[11] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. Int. Conf. Machine Learning*, 1998, pp. 242–250.
[12] M. Kaya and R. Alhajj, "Reinforcement learning in multiagent systems: A modular fuzzy approach with internal model capabilities," in *Proc. IEEE Int. Conf. Tools Artificial Intelligence*, Nov. 2002, pp. 469–474.
[13] M. L. Littman, "Markov games as a framework for multi agent reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 1994, pp. 157–163.
[14] Y. Nagayuki, S. Ishii, and K. Doya, "Multi-agent reinforcement learning: An approach based on the other agent's internal model," in *Proc. IEEE Int. Conf. Multiagent Systems*, Jul. 2000, pp. 215–221.
[15] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auton. Robotics*, vol. 8, no. 3, 2000.
[16] T. W. Sandholm and R. H. Crites, "Multi agent reinforcement learning in the iterated prisoner's dilemma," *Biosystems*, vol. 37, pp. 147–166, 1995.
[17] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proc. Int. Conf. Very Large Databases*, 1995, pp. 432–443.
[18] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proc. ACM SIGMOD Int. Conf. Management Data*, 1996, pp. 1–12.
[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[20] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. Int. Conf. Machine Learning*, 1993, pp. 330–337.
[21] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.
[22] S. Zhang, C. Zhang, and X. Yan, "Post-mining: Maintenance of association rules by weighting," *Inform. Syst.*, vol. 28, no. 7, pp. 691–707, 2003.

On the Development of a Computer-Assisted Testing System With Genetic Test Sheet-Generating Approach

Gwo-Jen Hwang, Bertrand M. T. Lin, Hsien-Hao Tseng, and Tsung-Liang Lin

***Abstract*—Since the last decade, computer-assisted testing has proven to be an efficient and effective way to evaluating students' learning status such that proper tutoring strategies can be adopted to improve their learning performance. A good test will not only help the instructor evaluate the learning status of the students, but also facilitate the diagnosis of the problems embedded in the students' learning process. One of the most important and challenging issues in conducting a good test is the construction of test sheets that can meet various assessment requirements. A previous study has indicated that selecting test items to best fit multiple assessment requirements can be formulated as a mixed integer programming model. The problem is known to be NP-hard in the literature and, hence, computational challenges hinder the development of efficient solution methods. As a sequel, we instead seek quality approximate solutions in a reasonable time. Two approximation methods based upon a genetic approach are developed. Statistics from a series of computational experiments indicate that our approach is able to efficiently generate near-optimal combinations of test items that satisfy the specified requirements or constraints.**

***Index Terms*—Computer-assisted testing, genetic algorithm (GA), mixed integer programming, test sheet generating.**

I. Background and Motivations

In recent years, educators have reported the importance of conducting an interactive and personalized tutoring process, which is helpful toward the training of creativity and the improvement of learning performance in children. The need for interactive and personalized tutoring environments has encouraged the development of computer-assisted-instruction (CAI) systems which are able to record the learning status of each student and provide adaptive subject materials and practice drills. Therefore, it is very important to precisely determine the learning status of each student so that proper tutoring strategies can be applied accordingly [10], [17]. A high-quality test is the major criterion for determining the learning status of students.

Computer-based tests have been proven to be more effective and efficient than traditional paper-and-pencil tests due to several reasons: First, the test sheets can be composed dynamically based on the practical requirements; second, more plentiful test items can be presented in multimedia styles; third, the student testing portfolio can be recorded and analyzed to improve their learning performance [5], [15], [19].

The key to a high-quality test not only depends on the quality of test items, but also the way the test sheet is constructed [10], [14]. As the number of test items in an item bank is usually large and the number of feasible combinations to form test sheets thus grows exponentially, it is very difficult to find an *optimal* test sheet in a timely manner [3], [6], [11], [12]. Such an issue is likely to grow in importance owning to the rapid advent of Internet technologies and the fast growth of network-

Manuscript received June 15, 2003; revised October 28, 2004. This work was supported by the National Science Council of the Republic of China under Contracts NSC-91-2520-S-260-003 and NSC92-2524-S-260-002. This paper was recommended by Guest Editor S. H. Rubin.

G.-J. Hwang is with the Department of Information and Learning Technology, National University of Tainan, Tainan, Taiwan 700, R.O.C. (e-mail: gjhwang@mail.nutn.edu.tw).

