree) for each test item.

- The management of the item bank to reduce the redundancy and to maintain the consistency and integrity of the test items.
- An efficient algorithm for generating test sheets that meet multiple assessment requirements.

Since it is difficult to simultaneously satisfy multiple requirements (or constraints) in selecting test items, most computerized testing systems generate test sheet randomly (which will be called as "random selection" in the following discussions).

In Hwang (2003b), a multiple criteria test sheet generating problem is formulated as a dynamic programming problem to minimize the distance between the parameters (e.g., discrimination, difficulty, etc.) of the generated test sheet and the objective values subject to the distribution of concept weights.

Although the dynamic programming approach has taken multiple requirements into consideration, in practical applications, more criteria need to be considered. For example, a teacher might like to assign a range of test time instead of giving an objective test time and usually a teacher will assign an expected lower bound for each concept weight instead of giving a distribution of concept weights. Moreover, the object of a group test is to discriminate the status of the students. This implies that the discrimination degree of the entire test sheet needs to be maximized. Another problem of the approach proposed in Hwang (2003b) is the possibly long execution time. As the time-complexity of the dynamic programming approach is exponential in terms of input length, the actual execution time will become unacceptably long if the number of candidate test items is large. Therefore, the approach is suitable to generate test sheets for the tests with a short test time and a small number of candidate test items, e.g., unit tests. For a test that contains a large number of candidate test items and requires a longer test time (i.e., a larger number of test items are to be selected), efficient algorithms are required.

In the following sections, we shall propose a mixed integer programming model to formulate the problem of finding a set of test items that fit the multiple assessment requirements. As the problem is $\mathscr{NP}$-hard, we propose two heuristic algorithms to find quality approximate solutions.

3. Mixed integer programming model

In this section, we formulate the problem under study as a mixed integer program, in which a set of decision variables is introduced and an objective function are defined using these decision variables. Instantiation of decision variables dictates a solution or decision and an objective function value reflects the quality of the decision outcome. During the solution seeking session, several sets of constraints are specified to reflect the physical or logical limitations that could occur in the decision-making process. In a mixed integer programming problem, some of the variables involved are required to be integers. We consider the following example to demonstrate a basic form of mixed integer programs. Suppose there are two products to produce under a weekly capacity of 142 man-hours. The profits of product one and product two are 5 and 2, respectively. A unit of product one will requires 7 man-hours, and a unit of product two requires 4 man-hours. While product one must be integral, product two can be fractional. The weekly demand of product one is 19. The problem seeking the most profitable combination of product can be expressed as

$$\begin{aligned}
\text{Maximize} \quad & z = 5x_1 + 2x_2 \\
\text{Subject to} \quad & 7x_1 + 4x_2 < 142, \\
& x_1 \leqslant 14, \\
& x_1, x_2 \geqslant 0,\ x_1 \text{is an integer.}
\end{aligned}$$

In the formulation, x_1 and x_2 are decision variables indicating the units of two products to produce, z is the object function, and $7x_1 + 4x_2 < 142$, $x_1 \leqslant 14$, $x_1, x_2 \geqslant 0$ and x_1 is an integer are the specified constraints. Moreover, variable x_1 is integer and variable x_2 is non-negative real.

Mixed integer programs have been widely adopted to model real-word applications (Hillier & Lieberman, 2001) since the past few decades. Some success stories were reported in Bermon and Hood (1999), Katok and Ott (2000) and Ambs et al. (2000).

Now, we turn to the test-sheet composition problem. Suppose, in an item bank, we have n candidate test items $Q_1, Q_2, \ldots, Q_n$, from which a subset will be selected for composing a test sheet. The test is designed to be related to m concepts, $C_1, C_2, \ldots, C_m$. The variables used in our proposed model are defined as follows:

- Decision variables: x_i is 1 if item i is selected; 0, otherwise.
- Coefficient d_i: degree of discrimination of item Q_i.
- Coefficient r_{ij}: degree of association between item Q_i and concept C_j.
- Coefficient t_i: expected time needed for answering item Q_i.
- Right-hand side h_j lower bound on the expected relevance of concept C_j.
- Right-hand side l: lower bound on the expected time required for answering all of the selected items.
- Right-hand side u: upper bound on the expected time required for answering all of the selected items.

