Algorithm for Uniform Test Assembly Using a Maximum Clique Problem and Integer Programming

Takatoshi Ishii[1(✉)] and Maomi Ueno[2]

[1] Tokyo University of Science, Tokyo, Japan
t.ishii@rs.tus.ac.jp
[2] University of Electro-Communications, Tokyo, Japan
ueno@ai.is.uec.ac.jp

Abstract. Educational assessments occasionally require "uniform test forms" for which each test form consists of a different set of items, but the forms meet equivalent test specifications (i.e., qualities indicated by test information functions based on item response theory). For uniform test assembly, one of most important issues is to increase the number of assembled tests. This study proposes a new algorithm, RIPMCP, to improve the number of assembled tests. RIPMCP applies a maximum clique algorithm and integer programming for assembling uniform tests. RIPMCP requires less computational space resources, thus, the proposal can assemble a greater number of tests than the previous methods on the same computational environment. Finally, we demonstrate the advantage of the proposal using simulated and actual data.

Keywords: Uniform test assembly · Maximum clique problem · Integer programming

1 Introduction

ISO/IEC 23988:2007 [6] is a global standard on the use of IT to deliver assessments to the examinees, and it recommends the use of *uniform test forms*, which are also called *parallel test forms* to secure the test reliability. Uniform test forms are the set of test forms for which each form comprises a different set of items but which must have equivalent specifications such as equivalent amounts of test information based on the item response theory [1,8], equivalent question area, equivalent average test score, and equivalent time limits. By providing different forms for each examinee, the e-testing systems employing uniform tests protect the security of tests and test items. Thus, the number of tests should be larger than the number of examinees, and one of most important issues in uniform test assembly is to increase the number of assembled tests.

To increase the number of assembled tests, Ishii and Ueno proposed a maximum clique problem (MCP) for test assemblies [4,5]. MCP is a combinational optimization in a graph. They proposed a graph in which the vertices

E. André et al. (Eds.): AIED 2017, LNAI 10331, pp. 102–112, 2017.
DOI: 10.1007/978-3-319-61425-0_9

are described as the generated tests and the edges are the satisfaction of the testsf constraints. In this graph, the maximum clique indicates the uniform tests with the maximum number of tests. From extracting the maximum clique, these methods assemble a greater number of uniform tests than any other traditional method [9–11]. However, these methods have a major computational space cost. Thus, there the number of assembled tests is restricted by the calculation cost.

In this paper, we propose a new algorithm, RIPMCP, to reduce the computational space cost and to increase the number of assembled tests. RIPMCP is a similar algorithm to the previous algorithm [4,5]. The major difference between RIPMCP and the previous algorithm is that RIPMCP generates graph structures for maximum clique searching by solving the integer programming. From this graph generation, RIPMCP can assemble tests with lower computational space cost than the previous method [4,5]. Thus, RIPMCP can assemble a greater number of assembled tests than the previous methods using the same computational environment. Therefore, the proposed algorithm can utilize the item pool more efficiently than traditional methods. Finally, we demonstrate the effectiveness of the proposed methods using simulated and actual data.

2 Item Response Theory

Many previous studies (such as [4,5,9–11]) use item response theory (IRT) [1,8] to measure the quality of tests for uniform test assembly. This section provides IRT equations to prepare the later description.

IRT, which describes the relation between item characteristics and examinee abilities, can measure examinee abilities on the same scale even when the examinees are taking different tests. For IRT, u_{ij} denotes the response of item i $(= 1, \ldots, n)$ on examinee j $(= 1, \ldots, m)$ as

$$u_{ij} = \begin{cases} 1 \text{ If the } j\text{th examinee answers} \\ \quad i\text{th item correctly,} \\ 0 \text{ Otherwise.} \end{cases}$$

In the two-parameter logistic model which is one of the most popular IRT models, the probability of a correct answer to item i by examinee j with ability $\theta_j \in (-\infty, \infty)$ is assumed as

$$\mathrm{p}_i(\theta_j) \equiv \mathrm{p}(u_{ij} = 1 | \theta_j) = \frac{1}{1 + \exp(-1.7a_i(\theta_j - b_i))},$$

where $a_i \in [0, \infty)$ is the ith item's discrimination parameter, and $b_i \in (-\infty, \infty)$ is the ith item's difficulty parameter. This probability is called the item characteristic curve (ICC).