B. M. T. Lin is with the Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan 300, R.O.C.

H.-H. Tseng and T.-L. Lin are with the Department of Information Management, National Chi Nan University, Pu-Li, Taiwan 545, R.O.C.

Digital Object Identifier 10.1109/TSMCC.2004.843184

based educational systems and online learning population. Along with the growth of distance learning through the Internet, computer-based assessment systems are also becoming demanding.

Although many computer-assisted testing systems have been proposed, few of them have addressed the problem of systematically composing test sheets for multiple assessment requirements [2], [17]. Most of the existing systems construct a test sheet by manually or randomly selecting test items from their item banks. Such manual or random test item selecting strategies are inefficient and usually are not able to simultaneously meet multiple assessment requirements. Some previous investigations attempted to employ a dynamic programming algorithm to find an optimal composition of the test items [11]. As the time complexity of the dynamic programming algorithm is exponential in terms of the size of input data, the required execution time will become unacceptably long if the number of candidate test items is large.

To cope with the increasingly hard situations encountered in developing optimal test sheets, we shall present two mixed integer programming models to formulate the problems of finding a set of test items that fit multiple assessment requirements. As the problems are strongly NP-hard, we propose two genetic algorithms [4], [7], [13], [16]–[18] to find quality approximate solutions in acceptable time. Computational experiments will be also presented to study the performances of the proposed algorithms.

II. Mixed Integer Programming Models

In an item bank, a subset of n candidate test items $Q_1, Q_2, \ldots, Q_n$ will be selected for composing a test sheet. In the following subsections, we shall present two models that formulate the test sheet-generating problem under different assessment considerations. The first model was proposed by [12] that is aimed at optimizing the discrimination degree of the generated test sheets with a specified range of assessment time and some other multiple constraints. The second model proposed in this paper formulates the optimization of discrimination degree of the generated test sheets with a fixed number of test items as the major constraint.

A. *Specified Length of Assessment Time (SLAT) Problem*

In the SLAT problem, the major consideration is to confine the length required by the students to answer the selected items. Assume there are n items in the item bank and m concepts are involved in the test. The variables used in the formulated models are defined as follows:

- Decision variables $x_i, 1 \le i \le n : x_i$ is 1 if test item i is selected; 0, otherwise.
- Coefficient $t_i, 1 \le i \le n$: Expected time needed for answering item Q_i.
- Coefficient $d_i, 1 \le i \le n$: Degree of discrimination of Q_i.
- Coefficient $r_{ij}, 1 \le i \le n, 1 \le j \le m$: Degree of association between Q_i and concept C_j.
- Right-hand side $h_j, 1 \le j \le m$: Lower bound on the expected relevance of C_j.
- Right-hand side l: Lower bound on the expected time needed for answering the selected items.
- Right-hand side u: Upper bound on the expected time needed for answering the selected items.

Objective function

$$\text{Maximize } Z = \sum_{i=1}^{n} d_i x_i \bigg/ \sum_{i=1}^{n} x_i$$

Subject to

$$\sum_{i=1}^{n} r_{ij} x_i \ge h_j, \quad j = 1, 2, \ldots, m \tag{1}$$

$$\sum_{i=1}^{n} t_i x_i \ge l \tag{2}$$

$$\sum_{i=1}^{n} t_i x_i \le u; \quad x_i = 0 \text{ or } 1, \quad i = 1, 2, \ldots, n. \tag{3}$$

In the above formula, binary variable x_i reflects the decision about whether test 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. Constraint sets (2) and (3), respectively, specify the lower and upper limits on the time needed to answer the selected items. In the objective function, $\sum_{i=1}^{n} d_i x_i$ is the total discrimination summing over the selected test items and $\sum_{i=1}^{n} x_i$ is the number of test items selected. Therefore, the objective of this model aims to select a subset of test items such that average discrimination is maximized.

B. *Fixed Number of Test Items (FNTI) Problem*

In the FNTI problem, the number of test items is specified and fixed as $q\_\text{num} \le n$. The variables used in this model are given as follows.

- Decision variables: x_i is an integer variable that reflects the decision about which test item would be selected and designated as question $i, 1 \le x_i \le n, i = 1, 2, \ldots, q\_\text{num}$.
- Right-hand side $h_j, 1 \le j \le m$: lower bound on the expected relevance of concept C_j.