A mixed integer programming model for selecting test items from the item bank to meet multiple assessment requirements is therefore defined as

$$\begin{aligned}
\text{Maximize} \quad & \mathrm{Z} = \textstyle\sum_{i=1}^{n} d_i x_i \Big/ \sum_{i=1}^{n} x_i \\
\text{Subject to} \quad & \textstyle\sum_{i=1}^{n} r_i j x_i \geqslant h_j, \quad j = 1,2,\ldots,m, && (1) \\
& \textstyle\sum_{i=1}^{n} t_i x_i \geqslant l, && (2) \\
& \textstyle\sum_{i=1}^{n} t_i x_i \leqslant u, && (3) \\
& x_i = 0 \text{ or } 1,\ i = 1,2,\ldots,n.
\end{aligned}$$

In the above formula, n is the number of candidate test items in the item bank, and binary variable x_i reflects the decision about item i is included or not. Constraint set (1) indicates that the selected items must have a total relevance no less than the expected relevance to each concept to be addressed. Constrain sets (2) and (3), respectively, specify the lower and upper limits on the time required to answer the selected items. In the objective function of Z, $\sum_{i=1}^{n} d_i x_i$ is the total discrimination summing over the selected items and $\sum_{i=1}^{n} x_i$ is the total number of selected items. Therefore, the objective of this model aims to select a subset of items such that the average discrimination is maximized.

Mixed integer programming models are useful for formulating real-world applications. Therefore, a variety of solution methods have been proposed in the open literature to compose solutions for such models. Theoretical as well as practical works in this aspect abound (Hillier & Lieberman, 2001). Therefore, our formulation not only represents a real-life issue in a formal mathematical form but also make known methods or commercial software products available for solving the problem. See Linderoth and Savelsbergh (1999), a comprehensive computational study. The only point that deserves our further consideration is the cost of software and the time needed for finding an *optimal* solution.

The complexity of the studied problem originated from the classical knapsack problem, in which a knapsack and a set of stones are available and the goal is to select a subset of stones such that the total value of the selected stones is maximum subject to the capacity constraint of the knapsack. This problem is already known to be $\mathcal{NP}$-hard (Garey & Johnson, 1979). The test sheet composition problem can be regarded as a kind of knapsack problem even if only test time and discrimination degrees are considered. Apparently, the test-sheet composition problem is far more sophisticated than the knapsack problem. As a sequel, it is very *unlikely* to develop efficient algorithms that can produce optimal solutions within an acceptable amount of time (Garey & Johnson, 1979). In the next section we circumvent to develop heuristic procedures that can compose approximate solutions of a certain degree of quality in a reasonable time.

4. A heuristic algorithms for test-sheet construction

In this section, we shall propose two test-sheet constructing algorithms, *FTF* (Feasible Time First) and *CLF* (Concept Lower-bound First). In *FTF*, we attempt to select a set of test items to meet the upper bound and lower bound on the expected answering time first, and then substitute the selected test items with the candidate test items to meet the lower bound on the expected relevance of each concept. In *CLF*, the lower bound on the expected relevance of each concept is satisfied first. The algorithms are outlined in the following. For pseudo-codes of detail steps, the reader is referred to Appendix A.

4.1. FTF algorithm

Input: test items $Q_1, Q_2, \ldots, Q_n$, concepts, $C_1, C_2, \ldots, C_m$.
$d[i]$: degree of discrimination of Q_i;
$r[i,j]$: degree of association between Q_i and C_j;
$t[i]$: expected time needed for answering Q_i;
$h[j]$: lower bound on the expected relevance of concept C_j;
l: lower bound on the expected time needed for answering the selected items;
u: upper bound on the expected time needed for answering the selected items;
Step 1. Sort all test items in non-increasing order of $d[i]$'s.
Step 2. Create index $IC[j][i]$ for Concept C_j with non-increasing order of $r[i][j] \times d[j]$.

Step 3. For $l\_idx = l$ to $(l + u)/2$ do Step 3.1 to 3.5 and report the solution with the best discrimination degree.

Step 3.1. Find a set of test items that has a feasible expected answering time T. While the total time needed for answering the selected test items is less than l_idx, select the candidate test item with the largest discrimination degree and accordingly update the total time.

Step 3.2. Substitute the selected test items with candidate test items to meet the relevance lower bound of each concept. Check each concept C_j, in the order of $j = 1$ to m. If concept C_j's total relevance is less than the lower bound $h[j]$, then find a candidate test item i and a selected test item s. If replacing test item s with item i will increase concept C_j's total relevance and the total answering time will remain within the allowed range, and the total relevance for C_1 to C_{j-1} remain within the allowed range of expected relevance, then perform the substitution operation.

Step 3.3. If there exists any unsatisfied $h[j]$ and T is smaller than u, then select $h[j]$'s relevant test item ls with the highest discrimination degree under the prompt that $T + t[ls]$ remains no greater than upper bound.

Step 3.4. Unselected any test item ds with a degree of discrimination smaller than the average discrimination under the prompt that $l < T - t[ds] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting ds.

Step 3.5. If $T < u$ then select any test item ls with a degree of discrimination greater than the average discrimination under the prompt that $l < T - t[ls] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting ls.

