A RANDOMIZED ROUNDING APPROACH FOR OPTIMIZATION OF TEST SHEET COMPOSING AND EXPOSURE RATE CONTROL IN COMPUTER-ASSISTED TESTING

Chu-Fu Wang
Department of Computer Science
National Pingtung University of Education
cfwang@mail.npue.edu.tw

Chih-Lung Lin (Corresponding author)
Department of Computer Science
National Pingtung University of Education
clin@mail.npue.edu.tw

Jien-Han Deng
Department of Computer Science
National Pingtung University of Education
leslie1101kimo@yahoo.com.tw

ABSTRACT
Testing is an important stage of teaching as it can assist teachers in auditing students' learning results. A good test is able to accurately reflect the capability of a learner. Nowadays, Computer-Assisted Testing (CAT) is greatly improving traditional testing, since computers can automatically and quickly compose a proper test sheet to meet user requirements. For example, the users can specify the number of test items to be selected in the test sheet, the average difficulty with respect to the test sheet can be restricted within a lower bound and an upper bound, and the generated test sheet is able to cover each basic concept in the scope of the testing subject. In order to design an algorithm for test sheet composing in a CAT system to meet the above objectives, we model it as a 0-1 integer optimization problem and then transform it to a dominating set selection problem of graph theory. A Multi-stage Test Sheet Composing Algorithm (MTSCA) is proposed to give a near optimal solution to this optimization problem. Due to the fact that exposure rate control is also an important issue in test sheet composing, our proposed MTSCA adopts a randomized rounding technique to reduce the average item exposure rate. The simulation results show that the performance of the MTSCA can not only achieve high average discrimination in the generated test sheet, but the item exposure rate can be properly controlled as well.
Keywords: Computer-assisted testing, test sheet composing, randomized rounding, dominating set

INTRODUCTION

As information technology evolves, it gives fresh impetus to the educational field. For instance, the development of a computer-assisted testing (CAT) system can replace paper-and-pencil tests in many situations and provide the benefits of time and cost saving. A proper test can help teachers diagnose students' learning problems in a school setting. In addition, when applied to a company setting, a well-designed test will assist supervisors in better understanding the performance of their new employees'. Due to the broad range of applications for the CAT system, it has received much attention from researchers. Several new theories and tools for testing have been proposed in recent decades to enhance the capabilities of the system. For example, the Item Response Theory (IRT) can accurately assess the examinee's trait level using fewer test items and less time than the traditional approach. On the other hand, the test sheet composing techniques have also been enhanced in recent years.

Traditionally, a test sheet is composed by selecting the test items from the item bank in a random fashion. Although this method can be used to compose a test sheet quickly, the quality of randomly generated test sheets is generally poor. Moreover, if the users issue demands on the generated results, this method will fail to meet their requirements. These requirements may include specification of the number of test items, the range of the average difficulty of a test sheet having to fall within given upper and lower bounds, the test sheet having to cover each basic concept in the scope of the testing subject, and the average discrimination of the generated test sheet needing to be as high as possible. However, due to the fact that there are usually thousands of test items in an item bank, to automatically compose the best (such as the greatest average discrimination) test sheet among every possible test sheet combination that meets the user's requirements is usually time consuming. Thus the algorithms designed to meet the objectives discussed above are not easy to design.

As artificial intelligence (AI) techniques and computer algorithms are being brought into CAT systems, the intelligence of the developed systems has been dramatically enhanced. Now, a CAT system can not only

automatically organize a test sheet to satisfy users' requirements efficiently, but can also help assess the examinee's trait level more accurately than traditional methods (Hwang et al., 2006; Yin et al., 2006; Hwang et al., 2005). Among the related research, Hwang et al. (Hwang et al., 2006) and Yin et al. (Yin et al., 2006) adopted artificial intelligence techniques to optimize test sheet selection according to different criteria, which can efficiently compose a near-optimal test sheet from large item banks. These research results provide a good approach for test sheet composition, though one of the key issues, the test items' exposure rate control, remains unconsidered. The main objective for developing computer algorithms to optimize test sheet composition is to find a solution (a test sheet) as close to the optimum as possible. However, this has the drawback that some test items will be selected into the test sheet more frequently than others if the same composition demand is invoked several times. In the worst case, the algorithms may output almost the same test sheet each time if the specified user requirements are the same, resulting in the average exposure rate of an item bank being too high. Therefore exposure rate control (Sympson & Hetter, 1985; Barrada et al., 2007; Chang & Ying, 1999; Chang et al., 2001) is another important issue worth considering when developing test sheet composing algorithms.

In this study, we consider a test sheet composing optimization problem and take exposure rate control into consideration. In the considered problem, we aim to compose a test sheet with the greatest average discrimination among every possible test sheet that satisfies the following requirements. The first requirement is that the number of test items is equal to a given specified value k. Secondly, the average difficulty of the test sheet must be greater than a given lower bound (p_l) and also less than a given upper bound (p_u). The third requirement is to force the generated test sheet to cover each basic concept in the range of the testing subject. First, we give a mathematical formulation of the above test sheet composing optimization problem. Then the considered problem is transformed to a *dominating set problem* (Haynes et al., 1998), of the graph theory, which is an older branch of discrete mathematics and has plenty of excellent results. In addition, a randomized rounding technique is adopted in our algorithm design to limit the average exposure rate of the item bank. A Multi-stage Test Sheet Composing Algorithm (MTSCA) with randomized rounding is proposed to give a near-optimal test sheet solution to the optimization problem.

The organization of this paper is as follows. First, we describe some relevant research about the existing test sheet composing algorithms and the exposure rate control techniques developed so far. Second, the formal problem description and the problem transformation are specified. Then the proposed algorithm MTSCA and the simulation results are shown. The concluding remarks are stated in the last section.

Relevant research

Recent published research work related to test sheet composing is twofold. The first category of research aims to design efficient algorithms to generate proper test sheets to meet the user's specified requirements, while the second category aims to design techniques to limit the item exposure rate of an item bank. However, these two areas of research fail to consider the above two issues jointly, though both are important for test sheet composition. In the following, we describe each of the two categories of relevant research in detail.

In a CAT system, composing a test sheet using a comprehensive computer algorithm is a more proper approach than manually or randomly selecting test items from an item bank. In (Hwang et al., 2006), they used a tabu-based algorithm to generate test sheets for multiple assessment criteria. Their work aimed at optimizing the average discrimination of the generated test sheets. The constraints of their considered model include the following two criteria: (1) the selected test items must have a total expected relevance of each concept to be learned which is greater than a given lower bound; (2) the selected test items must also have a total expected assessment time for answering the selected items which is bounded by a specified range of assessment times. A tabu search is one of the efficient heuristic algorithms for finding near-optimal solutions for optimization problems, such as the traveling salesman problem, the network planning problem, the job-shop scheduling problem, etc. Generally, these problems are NP-hard, and no polynomial time algorithms exist to solve them to optimally. The tabu search consists of the following features: configuration, a move function, neighborhood definition, tabu restriction, aspiration level, and stopping criteria. It starts with a randomly generated configuration (that is, a test sheet in the considered test sheet composing problem). And then it will iterativelymake the best move from the current configuration to a new configuration whose objective function value is the greatest among the current configuration’s neighbor set according to the designed move function.

	difficulty	discrimination
item 1	0.6	0.7
item 2	0.3	0.2
item 3	0.7	0.7
item 4	0.6	0.7
item 5	0.4	0.5
item 6	0.75	0.82
item 7	0.5	0.65
item 8	0.2	0.3
item 9	0.8	0.8
item 10	0.1	0.2

(K=2)

	difficulty	discrimination
item 2	0.3	0.2
item 10	0.1	0.2
item 8	0.2	0.3
item 5	0.4	0.5
item 7	0.5	0.65
item 1	0.6	0.7
item 3	0.7	0.7
item 4	0.6	0.7
item 9	0.8	0.8
item 6	0.75	0.82

Figure 1. A numerical example illustrating the a-STR method (N=10, K=2)

During the iterative configuration refinement movement, in order not to revisit recently visited configurations, the tabu search design includes a feature called the tabu restriction, which records recently visited configurations into a tabu list when each move is taken. The method also uses an aspiration level to relax the tabu restriction by accepting a refinement move that violates the tabu restriction in order to attain a better solution. The authors designed a tabu-search method to solve the test sheet composing optimization problem taking into consideration the above criteria.