Objective function

$$\text{Maximize } Z = \sum_{i=1}^{q\_\text{num}} d_{x_i}$$

Subject to

$$\sum_{i=1}^{q\_\text{num}} r_{ij} \ge h, \quad j = 1, 2, \ldots, m \tag{4}$$

$$x_1 \ge 1 \tag{5}$$

$$x_{i+1} > x_i, \quad 1 \le i \le q\_\text{num} - 1. \tag{6}$$

In the above formula, constraint set (4) indicates the selected test items must have a total relevance that is no less than the expected relevance to each concept to be covered. Constraint sets (5) and (6) indicate that no test item can be selected twice or more. In the objective function, $\sum_{i=1}^{q\_\text{num}} d_{x_i}$ is the total discrimination summing over the selected test items. Therefore, the objective of this model seeks to select a fixed number of test items such that the total discrimination is maximized.

III. Genetic Algorithms for Test Sheet Generation

In this section, we shall propose two genetic algorithms (GAs), concept lower-bound first genetic approach (CLFG) and feasible item first genetic approach (FIFG) to find quality approximate solutions for the SLAT and FNTI problems. In CLFG, we shall select a set of test items to meet the lower bound on the expected relevance of each concept first, and then substitute some of the selected test items with the candidate test items to meet the upper bound and lower bound of the expected answering time. In FIFG, we confine the number of test items of the test sheet first, and then substitute some of the selected items with the candidate items to meet the lower bound on the expected relevance of each concept.

A. *Concept Lower-Bound First Genetic (CLFG) Approach*

To cope with the SLAT problem, we propose the CLFG approach as follows:

Input: test items $Q_1, Q_2, \ldots, Q_n$ and concepts $C_1, C_2, \ldots, C_m$.

1) Step 1. Create Population (Encode): Let variable S denote the set of initially generated chromosomes and variable K be the size of the population in S. Chromosome S_k is represented as an n-bit binary string $[x_{k1}, x_{k2}, \ldots, x_{kn}]$ consisting of n genes, where x_{ki} is either 1 or 0 indicating that the test item is currently selected or not. An initial set of binary strings, such as $X = [0, 0, 1, \ldots, 0]$, is randomly generated to represent the status of each test item.

2) Step 2. Fitness Ranking: To satisfy the constraints for the lower bound on the expected relevance of each concept, we define a penalty function to approximate the constraints as $R = \mathrm{dc} \times ipt$, where R is a penalty score, $\mathrm{dc} = \sum_{j=1}^{m} \max\{h_j - \sum_{i=1}^{n} r_{ij}x, 0\}$ is the sum of deviations between the relevance of each concept in the currently selected test items and the corresponding lower bound, and *ipt* is the penalty weight defined by the instructor.

Moreover, two constraints are needed to specify the penalty values when the total testing time of the selected test items is less than the lower bound or greater than the upper bound. For the selected test items that have a total testing time of less than the lower bound, the penalty function is $\alpha = w \times dtl \times ipt\_l$, where $w = \sum_{i=1}^{n} X_i d_i/\mathrm{average}(u, l)$ represents the average discrimination weight of a chromosome, $dtl = \max\{l - \sum_{i=1}^{n} t_i x_i, 0\}$ and ipt_l are the user-defined penalty weight penalizing the violation of lower bound constraint.

For the selected test items that have a total testing time greater than the upper bound, the penalty function is $\beta = w \times dtu \times ipt\_u$, where $w = \sum_{i=1}^{n} X_i d_i/\mathrm{average}(u, l)$ represents the average discrimination weight of chromosome $dtu = \max\{\sum_{i=1}^{n} t_i x_i - u, 0\}$, and ipt_u is a user-defined penalty weight for the case where the upper bound constraint is violated. The evaluation function is aggregated from the aforementioned weights and defined as $v(S_k) = (\sum_{i=1}^{n} d_i x_i - \alpha - \beta - R)/\sum_{i=1}^{n} x_i$.