4.2. CLF algorithm

Input: test items $Q_1, Q_2, \ldots, Q_n$, concepts, $C_1, C_2, \ldots, C_m$.
$d[i]$: degree of discrimination of Q_i;
$r[i,j]$; degree of association between Q_i and C_j;
$t[i]$: expected time needed for answering Q_i;
$h[j]$: lower bound on the expected relevance of concept C_j;
l: lower bound on the expected time needed for answering the selected items;
u: upper bound on the expected time needed for answering the selected items;

Step 1. Sort all test items in non-increasing order of $d[i]$'s.

Step 2. Create index $IC[j][i]$ for Concept C_j in non-increasing order of $r[i][j] \times d[j]$.

Step 3. Create index IT by sorting test items in non-increasing order of $t[i]$'s.

Step 4. Find a set of test items that meet the relevance lower bound of each concept. While there exists any concept whose total relevance is less than the lower bound, select a relevant candidate test item with the largest discrimination degree.

Step 5. Substitute selected test items with candidate test items to meet the expected answering time. While the total answering time for the selected test items is less than the lower bound, find a candidate test item with the longest test time to replace a selected test item with the shortest test time under the constraint that the total relevance of every concept will remain greater than or equal to the corresponding lower bound. While the total answering time for the selected test items is greater than the upper bound,

find a candidate test item with the shortest test time to replace a selected test item with the longest test time under the constraint that the total relevance of every concept will remain greater than or equal to the corresponding lower bound.

Step 6. Unselected any test item ds with a degree of discrimination smaller than the average discrimination under the prompt that $l < T - t[ds] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting ds.

Step 7. If $T < u$ then select any test item ls with a degree of discrimination greater than the average discrimination under the prompt that $l < T - t[ls] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting ls.

It is clear that the above two algorithms are not aimed to provide optimal test sheets. They work by iteratively improving a draft until no more gains are further attainable. In the sense of local search, the algorithms might be trapped in a local optimum. In trading with the exceedingly long time required by methods that guarantee optimal solutions, heuristics usually provide satisfactory, although not optimal, solutions with reasonable responsiveness. In the following section, we shall use several test cases to examine the execution time and solution quality of the two heuristics.

5. Experiments and evaluation

To evaluate the performance of the proposed algorithms, two experiments have been conducted to compare the execution time and the solutions of four solution-seeking strategies, FTF, CLF, random selection and exhaustive search. The platform of the experiments is a personal computer with a Pentium III 1.0 GHz CPU and 256 MB RAM. The programs are coded in Java.

In total, eight item banks were used in the experiments. The test banks are stored in a Microsoft Access database. Table 1 gives the features of the item banks used in the experiments. Columns entitled N indicate the numbers of test items in the item banks, *IT*-time is the time elapsed for creating the index file by arranging the test items in non-increasing order of $t[i]$'s, IC-time is the time required for creating the index file $IC[j][i]$ in non-increasing order of $r[i][j] \times d[j]$.

The experiment was conducted by applying *FTF* and *CLF* 20 times on each item bank with the average execution time and discrimination degree recorded. Tables 2–4 show the experimental results for lower bounds of test time set to be 30, 60 and 120 minutes, respectively. It can be seen

Table 1
Features of the item banks

Item bank	N	Loading time (s)	*IT*-time (s)	*IC*-time (s)	Average discri.
1	25	5.067	<0.001	<0.001	0.63267
2	30	5.308	<0.001	<0.001	0.65331
3	40	5.217	<0.001	<0.001	0.66602
4	250	8.522	0.004	0.020	0.60985
5	500	8.703	0.010	0.060	0.60920
6	1000	13.599	0.020	0.230	0.61208
7	2000	28.361	0.121	0.931	0.61339
8	4000	60.887	0.450	3.756	0.61534

that for most cases, it is time-consuming to derive optimum solutions. For $N = 30$ and $l = 120$, it takes more than 3 hours (i.e. 187 min) to find the optimal solution. Such a lengthy process is obviously unacceptable. When the values of N and l increase, it becomes very unlikely to find optimal solutions in a reasonable time. This fact stimulates the need for the design of heuristic algorithms for deriving approximate solutions of a certain quality level. In Tables 2 and 3, the solutions given by *FTF* and *CLF* are pretty close to the optimum ones. In terms of run times and discrimination deviations, our proposed methods demonstrate not only convincing effectiveness but also impressive efficiency.

