Hybrid Maximum Clique Algorithm Using Parallel Integer Programming for Uniform Test Assembly

Kazuma Fuchimoto, Takatoshi Ishii, and Maomi Ueno, *Member, IEEE*

***Abstract*—Educational assessments often require uniform test forms, for which each test form has equivalent measurement accuracy but with a different set of items. For uniform test assembly, an important issue is the increase of the number of assembled uniform tests. Although many automatic uniform test assembly methods exist, the maximum clique algorithm (MCA)-based method is known to assemble the greatest number of uniform tests with the highest measurement accuracy based on the item response theory. In that method, the graph is constructed by sequentially adding a randomly formed test as a vertex without considering the graph structure. However, an important difficulty is its high space complexity, which interrupts search cliques with more than a hundred thousand vertices. To overcome this difficulty, this article proposes a new uniform test assembly algorithm: hybrid maximum clique algorithm using parallel integer programming. The first step searches a maximum clique that is as large as possible up to computer memory limitations using a state-of-the-art MCA with low time complexity but with high space complexity. The second step repeatedly searches a vertex connected with all vertices of the current maximum clique from the remaining vertices using integer programming with low space complexity but with high time complexity. The proposed method constructs a larger number of tests than the traditional methods do. Finally, we use simulation and actual data experiments to demonstrate the effectiveness of the proposed method. Results show that our method assembles a 1.5–2.7 times greater number of uniform tests than traditional methods can.**

***Index Terms*—E-testing, integer programming (IP), item response theory (IRT), maximum clique problem (MCP), uniform test assembly.**

I. Introduction

RECENTLY, automatic assemblies of *uniform test forms* (also called *parallel test forms*), for which each form has equivalent measurement accuracy but with a different set of items, have become popular. For example, uniform test forms are necessary when a testing organization administers tests in different time slots. To achieve this, uniform test forms are assembled, in which all forms have equivalent qualities so that examinees who have used different test forms can be evaluated objectively using the same scale. Even if different examinees with the same ability take different tests, their test scores should be guaranteed to be equivalent.

The most important task of uniform test assembly is how to increase the number of assembled tests to as great a degree as possible. Test organizations allocate different test forms to each examinee for securing the contents of the items. Therefore, the number of test forms should be greater than the number of examinees. For example, almost public examinations in Japan need more than a hundred thousand forms because more than 100 000 examinees take each of them every year.

Many automatic uniform test assembly methods have been proposed for the purpose. They construct test forms to satisfy given test constraints such as the number of test items from the respective question areas, average test scores, and ability measurement accuracy to provide equivalent test qualities (see, e.g., [1]–[24]).

Earlier studies assessed the formalization of a test assembly as a combinational optimization problem. The test construction is searching for a combination of items that satisfies given test constraints from a given item pool.

To assemble uniform tests, van der Linden [25] proposed the big shadow test method (BST) using integer programming (IP). This method sequentially assembles uniform test forms by minimizing qualitative differences between a current assembling 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 entails two crucially important difficulties. First, the ability measurement accuracy decreases with the assembled order of test forms. Second, this method does not maximize the number of uniform test forms from the item pool.

To equilibrate the ability measurement accuracy, van der Linden and Ameda [15], Sun *et al.* [18], Songmuang and Ueno [19], Boekooi-Timminga [26], Armstrong *et al.* [27], Armstrong *et al.* [28], Chang and Shiu [29], and Pereira and Vila [30] formulated an assembly of uniform tests as a large-scale IP that directly minimizes the differences of the ability measurement accuracies among tests. However, these optimizations have extremely large computational costs.

Sun *et al.* [18] proposed the use of a genetic algorithm for uniform test assembly that simultaneously assembles uniform test forms as minimizing differences among the qualities of

Manuscript received November 19, 2020; revised March 16, 2022; accepted March 22, 2022. Date of publication March 30, 2022; date of current version June 20, 2022. This work was supported by the Japan Society for the Promotion of Science Grants-in-Aid for Scientific Research under Grant JP19H05663 and Grant JP19K21751. Parts of this research were reported previously in an earlier conference paper published in DOI: 10.1007/978-3-319-61425-0_9. *(Corresponding author: Kazuma Fuchimoto.)*

Kazuma Fuchimoto and Maomi Ueno are with the University of Electro-Communications, Tokyo 182-8585, Japan (e-mail: fuchimoto@ai.lab.uec.ac.jp; ueno@ai.is.uec.ac.jp).