3) Step 3. Selection: The roulette wheel approach is adopted in the fitness-proportional selection procedure, which selects a new population with respect to the probability distribution based on fitness values. The probability that chromosome S_k is selected and defined as $p_k = v(S_k)/V$, where

$$V = \sum_{k=1}^{\mathrm{pop\_size+offspring\_size}} v(s_k).$$

4) Step 4. Crossover: The one-cut-point method is used to perform the "crossover" operation by randomly selecting a cut point and exchanging the right parts of two parents to generate offsprings. In this application, the value of the crossover rate is 0.2, which was derived from the results of a series of preliminary experiments.

5) Step 5. Mutation: Mutation alters one or more genes with the mutation rate $P = n^{-1}$. A sequence of real random numbers $y_1, y_2, \ldots, y_{nk}$ is then generated with each y_i to be a real number in [0, 1]. If $y_i, 1 \le i \le nk$ is greater than P, then the rth, $r = i - (\lceil i/n \rceil - 1)n$, bit of the $\lceil i/n \rceil$ chromosome will be complemented.

Steps 2 to 5 constitute a generation. In our procedure, the whole process iterates generation by generation until either no better solution was attained within the most recent ten generations or 1500 generations have been examined. When the procedure stops, the best solution encountered is reported.

B. Feasible Item First Genetic (FIFG) Approach

To cope with the FNTI Problem, we propose a CLFG approach. The GA differs from the previous one in representation, fitness function, and mutation scheme. Therefore, we introduce these parts only.

1) Step 1. Create Population (Encode): Let variable K be the number of the chromosomes in S, the initial population, and variable S_k be the kth chromosome of S. Chromosome S_k is represented as $[x_{k1}, x_{k2}, \ldots, x_{k,q\_\mathrm{num}}]$, consisting of q_num genes, each of which denotes a selected item. A set of integers is randomly generated to represent the test item numbers, for example, $X = [25, 908, \ldots, 113]$. Note that $x_i \ne x_j$ for $1 \le i \ne j \le q\_\mathrm{num}$.

2) Step 2. Fitness Ranking: To satisfy the constraints of the lower bound on the expected relevance of concept, we define a penalty function $R = \mathrm{dc} \times ipt$, where R is a penalty score, $\mathrm{dc} = \sum_{j=1}^{m} \max\{h - \sum_{i=1}^{n} r_{x_i j}, 0\}$, which is the sum of distances between the relevance of each concept for the currently selected test items and the corresponding lower bound, and *ipt* is the user-defined penalty weight. The evaluation function is defined as

$$v(S_k) = \sum_{i=1}^{n} d_{x_i} - R.$$

3) Step 5. Mutation: "Mutation" operation alters one or more genes with the mutation rate $P = n^{-1}$. A sequence of real random numbers $y_1, y_2, \ldots, y_{q\_\mathrm{num} \times k}$ is then generated with y_i being a real number in [0, 1], for $i = 1$ to $q\_\mathrm{num} \times k$. A random number selected from 1 to n is used to replace the value of ith gene if $y_i < P$.

TABLE I
BRIEF DESCRIPTION OF EACH ITEM BANK

Item Bank	N	Loading time (second)	Average Discrimination
1	25	5.067	0.63267
2	30	5.308	0.65331
3	40	5.217	0.66602
4	250	8.522	0.60985
5	500	8.703	0.60920
6	1,000	13.599	0.61208
7	2,000	28.361	0.61339
8	4,000	60.887	0.61534

IV. Experiments and Evaluation

To evaluate the performance of the proposed algorithms, two experiments have been conducted to compare the execution time and the solution quality of four solution-seeking strategies: CLFG, FIFG, random selection, and exhaustive search. The random selection program generates the test sheet by selecting test items randomly to meet the constraints of time interval or number of test items, while the exhaustive search program examines every feasible combination of the test items to find the optimal solution. Eight item banks of K7 to K9 mathematics courses have been employed in the experiments. Table I shows a brief description of each item bank, where N indicates the total number of test items. The platform of the experiments is a personal computer with a Pentium III 1.0-GHz CPU and 256-MB random-access memory (RAM). The programs are coded in Java Language.

The experiment is conducted by applying CLFG and FIFG twenty times on each item bank with the average execution time and discrimination degree recorded. Tables II–IV show the experimental results for the lower bounds of testing time being 30, 60, and 120 min, respectively. It can be seen that for most cases, it is time-consuming to derive optimal solutions. For $N = 30$ and $l = 60$, it takes nearly 3 h (i.e., 187 min) to find an 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 reasonable time. This indicates the need for heuristic algorithms to derive approximate solutions at a certain quality level.