Table 2
Experimental results for $l = 30$

N	Random selection		*FTF*		*CLF*		Optimum solution	
	Time (s)	Discri.	Time (s)	Discri.	Time (s)	Discri.	Time (min)	Discri.
25	0.03	0.63704	0.06	0.75466	0.03	0.75466	5	0.75466
30	0.03	0.69388	0.07	0.81812	0.03	0.81812	187	0.81812
40	0.03	0.64978	0.09	0.88001	0.03	0.86188	163,840	0.88144
250	0.03	0.54248	0.07	0.93936	0.05	0.91161	$>10^6$	N/A
500	0.03	0.60500	0.08	0.95161	0.10	0.93046	N/A	N/A
1000	0.03	0.69753	0.09	0.95253	0.29	0.94831	N/A	N/A
2000	0.03	0.54342	0.21	0.95322	1.08	0.95156	N/A	N/A
4000	0.03	0.48540	0.56	0.95649	4.23	0.95256	N/A	N/A

Table 3
Experimental results for $l = 60$

N	Random selection		*FTF*		*CLF*		Optimum solution	
	Time (s)	Discri.	Time (s)	Discri.	Time (s)	Discri.	Time (min)	Discri.
30	0.03	0.64321	0.08	0.70726	0.03	0.70726	187	0.70726
40	0.03	0.63240	0.08	0.80639	0.04	0.79604	163,840	0.80639
250	0.03	0.62150	0.16	0.90191	0.05	0.89935	$>10^6$	N/A
500	0.03	0.55859	0.12	0.94052	0.10	0.91458	N/A	N/A
1000	0.03	0.63284	0.16	0.95037	0.28	0.93440	N/A	N/A
2000	0.03	0.60746	0.24	0.95166	1.08	0.94513	N/A	N/A
4000	0.03	0.62240	0.63	0.95219	4.23	0.94712	N/A	N/A

Table 4
Experimental results for $l = 120$

N	Random selection		*FTP*		*CLF*		Optimal solution	
	Time (s)	Discri.	Time (s)	Discri.	Time (s)	Discri.	Time (min)	Discri.
250	0.03	0.59964	0.22	0.88729	0.05	0.87227	$>10^6$	N/A
500	0.03	0.66515	0.16	0.92266	0.10	0.89674	N/A	N/A
1000	0.03	0.62918	0.11	0.93825	0.28	0.91799	N/A	N/A
2000	0.03	0.59838	0.23	0.94162	1.08	0.93358	N/A	N/A
4000	0.03	0.61402	0.60	0.94654	4.23	0.94171	N/A	N/A

To better understand the trends of the solution quality and run time of various solution approaches, we further visualize the numerical results in line charts as shown in Figs. 1–3. It can be recognized that *FTF* and *CLF* have achieved much better results than random selection, which is commonly adopted in real-world situations. In fact, the results of *FTF* and CLF are very close to those known optimum solutions. For the cases with more than 250 candidate test items, the execution time required by any application of exact methods for composing an optimal test sheet is over 1,000,000 min, which are of course unacceptable. However, *FTF* and *CLF* can still generate test sheets with degrees of discrimination greater than 0.9. Such test sheets would be very useful for evaluating the learning status of students. An important observation to address is related to the guaranteed performance. When a larger test bank is deployed, the two heuristics can construct test sheets with higher discrimination degrees. To discriminate the relative effectiveness between *FTF* and *CLF*, we have the observation that *FTF* outperforms *CLF* for all test cases.

Fig. 4 further depicts the line charts concerning the execution times required by *FTF* and *CLF* and the time needed for finding optimum solutions. When the number of candidate test items exceeds 40, it is almost impossible to find an optimal solution, while *FTF* and *CLF* can find near-optimal solution in a very short time (less than 0.1 s). Moreover, when more candidate items are

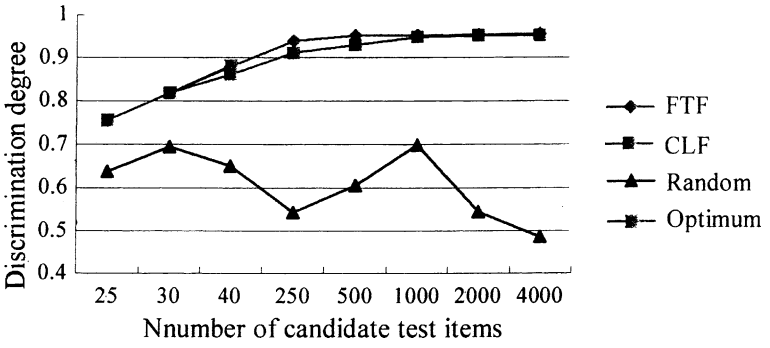

Fig. 1. Solution qualities for $l = 30$.