Using this probability, we can define the item reliability that measures how accurately the item can estimate the examinee's ability levels θ. The ith item reliability $\mathrm{I}_i(\theta_j)$ based on the two-parameter logistic model is defined as

$$\mathrm{I}_i(\theta) = a_i^2 \mathrm{p}_i(\theta)(1 - \mathrm{p}_i(\theta)).$$

This function is a Fisher information metric calculated from the ICC. Furthermore, on the condition called the local independence, the probability of one item being used is not related to any other item(s) being used and that response to an item is each and every examinees' independent decision, the test reliability of tests is described as the sum of the information functions of the items in the test form. The test information function $\mathrm{I}_{Test}(\theta)$ of a test $Test$ is defined as

$$\mathrm{I}_{Test}(\theta_j) = \sum_{i \in Test} \mathrm{I}_i(\theta_j).$$

By using this measure, a test administrator can estimate how accurate a test form is. In traditional uniform test assembly methods (e.g. [9,11]), the test information function is treated discretely: the test information function has been compared on some points $\Theta = \{\theta_1, \ldots, \theta_k, \ldots, \theta_K\}$ in ability level θ. In this paper, we treat the test information function in the same way.

3 Maximum Clique Algorithm for Uniform Test Assembly

Ishii and Ueno proposed the MCP for uniform test assembly [4,5]. The clique problem is a combinational optimization in graph theory [2,7]. A graph is represented as a pair $G = \{V, E\}$, where V denotes a set of vertices, and E denotes a set of edges. The clique problem seeks a special structure called the *clique* from a given graph. A clique is a set of vertices for which each pair of vertices is connected. The MCP searches for the clique which has the maximum number of vertices in the given graph. Letting $G = \{V, E\}$ be a finite graph and letting $C \subseteq V$ be the clique, then the MCP is formally defined as follows:

$$\begin{aligned} &\textbf{maximize } |C| \\ &\textbf{subject to} \\ &\qquad \forall v, w \in C,\ \{v, w\} \in E \\ &\qquad \text{(clique constraint)}. \end{aligned} \tag{1}$$

In this study [5], they employed the MCP to search for the maximum number of uniform tests. In general, uniform tests are defined as a set of tests that has following specifications:

1. any test satisfies all test constraints;
2. any two tests satisfy the overlapping constraint. (i.e. any two test forms have fewer overlapping items than the allowed number in the overlapping constraint).

Accordingly, the assembling of the maximum number of uniform test forms can be described as the maximum clique extraction from the following *corresponding graph*:

$$V = \left\{ \begin{array}{l} s : s \in S,\ \text{Feasible test s satisfies all test constraints} \\ \quad \text{except for the overlapping constraint from a given item pool} \end{array} \right\}$$

$$E = \left\{ \{s, s'\} : \text{The pair of s and s' satisfies the overlapping constraint} \right\}.$$

Here, the test constraints include a constraint for the number of items and a test information function. Letting L_{θ_k} be a lower bound and letting U_{θ_k} be a upper bound for test information function on $\mathrm{I}_{Test}(\theta_k)$, a constraint for the test information function is written as the following equation:

$$L_{\theta_k} \leq \mathrm{I}_{Test}(\theta_k) \leq U_{\theta_k}. \tag{2}$$

In addition, if we let O be the allowed number in the overlapping constraint and both s and s' be tests which are the sets of items, the overlapping constraint is defined as follows:

$$\forall s, \forall s' \in V, \tag{3}$$
$$|s \cap s'| \leq \mathrm{O}. \tag{4}$$

The proposed MCP seeks the maximum set of feasible test forms in which any two test forms satisfy the overlapping constraint. Therefore, this set of tests is the maximum number of uniform tests from a given item pool.

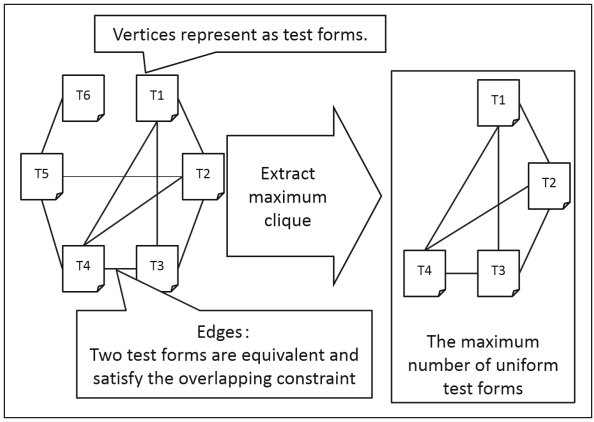

Fig. 1. Maximum clique algorithm for uniform test assembly.