In (Yin et al., 2006), they used a newly developed heuristic approach called *particle swarm* (Kennedy & Eberhart, 1995) to compose near-optimal serial k test sheets. Particle swarm is also an efficient heuristic algorithm to deal with the NP-hard problem by iteratively refining an initial solution to achieve a near-optimal solution. It is a biologically inspired algorithm which models the flocking behavior of bird. Initially, the method generates a set of solutions (test sheets) called swarms. In each iteration step, the algorithm will update the set of swarms according to the flying velocities and the fitness function until a stopping condition is met. For more details of the procedure, one can refer to the study of (Kennedy & Eberhart, 1995). Yin et al. proposed test sheets composing optimization problem called the STSC (Serial Test Sheet Composition) problem. This problem aims to minimize the difference between the average difficulty of each test sheet and a user specified difficulty value. The constraints are the same as for the research work in (Hwang et al., 2006) (i.e., the specified range of assessment time and the relevant concept constraint), plus a new constraint to limit the number of test items in common between any two generated test sheets being no more than a user specified value.

The above research used comprehensive computer algorithms (such as tabu search and particle swarm) to develop test sheet composition methods. Their simulation results showed that these developed algorithms can efficiently generate good-quality near-optimal test sheets under different considered problem models. For more details, one may refer to these two research works (Hwang et al., 2006; Yin et al., 2006). There are many other modern heuristic techniques (Bertsimas et al., 1999; Kennedy & Eberhart, 1995; Michalewicz & Fogel, 2002; Aart & Lenstra, 1997) which can be applied to solve the test sheet composing optimization problem, including, the genetic algorithm, simulated annealing, the ant colony optimization algorithm, neural networks, fuzzy systems, etc. The readers may refer to (Linden, 2005) for a discussion of more modern heuristic techniques which are being applied in the test sheet design field. As discussed earlier, however, these do not take exposure rate control into consideration; thus, the average exposure rate of the item bank may be too high, which could consequently endanger the accuracy of the test items used in the future.

In the following we describe some existing exposure rate control techniques for testing. Due to the fact that exposure rate control is an important security issue for testing theory, many researchers have developed methods to prevent a test item from being selected too frequently (Sympson & Hetter, 1985; Barrada et al., 2007; Chang & Ying, 1999; Chang et al., 2001; Revuelta & Ponsoda, 1998; Wang & Chang, 2011) to prevent a test item from being selected too frequently. The simplest approach is to associate each test item with an exposure value. Each time a test item is selected for testing, its exposure value will be increased. Thus the current exposure value with respect to each test item can be an important parameter for considering the test items being selected into the test sheet for examinees' assessment. For more comprehensive methods,

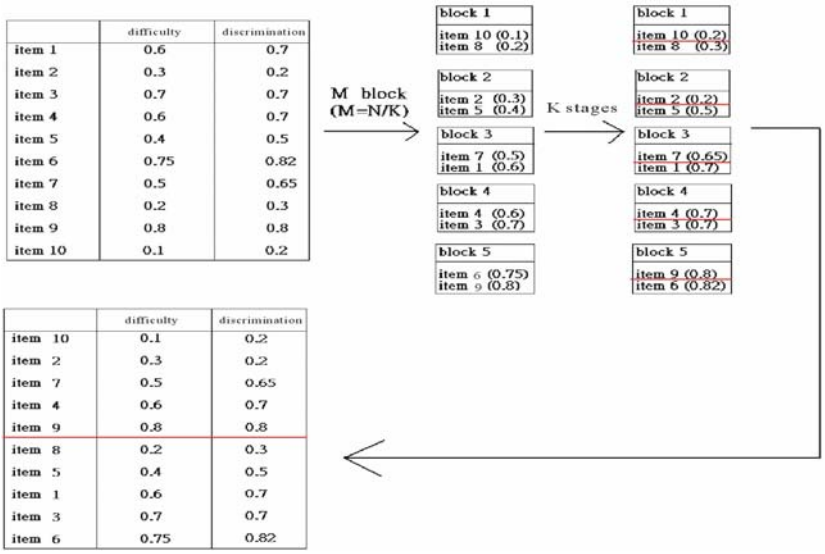


Figure 2. A numerical example illustrating the ba-STR method (N=10, K=2)

Chang and Yin (Chang & Ying, 1999) proposed a multi-staged approach called the *a-stratified selection method* (a-STR) for a Computer Adaptive System, where a denotes the value of discrimination. Let the test length be L; that is, the Computer Adaptive System will select L test items from the item bank for serial testing. The testing chooses the test items with higher discrimination value and a difficulty value (or the examinee's estimate trait $\hat{\theta}$) as close as possible to the examinee's true score θ. In order to meet this objective, the a-STR method firstly partitions the test items into K levels (stages) according to the discrimination value. Then the system will select n_i test items according to the user's requirement from the i-th ($1 \le i \le K$) level, respectively, such that $n_1 + n_2 + ... + n_K = L$.

For example, as shown in Figure 1, there are 10 (N=10) test items in the item bank, and the difficulty and discrimination values with respect to each test item are shown in the left part of Figure 1. Assume the number of stages is 2 (K=2) and the test length is 4 (L=4). Then firstly, the a-STR method will partition the test items into 2 groups (the upper and lower groups) according to the discrimination value in non-decreasing order (see the right part of Figure 1). Note that the lower (higher) group is the test item group with higher (lower) discrimination values, respectively. Then n_i ($1 \le i \le K$) test items in group i will be selected, for which their difficulty values are close to the specified difficulty. For instance, if $n_1 = 3, n_2 = 1$ and the specified difficulty is 0.4, then Item_5, Item_7, and Item_2 are selected from the first group, and test Item_1 is selected from the second group. Though the a-STR method can control the exposure rate, there still remain many drawbacks with such an approach.

In (Chang et al., 2001), Chang et al. refined the a-STR method and proposed an a-Stratified with b blocking method (ba-STR) to improve the quality of item exposure rate control and to reduce the mean squared errors. In general, the discrimination value and the difficulty value of a test item are positively correlated. Based on this result, the ba-STR method firstly partitions the item bank according to the difficulty value instead of the discrimination value, which the a-STR method does. The detailed method can be illustrated using the numerical example in Figure 2. Assume that there are 10 (N=10) test items in an item bank, which is shown in the upper-left part of Figure 2. We also assume that ba-STR uses 2 (K=2) stages and the testing length is equal to 4 (L=4). Firstly, sort the test items according to the difficulty values in non-decreasing order and then partition these test items into M blocks, where M=N/K=5. In each block, the ba-STR again sorts the test items according to the discrimination value in non-decreasing order, and forms each block into 2 (K=2) subgroups, as shown in the far-right part of Figure 2. Finally, collect the test items in the i-th ($1 \le i \le K$) subgroup of each block to form a group (the i-th group), as shown in the bottom-left part of Figure 2. The rest of the operation to select L test items for adaptive testing is the same as the method that is described in a-STR.

The other approach to suppressing overexposed items is the progressive strategy (Revuelta & Ponsoda, 1998; Wang & Chang, 2011). The primary idea of this approach is to add a stochastic component into the item selection method to avoid frequently choosing the highest information items in an IRT-based CAT. Recently,

Wang and Chang (Wang & Chang, 2011) proposed two item selection methods, the restrictive progressive method and the restrictive threshold method, to include additional stochastic components in the item selection method so as to increase the usage of underexposed items in the item bank. In the restrictive threshold method, a threshold value (δ) is given to extend the candidate set of selection items instead of the highest information item only. A larger (lower) value of δ will result in a larger (smaller) size of the candidate item set. The candidate item set (S_c) is the collection of items such that the respective information falls within the information interval $[\max_{\inf o} - \delta, \max_{\inf o}]$, where $\max_{\inf o}$ denotes the maximum information value among all items. That is, set S_c collects the items with an information value close to the maximum such that the information difference between them is less than or equal to δ. Then the method will randomly select one of the items in the candidate item set S_c to be the chosen item, and put it into the test sheet. In this way, the exposure rates of the items may tend to be uniform. For the restrictive progressive method and a detailed description of the method, one can refer to the literature (Wang & Chang, 2011). The above research works for controlling the exposure rate fit into Computer Adaptive Testing, but most of them are not suitable for single test sheet composition. Besides, the aims of these approaches are not to optimize some parameters; thus, they may not be directly applied to our considered problem model.

