Maximum Clique Algorithm and Its Approximation for Uniform Test Form Assembly

Takatoshi Ishii, Pokpong Songmuang, and Maomi Ueno, *Member, IEEE*

Abstract—Educational assessments occasionally require uniform test forms for which each test form comprises 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). We propose two maximum clique algorithms (MCA) for uniform test form assembly. The proposed methods can assemble uniform test forms with allowance of overlapping items among uniform test forms. First, we propose an exact method that maximizes the number of uniform test forms from an item pool. However, the exact method presents computational cost problems. To relax those problems, we propose an approximate method that maximizes the number of uniform test forms asymptotically. Accordingly, the proposed methods can use the item pool more efficiently than traditional methods can. We demonstrate the efficiency of the proposed methods using simulated and actual data.

Index Terms—Maximum clique problem, item response theory, uniform test forms, test assembly

1 Introduction

Educational assessments occasionally require *uniform test forms* (which is also called *parallel test forms*) for which each form comprises a different set of items but which still must have equivalent specifications such as equivalent amounts of test information based on item response theory (IRT), equivalent question area, equivalent average test score, and equivalent time limits. For example, uniform test forms are necessary when a testing organization administers a test in different time slots. To achieve this, uniform test forms are assembled in which all forms have equivalent qualities so that examinees who have taken different test forms can be evaluated objectively using the same scale.

Recently, automatic assembly for test forms has become popular. Automatic assembly assembles test forms to satisfy given test constraints, such as the numbers of test items, the numbers of items from each set of contents, amounts of test information, average test scores, and/or etc., to provide equivalent qualities [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17].

In earlier studies, a test assembly was formalized as a combinational optimization problem. For example, van der Linden [18] proposed the big shadow test method using linear programming (LP). This method sequentially assembles uniform test forms by minimizing qualitative differences between a current assembled test form and the remaining set of items in an item pool. Although this method assembles uniform test forms in a practically acceptable time, it presents two problems. First, qualitative differences increase with the assembled order of test forms. Second, this method does not maximize the number of uniform test forms from the item pool.

To alleviate or ameliorate the first problem, Sun et al. [19] proposed the use of a genetic algorithm (GA) for uniform test assembly that simultaneously assembles uniform test forms as minimizing differences among the qualities of assembled test forms and user-determined values. Furthermore, Songmuang and Ueno [20] applied the bees algorithm (BA) to uniform test form assembly and thereby improved the performance of the method proposed by Sun et al. [19]. Although these methods [18], [19], [20] demonstrated effective performance for minimizing the qualitative differences among assembled test forms, no method maximizes the number of uniform test forms from the item pool. These methods do not allow the item pool to be used efficiently to the greatest degree possible.

To maximize the number of test forms, Belov and Armstrong [21] proposed a uniform test assembly method based on maximum set-packing problems (MSP). Moreover, Belov proposed a random test assembly method to improve the tractability of maximizing the number of uniform test forms. However, these methods [21], [22] cannot assemble uniform test forms with overlapping items, where overlapping items mean common items among multiple test forms. In the non-overlapping conditions, each item is used only once on assembled test forms. Therefore, the non-overlapping condition strongly restricts the number of assembled test forms. Consequently, the non-overlapping condition interrupts the efficient uses of the item pool.

Our study is conducted to assess a proposed uniform test form assembly method that maximizes the number of assembled test forms with overlapping conditions. To achieve this goal, we apply the maximum clique algorithm

- *T. Ishii and M. Ueno are with the University of Electro-Communications. E-mail: {ishii, ueno}@ai.is.uec.ac.jp.*
- *P. Songmuang is with Thammasat University. E-mail: pokpong@cs.tu.ac.th.*

Manuscript received 10 May 2013; revised 25 Nov. 2013; accepted 19 Dec. 2013; date of publication 1 Jan. 2014; date of current version 6 May 2014.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TLT.2013.2297694

(MCA), which solves the maximum clique problem (MCP). We propose an exact method based on maximum clique problem (ExMCP) for the maximum number of uniform test forms from the item pool.

The unique feature of ExMCP is that it generalizes Belov and Armstrong's method [21] to maximize the number of uniform test forms with an overlapping condition. Therefore, theoretically, ExMCP can assemble a greater number of test forms than when using traditional methods (e.g., [18], [19], [20], [21]). In fact, ExMCP is expected to use the item pool more efficiently than traditional methods do.

However, the computational time and space costs of ExMCP increase exponentially with the number of *feasible test forms* (i.e., a set of test forms that satisfies all test constraints except for the overlapping constraint from a given item pool). Therefore, it is difficult to use ExMCP for a large item pool.

To relax this problem, we propose RndMCP by approximating ExMCP using a random search approach, such as that explained in an earlier report of the literature [23]. RndMCP maximizes the number of uniform test forms asymptotically from the item pool with overlapping conditions and assembles a greater number of test forms than those assembled using traditional methods (e.g., [18], [21]). In addition, RndMCP seeks the maximum number of uniform test forms more efficiently than traditional random search methods [19], [20] do because the search space of RndMCP is more restrictive than those of the traditional methods.

Moreover, some experiments were conducted to evaluate the proposed methods. Results show that the proposed methods assemble a greater number of uniform test forms than the traditional methods do.

2 Item Response Theory

Most previous studies of test form assembly use item response theory to measure the quality of test forms [18], [20], [24], [25], [26], [27].

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 test forms. For IRT, u_{ij} denotes the response of item $i(1,\ldots,n)$ on examinee $j(1,\ldots,m)$ as

$$u_{ij} = \begin{cases} 1, & \text{If } j\text{th examinee answers} \\ & i\text{th item correctly,} \\ 0, & \text{Other Cases.} \end{cases}$$

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 $\theta \in (-\infty, \infty)$ is assumed as

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

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.

Using this correct probability, we can define the item information that measures how accurately an item can estimate the examinee's ability levels θ.

The ith item information function $I_i(\theta)$ based on the two-parameter logistic model is defined as

$$I_i(\theta) = a_i^2 p_i(\theta)(1 - p_i(\theta)).$$

To estimate how much accuracy a test form has, a test administrator monitors test information functions. We can define the test information similarly to the item information function. The test information function is the sum of the information functions of the items in the test form. The test information function $I(\theta)$ of a test form $Test$ is defined as

$$I(\theta_j) = \sum_{i \in Test} I_i(\theta_j).$$

Almost all traditional methods use this test information function as the test form quality. Accordingly, their uniform test assemblies are implemented by minimizing the difference between the assembled test information functions. (In practice, they compare the test information function on some points $\Theta = \{\theta_1, \ldots, \theta_k, \ldots, \theta_K\}$ in ability level θ, and minimize each difference of test information $I(\theta_1), \ldots, I(\theta_k), \ldots, I(\theta_K)$.)

In fact, the test assembly methods were presumed to be implemented after each item's IRT parameters had been collected in the item pool.

3 Traditional Methods of Uniform Test Form Assembly

This section introduces some traditional methods.

3.1 Big Shadow Test Method

For uniform test forms, van der Linden [18] proposed a big shadow test method using linear programming. This method assembles test forms sequentially by minimizing the difference of test information functions between a current assembled test form and a set of items remaining in the item pool. They called the set of remaining items a *shadow test*.

This method assembles equivalent test forms to solve an optimization problem as follows:

$$\begin{aligned} &\textbf{minimize } y \\ &\textbf{subjectto} \\ &\qquad \sum_{k=1}^{K} \sum_{i=1}^{n} |I_i(\theta_k)x_i - T(\theta_k)| \le y, \qquad (1) \\ &\qquad \sum_{k=1}^{K} \sum_{i=1}^{n} |I_i(\theta_k)z_i - T_{ST}(\theta_k)| \le y, \end{aligned}$$

where

$$y \ge 0$$

$$x_i = \begin{cases} 1, & \text{If } i\text{th item is selected} \\ & \text{into test form,} \\ 0, & \text{Otherwise,} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{If } i\text{th item is selected} \\ & \text{into shadow test form,} \\ 0, & \text{Otherwise,} \end{cases}$$

denotes a distribution of the ability level θ_k, and $T_{ST}(\theta k)$ denotes a target value of information function at ability level θ_k for the shadow test form. The test quality constraints without information function are included in the constraints of Problem (1).