Figure 1 presents an example of uniform test form assembly using the MCP. The graph G has six feasible test forms T1–T6 with nine satisfactions of the overlapping constraint and the maximum uniform tests $C_{max} = \{\mathrm{T1}, \mathrm{T2}, \mathrm{T3}, \mathrm{T4}\}$, that is the maximum number of tests in which any pair of tests satisfies the given overlapping constraint.

Unfortunately, this problem for assembling uniform tests cannot be solved exactly because it has heavy computational time and space costs. To solve the problem, in previous work [5] an approximate algorithm called RndMCP was proposed.

This algorithm has the following three parameters for computational costs:

$\mathbf{C_1}$ is the number of feasible tests assembled in Step 1;
$\mathbf{C_2}$ is the time limit of Step 3;
$\mathbf{C_3}$ is the total time limit of the test assembly.

This algorithm contains the following four stpdf.

Step 1: This step assembles $\mathbf{C_1}$ feasible tests, and stores those tests.
Step 2: This step builds graph structures by checking the number of overlapping items between any two stored tests. If the number of overlapping items between two vertices (tests) is less than a given O, those vertices are connected.
Step 3: This step extracts the maximum clique from the structure built by **Step 2**. **Step 3** is aborted by the calculation time $\mathbf{C_2}$. By comparing the size of the extracted clique and current maximum clique, this step stores the larger clique as the current maximum clique.
Step 4: If the calculation time is less than $\mathbf{C_3}$, go to **Step 1**; otherwise return the current maximum clique.

RndMCP repeatedly extracts the maximum number of uniform tests from a subgraph of the global corresponding graph. Therefore, in the case where $\mathbf{C_1}$ is larger than the size of the maximum clique in the global corresponding graph, RndMCP asymptotically extracts the maximum clique as the maximum number of uniform test forms from the global corresponding graph.

The computational time cost of RndMCP is $\mathbf{C_3}$ and the space cost is $O(\mathbf{C_1}^2)$. Therefore, it is possible to extract uniform tests in a limited computing environment by controlling computational time and space costs.

However, when this algorithm assembles $|C|$ uniform tests, this algorithm requires at least $O(|C|^2)$, because the extracted uniform tests are a subset of the $\mathbf{C_1}$ tests assembled by **Step 1**. Therefore, this algorithm has a problem of requiring a calculation cost proportional to the square of the number of configuration tests.

4 Uniform Test Assembly Using the Maximum Clique Algorithm and Integer Programming

To reduce the computational space cost and to increase the number of assembled tests, we propose a new algorithm: RIPMCP. By employing integer programming to generate a subgraph, the proposal divides the extraction of the maximum clique from the global graph into repeated extractions from subgraphs.

In the corresponding graph, the edges describe the satisfactions of the overlapping constraint which implies that there are fewer overlap items between two connected vertices. Therefore, the searching of a vertex connected with a certain vertex becomes an optimization problem with a constraint for the connection.

RIPMCP has the same constraint parameters for computational costs as Ishii and Ueno's method [4,5], and contains the following five stpdf.

Step 1: This step sets the current searching clique Q as empty, and the current maximum clique Q_{max} as empty.

maximize

$$\sum_{i=1}^{n} \lambda_i x_i \tag{5}$$

where

$$x_i = \begin{cases} 1 \text{ If the } i\text{th item is selected in the feasible test,} \\ 0 \text{ Otherwise.} \end{cases}$$

subject to

$$\sum_{i=1}^{n} x_i = \mathrm{M} \tag{6}$$

$$L_{\theta_k} \leq \sum_{i=1}^{n} \mathrm{I}_i(\theta_k) x_i \leq U_{\theta_k}, (k = 1 \ldots K) \tag{7}$$

$$\sum_{i=1}^{n} \mathrm{x}_{i,r} x_i \leq \mathrm{O}, (r = 1 \ldots |Q|) \tag{8}$$

In this problem, t_i is the average response time of item i,

$$\mathrm{x}_{i,r} = \begin{cases} 1 \text{ If the } i\text{th item is selected in the } r\text{th test} \\ \quad \text{in the current searching clique } Q, \\ 0 \text{ Otherwise.} \end{cases}$$

Therein, coordinates $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote random variables distributed uniformly on $[0, 1]$. Here $\lambda_i (0 \leq i \leq n)$ are re-sampled after each problem is solved.