It can be seen that test sheets with near-optimal discrimination degrees can be obtained in a much shorter time by employing CLFG than by the random selection approach. The line charts also show that the

TABLE II
EXPERIMENTAL RESULTS FOR $l = 30$

N	CLFG		Random Selection		Optimum Solution	
	Time (sec)	Degree of Discri.	Time (sec)	Degree of Discri.	Time (min)	Degree of Discri.
25	0.13275	0.754664	0.03	0.63704	5	0.75466
30	0.14265	0.818120	0.03	0.69388	187	0.81812
40	0.27880	0.880276	0.03	0.64978	163,840	0.88144
250	0.96815	0.943386	0.03	0.54248	$>10^6$	N/A
500	1.98875	0.952377	0.03	0.60500	N/A	N/A
1,000	3.75490	0.957359	0.03	0.69753	N/A	N/A
2,000	7.96650	0.956658	0.03	0.54342	N/A	N/A
4,000	23.89610	0.957550	0.03	0.48540	N/A	N/A

TABLE III
EXPERIMENTAL RESULTS FOR $l = 60$

N	CLFG		Random Selection		Optimum Solution	
	Time (sec)	Discri.	Time (sec)	Discri.	Time (min)	Discri.
30	0.13210	0.707622	0.03	0.64321	187	0.70726
40	0.22985	0.806201	0.03	0.63240	163,840	0.80639
250	1.64565	0.924587	0.03	0.62150	$>10^6$	N/A
500	2.85260	0.942419	0.03	0.55859	N/A	N/A
1,000	4.36935	0.950709	0.03	0.63284	N/A	N/A
2,000	10.12005	0.952650	0.03	0.60746	N/A	N/A
4,000	27.90960	0.954701	0.03	0.62240	N/A	N/A

TABLE IV
EXPERIMENTAL RESULTS FOR $l = 120$

N	CLFG		Random Selection		Optimum Solution	
	Time (sec)	Discrimination	Time (sec)	Discrimination	Time (min)	Discrimination
250	2.93420	0.896015	0.03	0.59964	$>10^6$	N/A
500	4.01775	0.927922	0.03	0.66515	N/A	N/A
1,000	6.37270	0.940930	0.03	0.62918	N/A	N/A
2,000	14.80980	0.944458	0.03	0.59838	N/A	N/A
4,000	35.31320	0.947673	0.03	0.61402	N/A	N/A

results of CLFG are very close to the known optimal solutions. For each case with more than 250 candidate test items, the execution time for finding optimal solutions is more than 1 000 000 min, which is not acceptable, while CLFG can still generate test sheets with a degree of discrimination of greater than 0.9 min.

Figs. 1 and 2 depict the chart concerning the execution time of CLFG and that of finding optimum solutions. When the number of candidate test items exceeds 40, it is almost impossible to find an optimal solution, while CLFG can find near-optimal solutions in a very short time (less than 1 min).

Moreover, the statistics show that CLFG can efficiently select proper test items from an item bank containing two 500-candidate test items. Even when the number of test items in the item bank increases to 4000, the execution time of CLFG is still acceptable (about 25 to 35 s).

It is also interesting to compare the performances of CLFG and FIFG although they are used to solve different problems with different GA representations. In Tables V–VII, the experiment results of FIFG and CLFG are given to compare the execution time and discrimination degree of each generated test sheet. In each table, $\mathrm{dc} = \sum_{j=1}^{m} \max\{h_j - \sum_{i=1}^{n} r_{ij}x, 0\}$ is the sum of distance between the relevance of each concept for the currently selected test items and the corresponding lower bound, and q_num is the number of test items selected in the generated test sheet.

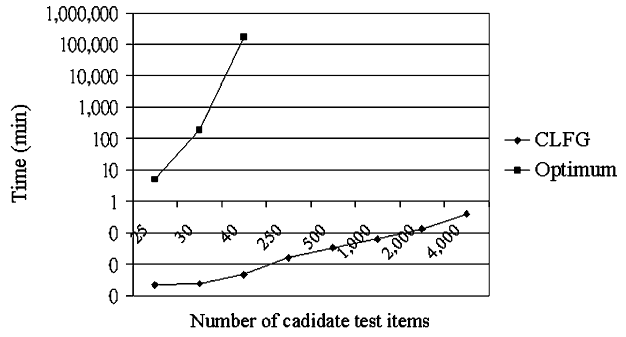

Fig. 1. Runtimes of CLFG and Optimum for $l = 30$.