PROBLEM DESCRIPTION AND FORMULATION

In this paper, a test sheet optimization problem is considered. Assume a user specifies the following parameters: (1) the number of test items in a test sheet is m; (2) the average difficulty of the test sheet is greater than alower bound p_l and is less than an upper bound p_u; (3) the number of basic concepts to be learned in the testing scope is k. Then our considered problem aims to optimize the average discrimination of the test sheet, such that the number of test items in the test sheet is equal to m, the average difficulty of the test sheet is bounded within the interval (p_l, p_u), and all the k basic concepts have to be covered by the generated test sheet. In the following, we firstly describe the notations and parameters that are discussed throughout this paper. Then, a special structure called the *Item Relationship Graph* (IRG) is introduced for transforming the considered problem into a graph theory optimization problem. Some preliminary results of IRG are also given. Finally, a mathematical formulation for our considered problem is provided.

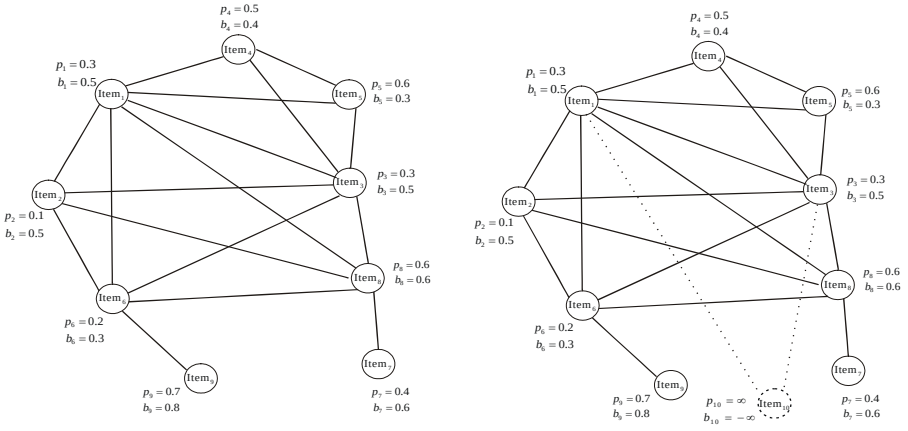
The characteristic vector of a test item

Let Item_1, Item_2, …, Item_n be the test items in the scope of testing and the number of these test items be n. Let $e_1, e_2, \ldots, e_k$ be the basic concepts to be learned in the testing scope. Set C, called the *basic concept set* is the collection of these k basic concepts; that is, $C = \{e_1, e_2, ..., e_k\}$. For each test item Item_i $(1 \le i \le n)$, there are three associated parameters namely, the discrimination value b_i, the difficulty value p_i, and $C_i = \{e_{i1}, e_{i2}, ..., e_{it}\} \subseteq C$, called the *basic concept subset* of C to denote the basic concepts being covered by test item Item_i. That is, if the test item Item_i is placed in the test sheet, then the t basic concepts $e_{i1}, e_{i2}, \ldots,$ and e_{it} can be used to assess whether or not the examinees have learned these concepts. Thus, a three-tuple vector (b_i, p_i, C_i) called the *test item characteristic vector* is associated with each test item Item_i $(1 \le i \le n)$, and is given in advance. Note that the discrimination value and the difficulty value can be obtained by using some existing item analysis techniques.

The Item Relationship Graph (IRG)

An Item Relationship Graph (IRG) $G = (V, E)$ with respect to the testing is defined as follows. The vertex set V is the collection of all test items within the testing scope, thus V={Item_1, Item_2, …, Item_n}. For any two test items Item_i and Item_j in V, let the corresponding basic concept subsets be $C_i = \{e_{i1}, e_{i2}, ..., e_{it1}\}$ and $C_j = \{e_{j1}, e_{j2}, ..., e_{jt2}\}$, respectively. If the vertices Item_i and Item_j in V are said to have an edge between them, if $C_i \cap C_j \neq \phi$, this means the two test items have basic concepts in common. Thus the edge set E is defined $E = \{(\text{Item}_i, \text{Item}_j) \mid \text{if } C_i \cap C_j \neq \phi \text{ and } i \neq j, \forall \text{Item}_i, \text{Item}_j \in V\}$. As shown in Figure 3(a), there are 9 test items in the testing scope. Assume that the basic concept set in the testing scope is $C = \{e_1, e_2, e_3, e_4, e_5\}$. The discrimination value and the difficulty value with respect to each test item are also shown in the figure. Suppose the basic concept subset C_i with respect to Item_i $(1 \le i \le 9)$ is as follows: $C_1 = \{e_1, e_2, e_3\}$, $C_2 = \{e_3\}$, $C_3 = \{e_1, e_2, e_3\}$, $C_4 = \{e_1\}$, $C_5 = \{e_1\}$, $C_6 = \{e_3, e_5\}$, $C_7 = \{e_4\}$, $C_8 = \{e_3, e_4\}$, and $C_9 = \{e_5\}$. Since $C_4 \cap C_1 \neq \phi$, $C_4 \cap C_3 \neq \phi$, and $C_4 \cap C_5 \neq \phi$, then $(\text{Item}_4, \text{Item}_1) \in E$, $(\text{Item}_4, \text{Item}_3) \in E$, and $(\text{Item}_4, \text{Item}_5) \in E$.

Now we define an augmented graph of IRG, called the A-IRG ($G_A = (V_A, E_A)$) as follows. A test item Item$_i$, whose basic concept subset C_i only contains a single basic concept, is called an *identity test item*. That is, Item$_i$ is an identity test item, if $|C_i| = 1$. For the example in Figure 3, the test items Item$_2$, Item$_4$, Item$_5$, Item$_7$,



(a) the IRG (b) the A - IRG

Figure 3. An example of an IRG

and Item$_9$ are all identity test items. In the case of a basic concept $e_t \in C$ not being contained by any identity test item Item$_l$ such that $C_l = \{e_t\}$, then e_t is called a *missing basic concept*. Let C_M be the collection of all missing basic concepts. Then, for each missing basic concept $e_t \in C_M$, a pseudo vertex Item$_t$ is added to V_A, and the characteristics vector with respect to Item$_t$ is let to be $(-\infty, \infty, \{e_t\})$. The edge set E_A of the A-IRG is then obtained by the same definition of the edge set in the IRG. For the example in Figure 3, since C_2, C_4, C_5, C_7, and C_9 are all identity basic concept subsets and $C_2 \cup C_4 \cup C_5 \cup C_7 \cup C_9 = \{e_1, e_3, e_4, e_5\}$, then e_2 is a missing basic concept. We then add a pseudo vertex Item$_{10}$ in G_A and let the test item characteristic vector with respect to Item$_{10}$ be $(b_{10}, p_{10}, C_{10}) = (-\infty, \infty, \{e_2\})$. The resulting A-IRG with respect to the IRG in Figure 3(a) is shown in Figure 3(b). In the following we give some preliminary results related to the augmented IRG, and describe the relationship between the basic concepts covering constraint and the *dominating set* of graph theory. A vertex subset $D \subseteq V_A$ of the A-IRG is called a dominating set, if for each vertex Item$_i \in V_A$, either Item$_i \in D$ or a vertex Item$_t \in D$ exists, such that (Item$_i$, Item$_t$) $\in E_A$ (or simply, Item$_i \in N(D)$, where $N(D)$ denotes the collection of all neighboring vertices of D). For the example in Figure 3(b), subsets {Item$_1$, Item$_6$, Item$_8$}and {Item$_3$, Item$_8$, Item$_9$} are two dominating sets of the A-IRG G_A. The following theorem gives the relationship between the basic concept covering constraint of our considered problem with the dominating set in G_A.

Theorem 1. For any dominating set in an A-IRG, the corresponding test sheet meets the basic concept covering constraint and vice versa.

Proof.

Let D be a dominating set in the A-IRG and C be the basic concept set in the testing. For each basic concept $e_t \in C$, there exists an identity test item Item$_t \in V_A$ such that $C_t = \{e_t\}$. According to the definition of the A-IRG and the dominating set, the Item$_t$ is dominated by D. Thus, $e_t \in \bigcup_{\text{Item}_i \in D} C_l, \forall e_t \in C$. That is, the set D meets the basic concept covering constraint.