The inequality $\sum_{k=1}^{K}\sum_{i=1}^{n}|I_i(\theta_k)x_i - T(\theta_k)| \leq y$ in Problem (1) minimizes the difference of test information functions between the currently assembled test form and the target value $T(\theta k)$. The inequality $\sum_{k=1}^{K}\sum_{i=1}^{n}|I_i(\theta_k)z_i - T_{ST}(\theta_k)| \leq y$ in Problem (1) minimizes the difference of test information functions between the set of remaining items and the target value $T_{ST}(\theta k)$. That is to say, Problem (1) simultaneously minimizes the difference of test information functions between a current assembled test form and a set of items remaining in the item pool.

Solving Problem (1) assembles an equivalent test form one by one. For uniform test forms, this method repeatedly solves this problem. Let R be the user-determined desired number of assembled test forms. The algorithm of this method is shown as Algorithm 1.

Algorithm 1 Big Shadow Test method [18].

Require: Item pool and test constraints
Ensure: Uniform test forms
Step 1:Set $U := \phi$.
Step 2:Solve Problem (1): if successful, the resulting test $\rightarrow \tilde{x}$; otherwise, go to Step 4.
Step 3:Add $\tilde{x}$ to the set U, if $|U| < R$ go to Step 2.
Step 4:Return U.

This method decomposes the uniform test form assembly into R times assemblies. This decomposition reduces computational costs, but increases the qualitative difference between the first assembled test form and the last test form.

3.2 Genetic Algorithm for Uniform Test Form Assembly

To reduce the qualitative difference among the assembled test forms, Sun et al. [19] proposed a uniform test form assembly using the genetic algorithm. This method simultaneously assembles test forms by minimizing the difference between the qualities of assembled test forms and the target value $T(\theta_k)$.

Then the method assembles the uniform test forms by solving an optimization problem as follows:

$$\begin{aligned} &\textbf{minimize } y \\ &\textbf{subject to} \\ &\sum_{k=1}^{K}\sum_{i=1}^{n}|I_i(\theta_k)x_{ir} - T(\theta_k)| \leq y, \\ &(r = \{1, 2, \ldots, R\}), \end{aligned} \tag{2}$$

where

$$x_{ir} = \begin{cases} 1, & \text{If } i\text{th item is selected} \\ & \text{into } r\text{th test form,} \\ 0, & \text{Otherwise.} \end{cases}$$

The test quality constraints without an information function are included in the constraints of Problem (2), which minimizes the difference of test information functions simultaneously and directly among all test forms. The term $\sum_{k=1}^{K}\sum_{i=1}^{n}|I_i(\theta_k)x_{ir} - T(\theta_k)|$ in Problem (2) is the difference between the quality of rth test form and target value $T(\theta_k)$. This term is called the *fitting error*. That is, Problem (2) minimizes all the test form fitting errors.

This method approximately solves Problem (2) using GA. More specifically, a vector $\mathbf{x} = (x_{11}, x_{21}, \ldots, x_{N1}, x_{12}, x_{22}, \ldots, x_{N2}, \ldots x_{NR})$ in Problem (2) is represented as population for the GA operations (i.e., random sampling, fitting, crossover, and mutation).

This method repeats the GA operations to the vector $\mathbf{x}$, and seeks a better solution that has less qualitative difference among the assembled test forms.

3.3 Bees Algorithm for Uniform Test Form Assembly

Songmuang and Ueno [20] proposed a uniform test form assembly method using the bees algorithm.

This method has the following two steps:

1. (Satisfying test constraints)
   Step 1 assembles test forms only to minimize the fitting errors of each test form. To achieve this, step 1 solves the following optimization problem:

$$\textbf{minimize} \quad \sum_{k=1}^{K}\sum_{i=1}^{n}|I_i(\theta_k)x_i - T(\theta_k)|, \tag{3}$$

   where

$$x_i = \begin{cases} 1, & \text{If } i\text{th item is selected} \\ & \text{into test form,} \\ 0, & \text{Otherwise.} \end{cases}$$

   The test quality constraints without information function are included in the constraints of Problem (3).

   This step repeats solving Problem (3) L times and assembles L feasible test forms that satisfy the test constraints without overlapping constraints.
2. (Equating test forms)
   Step 2 extracts the most equivalent set of test forms from the assembled L feasible test forms in Step 1.

   To achieve this, step 2 solves the following optimization problem:

$$\textbf{minimize} \sqrt{\frac{1}{\sum_{l=1}^{L} s_l + 1}\sum_{l=1}^{L} s_l(e - \mu_S)^2}, \tag{4}$$

where

$$s_l = \begin{cases} 1, & \text{If } l\text{th test form is selected} \\ & \text{into the set of test forms,} \\ 0, & \text{Otherwise,} \end{cases}$$

$$e = \sum_{k=1}^{K} \sum_{i=1}^{n} |I_i(\theta_k)x_i - T(\theta_k)|,$$

$$\mu_S = \frac{1}{\sum_{l=1}^{L} s_l + 1} \sum_{l=1}^{L} s_l e.$$

The objective function in Problem (4) is a standard deviation of fitting errors e among the extracted test forms (indicated by vector $\mathbf{s} = (s_1, s_2, \ldots, s_L)$). The overlapping constraint is included in optimization constraints of Problem (4).

The test forms extracted by solving Problem (4) are the uniform test forms. These satisfy all qualitative constraints and overlapping constraints.

Both A and B steps use the bees algorithm to solve each Problem (3), (4). The vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in Problem (3) and the vector $\mathbf{s} = (s_1, s_2, \ldots, s_L)$ in Problem (4) are represented as populations. This method repeats the BA operations (i.e., generates neighbor solutions, evaluates the fitness of the solutions, and learns the fitness distribution for the solutions. For more details, see [20]) to seek a better solution. Accordingly, this method assembles uniform test forms and approximately minimizes the qualitative difference among the uniform test forms.

3.4 Maximum Set-Packing Problem for Uniform Test Form Assembly

Although previously introduced methods [18], [19], [20] showed effective performance to minimize difference of qualities among the uniform test forms, none of these methods was able to guarantee the maximum number of test forms from an item pool. These methods do not allow the item pool to be used efficiently to the greatest degree possible.

To maximize the number of uniform test forms from the item pool, Belov and Armstrong [21] proposed a uniform test assembly method based on maximum set-packing problems.

The maximum set-packing problem is an optimization problem. Let S be a finite set, $\mathbf{S} = \{s; s \subseteq S\}$, and let $|\mathbf{S}|$ be the number of elements in $\mathbf{S}$. Given those assumptions, maximum set-packing problem is defined as

$$\begin{aligned} &\textbf{maximize } |\mathbf{S}| \\ &\textbf{subject to} \\ &\qquad \forall v, w \in \mathbf{S}, v \cap w = \emptyset. \end{aligned} \tag{5}$$

In this method, a uniform test form assembly problem is represented as a kind of MSP. Letting S be a given item pool, and $\mathbf{S}$ be a set of uniform test forms, MSP for uniform test form assembly is the following:

$$\begin{aligned} &\textbf{maximize } |\mathbf{S}| \\ &\textbf{subject to} \\ &\qquad v \in \mathbf{S}: \\ &\qquad \text{test form } v \text{ satisfy all test constraints} \\ &\qquad \forall v, w \in \mathbf{S}, v \cap w = \emptyset \\ &\qquad \text{(non-overlapping constraint).} \end{aligned} \tag{6}$$

Therefore, this method guarantees the maximum number of uniform test forms with a non-overlapping condition.