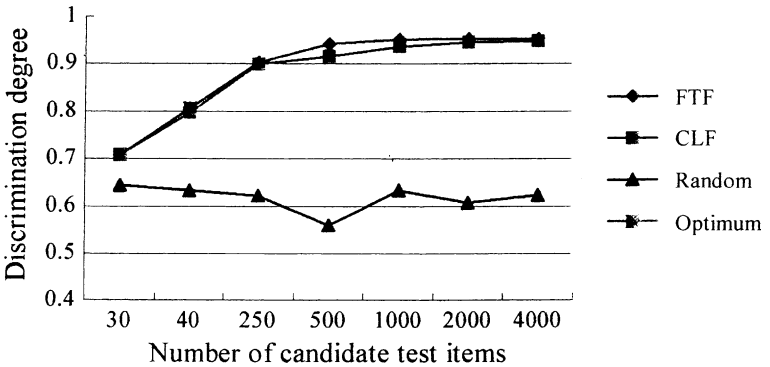

Fig. 2. Solution qualities for $l = 60$.

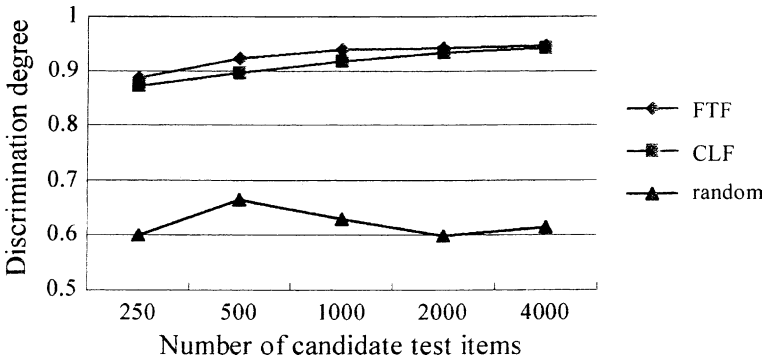

Fig. 3. Solution qualities of *FTF*, *CLF* and *Random* for $l = 120$.

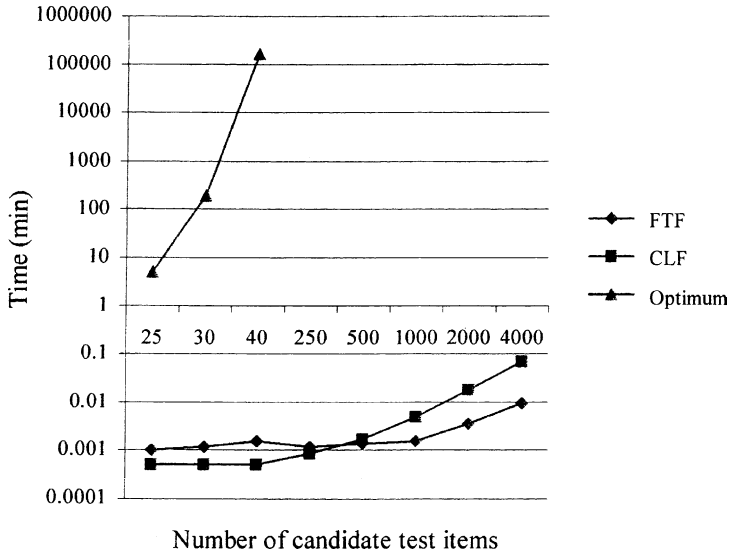

Fig. 4. Run times of *FTP*, *CLF* and *Optimum* for $l = 30$.

considered, the performance of *FTF* is slightly better than *CLF*, regardless of the expected length of testing time.

Although the two heuristics are both based upon an iterative improvement policy, they improve the solutions from different angles. Therefore, it is interesting to know the relationship between execution times and improvement approaches. Fig. 5 shows the execution times of *FTF* and *CLF* for $l = 120$. In fact, a longer test time usually implies a larger number of test items will be selected to construct a test sheet. It can be seen that *FTF* achieves much better performances than *CLF* for the cases involving more candidate items and more selected test items. Algorithm *CLF* takes a longer time mainly due to its Step 5 that incurs a great number of substution operations. In summary, we can conclude that *FTF* outperforms *CLF* in the aspects regarding execution time and degree of discrimination.

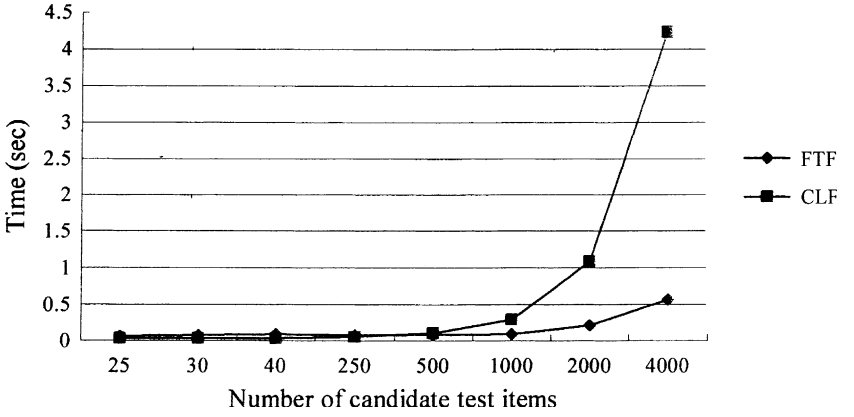

Fig. 5. Run times of *FTF* and *CLF* for $l = 120$.