Conversely, we will show that given any vertex subset $S \subseteq V_A$ that satisfies the basic concept covering constraint, then S must be a dominating set in G_A. Suppose not, then there exists a vertex $v \in V_A$, but $v \notin S$ and $v \notin N(S)$. Suppose that $C_v = \{e_{i1}, e_{i2}, ..., e_{il}\}$. Since $e_{it} \in C_v$, $(1 \le t \le l)$, according to the definition of edge

in the A-IRG, we have $e_{it} \notin \bigcup_{\text{Item}_t \in S} C_t$. Since S does not cover any one of the l basic concepts e_{it} $(1 \le t \le l)$, then it does not meet the basic concept covering constraint, which violates the assumption. Thus set S is a dominating set. Q.E.D.

Note that the above theorem gives the fact that the basic concepts covering the constraints of our problem model can be completely eliminated by finding a dominating set in the A-IRG instead. Thus we have transformed the test sheet composing optimization problem into a dominating set finding optimization problem in graph theory. Due to graph theory being a field with fertile theoretical results, this transformation builds a bridge from the test sheet composing problem to graph theory.

Mathematical formulation

Let $V = \{\text{Item}_1, \text{Item}_2, \ldots, \text{Item}_n\}$ be the collection of test items in the testing scope. And let $C = \{e_1, e_2, \ldots, e_k\}$ be the basic concept set. For each test item Item_i $(1 \le i \le n)$, a test item characteristic vector (b_i, p_i, C_i) is associated with it, where b_i denotes the discrimination value, p_i denotes the difficulty value, and C_i denotes the basic concept subset. Let graph $G_A = (V_A, E_A)$ be the augmented item relationship graph with respect to the above test items. Suppose a user issues a demand for test sheet composition and specifies that the test sheet size be m and the upper (lower) bound of the average difficulty of the test sheet be p_u (p_l). The considered problem aims to determine a vertex subset $S^* \subseteq V_A$ and its average discrimination value maximizes every possible vertex subset that meets the following three constraints.

1. The set S^* satisfies the difficulty constraint; that is, $p_l \le \dfrac{\sum_{\text{Item}_i \in S^*} p_i}{|S^*|} \le p_u$.

2. $|S^*| = k$.

3. S^* is a dominating set.

As discussed above, we give a formal mathematical formulation of our considered problem, called the Test Sheet Composing Optimization Problem (TSCOP) as follows.

$$\text{Max.} \frac{\sum_{1 \le i \le n} x_i \cdot b_i}{\sum_{1 \le i \le n} x_i} \qquad (1)$$

s.t.

$$p_l \le \frac{\sum_{1 \le i \le n} x_i \cdot p_i}{\sum_{1 \le i \le n} x_i} \le p_u \qquad (2)$$

$$\sum_{1 \le i \le n} x_i = m \qquad (3)$$

$$\sum_{\{j | \text{Item}_j \in N(\text{Item}_i)\}} x_j > 0, \forall \text{Item}_i \in V_A \qquad (4)$$

$$x_i \in \{0,1\}, \forall \text{Item}_i \in V_A \qquad (5)$$

In the above mathematical programming model, each test item Item_i $(1 \le i \le n)$ corresponds to a 0-1 variable x_i. If x_i is set to be 1, this means Item_i is selected into the test sheet; otherwise $x_i = 0$ (see Equation (5)). Equation (2) guarantees the test sheet will meet the difficulty constraint. Equation (3) gives the test sheet size constraint. Finally, Equation (4) ensures that the neighboring set with respect to each test item Item_i in A-IRG must have at least one test item selected into the test sheet, which guarantees that the resulting test sheet will meet the dominating set constraint.

Exposure rate control

Assume a testing system has generated t test sheets $S_1, S_2, \ldots,$ and S_t for testing so far. Suppose the test item Item_i appears x times in the t test sheets. Then the exposure rate $Xposure_i$ with respect to test item Item_i is defined as, $Xposure_i = x/t$, and the average exposure rate $Avg\_Xposure(t)$ with respect to the t test sheets is defined as follows:

$$Avg\_Xposure(t) = \frac{\sum_{\text{Item}_i \in S_1 \cup S_2 \cup ... \cup S_t} Xposure_i}{|S_1 \cup S_2 \cup ... \cup S_t|} \quad (6)$$

In the next section, we describe the proposed algorithm, which can generate a near-optimal test sheet for our considered problem for which the average exposure rate will not be too high compared to other conventional algorithms.

The proposed algorithm

The main objective of the TSCOP is to optimize the average discrimination of the generated test sheet; however, the other objective is to decrease the average exposure rate of the testing. The above two objectives conflict with each other, since the previous one tends to always select the "better" test items into the test sheets, but the latter one tends to balance the chance of each test item being selected into the test sheets in order to control the average exposure rate such that it is below a certain level. In this paper, we propose a Multi-stage Test Sheet Composing Algorithm (MTSCA) using the randomized rounding technique in our algorithm design. The main approach of the MTSCA is that, first, we relax the integer programming optimization problem TSCOP to be a linear programming optimization problem, which can be solved to optimize the result by using some well-known polynomial time algorithms (for example, the Ellipsoid method or Interior-point algorithm). Then, we use the randomized rounding technique on the above optimum linear solution to round it to be an integer solution with random fashion. Due to the resulting integer solution possibly not being feasible, a feasibility modification process is also proposed to modify the current solution so that it can be feasible. Then the corresponding test sheet with the resulting solution will be the generated test sheet of our proposed MTSCA. Since the above MTSCA adopts randomized rounding, the generated test sheet will be different each time the algorithm is invoked. A detailed algorithm description of the MTSCA is provided as follows.

The Multi-stage Test Sheet Composing Algorithm (MTSCA)

Let set $V = \{\text{Item}_1, \text{Item}_2, \ldots, \text{Item}_n\}$ and $C = \{e_1, e_2, ..., e_k\}$ be the test item set and the basic concept set, respectively. Let (b_i, p_i, C_i) be the test item characteristic vector with respect to test item $\text{Item}_i \in V$. Assume the user specifies the test sheet size to be k and the upper (lower) bound of the average difficulty of the generated test sheet to be p_u (p_l). Firstly, we will construct the A-IRG $G_A = (V_A, E_A)$ with respect to the above test items. The proposed MTSCA consists of the following three stages.

Stage 1. Solving the linear programming problem model

First, we relax the integer decision variables $x_i, \forall \text{Item}_i \in V_A$ of the integer programming problem model TSCOP to be real decision variables. The resulting linear programming model is as follows:

$$\text{Max.} \frac{\sum_{1 \le i \le n} x_i \cdot b_i}{\sum_{1 \le i \le n} x_i} \quad (7)$$

s.t.

$$p_l \le \frac{\sum_{1 \le i \le n} x_i \cdot p_i}{\sum_{1 \le i \le n} x_i} \le p_u \quad (8)$$

$$\sum_{1 \le i \le n} x_i = m \quad (9)$$

$$\sum_{\{j | \text{Item}_j \in N(\text{Item}_i)\}} x_j > 0, \forall \text{Item}_i \in V_A \quad (10)$$

$$x_i \in R, \forall \text{Item}_i \in V_A \quad (11)$$

Generally, the above model can be easily solved using some existing polynomial time algorithms, such as the well known Ellipsoid method or Interior-point algorithm. For detailed solution steps, one can refer to some Combinatorial Optimization works (e.g., Aarts & Lenstra, 1997), so we omit the detail here. Let the resulting optimum solution be $(x_1^*, x_2^*, ..., x_n^*)$, where $x_i^* \in R, 1 \le i \le n$.

Stage 2. Randomized rounding

This stage rounds the above real number solution $(x_1^*, x_2^*, ..., x_n^*)$ to be 0-1 integer solution $(\hat{x}_1^*, \hat{x}_2^*, ..., \hat{x}_n^*)$ according to the following rules.