3.5 Random Test Assembly Based on Linear Programming

Almost all uniform test form assemblies use LP (i.e., [18] [19], and others) because LP has several powerful solvers (i.e., [28]). However, in Belov and Armstrong's method [21], a uniform test form assembly is implemented as a kind of MSP. Nevertheless, the MSP is not tractable for practical use because it has no useful and practical solvers.

To resolve this problem, Belov proposed a random test assembly method with low probability of overlapping items [22].

This method assembles equivalent test forms as follows:

$$\begin{aligned} &\textbf{maximize} \quad \sum_{i=1}^{n} \lambda_i x_i, \\ &\textbf{where} \quad x_i = \begin{cases} 1, & \text{If } i\text{th item is selected} \\ & \text{into the test form,} \\ 0, & \text{Otherwise.} \end{cases} \end{aligned} \tag{7}$$

Therein, coordinates $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote random variables distributed uniformly on $[0, 1]$. All test constraints except the overlapping constraint are included in the constraints in Problem (7). $\lambda_i (0 \leq i \leq n)$ are resampled each Problem (7) is solved.

From Sepian's inequality, Problem (7) is guaranteed to assemble uniform test forms with low probability of overlapping items theoretically.

Using this theory, Belov proposed an algorithm (shown in Algorithm 2) for uniform test form assembly with a non-overlapping condition.

Algorithm 2 Uniform test form assembly with non-overlapping condition [22].

Require: Item pool and test constraints
Ensure: Set of non-overlapping test forms
Step 1:Set $R := \phi$.
Step 2:Set $G := \phi$.
Step 3:Solve Problem (7): if successful, the resulting test $\rightarrow \tilde{x}$; otherwise, go to Step 5.
Step 4:Add $\tilde{x}$ to the set G and withdraw the entire items in $\tilde{x}$ from the item pool; go to Step 3.
Step 5:If $|G| > |R|$; then set $R := G$.
Step 6:Return the entire items of non-overlapping tests from G to pool.
Step 7:If the number of iterations is not exceeded, then go to Step 2.
Step 8:Return R.

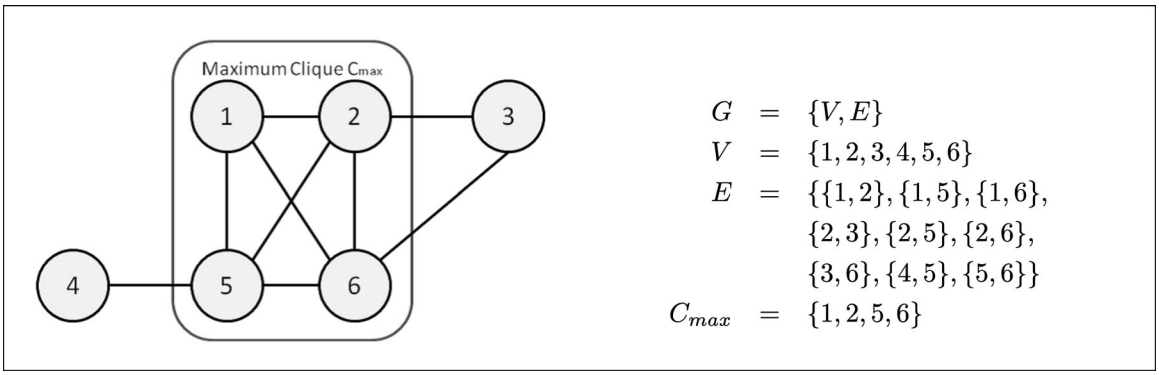


Fig. 1. Maximum clique in graph G.

To assemble uniform test forms, Algorithm 2 repeatedly assembles a test form by solving Problem (7), and seeks the uniform test forms from the combinations of assembled test forms.

Because Problem (7) assembles uniform test forms with lower overlapping items, this algorithm is expected to assemble numerous uniform test forms. Accordingly, this algorithm asymptotically maximizes the number of assembled test forms. Moreover, this method is more tractable than the previous method [21] because this method decomposes uniform test assembly into repeated LP solving.

However, this algorithm and the previous method [21] assemble uniform test forms only with non-overlapping conditions. The non-overlapping conditions strongly restrict the number of assembled test forms.

The non-overlapping conditions therefore interrupt the efficient uses of the item pool.

4 Maximum Clique Algorithm for Uniform Test Form Assembly

Traditional methods can not maximize the number of uniform test forms from the item pool in overlapping conditions. To solve this problem, we propose new methods.

4.1 Maximum Clique Problem

We apply the maximum clique algorithm to assemble the maximum number of uniform test forms. The MCA is an algorithm to solve the maximum clique problem, which is a well-known combinational optimization problem in graph theory [29], [30], [31], [32].

As described in this paper, 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 maximum clique problem seeks a special structure called *maximum clique* from a given graph. A *Clique* is a set of vertices for which each pair of vertices is connected. The maximum clique is 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 clique, then the maximum clique problem is formally defined as follows:

$$\begin{aligned} &\textbf{maximize} && |C| \\ &\textbf{subject to} && \\ &&& \forall v, w \in C, \{v, w\} \in E \\ &&& (\text{clique constraint}). \end{aligned} \tag{8}$$

Fig. 1 presents an example of the maximum clique. Graph G has six vertices V with nine edges E. The maximum clique $C_{max} = \{1, 2, 5, 6\}$ has four vertices and six edges.

4.2 Maximum Clique Algorithm for Uniform Test Form Assembly

In our study, the maximum number of uniform test forms is assembled to solve the maximum clique problem.

We assemble the following Uniform test forms:

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

Accordingly, the maximum number of uniform test form assembly can be described as the maximum clique extraction from a graph:

$$V = \left\{ \begin{array}{lll} s & : & s \in S, \text{Feasible test form } s \\ & & \text{satisfies all test constraints} \\ & & \text{excepting the overlapping} \\ & & \text{constraint from a given} \\ & & \text{item pool} \end{array} \right\},$$

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

This maximum clique problem seeks the maximum set of feasible test forms in which any two test forms satisfy the overlapping constraint. This set is the maximum uniform test forms. Therefore, this optimization problem theoretically maximizes the number of uniform test forms. Fig. 2 presents an example of uniform test form assembly using the maximum clique problem. The graph G has six feasible test forms T1-T6 with nine satisfactions of overlapping

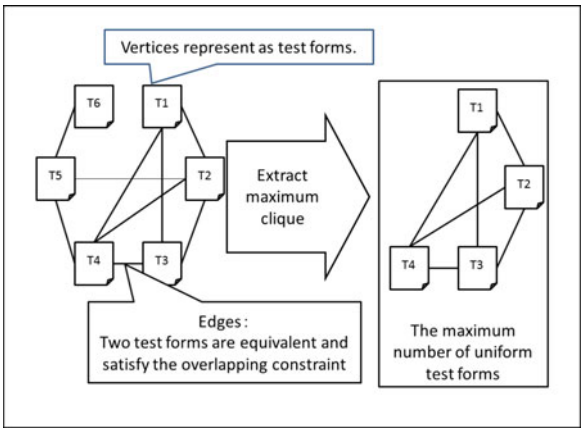


Fig. 2. MCA for uniform test assembly.

constraint and the maximum number of uniform test forms $C_{max} = \{\mathrm{T1}, \mathrm{T2}, \mathrm{T3}, \mathrm{T4}\}$.

Belov and Armstrong's method [21] is a special case of this maximum clique problem when $E = \{\{v, w\} : v \text{ and } w \text{ have no overlap items } (v \cap w = \emptyset)\}$. Therefore, our method generalizes Belov and Armstrong's method by relaxing the overlapping constraint.

4.3 Exact Solution: ExMCP

We propose a uniform test assembly algorithm; ExMCP, which exactly solves the maximum clique problem described in Section 4.2. Therefore, ExMCP theoretically maximizes the number of uniform test forms.