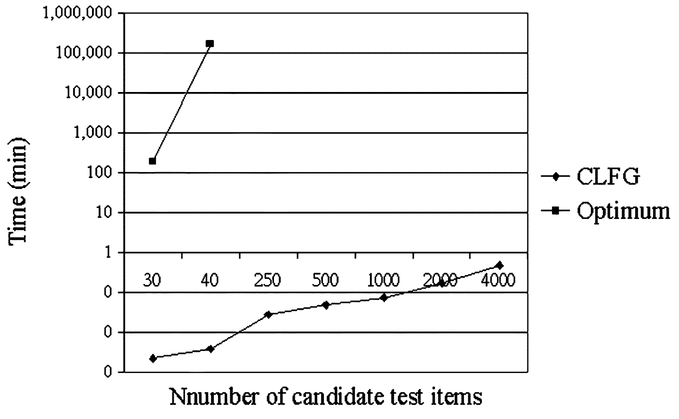

Fig. 2. Runtimes of CLFG and Optimum for $l = 60$.

TABLE V
EXPERIMENTAL RESULTS FOR $q\_\mathrm{num} = 18$

N	FIFG			CLFG	
	Time (sec)	Discrimination	*dc*	Time (sec)	Discrimination
250	0.5738	0.914176	0.29	0.9682	0.943386
500	0.9243	0.927987	0.14	1.9888	0.952377
1,000	1.2223	0.944507	0.18	3.7549	0.957359
2,000	1.4835	0.949120	0.16	7.9665	0.956658
4,000	2.0327	0.950004	0.16	23.8961	0.957550
8,000	1.9286	0.950937	0.21	42.6110	0.964375
16,000	1.3444	0.953121	0.17	145.3690	0.969500

From Tables V–VII, it can be seen that the discrimination degrees reported by FIFG and CLFG are pretty close to each other. Sometimes the discrimination degree of FIFG even transcends CLFG with less time elapsed. The impacts that the number of candidate test items can impose on the runtime of FIFG are not significant. That is, as the number of candidate test items increases, FIFG still can demonstrate an impressive performance.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed two genetic algorithms: CLFG and FIFG to cope with the test sheet-generating problems. Experimental results show that test sheets with near-optimal discrimination degrees can be obtained in a much shorter time by employing our approaches.

The two algorithms have been embedded in a CAI system, Intelligent Tutoring, Evaluation, and Diagnosis (ITED-II), to provide a more informative, flexible, and capable tool for the instructors and learners [9]. The testing subsystem of ITED-II accepts assessment requirements and reads the test items from the item bank to generate test sheets. After

TABLE VI
EXPERIMENTAL RESULTS FOR $q\_\mathrm{num} = 36$

N	FIFG			CLFG	
	Time (sec)	Discrimination	*dc*	Time(sec)	Discrimination
250	3.8155	0.907841	0.28	1.6457	0.924587
500	5.3176	0.925028	0.11	2.8526	0.942419
1,000	6.0868	0.941465	0.24	4.3694	0.950709
2,000	6.1043	0.947262	0.21	10.1201	0.952650
4,000	6.3372	0.948721	0.10	27.9096	0.954701
8,000	6.4622	0.950386	0.19	46.8070	0.959770
16,000	4.6641	0.953367	0.18	162.6740	0.964322

TABLE VII
EXPERIMENTAL RESULTS FOR $q\_\mathrm{num} = 72$

N	FIFG			CLFG	
	Time (sec)	Discrimination	*dc*	Time (sec)	Discrimination
250	13.3372	0.905967	0.16	2.9342	0.896015
500	13.1369	0.922862	0.14	4.0178	0.927922
1,000	13.8198	0.939216	0.14	6.3727	0.940930
2,000	13.8575	0.944648	0.18	14.8098	0.944458
4,000	13.5064	0.946722	0.17	35.3132	0.947673
8,000	12.8459	0.947982	0.14	56.2110	0.956102
16,000	13.3562	0.951511	0.17	246.9150	0.960396

conducting a test, the test results are transmitted to the tutoring subsystem for arranging adaptive subject materials. The commercial version of ITED II is funded by an e-learning company and is scheduled for release in November 2005. This version will incorporate the following development strategies.