6. Conclusions and future work

In this paper, we have considered the composition issue of test sheets from test banks. The problem seeks to select items to form a test sheet such that the discrimination degree can be attained to a certain level. We have proposed a mixed integer programming model to formulate this real-world problem that incorporates multiple assessment requirements. As the studied problem is $\mathcal{NP}$-hard, it is unlikely to solve it by any known algorithm in a reasonable time for practical applications. To cope with the situations, two heuristic algorithms, *FTF* and *CLF*, have been developed for producing approximate solutions. The two algorithms deploy different approaches to adjusting, subject to the specified requirement, a preliminary test sheet in an iterative fashion. From the experimental results, it can be clearly evinced that our approach is able to meet the specified requirements by efficiently finding near-optimal combinations of the test items. While any exact algorithm takes an exceptionally long time to come up with an optimal test sheet, the two heuristics can produce test sheets of high discrimination degrees in a few seconds. In practice, instructors may select test items manually or use computers to perform a random draw. Both approaches cannot perfectly take time and quality into account simultaneously. Our algorithm have provided a systematic way to fulfilling the multiple criteria. Moreover, algorithm *FTF* can achieve a much better performance while the problem scale is large. Therefore, algorithm *FTF* is recommended to be a good solution method for coping with the test-sheet generation problem.

The performances of our proposed algorithms convincingly suggest the possible realization of workable testing systems. The FTF algorithm has been applied to several practical applications. An ASP (Active Service Page) Design course, a K-6 Natural Science course and a K-9 Mathematics course have been developed. For further extensive applications, we are working out collaborative plans with some local test bank providers, who maintain large-scale test banks to which students as well as instructors can have access through the world-wide-web under a micro-payment mechanism. No matter which specific application domain our algorithm will be deployed to provide the decision aids, we expect to establish a system, rather than an algorithm or program, that is more informative, flexible, and capable for the instructors.

Acknowledgement

This study is supported in part by the National Science Council of the ROC under Contract No. NSC-91-2520-S-260-002.

Appendix A. Pseudo-codes of algorithms *FTF* and *CLF*

A.1. FTF algorithm

Input: test items $Q_1, Q_2, \ldots, Q_n$, concepts $C_1, C_2, \ldots, C_m$.
$d[i]$: degree of discrimination of Q_i;
$r[i,j]$: degree of association between Q_i and C_j;
$t[i]$: expected time needed for answering Q_i;
$h[j]$: lower bound on the expected relevance of concept C_j;
l: lower bound on the expected time needed for answering the selected items;
u: upper bound on the expected time needed for answering the selected items;

Step 1. Use DB input test items and sorting test items in non-increasing order of $d[i]$'s

Step 2. Create index $IC[j][i]$ for Concept C_j in non-increasing order of $r[i][j] \times d[j]$

Step 3. For $l\_idx = l$ to $(l + u)/2$ do Step 3.1 to 3.5 and report the solution with the best discrimination degree

Step 3.1. Find a set of test items with feasible expected answering time

```
For i = 1 to n do
      x[i] = 0; initially no test item is selected
End if
D = 0; accumulated degree of discrimination of the selected test items
T = 0; total time needed for answering the selected test items
N = 0; number of selected test items
For j = 1 to m do
 Acc_C[j] = 0; total relevance for selected test items to Concept C_j
End if
k = 1; select the test items that have maximum degree of discrimination first
While (T < l_idx) do
 If (x[k] = 0) Then
        If ((T + t[k]) ⩽ u) Then
            T = T + t[k]
            x[k] = 1
            For j = 1 to m do
                Acc_C[j] = Acc_C[j] + r[k][j]
            End For
            N = N + 1
            D = D + d[k]
```

End If
$k = k + 1$
End If
End While

Step 3.2. Substitute selected test items with candidate test items to meet the relevance lower bound of each concept

For $j = 1$ to m do; for each Concept C_j
$s = 1$; index for selected test items
$i = n$; index for candidate test items
While ($Acc\_C[j] < h[j]$) do
While ($x[IC[j][s]] = 0$) do; find next selected test item
$s = (s + 1) \bmod n$; check $s + 1 = n$ then s need not mod n
End While

While ($x[i] = 1$) do; find next candidate test item
$i = (i - 1)$; check $i - 1 = 0$ then $i = n$
End While