ExMCP consists of the following three steps:

1. (Assembling feasible test forms)
   Step 1 assembles all feasible test forms. We use branch and bound technique (e.g., [33]) to assemble the feasible test forms using test constraints except for the overlapping constraint. Finally, Step 1 stores the feasible test forms in system memory.
2. (Generating a graph which corresponds to a set of feasible test forms with overlapping items)
   Step 2 generates the corresponding graph by counting overlapping items among each pair of feasible test forms. The feasible test forms are represented as vertices and satisfactions of the overlapping constraint are represented as edges.
   Thereby, only if a pair of test forms has fewer common items than the overlapping constraint do two vertices representing the pair of test forms have an edge.
3. (Extracting the maximum clique from the graph)
   Step 3 extracts the maximum clique from the graph generated in Step 2. The extracted maximum clique represents the maximum number of uniform test forms that satisfy all test constraints including the overlapping constraint.

TABLE 1
Computational Environment

CPU	Intel(R) Core i7(R) 3930K 3.2 GHz
System Memory	64.0 GB
OS	Windows 7 SP1 64 bit

TABLE 2
Test Information Constraint for Experiment of ExMCP

Information Function (Lower Bound /Upper Bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0$	$\theta = 1.0$	$\theta = 2.0$
$0.7/1.5$	$0.8/1.6$	$0.8/1.6$	$0.8/1.6$	$0.7/1.5$

To obtain the maximum clique, we employ Nakanishi and Tomita's algorithm [31], which is the fastest exact algorithm in MCA.

ExMCP guarantees extraction of the maximum number of uniform test forms with overlapping conditions from all combinations of feasible test forms from an item pool. However, the computational time and space costs of ExMCP are $O(2^F)$ and $O(F^2)$ when F is the number of feasible test forms from an item pool. Consequently, ExMCP is not available for large item pools.

4.4 Performance of ExMCP

To demonstrate the performance of ExMCP, we conduct the following experiment. We make a comparison of the number of assembled test forms of ExMCP to those of traditional methods [18], [19], [20].

Table 1 presents details of the computational environment used for this experiment.

We use two simulated item pools. The items in the item pools have the discrimination parameter a and the difficulty parameter b based on item response theory. The items have the discrimination parameter $a = 1$. The difficulty parameter b is distributed as $b \sim N(0, 1^2)$. The simulated item pools have the total number of items $I = 20 \text{ and } 30$. (ExMCP can not accommodate large item pools because it entails high computational costs.)

We set the test constraints as follows:

1. The test includes four items.
2. The allowed numbers of overlapping items are $0, 1 \text{ and } 2$,

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

For traditional methods [18], [19], [20], we determine the target value of information function $T(\theta_k)$ as follows:

$$T(\theta_k) = \{(\text{Lower bounds of information function}) + (\text{Upper bounds of information function})\}/2.$$

(The target value $T(\theta_k)$ is an average value of the lower and upper bounds of information function.) We used CPLEX [28] for the linear programming method in Linden's method.

Table 3 shows the averages (Avg.) and the standard deviations (SD) of the assembled test forms for each method, the item pool size and the overlapping constraint (maximum number of overlap items). "OC" denotes the overlapping constraint. "EM" denotes the proposed method ExMCP, "BST" denotes Linden's method [18], "GA" denotes Sun's method[19], and "BA" denotes Songmuang's method[20].

Table 4 shows the number of times that the proposed method generates more uniform test forms than the traditional methods do. "vsBST" signifies a comparison to Linden's method [18]. "vsBA" signifies a comparison to

TABLE 3
Versus the Traditional Method: Average of the Number of Test Forms

Item Pool Size	OC	BST		GA		BA		EM	
		Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
20	0	1.56	0.80	1.89	1.04	1.91	0.99	1.98	1.07
	1	3.41	1.74	5.02	3.29	4.80	2.82	5.91	3.92
	2	4.26	1.70	18.04	13.03	15.62	9.35	27.22	20.67
30	0	2.46	0.88	3.01	1.11	3.03	0.99	3.36	1.24
	1	6.29	1.34	9.68	4.77	9.68	3.62	14.97	7.52
	2	6.95	0.50	33.03	19.72	31.19	9.86	98.63	61.00

Sun's method [19]. "vsGA" signifies a comparison to Songmuang's method [20]. "$>$" denotes the times that ExMCP assembles fewer uniform test forms than the target traditional method does. "$=$" denotes the times that ExMCP assembles as many uniform test forms as the target traditional method does. "$<$" denotes the times that ExMCP assembles a greater number of uniform test forms than the target traditional method does.

Table 3 shows no significant difference. However, Table 4 shows that in any condition, ExMCP assembles as many or more uniform test forms than the traditional methods do. Moreover, the "$<$" cases increases with item pool size or overlapping constraints.

That is to say, ExMCP increases the number of assembled test forms more efficiently than the traditional methods do. This efficiency increases with the number of feasible test forms (or the scale of test assembly).

However, ExMCP presents computational cost problems. In this experiment, the item pool sizes are limited by computational costs of ExMCP. Accordingly, the result shows that ExMCP is not available for large item pools.

These results are summarized as follows:

1. ExMCP assembles a greater number of uniform test forms than the traditional methods do.
2. The efficiency of ExMCP increases with the scale of test assembly.
3. However, ExMCP presents computational costs problems in large scale test assembly.

4.5 Approximate Solution: RndMCP

To relax the computational cost problem, we approximate ExMCP using a random search approach. This method is designated as RndMCP, which maximizes the number of uniform test forms asymptotically.

RndMCP cannot employ the usual approximation for MCA. The usual approximations require a global structure of the target graph. For example, [32], [34], [35] use each vertices degree for a given graph. The corresponding graph in the proposed method cannot use the global structure because the vertices in the corresponding graph are too numerous to store. Consequently, RndMCP approximates ExMCP without using the global structure.

Although, RndMCP consists of three steps similar to those of ExMCP, RndMCP repeats the three steps using a random search approach until it satisfies the three following constraints for computational costs:

- $\mathbf{C_1}$ is the number of feasible test forms 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.

Details of steps are the following:

1. (Assembling feasible test forms randomly)
   Step 1 randomly assembles feasible test forms. Step 1 continues this step until the number of feasible test forms reaches $\mathbf{C_1}$. Finally, Step 1 stores the feasible test forms into the system memory.
2. (Generating a graph that corresponds to a set of feasible test forms with overlapping items)
   Step 2 generates the corresponding graph by counting the overlapping items among feasible test forms similarly to ExMCP.
3. (Extracting the maximum clique)
   Although Step 3 extracts the maximum clique from the graph similarly to ExMCP, the computation time of this step is limited by $\mathbf{C_2}$.
4. (Controlling the computation time)
   Step 4 compares the current largest clique and the result of Step 3. Step 4 stores the larger clique as the largest clique. If the computation time is less than $\mathbf{C_3}$, then jump to Step 1.

The computational time cost of RndMCP is $\mathbf{C_3}$, and the space cost of RndMCP is $O({\mathbf{C_1}}^2)$. By controlling the computational time and space costs, RndMCP relaxes the computational costs problem in ExMCP.

RndMCP repeatedly extracts the maximum number of uniform test forms from subsets that are sampled randomly from all of feasible test forms. Therefore, it assembles the maximum number of uniform test forms asymptotically.