Takatoshi Ishii is with the Sundai Institute of AI for Education, SATT Company Ltd., Tokyo 101-0061, Japan (e-mail: t_ishii@satt.jp).

Digital Object Identifier 10.1109/TLT.2022.3163360

assembled test forms and user-determined values. Songmuang and Ueno [19] applied the Bees algorithm to uniform test form assembly and thereby improved the performance of the method reported by Sun *et al.* [18].

Chang and Shiu [29] reformulated optimization as a bin-packing problem and proposed an approximate algorithm based on variable neighborhood search heuristic.

Although these methods demonstrated effective performance for minimizing qualitative differences among the assembled test forms, they are not guaranteed to maximize the number of uniform test forms from the item pool.

To maximize the number of test forms, Belov and Armstrong [31] proposed a uniform test assembly method based on maximum set-packing problems. This method divides the item pool into the maximum number of item sets (as tests) that match given test constraints. Nevertheless, this method [31] cannot assemble uniform test forms with overlapping items, where overlapping items mean common items among multiple test forms. In the nonoverlapping conditions, each item is used only once for assembled test forms. Therefore, the nonoverlapping condition strongly restricts the number of assembled test forms.

To resolve this difficulty, Ishii *et al.* [20], [21] formalized the uniform test assembly with overlapping conditions as a maximum clique problem (MCP), which is a combinational optimization in the graph theory. The method constructs a graph in which the vertices and the edges represent tests satisfying the test constraint and the satisfaction of overlapping constraints, hereinafter called the graph for uniform test assembly as "*corresponding graph*." The method also extracts the maximum clique as uniform test forms from the corresponding graph. However, the number of possible constructed test forms (the graph size) superexponentially increases as the number of overlapping items increases. Furthermore, this method superexponentially increases the computational time with increasing item pool size because the MCP is NP-complete. Therefore, it is difficult to extract the maximum clique from the graph for a large item pool with overlapping constraints because the corresponding graph size becomes too large to store in memory.

To relax these difficulties, they proposed an approximation using a random search approach (random maximum clique problem (RndMCP) algorithm[20], [21]). This method repeatedly constructs graphs by sequentially and randomly assembling as many tests (as vertices) as possible. The method also extracts the maximum clique from those graphs. In other words, this method repeatedly samples a random subgraph from the global corresponding graph and extracts a maximum clique from the subgraph. This approximation allows the RndMCP, which is to assemble 10–1000 times the number of uniform tests than traditional methods do.

However, the RndMCP still has high computational costs. It cannot assemble a sufficient number of tests for large-scale testing. Actually, the RndMCP employs a maximum clique algorithm (MCA). The time complexity of MCAs has been improved up to $\mathrm{O}(2^{0.19171|V|})$ using state-of-the-art methods [32], where V represents the vertices of the subgraph. Nevertheless, the salient difficulty is their space complexity $\mathrm{O}(|V^2|)$. It costs only polynomial order but disables the searching of cliques with more than a hundred thousand vertices because of computer memory limitations. Therefore, it limits the number of assembled uniform tests to a hundred thousand.

For this article, to reduce the high time complexity of the RndMCP, we propose a new algorithm, i.e., random integer programming maximum clique problem (RIPMCP), which seeks a vertex connected with all vertices of the current clique using IP. This method has lower space complexity $\mathrm{O}(|V|)$ than the RndMCP's space complexity $\mathrm{O}(|V|^2)$. However, it has higher time complexity $\mathrm{O}(|V| \cdot 2^n)$ (where n represents the item pool size) than RndMCP's time complexity of $2^{0.19171|V|}$. The advantage of space complexity increases the number of assembled uniform tests for limited memory. However, the improvement of the RIPMCP is expected to be limited because of the high time complexity $\mathrm{O}(|V| \cdot 2^n)$.

To relax the high time complexity of the RIPMCP, we propose a new two-step parallel algorithm: hybrid maximum clique algorithm with parallel integer programming (HMCAPIP). The first step seeks a maximum clique that is as large as possible up to the computer memory limit using the RndMCP with low constant time, but with high space complexity. The second step repeatedly seeks a vertex connected with all vertices of the current clique from the remaining vertices using IP with low space complexity but with high time complexity. However, the second step using IP has higher time complexity than the first step has. To relax the difficulty, our method parallelizes the second step. Specifically, to parallelize the second step efficiently, the main idea is to repeat the parallel search of the vertices connected with all vertices of the current clique using IP up to a determined number. Subsequently, the second step seeks a maximum clique from the found vertices. The found maximum clique combines with the current clique. Consequently, the HMCAPIP can assemble a greater number of uniform tests than the RIPMCP can. It does so by dividing the computational cost.