Fig. 2. Integer programming problem for assembling the feasible test.

Step 2: This step assembles $\mathbf{C_1}$ tests by solving the integer programming problem. Figure 2 shows the integer programming problem. This problem contains the test constraints and overlapping constraints. The solution vertex (test) is feasible and has fewer overlapping items between each test in the current searching clique. Then, this step stores those tests.

Step 3: This step builds graph structures by checking the number of overlapping items between any two stored tests. If the number of overlapping items between two vertices (tests) is lower than a given O, those vertices are connected.

Step 4: This step extracts the maximum clique from the structure built by **Step 3**. **Step 4** is aborted by the calculation time $\mathbf{C_2}$. Then, this step adds the maximum clique solution to the current searching clique Q. By comparing the size of the current searching clique Q and the current maximum clique Q_{max}, this step stores the larger clique as the current maximum clique Q_{max}.

Step 5: If the calculation time is less than $\mathbf{C_3}$, go to **Step 2**; otherwise output the current maximum clique Q_{max}. If the integer programming is in **Step 2**, go to **Step 1**.

RIPMCP is similar to the RndMCP algorithm. RndMCP randomly assembles feasible tests as vertices and searches for the maximum clique from those vertices. The proposal randomly assembles feasible tests but those tests are connected to all vertices in the current searching clique Q. Thus, $Q\cup$ the clique in those feasible tests is also the clique in the global corresponding graph. Then, the

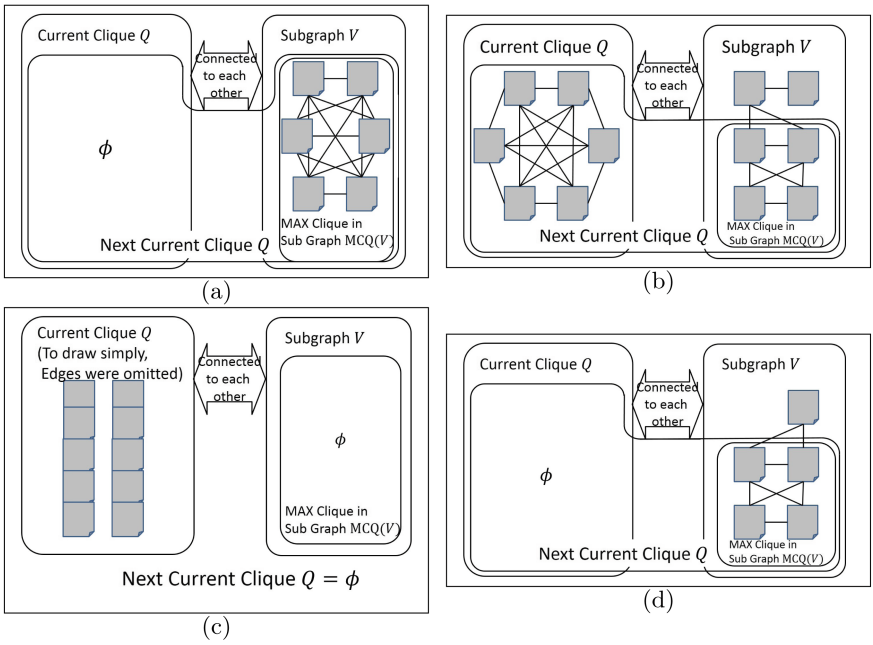

Fig. 3. Searching image of the proposed method.

proposal searches for the maximum clique from those vertices, and adds that maximum clique to the current searching clique Q.

Figure 3 presents a searching image of the proposal. In Fig. 3, the search will be conducted in the order (a)→(b)→(c)→(d). First, the proposed method sets $Q := \phi$ and randomly assembles the $\mathbf{C_1}$ tests V that satisfy the given test constraint by solving the integer programming problem. Then, the method constructs the corresponding graph G by checking the overlap constraint among assembled tests in V. In Fig. 3(a), the proposal searches for the maximum clique in the random tests V. The proposal adds the found clique to the current clique Q. For the first time, this step is the same as the first step of the RndMCP method.