For each solution value $x_i^*, 1 \le i \le n$,

Case 1 if $x_i^* \ge 1$, then $\hat{x}_i^* = 1$;

Case 2 if $x_i^* \le 0$, then $\hat{x}_i^* = 0$;

Case 3 if $0 < x_i^* < 1$, then we randomly choose a real number t from range [0,1]. In case of $t \le x_i^*$, then $\hat{x}_i^* = 1$; otherwise, $\hat{x}_i^* = 0$. For example, assume $x_i^* = 0.7$ and the random number t=0.5. Since $t \le x_i^*$, we set the resulting value $\hat{x}_i^*$ to be 1.

Perform the above randomized rounding process on each of the solution values $x_i^*, 1 \le i \le n$, and an integer solution $(\hat{x}_1^*, \hat{x}_2^*, ..., \hat{x}_n^*)$ will then be obtained. Due to the above solution perhaps not being feasible with respect to the TSCOP, the following stage will modify the resulting integer solution $(\hat{x}_1^*, \hat{x}_2^*, ..., \hat{x}_n^*)$ so that it becomes feasible solution.

Stage 3. Performing the feasibility modification process

In the case of the obtained solution $(\hat{x}_1^*, \hat{x}_2^*, ..., \hat{x}_n^*)$ after performing Stages 1 and 2 being feasible for the TSCOP, then the execution of Stage 3 is omitted. On the other hand, if the solution is infeasible, then the three steps of Stage 3 have to be performed in order to obtain a feasible solution. The three steps include, the *difficulty feasibility modification*, the *dominating feasibility modification*, and the *test sheet size feasibility modification*, and are stated as follows.

1. The difficulty feasibility modification:

Let set S be the test item set in graph G_A with respect to the current solution $(\hat{x}_1^*, \hat{x}_2^*, ..., \hat{x}_n^*)$. That is, $S = \{\text{Item}_i \mid \text{if } \hat{x}_i^* = 1\}$. If S meets the difficulty constraint, then skip this step. Otherwise, if the average difficulty of S is less than the specified lower bound p_l, then we let set U={Item_{i1}, Item_{i2},…, Item_{il}} be the collection of test items not currently selected in S, and its difficulty value is greater than the lower bound (that is, if $\text{Item}_{it} \in U$, then $\text{Item}_{it} \in V_A - S$ and $p_{i_t} > p_l$). Choose the test item Item_{it} with the highest discrimination value in set U, and then put it into S. Then update sets S and U by $S \cup \{\text{Item}_{it}\}$ and U=U-{Item_{it}}. Repeat the above process until the average discrimination value of S is greater than the lower bound p_l. On the other hand, if the average discrimination of the current solution after performing Stages 1 and 2 is greater than the specified upper bound p_u, then the operations are similar to the above description except that set U={Item_{i1}, Item_{i2},…, Item_{il}} is set to be the collection of test items not currently selected in S, and its difficulty value is less than the upper bound.

2. The dominating feasibility modification:

If set S is not a dominating set in graph G_A, then the following operations will be performed. Let set $U = V_A - S$ and let the test item Item_{it}^* be the highest discrimination value in set U. If adding Item_{it}^* will not violate the difficulty constraint and can enhance the basic concept coverage, then $S = S \cup \{\text{Item}_{it}^*\}$ and $U = U - \{\text{Item}_{it}^*\}$. Otherwise, we just drop it from set U; that is, $U = U - \{\text{Item}_{it}^*\}$. Repeat the above process until set S becomes a dominating set.

3. The test sheet size feasibility modification:

If the current test sheet size is greater than the user specified number k; that is $|S| > k$, then some test items have to be removed. Firstly, we sort the test items in S according to the discrimination value in nondecreasing order, and let Item_{i1}, Item_{i2},…, $\text{Item}_{i|S|}$ be the resulting order. Then, starting from the first test item Item_{i1}, check whether or not removing this item from S will violate the difficulty constraint or the dominating constraint. In case of either one of the constraints being violated, then skip this item and do nothing; otherwise, remove Item_{i1} from S and let the resulting set S be S-{Item_{i1}}. Continue the above process sequentially on the test item list until $|S| = k$, and then output the resulting solution set S.

On the other hand, in case of $|S| < k$, then some test items which have not been selected in set S have to be added to S. Similar to the above case ($|S| > k$), the candidate test item checking list is no longer S but $U = V_A - S$. Then sort the test items in U according to the discrimination value in nonincreasing order, and

decide whether or not to and the current item in the list to S until $|S| = k$.

```
Procedure Feasible-modification-process;
{
/* Step 1. The difficulty feasibility modification */
  let S be the test item set in G_A with respect to solution (x̂*_1, x̂*_2,...x̂*_n);
  if (S does not meet the difficulty constraint) then {
    repeat
      if (the average difficulty value of S < p_l ) then
         U = {Item_it | Item_it ∈ V_A − S and p_it > p_l};
      else U = {Item_it | Item_it ∈ V_A − S and p_it < p_l};
      if (U = φ ) then randomly remove some test items from S
              until the resulting U becomes nonempty;
      let Item*_it be the test item with the greatest discrimination value in U;
       S = S ∪ {Item*_it};
       U = U ∪ {Item*_it};
    until (S meets the difficulty constraint);
     }
/* Step 2. The dominating feasibility modification */
  repeat
    U = V_A − S ;
    let Item*_it be the test item with the greatest discrimination value in U;
    if (add Item*_it into S will not violate the difficulty constraint and can enhance the basic concept
            coverage) then S = S ∪ {Item*_it};
    U = U − {Item*_it};
  until (S becomes a dominating set);
/* Step 3. The test sheet size feasibility modification */
  if (|S| > k ) then {
     sorting the test items in S according to the discrimination value in nondecreasing order, and let the
          resulting order list be L = (Item_i1, Item_i2,..., Item_i|S|);
     repeat
       remove the first item Item_j from L;
       if ( S − {Item_j} does not violate the difficulty and dominating constraints) then S = S − {Item_j};
     until (|S| = k );
  }
  else if (|S| < k ) then {
     sorting the test items in V_A − S according to the discrimination value in nonincreasing order, and let
          the resulting order list be L = (Item_i1, Item_i2,..., Item_i|V_A−S|);
     repeat
       remove the first item Item_j from L;
       if ( S ∪ {Item_j} does not violate the difficulty and dominating constraints) then S = S − {Item_j};
     until (|S| = k );
  }
 return(S);
```

Figure 4. The complete procedure for the feasibility modification process

```
Algorithm Multi-staged Test Sheet Composing;
{
 construct the A-IRG G_A = (V_A, E_A);
 /* Stage 1. Solving the linear programming problem model */
    Solve the linear programming model and let the resulting solution be (x_1^*, x_2^*, ..., x_n^*);
  /* Stage 2. Randomized rounding */
    for each solution value x_i^* in (x_1^*, x_2^*, ..., x_n^*) do {
     if ( x_i^* ≥ 1) then x̂_i^* = 1;
     else if ( x_i^* ≤ 0 ) then x̂_i^* = 0;
     else { choose a random number t from [0,1];
        if ( t ≤ x_i^* ) then x̂_i^* = 1; else x̂_i^* = 0;
        }
    }
  /* Stage 3. Performing the Feasibility-modification-process */
    S=Feasible-modification-process();
 output(S);
}
```

Figure 5. The complete procedure for the Multi-stage Test Sheet Composing Algorithm (MTSCA)