TABLE 4
Versus the Traditional Method: Comparison of ExMCP and Traditional Methods

Item Pool Size	OC	vsBST			vsGA			vsBA		
		$>$	$=$	$<$	$>$	$=$	$<$	$>$	$=$	$<$
20	0	0	59	41	0	91	9	0	93	7
	1	0	26	74	0	37	63	0	44	56
	2	0	18	82	0	17	83	0	23	77
30	0	0	25	75	0	66	34	0	69	31
	1	0	4	96	0	2	98	0	10	90
	2	0	1	99	0	0	100	0	2	98

TABLE 5
Test Information Constraint for Experiment of RndMCP (Simulated Data)

Information Function (Lower Bound /Upper Bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0$	$\theta = 1.0$	$\theta = 2.0$
0.1/0.2	0.2/0.3	0.4/0.5	0.2/0.3	0.1/0.2

TABLE 6
$\mathbf{C_2}$ Time and Number of Tests

Item Pool Size	Overlapping Constraint=0				Overlapping Constraint=1				Overlapping Constraint=2			
	5min	10min	60min	360min	5min	10min	60min	360min	5min	10min	60min	360min
70	7	7	8	8	66	66	66	66	736	736	737	737
80	9	9	9	9	100	100	100	100	1461	1461	1461	1462
90	10	10	10	10	122	122	122	122	1954	1954	1955	1955
100	10	10	10	10	131	132	132	132	2318	2318	2319	2319
110	10	10	10	10	141	141	141	141	2628	2628	2629	2629
120	11	11	11	11	152	152	152	152	2903	2903	2903	2904

Moreover, this method seeks the maximum number of uniform test forms more efficiently than the traditional random search methods [19], [20] do because the search space of RndMCP is more restrictive than that of the traditional methods. The traditional methods have $O(2^F)$ search space size, but RndMCP (and ExMCP) has $O(2^{0.19171F})$ search space because this depends on Nakanishi and Tomita's MCA [31]. This size is an upper bound of the search space size of the maximum clique algorithm and might be more restricted as MCA research progresses.

4.6 Tradeoff between Computational Costs and the Number of Assembled Test Forms

RndMCP assembles uniform test forms with the computational cost constraints. Consequently, RndMCP presents a tradeoff between the number of assembled test forms and the computational cost constraints. In this section, we elucidate this tradeoff: How much cost is sufficient for practical use? To show that, we compare the number of assembled test forms in various computational cost constraints.

RndMCP has three cost constraints: $\mathbf{C_1}, \mathbf{C_2}$, and $\mathbf{C_3}$. We first show the relation of $\mathbf{C_2}$ and the number of uniform test forms by plotting. We use six simulated item pools and a test constraint.

The item pools have the total number of items $I=70, 80, 90, 100, 110,$ and 120. The items in simulated item pools have discrimination parameter a and difficulty parameter b based on item response theory. The discrimination parameter a is distributed as $a \sim U(0,1)$. The difficulty parameter b is distributed as $b \sim N(0,1^2)$.

The test constraints are the following:

1. The test includes four items.
2. The allowed numbers of overlapping items are $0, 1$ and 2.

The information constraints are presented in Table 5.

We determined $\mathbf{C_1} = 100,000$ and $\mathbf{C_3} = \mathbf{C_2}$. This $\mathbf{C_1}$ size is the maximum allocation of the used environment. We change $\mathbf{C_2}$ between $0 \leq \mathbf{C_2} \leq 360$ min, and plot the number of assembled test forms for $\mathbf{C_2}$.

Table 6 lists the number of assembled tests for item pool size, $\mathbf{C_2}$ time constraints, and Overlapping Constraints. From Table 6, the number of assembled test forms does not appear to increase with the time constraint $\mathbf{C_2}$. This result shows that the number of assembled test forms rapidly converges to the optimal solution, and that the time constraint $\mathbf{C_2}$ does not strongly affect the number of assembled test forms.

Next, we show the relations of $\mathbf{C_1}$, $\mathbf{C_3}$ and the number of test forms. We compare the number of assembled test forms using ExMCP and RndMCP with various $\mathbf{C_1}$ and $\mathbf{C_3}$ constraints.

We use a simulated item pool ($I = 120$) and the same test constraints as those used in the preceding experiment. We use the space constraints $\mathbf{C_1}$ as $1,000, 5,000, 10,000, 50,000,$ and $100,000$, and the time constraints as $\mathbf{C_2} = 60[s]$, $\mathbf{C_3} \leq 360$ min ($= 6$ hr).

Table 7 shows the number of assembled tests for $\mathbf{C_1}$ space constraints, $\mathbf{C_3}$ time constraints, and Overlapping Constraints.

In Table 7, the "Optimal" correspond to the number of assembled test forms by ExMCP. These numbers are the maximum numbers of test forms in the respective conditions.

From Table 7, it might be readily apparent that the number of assembled test forms did not increase greatly with time constraint C_3. Moreover, the number of assembled test forms increases with $\mathbf{C_1}$, and close to the "Optimal". However, in the case of "OC" $= 2$, the number of test forms does not reach the "Optimal" level even if $\mathbf{C_1} = 100,000$. These results demonstrate that $\mathbf{C_1} = 100,000$ is insufficient for "OC"$= 2$. Moreover, the different number between RndMCP and ExMCP increases with the scale of the test assembly.

These results are summarized as shown below.

1. The number of test forms assembled by RndMCP increases with space cost $\mathbf{C_1}$, and approaches the result of ExMCP.
2. The time costs $\mathbf{C_2}$ and $\mathbf{C_3}$ do not affect the number of assembled test forms.

TABLE 7
$\mathbf{C_3}$ Time and Number of Tests

C1 Size	Overlapping Constraint=0				Overlapping Constraint=1				Overlapping Constraint=2			
	5 min	10 min	60 min	360 min	5 min	10 min	60 min	360 min	5 min	10 min	60min	360 min
1000	11	11	11	11	114	114	114	115	702	702	705	705
5000	11	11	11	11	135	135	136	136	1728	1728	1739	1740
10000	11	11	11	11	140	140	142	142	2171	2171	2177	2181
50000	11	11	11	11	150	150	150	150	2721	2727	2736	2738
100000	11	11	11	11	150	153	153	153	0	2826	2834	2839
Optimal	11				153				2917			

TABLE 8
Details of Actual Item Pool

Item Pool Size	Parameter a			Parameter b		
	Range	Mean	SD	Range	Mean	SD
87	0.15–0.67	0.35	0.134	-2.09–4.55	0.73	1.625
93	0.19–0.69	0.43	0.122	-3.92–3.61	-0.79	1.196
104	0.13–1.10	0.59	0.213	-0.18–4.55	1.50	1.188
141	0.24–1.09	0.64	0.155	-1.41–3.91	0.60	0.855
158	0.15–3.08	0.44	0.255	-4.00–4.00	-1.12	1.434
175	0.12–0.93	0.39	0.139	-2.93–3.12	-0.25	1.113
220	0.16–0.92	0.46	0.155	-4.00–2.82	-1.28	1.098
Total: 978	0.12–3.08	0.46	0.198	-4.00–4.55	-0.22	1.572

4.7 Comparison of RndMCP and ExMCP

To show the performance of RndMCP, we present the following experiment. We compare the number of assembled test forms with ExMCP, RndMCP, and the traditional methods [18], [19], [20].

We use simulated and actual item pools. The simulated item pools have a total number of items $I = 70, 80, 90, 100, 110$ and 120. We set the simulated item parameter to have similar features to actual items. The items have a discrimination parameter a distributed as $a \sim U(0, 1)$, and the difficulty parameter b distributed as $b \sim N(0, 1^2)$.

We use the actual item pools with total numbers of items $I = 87$, 93, 104, 141, 158, 175, and 220. The distributions of item parameters a and b in item pool are given in Table 8.

We set the test constraint as follows:

1. The test includes four items.
2. The allowed numbers of overlapping items are $0, 1$ and 2,

and the information constraints listed in Table 9.

These constraints are configured to increase the number of assembled test forms to the constraints ID:1 $<$ ID:2 $<$ ID:3 for the actual item pools.

These actual item pools and test constraints were used in the synthetic personality inventory (SPI) examination [39]. The SPI is a popular aptitude test in Japan. This examination has seven contents areas and total number of 28 items (four items $\times$ seven contents areas). In fact, this examination was assembled by integrating the sub-tests assembled separately from each area.