If ((($l \leqslant T - t[IC[j][s]] + t[i]) \leqslant u$) and ($r[IC[j][s]][j] < r[i][j]$)) Then
$exchange\_flag = 1$; 1: substitution is acceptable
; 0: substitution is not acceptable
For $k = 1$ to $j - 1$ do
If ($Acc\_C[k] > h[k]$) Then
If (($Acc\_C[k] - r[IC[j][s]][k] + r[i][k]) < h[k]$) Then
$exchange\_flag = 0$
End if
End if

If $exchange\_flag = 1$ Then; Perform test item exchange
For $k = 1$ to m do
; Accumulate weights of relevant concepts
$Acc\_C[k] = Acc\_C[k] - r[IC[j][s]][k] + r[ID[i]][k]$
End For
$x[IC[j][s]] = 0$; Update flags
$x[ID[i]] = 1$
End if
$D = D - d[IC[j][s]] + d[ID[i]]$; Update discrimination degree
End If
End While
End For

Step 3.3. If there exists any unsatisfied $h[j]$ and T is smaller than u, then select $h[j]$'s relevant test item ls with highest discrimination degree under the prompt that $T + t[ls]$ remains smaller than upper bound.

Step 3.4. Unselected any test item *ds* with degree of discrimination smaller than the average discrimination under the prompt that $l < T - t[ds] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting *ds*.

Step 3.5. If $T < u$ then select any test item *ls* with degree of discrimination greater than the average discrimination under the prompt that $l < T - t[ls] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting *ls*.

A.2. CLF algorithm

Input: test items $Q_1, Q_2 \ldots, Q_n$, concepts $C_1, C_2, \ldots, C_m$.
$d[i]$: degree of discrimination of Q_i;
$r[i,j]$; degree of association between Q_i and C_j;
$t[i]$: expected time needed for answering Q_i;
$h[j]$: lower bound on the expected relevance of concept C_j;
l: lower bound on the expected time needed for answering the selected items;
u: upper bound on the expected time needed for answering the selected items;

Step 1. Use DB input test items and sorting test items in non-increasing order of $d[i]$'s.

Step 2. Create index $IC[j][i]$ for Concept C_j in non-increasing order of $r[i][j] \times d[j]$

Step 3. Create index IT by sorting test items in non-increasing order of $t[i]$'s

Step 4. Find a set of test items to meet the relevance lower bound of each concept

```
int I[j]; index for IC[j]
For i = 1 to n do
x[i] = 0; initially no test item is selected
        End if
D = 0; accumulated degree of discrimination of the selected test items
T = 0; total time needed for answering the selected test items
N = 0; number of selected test items
For j = 1 to m do
        Acc_C[j] = 0; total relevance for selected test items to Concept C_j
        I[j] = 1; initially each index for IC[j] pointing to the first position
End if
counter = 0; the number of concepts that satisfy their lower bounds of weight
While (counter < m) do
; if counter = m, then all of the concepts have satisfied their weight lower bounds.
        For j = 1 to m do
            If (Acc_C[j] < h[j]) Then
            x[IC[j][I[j]]] = 1
            N = N + 1
            T = T + t[IC[j][I[j]]]
            For k = 1 to m do
```

```
                Acc_C[k] = Acc_C[k] + r[IC[j]I[j]][k]
            D = D + d[IC[j][I[j]]]
            I[j] = I[j] + 1
        End If
    End For
    For j = 1 to m do
        If (Acc_C[j] ⩾ h[j]) Then
            counter = counter + 1
        End If
    End For
End While
```

Step 5. Substitute selected test items with candidate test items to meet the expected answering time

```
p = 1
q = n
While (T < l) or (T > u) do
        While (T < l) do
         If (x[p] = 0) Then
            x[p] = 1
            T = T + t[p]
            For j = 1 to m do
                Acc_C[j] = Acc_C[j] + r[p][j]
            End For
            p = p + 1
                D = D + d[p]
                N = N + 1
        End If
End While
While (T > u) then
        While (x[IT[q]] = 1) do
        q = q − 1; Find a candidate item with longest answering time
        If q ⩽ p Then
          q = n
              p = p + 1
          End If
        End While
While (x[IT[p]] = 0) do
p − p + 1; Find a selected item with shortest answering time
If p ⩾ q Then
        p = 1
        q = q − 1
End If
End While
```

exchange_flag = 1
For $j = 1$ to m do; Check if the substitution violates $h[j]$ constraint
If ($Acc\_C[j] - r[IT[p],j] + r[IT[q],j] < h[j]$) then
exchange_flag = 0
End If
If *exchange_flag* = 1 then
For $k = 1$ to m do
$Acc\_X[k] = Acc\_C[k] - r[IT[p],k] + r[IT[q],k]$
End For
$x[IT[p]] = 0$
$x[IT[q]] = 1$
$D = D - d[IT[p]] + d[IT[q]]$
Else
$p = p + 1$
End If
End While
End While

Step 6. Unselected any test item ds with degree of discrimination smaller than the average discrimination under the prompt that $l < T - t[ds] < u$ and $Acc\_C[j] \geqslant h[j]$ for it j = 1 to m after unselecting *ds*.