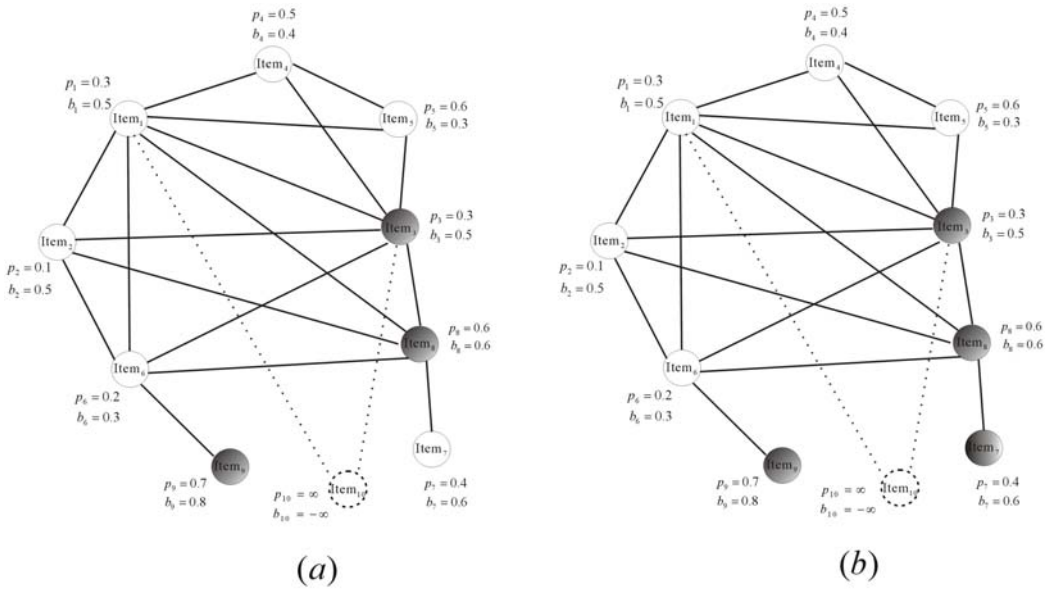


Figure 6. A numerical example to illustrate the solution steps of the MTSCA

According to our simulations, the test item set S after performing the processes of Stages 1 and 2 is very close to being a feasible solution. Thus Stage 3 generally only makes minor modifications to set S. The complete feasibility modification process and the proposed algorithm description are shown in Figures 4 and 5, respectively.

Numerical example

In this subsection, we use the problem instance shown in Figure 3(b). Assume that the user specifies the test sheet size to be 4 and the difficulty range as [0.4, 0.5]. Suppose the resulting solution after performing the processes of Stage 1 is (0, 0, 0.6, 0.8, 0.8, 0, 0, 0.9, 0.9, 0). In Stage 2, for each non-zero decision variable in the above solution, randomized rounding is performed, and we let the resulting solution be (0, 0, 1, 0, 0, 0, 0, 1, 1,

0). As shown in Figure 6(a), the corresponding test item set S for the above solution is {Item$_3$, Item$_8$, Item$_9$}, which is identified by gray nodes. Since S is not feasible, thus the modification processes in Stage 3 will be performed to modify S and make it feasible. In the difficulty feasibility modification process, due to the average difficulty value of S being equal to $(p_3 + p_8 + p_9)/3 = (0.3 + 0.6 + 0.7)/3 = 0.53 > 0.5 = p_u$, which violate the difficulty constraint, some test items in $V_A - S$ with a smaller difficulty value than $p_u (= 0.5)$ will be selected and added to S to lower the average difficulty value until the difficulty constraint is met.

In this example, the U={Item$_1$, Item$_2$, Item$_6$, Item$_7$}. Since Item$_7 \in U$ has the greatest discrimination value in U, then add it into S and the resulting S becomes {Item$_3$, Item$_7$, Item$_8$, Item$_9$} (see Figure 6(b)). The average difficulty value of S becomes (0.3+0.4+0.6+0.7)/4=0.5. Since S meets the difficulty constraint the difficulty modification process stops. In the dominating feasibility modification process, due to S being a dominating set in V_A, nothing will happen to it after performing this process. Finally, since $|S| = 4$, it meets the test sheet size constraint. Similarly, set S remains unchanged after performing the test sheet size feasibility modification process. Therefore, set {Item$_3$, Item$_7$, Item$_8$, Item$_9$} is the resulting output test sheet of our proposed MTSCA.

Time complexity analysis

Let the number of test items in the considered problem be n and the given size of the test sheet be k. The time complexity analysis for the feasibility modification process is as follows. Initially, the construction of the augmented graph will take O(n) computation steps in the worst case, since the number of basic concepts is much less than n in general. The Step 1 (difficulty feasibility modification) will firstly take $O(n \log n)$ computation steps to sort the test items according to the discrimination values in non-increasing order. The loop in this step will iteratively remove (or add) test items from (to) the solution until the difficulty constraint is met, and will be repeated at most O(n) times. Moreover, in each item remove (or add) step, the average discrimination value has to be updated, which will take O(n) computation times. Thus, Step 1 will totally take $O(n) \times O(n) = O(n^2)$ computation steps. Similar arguments can be made for Step 2; thus it takes $O(n^2)$ computation steps to adjust the current solution to meet the dominating constraint. For Step 3 (test sheet size feasibility modification), it firstly sorts the test items in either set S (the if case) or $V_A - S$ (the else case) according to the discrimination value in non-increasing order, which will take at most $O(n \log n)$ computation steps. Then for either of the above cases, the loop in Step 3 will repeatedly modify the solution by removing (or adding) at most k items. In addition, for each iteration step of the loop, the process has to check whether the updated solution will violate the difficulty and dominating constraints by taking O(n) computation steps in the worst case. Thus the sorting and loop in Step 3 take at most $O(n \log n) + O(n \cdot k)$ computation steps. Finally, summing the computation time of the above three steps, we conclude that the feasibility modification process (Stage 3 of the proposed algorithm MTSCA) will take $O(n^2) + O(n^2) + O(n \log n) + O(n \cdot k) = O(n^2)$ computation steps to modify the solution to becomes a feasible solution in the worst case.

Now, let us get back to the time complexity analysis of the proposed algorithm MTSCA. The first stage of MTSCA tries to use a polynomial time algorithm (such as the Ellipsoid method or Interior-point algorithm) to solve the linear programming model, which is known to take $O(n^3 \cdot L)$ computation steps, where L denotes the bit-length of the data. In Stage 2 (Randomized rounding) of MTSCA, it iteratively rounds the n solution values one by one into a 0-1 value in random fashion; thus Stage 2 will take O(n) computation steps. According to the above arguments, we have that Stage 3 (the feasibility modification process) takes $O(n^2)$ computation steps in the worst case. Summing the computation time of the above three stages, we have that the worst case running time of MTSCA is $O(n^3 \cdot L) + O(n) + O(n^2) = O(n^3 \cdot L)$. In the following, we demonstrate the performance of the MTSCA against other conventional methods through simulations.

Simulation results

Our experiments had two purposes, namely to compare the performance in terms of the average discrimination value and the average exposure rate of the proposed algorithm MTSCA with three different traditional test sheet composition methods. The compared methods, including the random selection method, the genetic algorithm (GA), and a GA modified method with exposure rate control called GA-exposure, are described as follows.

The compared algorithms

(1) The random selection method

Firstly, this method randomly selects some test items to form an initial test sheet. Due to the resulting test sheet perhaps not being feasible, the feasibility modification process proposed above is adopted to obtain a

feasible solution.

(2) The genetic algorithm

In this method, each test sheet solution is encoded by an n-bits binary string $X = (x_1, x_2, ..., x_n)$. In the case of $x_i = 1 (x_i = 0), 1 \le i \le n$, it stands for the test item $Item_i$ being (not being) selected in the test sheet. The fitness function F(X) in the GA is defined as follows. Function F(X) takes the average discrimination value ($w = \sum_{i=1}^{n} b_i x_i / \sum_{i=1}^{n} x_i$) as the main part, and some penalty functions are adopted to force the solution not to violate the constraints. The penalty functions include the coverage constraint violating penalty function (δ), and two difficulty constraint violating functions (α and β). The coverage constraint violating penalty function δ is defined as,

$$\delta = w \times (1 - \frac{a}{m}) \qquad (12)$$

where m denotes the number of basic concepts in the testing range and a denotes the number of basic concepts covered by the current solution

The lower bound (upper bound) of difficulty constraint violating penalty functions α (β) are defined as,

$$\alpha = w \times [p_l - \frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} x_i}]^+,$$

$$\beta = w \times [\frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} x_i} - p_u]^+,$$

Note that the function $[x]^+ = \max\{x, 0\}$. Combining the above functions, the fitness function of the GA is $F(X) = w - \alpha - \beta - \delta$.

The selection method of the GA in our simulation uses Roulette Wheel selection, and the crossover operation and mutation uses one-point crossover and bit-mutation, respectively. Similarly, in the case of the solution generated by the GA not being feasible, we again invoke the feasibility modification process to obtain a feasible solution.

(3) The GA-exposure algorithm

The GA-exposure is similar to the GA described above, except that the discrimination value b_i in the fitness function is replaced by $b_i / Xposure_i$, where $Xposure_i (1 \le i \le n)$ denotes the current exposure rate with respect to $Item_i$. The other operations (such as the selection method, the crossover operation, and the mutation operation) are all the same as in the GA. Once again the feasibility modification process is invoked if the generated solution is not feasible.