For the simulated experiments, we use the same computational environment as in the previous experiment presented in Table 1 and the time limitation of test assembly was 6 hr for all methods except for RndMCP. For RndMCP, we determine the computational cost constraint C_1 as 100,000, C_2 as 60 s, and C_3 as 1,400 s.

For the actual experiments, we use the computational environment presented in Table 10.

The time limitation of test assembly was 6 hr for all methods except for RndMCP. For RndMCP, we determine the computational cost constraint C_1 as 100,000, C_2 as 30 s, and C_3 as 6hr.

TABLE 9
Test Information Constraints for Comparison of RndMCP and Traditional Methods

ID	Information Function Lower Bound /Upper Bound				
	$\theta = -2.0$	$\theta = -1.0$	$\theta = 0$	$\theta = 1.0$	$\theta = 2.0$
1	0.1/0.2	0.2/0.3	0.4/0.5	0.2/0.3	0.1/0.2
2	0.0/0.2	0.1/0.3	0.5/0.3	0.1/0.3	0.0/0.2
3	0.0/0.4	0.1/0.5	0.7/0.3	0.1/0.5	0.0/0.4

TABLE 10
Computation Environment

CPU	Intel (R) Xeon (R) E5640 2.67 GHz
System Memory	12.0 GB
OS	Windows 7 SP1 64 bit

For traditional methods [18], [19], [20], we determined the same target value of information function $T(\theta_k)$ as the previous experiment in Section 4.4.

Table 11 shows the number of test forms assembled using the proposed methods and traditional methods from simulated item pools for the item pool sizes, the overlapping constraint (maximum number of overlap items and information constraints. In the table, "EM" denotes the proposed exact method: ExMCP, "RM" denotes the proposed approximate method: RndMCP, "BST" denotes Linden's method [18], "GA" denotes Sun's method [19], and "BA" denotes Songmuang's method [20].

In many cases, ExMCP failed the test assembly because it did not complete the calculations in 6 hr (†). Moreover, it was unable to assemble uniform test forms because the computational environment had an insufficient system (‡). In † cases, ExMCP detected a greater number of uniform test forms than any other method in a given time. In all cases, RndMCP assembled a greater number of uniform test forms than the traditional methods [18], [19], [20] did. In addition, the computational time of RndMCP is less than that of the other random search methods (e.g., [19], [20]) The computational time of RndMCP is $C_3 = 1{,}400$ s. The time limitations of the other random search methods are 6 hr. Results show that RndMCP provides more accurate results than the other random search methods do. Moreover, the different numbers of assembled test forms between the proposed method and the traditional methods increase with the number of assembled test forms (or the scale of assembly).

Table 12 shows the number of test forms assembled by RndMCP and the traditional methods from the actual item pools. Similar to the previous experiments, the number of test forms assembled by RndMCP is greater than those of the traditional methods. In all cases, RndMCP assembled greater quantities of uniform test forms than the traditional methods [18], [19], [20] did. Moreover, the different number of assembled test forms between RndMCP and the traditional methods increases with the scale of test assembly.

Figs. 3, 4, and 5 portray results obtained using the proposed method and the traditional methods from the item pool (I $= 141$). The horizontal axes show the test constraints. The vertical axes show the numbers of assembled test forms. The overlapping constraints in Figs. 3, 4, and 5 are $0, 1$ and 2.

From Figs. 3, 4, and 5, the number of assembled test forms by RndMCP are shown to increase more than those of the traditional methods as the range between the upper and lower bounds of test information increases. Accordingly, RndMCP uses an item pool more efficiently than the traditional methods do.

Actually, there is a difference in the number of high informative items among item pools. This difference causes

TABLE 11
Results for the Simulated Item Pool

Item Pool Size	OC	Constraint ID:1					Constraint ID:2					Constraint ID:3				
		BST	GA	BA	EM	RM	BST	GA	BA	EM	RM	BST	GA	BA	EM	RM
70	0	1	0	1	1	1	6	6	7	8†	7	7	7	7	8†	8
	1	2	0	1	2	2	17	26	48	66†	67	17	58	59	0‡	99
	2	3	0	2	3	3	17	66	214	736†	735	17	274	278	0‡	1767
80	0	2	1	2	2	2	7	8	8	9†	9	7	8	8	0‡	9
	1	11	2	11	12†	11	20	40	64	100†	100	20	74	78	0‡	131
	2	20	4	69	88†	88	20	82	242	1462†	1404	20	347	301	0‡	2825
90	0	2	1	2	2	2	8	7	8	10†	10	8	8	9	0‡	10
	1	13	3	11	13†	12	22	40	71	122†	119	22	83	86	0‡	156
	2	22	3	78	107†	107	22	81	251	1949†	1846	22	321	336	0‡	3634
100	0	2	1	2	2	2	8	7	8	10†	10	9	9	9	0‡	11
	1	13	3	11	12†	13	25	36	76	131†	130	25	88	87	0‡	173
	2	25	3	87	118†	118	25	80	292	2325†	2170	25	312	346	0‡	4288
110	0	2	1	2	2	2	8	8	9	10†	10	10	9	10	0‡	11
	1	13	3	11	13†	13	27	34	79	138†	137	27	86	92	0‡	195
	2	27	2	91	123†	123	27	70	308	2632†	2413	27	271	356	0‡	4938
120	0	2	2	2	2	2	9	6	9	11†	11	10	10	11	0‡	13
	1	13	2	10	13†	13	30	29	82	152†	150	30	92	102	0‡	229
	2	30	4	95	129†	127	30	68	336	2913†	2617	30	269	407	0‡	6006

†: *Maximum number of uniform test forms detected in 6 hr.*
‡: *insufficient memory problem interrupted the test construction*

the phenomenon that the item pool size does not necessarily increase the number of assembled test forms. For example, the number of assembled test forms from the item pool with 158 items is lower than that from the item pool with 104 items. In addition, even when a given test constraint is relaxed (i.e., a given test constraint changes from ID:1 to ID:2), then the number of assembled test forms from the item pool with 87 items does not increase.

Therefore, these results are summarized as follows:

1. ExMCP assembles the maximum number of uniform test forms, but it entails a computational cost problem.
2. Even when ExMCP fails a uniform test form assembly by computational cost problem, RndMCP assembles a greater number of uniform test forms than the traditional methods do. Actually, RndMCP relaxes ExMCP computational cost problems.
3. RndMCP assembled more quantities of uniform test forms in a shorter time than the other random search methods (e.g., [21], [20]) did. Results show that RndMCP provides more accurate results than the other random search methods do.
4. The differences of the number of assembled test forms between the proposed methods and traditional methods increase with the number of feasible test forms (or the scale of test assembly). For large-scale assembly, the proposed methods are more efficient than the traditional methods are.