1) The subject materials and item banks are designed to completely match the contents of textbooks for primary schools and junior high schools.
2) Several functions suggested by the primary school and junior high school teachers, including adaptive learning, adaptive testing, personalized learning diagnosis, and guiding, are provided.
3) A client program is delivered to the teachers and students as a low-price compact-disc read-only memory (CD-ROM) bundled to the textbooks. The trial CD-ROM contains limited functions to demonstrate part of the subject materials, test items, and learning diagnosis functions.
4) The users need to register to the server for accessing advanced functions and complete subject materials, which are charged by month, semester, or year.

Several other AI- or optimization-based technologies, such as Tabu search, Ant systems, and heuristic algorithms, could be maneuvered to develop more efficient test sheet generating approaches for very large item banks. To facilitate possible comparisons between different problem-solving approaches, the test sheet-generating programs and the database schema of the item bank are available from the corresponding author upon request.

REFERENCES

[1] C. Chou, “Constructing a computer-assisted testing and evaluation system on the world wide web-the CATES experience,” *IEEE Trans. Educ.*, vol. 43, no. 3, pp. 266–272, Aug. 2000.
[2] J. M. Feldman and J. Jones Jr., “Semiautomatic testing of student software under Unix(R),” *IEEE Trans. Educ.*, vol. 40, no. 2, pp. 158–161, May 1997.
[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freedman, 1979.
[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[5] A. V. Gonzalez and L. R. Ingraham, “Automated exercise progression in simulation-based training,” *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 6, pp. 863–874, Jun. 1994.
[6] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 7th ed. New York: McGraw-Hill, 2001.
[7] K. Hitomi, *Manufacturing Systems Engineering*, 2nd ed. London, U.K.: Taylor & Francis.
[8] S. Hopper, “Cooperative learning and computer-based instruction,” *Educ. Technol. Res. Develop.*, vol. 40, no. 3, pp. 21–38, 1996. 1992.
[9] G.-J. Hwang, “On the development of a cooperative tutoring environment on computer networks,” *IEEE Trans. Syst., Man, Cybern. Part C*, vol. 32, no. 3, pp. 272–278, Aug. 2002.
[10] ——, “A concept map model for developing intelligent tutoring systems,” *Comput. Educ.*, vol. 40, no. 3, pp. 217–235, 2003.
[11] ——, “A test sheet generating algorithm for multiple assessment requirements,” *IEEE Trans. Educ.*, vol. 46, no. 3, pp. 329–337, Aug. 2003.
[12] G. J. Hwang, T. L. Lin, and B. M. T. Lin, “An effective approach to the composition of test sheets from large item banks,” in *Proc. 5th Int. Congr. Industrial Applied Mathematics*, Sydney, Australia, July 7–11, 2003.
[13] J. T. Linderoth and M. W. P. Savelsbergh, “A computational study of search strategies for mixed integer programming,” *INFORMS J. Comput.*, vol. 11, no. 2, pp. 173–187, 1999.
[14] P. Lira, M. Bronfman, and J. Eyzaguirre, “MULTITEST II: A program for the generation, correction, and analysis of multiple choice tests,” *IEEE Trans. Educ.*, vol. 33, no. 4, pp. 320–325, Nov. 1990.
[15] J. B. Olsen, D. D. Maynes, D. Slawson, and K. Ho, “Comparison and equating of paper-administered, computer-administered, and computerized adaptive tests of achievement,” in *Proc. Annu. Meet. American Educational Research Association*, California, Apr. 16–20, 1986.
[16] H. R. Parsaei, *Genetic Algorithms and Engineering Design*. New York: Wiley, 1997.
[17] K. Rasmussen, P. Northrup, and R. Lee, “Implementing web-based instruction,” in *Web-Based Instruction*, B. H. Khan, Ed. Englewood Cliffs, NJ: Educational Technology, 1997, pp. 341–346.
[18] N. Singh, *Systems Approach to Computer-Integrated Design and Manufacturing*. New York: Wiley, 1996.
[19] H. Wainer, *Computerized Adaptive Testing: A Primer*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1990.