The numerical results

The first purpose of our experiment is to conduct a comparison of the performance results of our proposed algorithm MTSCA with the other test sheet composing methods in the average discrimination results estimations. For each simulation case of test item number n (n=500, 1000, 1500, 2000, …, 5000), we randomly generated 500 problem instances for simulation. The specified test sheet size k is set to 25 and the difficulty range is set to [0.5,1] and [0.4,0.6], respectively. Thus, there are totally 500 average discrimination values for each test sheet composition method. We then take the average of the 500 discrimination values. The performance results are shown in Figures 7-8 with different sets of specified difficulty range. These results demonstrate that the performance of our proposed method MTSCA is very close to that of the GA, which gains a high average discrimination value of the composed test sheets. The random selection method gains the worst discrimination value in these performance evaluations. We also conducted another experiment for evaluating the performance on the average discrimination value when the number of test items was fixed (n=2000) and the number of test sheets generated varied. As shown in Figures 9-10, the GA (random selection method) achieved the greatest (lowest) average discrimination values compared with the others in each of the evaluation cases. Our proposed method MTSCA is very close to the GA's performance.

The second purpose of our experiment was to evaluate the proposed method MTSCA against the other test sheet

composition methods on the average exposure rate comparison. In the experiment, the number of test items is fixed and set to 2000 (n=2000), and we randomly generated 500 problem instances for each given test sheet's value. Figures 11-12 show the performance results of these experiments. As shown in these figures, the GA has the worst performance results on the average exposure rates. Besides, the GA-exposure performs much better than the GA does; however, the exposure rates are greater than 0.6 in most cases. Among these methods, without a doubt, the random selection method achieves the best performance results, and our proposed method MTSCA performs very close to the random selection method. Based on the above simulation results, we conclude that the traditional test sheet optimization composition methods (such as the GA) only perform well on average discrimination comparisons, but perform worse on average exposure rate comparisons. In contrast, the random selection method performs well only on average exposure rate comparisons, but is the worst in the other comparison cases. However, our proposed method (the MTSCA) performs comparably on both the average discrimination values and the average exposure rate comparisons, making it, overall, the most effective method.

CONCLUDING REMARKS

In this paper, we propose a test sheet composing optimization problem called the TSCOP. We then transform the TSCOP to a dominating set optimization problem in graph theory. A randomized rounding based algorithm called the MTSCA is proposed to give a near optimal solution to the considered problem. The simulation results show that our proposed MTSCA performed better than other conventional test sheet composition methods on both average discrimination value and average exposure rate comparisons. In the future, the considered problem model will be extended to a multi-objective optimization problem, and we will try to design algorithms to enhance the generated results.

REFERENCES

Aarts E. & Lenstra J. K. (1997). *Local Search in Combinatorial Optimization*, John-Wiley & Sons Ltd.

Barrada J. R., Olea J., & Ponsoda V. (2007). Methods for restricting maximum exposure rate in computerized adaptative testing. *Methodology, 3*(1), 14 - 23.

Bertsimas D., Teo C., & Vohra R. (1999). On dependent randomized rounding algorithms. *Operations Research, 24*, 105 - 114.

Chang H. H. & Ying Z. (1999). A-stratified multistage computerized adaptive testing. *Applied Psychological Measurement, 23*(3) 211 - 222.

Chang H. H., Qian J., & Ying Z. (2001). a-stratified multistage computerized adaptive testing with b blocking. *Applied Psychological Measurement, 25*(4), 333 - 341.

Wang C. & Chang, H. H. (2011). Restrictive stochastic item selection methods in cognitive diagnostic computerized adaptive testing. *Journal of Educational Measurement, 48*(3), 255-273.

Hwang G. J., Yin P. Y., & Yeh S. H. (2006). A tabu search approach to generating test sheets for multiple assessment criteria. *IEEE Transactions on Education, 49*(1), 88 - 97.

Hwang G. J., Lin B. M. T., Tseng H. H., & Lin T. L. (2005). On the development of a computer-assisted testing system with genetic test sheet-generating approach. IEEE Transactions on Systems, Man, and Cybernetics, Part C: *Applications and Review, 35*(4), 590 - 594.

Haynes T. W., Hedetniemi S. T., & Slater P. J. (1998). *Fundamentals of Domination in Graphs*, Marcel Dekker, New York.

Haynes T. W., Hedetniemi S. T., & Slater P. J. (1998). *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York.

Kennedy J. & Eberhart R. C. (1995). *Particle swarm optimization.* Paper presented at the IEEE International Conference on Neural Networks, Perth, Australia.

Linden W. J. (2005). *Linear Models for Optimal Test Design*, Springer-Verlag.

Michalewicz Z. & Fogel D. B. (2002). *How to Solve It: Modern Heuristics*, Springer-Verlag.

Revuelta, J. & Ponsoda, V. (1998). A comparison of item exposure control methods in computerized adaptive testing, *Journal of Educational Measurement, 35*, 311-327.

Sympson J. B. & Hetter R. D. (1985). *Controlling item-exposure rates in computerized adaptive testing.* Paper presented at the 27th annual meeting of the Military Testing Association, San Diego, CA

Yin P. Y., Chang K. C., Hwang G. J., Hwang G. H., & Chan Y. (2006). A particle swarm optimization approach to composing serial test sheets for multiple assessment criteria, *Educational Technology & Society, 9*(3), 3 - 15.

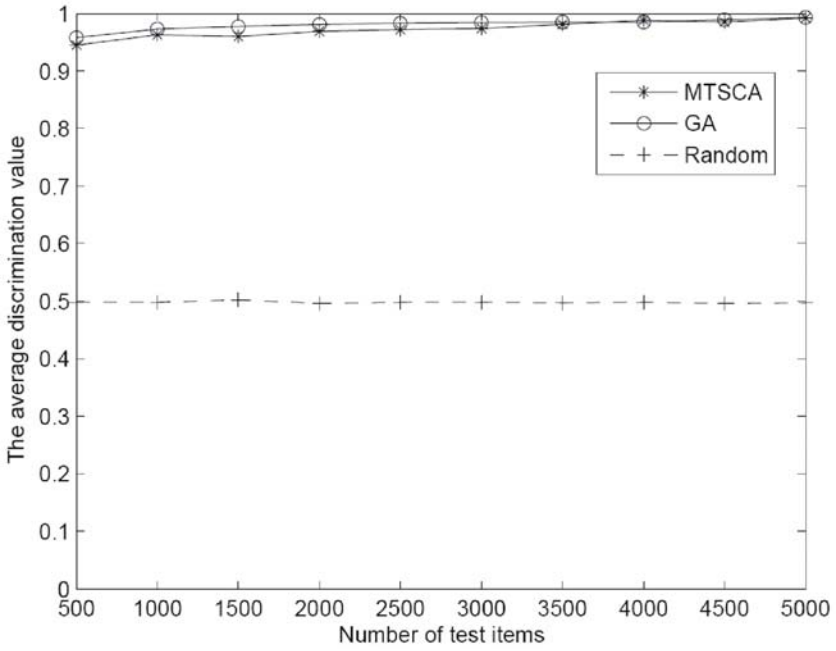


Figure 7. Performance evaluation on average discrimination value (difficulty range is [0.5,1])

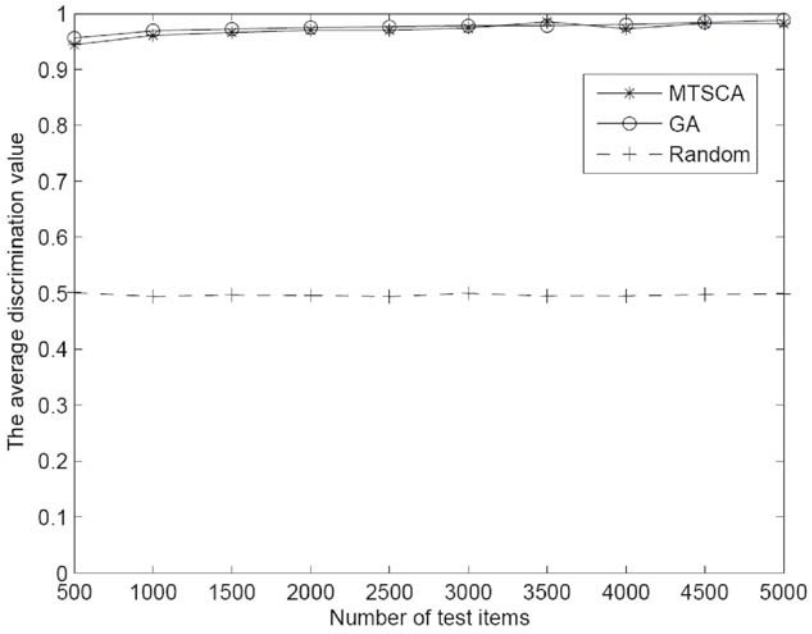


Figure 8. Performance evaluation on average discrimination value (difficulty range is [0.4,0.6])