Next, the proposal assembles the $\mathbf{C_1}$ tests V that satisfy the test constraint and the overlap constraint between all tests in the current clique Q by solving the integer programming problem. Then, the proposal constructs the graph structure and extracts the maximum clique in the graph. In case (b), a vertex in the found clique $\mathrm{MCQ}(V)$ has edges to all vertices in the current clique Q. Therefore, $Q \cup \mathrm{MCQ}(V)$ is a clique in the global corresponding graph. Thus, the method sets $Q \cup \mathrm{MCQ}(V)$ as the next current clique Q.

Third, the proposed method tries to assemble the $\mathbf{C_1}$ tests V in the same way. However, in case (c), the integer programming problem has no solution. In other words, the current clique Q is the maximal clique in the global corresponding

graph. To repeat these stpdf, the proposed method tries to search for the maximum cliques. Then, this algorithm initializes $Q = \phi$ (case (d)), and repeats those stpdf.

The computational cost of the proposal is the same as RndMCP. The computational time is $\mathbf{C_3}$ and the space cost is $O({\mathbf{C_1}}^2)$. However, the extracted uniform tests are not subsets of the $\mathbf{C_1}$ tests assembled by **Step 2**. Therefore, when this algorithm assembles $|C|$ uniform tests, this algorithm requires $O(|C|)$ space cost for storing clique vertices.

5 Experiments

To demonstrate the advantage of our proposal, we conducted an experiment. We compared the number of assembled test forms of our proposal with those of traditional methods [5,9–11].

We compared each method with the simulated and actual item pools that have 500–2000 items. The items in the simulated item pools have the discrimination parameter a and the difficulty parameter b based on IRT. We set the discrimination parameter a as $\log_2 a \sim N(0, 1^2)$, and the difficultly parameter $b \sim N(0, 1^2)$. Table 1 shows the details of the actual item pools.

We set the test constraints as follows.

1. The test includes 25 items.
2. The allowed numbers of overlapping items are 0 and 10.

The information constraint is described by the lower and upper bounds of the test information function $\mathrm{I}(\theta_k)$ and are listed in Table 2.

Table 1. Details of the actual item pool.

Item pool size	Parameter a			Parameter b		
	Range	Mean	SD	Range	Mean	SD
978	0.12–3.08	0.43	0.20	−4–4.55	−0.22	1.16

Table 2. Constraints for test assembly.

I(θ) (Lower bound/Upper bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0$	$\theta = 1.0$	$\theta = 2.0$
2/2.4	3.2/3.6	3.2/3.6	3.2/3.6	3.2/3.6

We used a time limitation of test assembly of 24 h for all methods. For RndMCP [5] and our proposal, we determined the computational cost constraint $\mathbf{C_1}$ as 100,000, $\mathbf{C_2}$ as 3 h, and $\mathbf{C_3}$ as 24 h. For BST [11] and our proposal, We used CPLEX [3] for the integer programming problem.

Table 3. The numbers of assembled tests for each methods.

item pool size	OC	BST	GA	BA	RndMCP	RIPMCP
500	0	12	3	5	10	18
	5	20	23	96	4380	20,547
1000	0	21	4	6	17	33
	5	40	17	104	46,305	58,760
2000	0	53	8	12	32	69
	5	80	22	104	96,876	102,666
978 (actual)	0	24	9	9	16	35
	5	39	283	371	40,814	55,658

Table 4. The number of assembled tests in 168 h.

	OC	RndMCP	RIPMCP
2000	5	96,949	134,383
	10	99,999	136,318
978 (actual)	5	45,955	96,787
	10	99,999	132,451

In the table, "BST" denotes big shadow method [11], "GA" denotes [10], "BA" denotes [9], and "RndMCP" denotes [5]. Our proposal is listed as "RIPMCP".

Table 3 shows the number of test forms assembled using our proposal and the traditional methods for the item pool size and the overlapping constraint. In traditional methods, with the exception of "OC = 0" cases, RndMCP assembles a greater number of test forms than the other methods. This is because the aim of BST, GA, and BA is not to maximize the number of assembled tests. In the case of "OC = 0," BST assembles a greater number of tests than RndMCP. Moreover, in the case of "item pool size = 2000, OC = 5," the number of assembled tests by RndMCP converged to 100,000. The reasons were that the $\mathbf{C_1}$ size was too small for this test assembly setting.