5 RndMCP for Large Item Pools

Finally, we demonstrate the performance of RndMCP for large item pools. We compare the number of assembled test forms with RndMCP and the traditional methods

TABLE 12
Results for Actual Item Pool

Item Pool Size	OC	Constraint 1				Constraint 2				Constraint 3			
		BST	GA	BA	RM	BST	GA	BA	RM	BST	GA	BA	RM
87	0	0	0	0	0	3	3	4	4	3	3	4	4
	1	0	0	0	0	16	10	19	29	14	11	20	27
	2	0	0	0	0	21	36	139	307	21	39	140	309
93	0	0	0	0	0	4	5	5	6	5	5	5	6
	1	0	0	0	0	23	16	33	51	23	16	33	51
	2	0	0	0	0	23	43	211	658	23	54	208	721
104	0	2	2	2	2	6	5	8	10	12	15	15	18
	1	6	5	9	10	26	26	71	131	26	171	140	369
	2	26	14	83	121	26	59	275	2088	26	590	394	8442
141	0	10	3	9	10	18	19	21	27	26	31	27	35
	1	35	5	70	150	6	122	188	589	35	506	239	1014
	2	35	20	268	2307	10	185	393	11426	35	1511	386	19095
158	0	0	0	0	0	6	1	5	6	6	4	7	8
	1	0	0	0	0	22	12	24	40	39	42	75	131
	2	0	0	0	0	39	50	137	316	39	94	279	4877
175	0	2	0	2	2	6	6	7	9	6	6	8	10
	1	12	1	13	15	43	53	96	186	43	65	100	193
	2	43	2	128	234	43	102	303	7030	43	103	283	7413
220	0	2	0	2	2	7	5	8	10	9	8	10	13
	1	8	2	7	17	54	20	87	177	54	57	124	282
	2	54	8	75	136	54	44	309	5889	54	114	334	9938

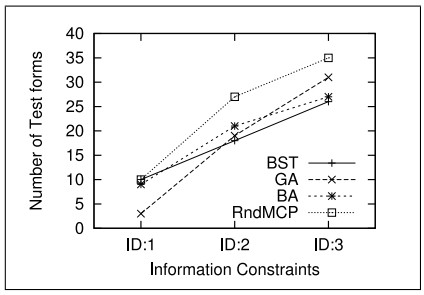


Fig. 3. Constraints and numbers of tests in overlap $= 0$ from actual item pool $I = 141$.

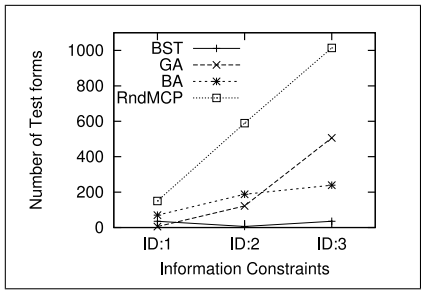


Fig. 4. Constraints and numbers of tests in overlap $= 1$ from actual item pool $\mathrm{I} = 141$.

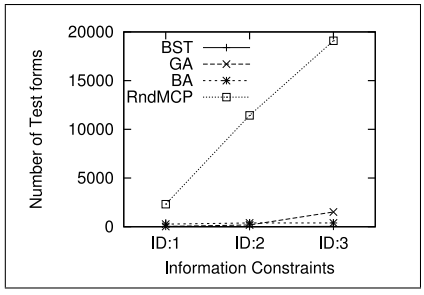


Fig. 5. Constraints and numbers of tests in overlap $= 2$ from actual item pool $\mathrm{I} = 141$.

[18], [19], [20]. We use simulated item pools and an actual item pool.

For this experiment, we use the computational environment presented in Table 1.

We use three simulated item pools with total numbers of items $I = 500, 1{,}000$, and $2{,}000$. The items in the simulated item pools have discrimination parameter $a \sim U(0, 1)$. The difficulty parameter b is distributed as $b \sim N(0, 1^2)$.

We use an actual item pool with a total number of items $I = 978$. This item pool is a summation of the actual item pools used in a previous experiment Section 4.7. Details of this item pool are listed in Table 8.

We set the test constraints as follows:

1. The test includes 25 items.
2. The numbers of allowed overlapping items are $0, 5$ and 10,

and test information constraint as Table 13, We determine these constraints according to the actual test setting[39].

For RndMCP, we determine the computational cost constraints $\mathbf{C_1} = 100,000$, $\mathbf{C_2} = 60$ s, and $\mathbf{C_3} = 24$ hr. All other assembly methods are also given 24 hr for calculation time.

TABLE 13
Test Information Constraint for Large Item Pools

Information Function (Lower Bound /Upper Bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0$	$\theta = 1.0$	$\theta = 2.0$
1.0/2.0	2.0/3.0	2.0/3.0	2.0/3.0	1.0/2.0

TABLE 14
Number of Tests from Large Item Pools

Item Pool Size	OC	Methods			
		BST	GA	BA	RM
500	0	12	3	5	10
	5	20	23	96	4380
	10	20	21	107	99983
1000	0	21	4	6	17
	5	40	17	104	46305
	10	40	19	105	100000
2000	0	53	8	12	32
	5	80	22	104	96876
	10	80	23	103	100000
978 (actual)	0	24	9	9	16
	5	39	283	371	40814
	10	39	286	381	100000

For the traditional methods [18], [19], [20], we determined the same target value of the information function $T(\theta_k)$ as the previous experiment Section 4.4.

Table 14 shows the number of test forms assembled using the RndMCP and the traditional methods for the item pool size and the overlapping constraint. With the exception of "OC=0" cases, the proposed method assembles more number of test forms than traditional methods do.

However, in the case of "OC=0", BST assembles the greatest number of tests. The reason is that the $\mathbf{C_1}$ size is too small for this test assembly setting. In the case of "OC=10", the numbers of assembled tests by RndMCP converged to 100,000. In the cases except for "OC=0", RndMCP assembles the most number of tests. Especially, the number of test forms assembled by RndMCP exponentially increases as the number of overlapping items increases. Therefore, by allowing item overlapping between assembled tests, RndMCP assembles a greater number of assembled tests than traditional methods do.

6 Conclusion

We proposed two uniform test form assembly methods, ExMCP and RndMCP, based on the maximum clique algorithm. The proposed methods exactly or asymptotically maximize the quantities of uniform test forms with an overlapping condition.

ExMCP generalizes Belov's method [21] for overlapping conditions. Furthermore, it maximizes the number of uniform test forms with overlapping conditions. However, ExMCP has computational cost problems. RndMCP approximates ExMCP using a random search approach to relax this computational cost problem. RndMCP assembles a greater number of uniform test forms than traditional methods (e.g., [18], [19], [20], [21]) do. Moreover, RndMCP provides more accurate results than other random search methods (e.g., [19], [20]) do.

To demonstrate these features, we conducted some experiments using simulated and actual data. Each experiment demonstrated that the proposed methods assemble a

greater number of uniform test forms than the traditional methods do. Moreover, the different numbers of assembled test forms between the proposed methods and the traditional methods increase with the number of feasible test forms (or the scale of test assembly). These results demonstrate that the proposed methods can assemble a greater number of uniform test forms than the traditional methods can.

In simulated experiments, more cases exist in which ExMCP cannot assemble uniform test forms because of computational cost problems. However in those cases, RndMCP assembles a greater number of uniform test forms than the traditional methods do. This result shows that RndMCP relaxes the computational cost problems of ExMCP.

In simulated experiments, the computational time of RndMCP is less than that of the other random search methods (e.g., [19], [20]). In actual experiments, RndMCP assembles a greater number of test forms than the traditional methods do, given equal time limitations. Therefore, RndMCP provides more accurate results than other random search methods (e.g., [19], [20]) do.

Results underscore the salient benefits of using the proposed methods.

However, in these experiments, we did not use a content constraint (such as the number of algebra items, geometry items and numbers and operations). Moreover, we employ only a 2PL-IRT model, but there are various IRT models such as 1 and 3 PL models. Wainer reports that those situations might produce unpredictable behavior [40]. Therefore, one course of future work is to evaluate the proposed method in those situations.

Furthermore, Wainer observed that the distribution of item exposure (use counts in assembled tests) followed Zip's law [41]. In other words, the same item is selected on nearly every form. Our methods also might suffer the bias of item exposure. Therefore, an important future challenge is to resolve the item exposure problem.

REFERENCES

[1] F.M. Lord, *Applications of Item Response Theory to Practical Testing Problems*. First ed., Routledge, July 1980.

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

[3] W.J. van der Linden and E. Boekkooi-Timminga, "*A Zero-One Programming Approach to Gulliksen's Matched Random Subtest Method*," Series Project Psychometrische Aspecten Van Item Banking, Dept. of Education, Univ. of Twente, 1986.

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

[5] E. Boekkooi-Timminga, "Simultaneous Test Construction by Zero-One Programming," *Methodika*, vol. 1, pp. 101-112, 1987.