Step 7. If $T < u$ then select any test item *ls* with degree of discrimination greater than the average discrimination under the prompt that $l < T - t[ls] < u$ and $Acc\_C[j] \geqslant h[j]$ for $j = 1$ to m after unselecting *ls*.

References

Ambs, K., Cwilich, S., Deng, M., Houck, D. J., Lynch, D. F., & Yan, D. (2000). Optimizing restoration capacity in the AT&T network. *Interfaces, 30*(1), 26–44.

Antao, B. A. A., Brodersen, A. J., Bourne, J. R., & Cantwell, J. R. (2000). Building intelligent tutorial systems for teaching simulation in engineering education. *IEEE Transactions on Education, 35*(3), 222–225.

Bermon, S., & Hood, S. J. (1999). Capacity optimization planning system (CAPS). *Interfaces, 29*(5), 31–50.

Chou, C. (2000). Constructing a computer-assisted testing and evaluation system on the world wide web-the GATES experience. *IEEE Transactions on Education, 43*(3), 266–272.

Fan, J.P., Tina, K.M., & Shue, L.Y. (1996). Development of knowledge-based computer-assisted instruction system. In *Proceedings of the 1996 international conference software engineering: education and practice*, Dunedin, New Zealand.

Feldman, J. M., & Jones, J. (1997). Semiautomatic testing of student software under Unix(R). *IEEE Transactions on Education, 40*(2), 158–161.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of $\mathcal{NP}$-completeness*. San Francisco, CA: Freedman.

Gonzalez, A. V., & Ingraham, L. R. (1994). Automated exercise progression in simulation-based training. *IEEE Transactions on Systems, Man and Cybernetics, 24*(6), 863–874.

Harp, S. A., Samad, T., & Villano, M. (1995). Modeling student knowledge with self-organizing feature maps. *IEEE Transactions on Systems, Man and Cybernetics, 25*(5), 727–737.

Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to operations research* (7th ed.). New York: McGraw-Hill.

Hwang, G. J. (1998). A tutoring strategy supporting system for distance learning on computer networks. *IEEE Transactions on Education, 41*(4), 1–19.

Hwang, G. J. (2003a). A concept map model for developing intelligent tutoring systems. *Computers & Education, 40*(3), 217–235.

Hwang, G. J. (2003b). A test sheet-generating algorithm for multiple assessment requirements. *IEEE Transactions on Education, 46*(3), 329–337.

Johnson, W. B., Neste, L. O., & Duncan, P. C. (1989). An authoring environment for intelligent tutoring systems. *IEEE International Conference on Systems, Man and Cybernetics, 2*, 761–765.

Katok, E., & Ott, D. (2000). Using mixed-integer programming to reduce label changes in the Coors aluminum can plant. *Interfaces, 30*(2), 1–12.

Khan, B. H. (1997). Web-based instruction (WBI): What is it and why is it?. In B. H. Khan (Ed.), *Web-based instruction* (pp. 5–18). Englewood Cliffs, NJ: Educational Technology.

Linderoth, J. T., & Savelsbergh, M. W. P. (1999). A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing, 11*(2), 173–187.

Lira, P., Bronfman, M., & Eyzaguirre, J. (1990). MULTITESTII: a program for the generation, correction; and analysis of multiple choice tests. *IEEE Transactions on Education, 33*(4), 320–325.

Olsen, J.B., Maynes, D.D., Slawson, D., & Ho, K. (1986). Comparison and equating of paper-administered, computer-administered and computerized adaptive tests of achievement. In: *Proceedings of the annual meeting of American educational research association*, California, April 16–20, 1986.

Rasmussen, K., Northrup, P., & Lee, R. (1997). Implementing Web-based instruction. In B. H. Khan (Ed.), *Web-based instruction* (pp. 341–346). Englewood Cliffs, NJ: Educational Technology.

Rowe, N. C., & Galvin, T. P. (1998). An authoring system for intelligent procedural-skill tutors. *IEEE Intelligent Systems, 13*(3), 61–69.

Sun, C. T., & Chou, C. (1996). Experiencing CORAL: design and implementation of distance cooperative learning. *IEEE Transactions on Education, 39*(3), 357–366.

Vasandani, V., & Govindaraj, T. (1995). Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. *IEEE Transactions on Systems, Man and Cybernetics, 25*(7), 1076–1096.

Wainer, H. (1990). *Computerized adaptive testing A primer*. Hillsdale, NJ: Lawrence Erlbaum.