On the other hand, the proposal assembled a greater number of tests than RndMCP in all cases. In more detail, the difference in the number of tests between RndMCP and our proposal was small in the situation that the number of assembled tests are nearly 100,000. This might be caused by the fact that the integer programming in our proposal takes a lot of time when the number of assembled tests is large. Moreover, the number of assembled test by RndMCP does not exceed $\mathbf{C_1}$; therefore, setting a larger time limit might increase the difference in the number of tests between RndMCP and our proposal in that situation.

To confirm this, finally we compared the number of assembled tests by RndMCP and our proposal, under the situation of setting the time limit as 168 h (7 days). For RndMCP [5] and our proposal, we determine the computational cost constraint C_1 as 100,000, C_2 as 3 h, and C_3 as 168 h (7 days). We examined item pools sizes 2000 and 978 and OC = 5 and 10.

Table 4 lists the number of assembled tests at time 168 h for both methods. Figure 4 plots the number of assembled tests for calculation time in the situation of item pool size 2000 and OC = 5. From Table 4 and Fig. 4, the proposed method can assemble a greater number of tests than RndMCP, and the difference in the number of assembled tests might increase with calculation time. In all situations, the number of assembled tests by RndMCP did not increase with calculation time; however, the number of assembled tests using our proposal did increase.

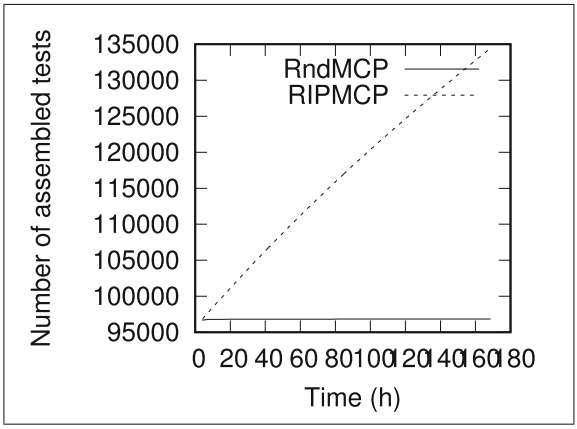

Fig. 4. Relation between the calculation time and the number of assembled tests.

6 Conclusion

We have proposed an algorithm that assembles a greater number of uniform tests than traditional methods. The proposal applies integer programming and the MCP for assembling uniform tests.

To demonstrate the performance of the proposed method, we have conducted an experiment using simulated and actual data. To summarize the results, the proposed method assembled a greater number of uniform tests than the traditional methods. Moreover, the results suggested that the difference in numbers of assembled tests between proposal and Ishii and Ueno's method [5] was increased by extending the calculation time.

Future works will include assessing this method in practical uses, and improving the algorithm to increase the number of assembled tests.

References

1. Baker, F., Kim, S.: Item Response Theory: Parameter Estimation Techniques, 2nd edn. Statistics: A Series of Textbooks and Monographs. Taylor & Francis (2004)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
3. ILOG: ILOG CPLEX User's Manual 11.0
4. Ishii, T., Songmuang, P., Ueno, M.: Maximum clique algorithm for uniform test forms. In: The 16th International Conference on Artificial Intelligence in Education, pp. 451–462 (2013)
5. Ishii, T., Songmuang, P., Ueno, M.: Maximum clique algorithm and its approximation for uniform test form assembly. IEEE Trans. Learn. Technol. **7**(1), 83–95 (2014)
6. ISO/IEC: ISO/IEC 23988:2007 Information technology - A code of practice for the use of information technology (IT) in the delivery of assessments (2007)

7. Karp, R.M.: Reducibility among combinatorial problems. Complex. Comput. **40**(4), 85–103 (1972)
8. Lord, F.M., Novick, M.R.: Statistical Theories of Mental Test Scores. Addison-Wesley Series in Behavioral Science. Addison-Wesley Pub. Co., MA (1968)
9. Songmuang, P., Ueno, M.: Bees algorithm for construction of multiple test forms in e-testing. IEEE Trans. Learn. Technol. **4**, 209–221 (2011)
10. Sun, K.T., Chen, Y.J., Tsai, S.Y., Cheng, C.F.: Creating irt-based parallel test forms using the genetic algorithm method. Appl. Measur. Educ. **2**(21), 141–161 (2008)
11. van der Linden, W.J.: Liner Models for Optimal Test Design. Springer, New York (2005)