[6] F.B. Baker, A.S. Cohen, and B.R. Barmish, "Item Characteristics of Tests Constructed by Linear Programming," *Applied Psychological Measurement*, vol. 12, no. 2, pp. 189-199, 1988.

[7] J.J. Ameda and W.J. van der Linden, "Algorithms for Computerized Test Construction Using Classical Item Parameters," *J. Educational Statistics*, vol. 14, pp. 279-290, 1989.

[8] T.A. Ackerman, "An Alternative Methodology for Creating Parallel Test Forms Using the IRT Information Function," *Proc. Ann. Meeting of the Nat'l Council on Measurement in Education*, Mar. 1989.

[9] J.J. Ameda, "Models and Algorithms for the Construction of Achievement Tests," PhD Dissertation, Univ. of Twente, 1990.

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

[11] J.J. Adema, "Methods and Models for the Construction of Weakly Parallel Tests," *Applied Psychological Measurement*, vol. 16, no. 1, pp. 53-63, 1992.

[12] L. Swanson and M.L. Stocking, "A Model and Heuristic for Solving Very Large Item Selection Problems," *Applied Psychological Measurement*, vol. 17, no. 2, pp. 151-166, 1993.

[13] H. Jeng and S. Shih, "A Comparison of Pair-Wise and Group Selections of Items Using Simulated Annealing in Automated Construction of Parallel Tests," *Psychological Testing*, vol. 44, no. 2, pp. 195-210, 1997.

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

[15] W.J. van der Linden and J.J. Adema, "Simultaneous Assembly of Multiple Test Forms," *J. Educational Measurement*, vol. 35, no. 3, pp. 185-198, Sept. 1998.

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

[17] P. Songmuang and M. Ueno, "Development of Practice of and Integrative E-Testing System," *Japanese Test Soc.*, vol. 4, no. 1, pp. 53-64, 2008.

[18] W.J. van der Linden, *Liner Models for Optimal Test Design*. Springer, 2005.

[19] K.-T. Sun, Y.-J. Chen, S.-Y. Tsai, and C.-F. Cheng, "Creating IRT-Based Parallel Test Forms Using the Genetic Algorithm Method," *Applied Measurement in Education*, vol. 2, no. 21, pp. 141-161, 2008.

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

[21] D.I. Belov and R.D. Armstrong, "A Constraint Programming Approach to Extract the Maximum Number of Non-Overlapping Test Forms," *Computational Optimization and Applications*, vol. 33, pp. 319-332, 2006.

[22] D.I. Belov, "Uniform Test Assembly," *Psychometrika*, vol. 73, no. 1, pp. 21-38, 2008.

[23] F.J. Solis and R.J.-B. Wets, "Minimization by Random Search Techniques," *Math. of Operations Research*, vol. 6, no. 1, pp. 19-30, 1981.

[24] W.J. van der Linden and E. Boekkooi-Timminga, "A Maximin Model for IRT-Based Test Design with Practical Constraints," *Psychometrika*, vol. 54, no. 2, pp. 237-247, June 1989.

[25] E. Boekkooi-Timminga, "The Construction of Parallel Tests from IRT-Based Item Banks," *J. Educational Statistics*, vol. 15, pp. 129-145, 1990.

[26] R.D. Armstrong, D.H. Jones, and Z. Wang, "Automated Parallel Test Construction Using Classical Test Theory," *J. Educational Statistics*, vol. 19, no. 1, pp. 73-90, 1994.

[27] R.D. Armstrong, D.H. Jones, and C.S. Kunce, "IRT Test Assembly Using Network-Flow Programming," *Applied Psychological Measurement*, vol. 22, no. 3, pp. 237-247, 1998.

[28] ILOG, ILOG CPLEX *User's Manual 11.0*, 2007.

[29] R.M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations*, vol. 40, no. 4, pp. 85-103, 1972.

[30] M.R. Garey and D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1990.

[31] H. Nakanishi and E. Tomita, "An $o(2^{0.19171n})$-Time and Polynomial-Space Algorithm for Finding a Maximum Clique," *The Special Interest Group Technical Reports of IPSJ*, vol. 2008, no. 6, pp. 15-22, 2008.

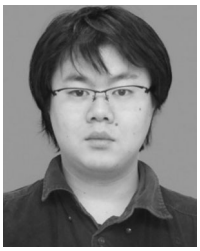
[32] X. Geng, J. Xu, J. Xiao, and L. Pan, "A Simple Simulated Annealing Algorithm for the Maximum Clique Problem.," *Information Sciences*, vol. 177, no. 22, pp. 5064-5071, 2007.

[33] J.J. Ameda, "Implementations of the Branch-and-Bound Method for Test Construction Problems," *Methodika*, vol. 6, pp. 99-117, 1992.

[34] Q. Zhang, J. Sun, and E. Tsang, "An Evolutionary Algorithm with Guided Mutation for the Maximum Clique Problem," *IEEE Trans. Evolutionary Computation*, vol. 9, no. 2, pp. 192-200, Apr. 2005.

[35] S. Balaji, V. Swaminathan, and K. Kannan, "A Simple Algorithm to Optimize Maximum Independent Set," *Advanced Modeling and Optimization*, vol. 12, no. 1, pp. 107-118, 2010.

[36] K. Katayama, A. Hamamoto, and H. Narihisa, "An Effective Local Search for the Maximum Clique Problem," *Information Processing Letters*, vol. 95, pp. 503-511, http://dx.doi.org/10.1016/j.ipl.2005.05.010, Sept. 2005.
[37] A. Singh and A.K. Gupta, "A Hybrid Heuristic for the Maximum Clique Problem," *J. Heuristics*, vol. 12, pp. 5-22, Jan. 2006.
[38] S. Balaji, V. Swaminathan, and K. Kannan, "A Simple Algorithm for Maximum Clique and Matching Protein Structures," *Int'l J. Combinatorial Optimization Problems and Informatics*, vol. 1, no. 2, pp. 2-11, 2010.
[39] Recruit, *Synthetic Personality Inventory (SPI)*, http://www.spi.recruit.co.jp/, 2014.
[40] H. Wainer, "Rescuing Computerized Testing by Breaking Zipf's Law," *J. Educational and Behavioral Statistics*, vol. 25, pp. 203-224, 2000.
[41] H. Wainer and Educational Testing Service, *CATS: Whither and Whence*. Educational Testing Service, 2000.
[42] T. Ishii, P. Songmuang, and M. Ueno, "A Method to Extract the Maximum Number of Test Forms Using Maxclique," *Proc. 23rd Ann. Conf. Japanese Soc. for Artificial Intelligence*, 2009.
[43] T. Ishii, P. Songmuang, and M. Ueno, "Maximum Clique Algorithm for Uniform Test Forms," *Proc. 16th Int'l Conf. Artificial Intelligence in Education*, 2013.



Takatoshi Ishii received the BEng and MEng degrees from the University of Electro-Communications in 2008 and 2011, respectively. He is currently a student of the doctor course at the University of Electro-Communications. His research interests include e-testing, data mining, and computer science.

Pokpong Songmuang received the BEng degree from Thammasat University in 2003, the MEng degree from the Nagaoka University of Technology in 2006, and the PhD degree in computer science from the University of Electro-Communications in 2010. He has been a lecturer at Thammasat University. His research interests include e-testing, data mining, and web technologies.

Maomi Ueno received the PhD degree in computer science from the Tokyo Institute of Technology in 1994. He has been a professor of the Graduate School of Information Systems at the University of Electro-Communications since 2013. He has also worked at the Tokyo Institute of Technology (1994-1996), Chiba University (1996-2000), and the Nagaoka University of Technology (2000-2007). He received best paper awards from the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), ED-MEDIA 2008, e-Learn 2004, e-Learn 2005, and e-Learn 2007. His research interests include e-learning, e-testing, e-portfolio, machine learning, data mining, Bayesian statistics, Bayesian networks, and so on. He is a member of the IEEE.