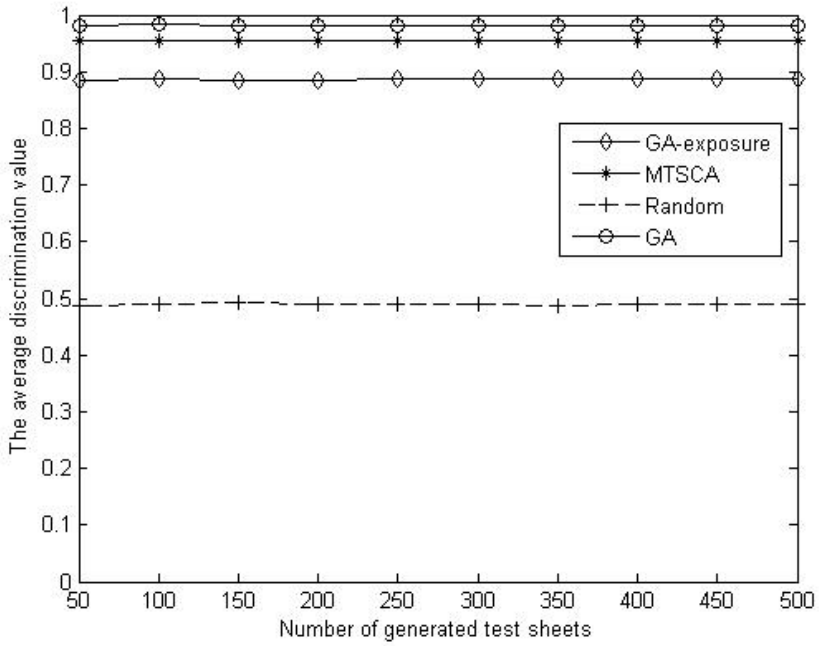


Figure 9. Performance evaluation on average discrimination value as the number of generated test sheets varied (n=2000 and difficulty range is [0.5,1])

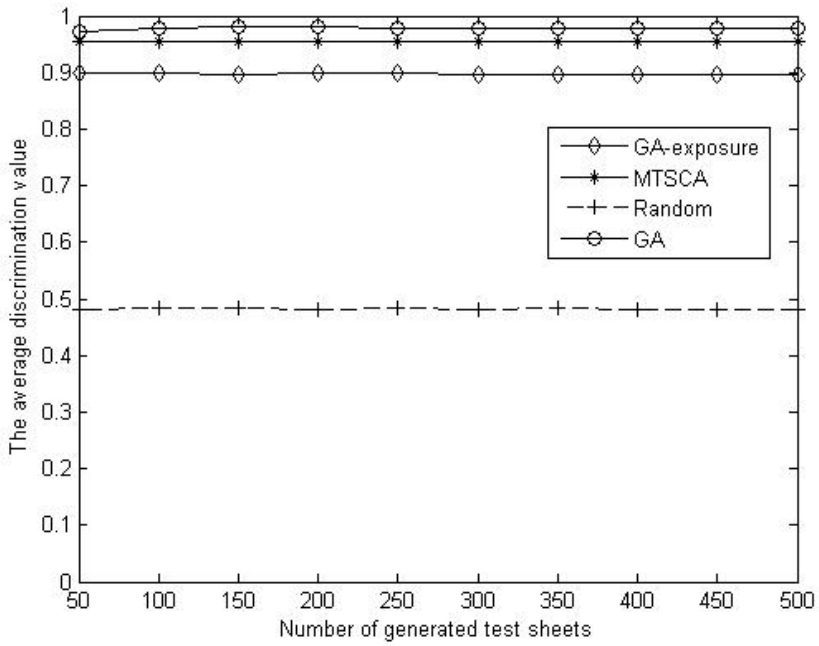


Figure 10. Performance evaluation on average discrimination value as the number of generated test sheets varied (n=2000 and difficulty range is [0.4,0.6])

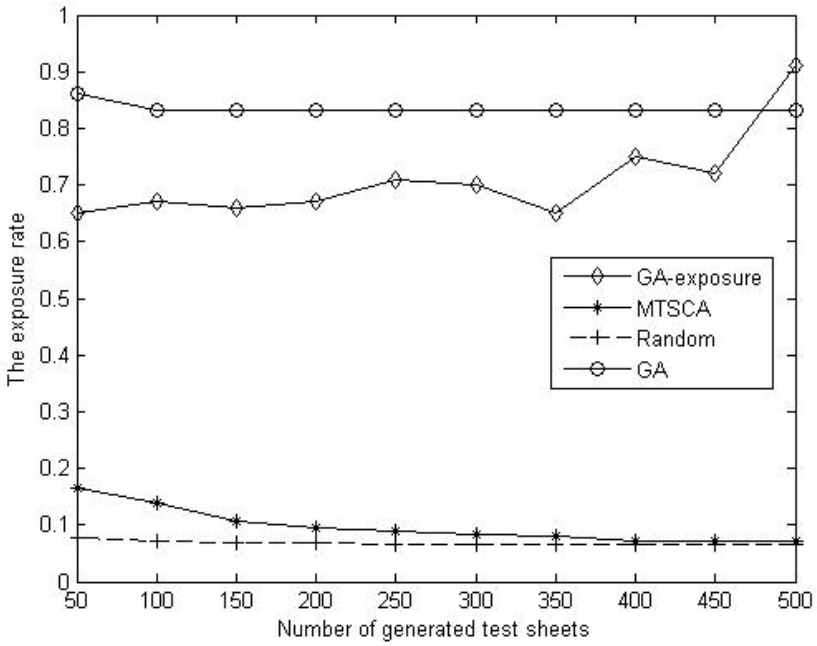


Figure 11. Performance evaluation on exposure rate (difficulty range is [0.5,1])

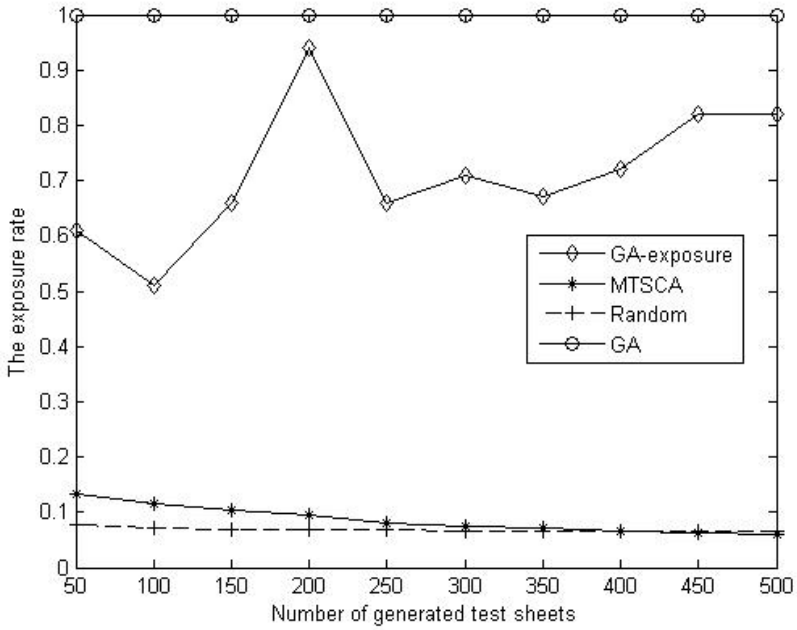


Figure 12. Performance evaluation on exposure rate (difficulty range is [0.4,0.6])