Finally, we demonstrate the performance of the proposed method using simulated and actual data. Results show that our method assembles a 1.5–2.7 times greater number of uniform tests than traditional methods can.

II. Item Response Theory

Most of the earlier studies of test form assembly employ item response theory (IRT) [33], [34] to evaluate the measurement accuracies of test forms (such as [19], [25]–[28], and [35]).

In fact, even when the examinees are using different test forms, IRT that describes the relation between item characteristics and examinee ability can measure examinee ability on the same scale. For IRT, u_{ij} denotes the response of item $i(=1,\ldots,n)$ on examinee $j(=1,\ldots,m)$ as

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

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

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

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

$$\mathrm{I}_i(\theta) = 1.7^2 a_i^2 p_i(\theta)(1 - p_i(\theta)). \quad (2)$$

This function is designated as Fisher information.

The test information function $\mathrm{I}_{\mathrm{Test}}(\theta)$ of a test form Test is defined as

$$\mathrm{I}_{\mathit{Test}}(\theta_j) = \sum_{i \in \mathit{Test}} \mathrm{I}_i(\theta_j). \quad (3)$$

The asymptotic standard error of estimating $\hat{\theta}$: $\mathrm{SE}(\hat{\theta})$ is the reciprocal of square root of the item/test information function at a given ability level $\hat{\theta}$

$$\mathrm{SE}_i(\theta) = \frac{1}{\sqrt{I_i(\theta)}} \quad (4)$$

$$\mathrm{SE}_{\mathit{Test}}(\theta) = \frac{1}{\sqrt{I_{\mathit{Test}}(\theta)}}. \quad (5)$$

Therefore, using the information function, a test administrator can estimate how much accuracy a test form has.

The test information function is a continuous function of the examinee ability and the item characteristic parameters. In traditional methods (see, e.g., [19], [25]–[28], and [35]), the function is treated as discretely to simplify the calculation. Regarded in greater detail, the test information values are treated discretely. They have been evaluated on some points $\Theta = \{\theta_1, \ldots, \theta_k, \ldots, \theta_K\}$ on the ability level θ. As described in this article, we treat the test information function similarly.

III. Traditional Methods of Uniform Test Assembly

This section introduces several conventional uniform test assembly methods.

A. Big Shadow Test Method

The most well-known uniform test assembly method is the *BST* using IP by van der Linden [25]. This method assembles test forms sequentially by minimizing the difference of test information functions between a current assembled test and a set of items remaining in the item pool. The set of remaining items is called the *shadow test*. This method solves the following optimization problem.

variables

$$y \geq 0$$

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

$$z_i = \begin{cases} 1, & \text{if the } i\text{-th item is selected} \\ & \text{into shadow test form} \\ 0, & \text{otherwise} \end{cases}$$

minimize

$$y$$

subject to

$$\sum_{k=1}^{K} |\sum_{i=1}^{n} \mathrm{I}_i(\theta_k)x_i - \mathrm{T}(\theta_k)| \leq \mathrm{M}y, \quad (6)$$

$$\sum_{k=1}^{K} |\sum_{i=1}^{n} \mathrm{I}_i(\theta_k)z_i - \mathrm{T}_{\mathit{ST}}(\theta_k)| \leq \mathrm{M}_{\mathit{ST}}y, \quad (7)$$

$$\sum_{i=1}^{n} x_i = \mathrm{M}, \quad (8)$$

$$\sum_{i=1}^{n} z_i = \mathrm{M}_{\mathit{ST}} \quad (9)$$

$$x_i + z_i \leq 1 \quad (i = 1, \ldots, n) \quad (10)$$

where

$$\mathrm{T}_{\mathit{ST}}(\theta_k) = \frac{M_{\mathit{ST}}}{M}\mathrm{T}(\theta_k). \quad (11)$$

M and M_{ST} are the number of items in the assembling test and the shadow test, respectively. $T(\theta_k)$ denotes a target value of the information function at the ability level θ_k for the assembling test. $\mathrm{T}_{\mathit{ST}}(\theta_k)$ denotes a target value of the information function at the ability level θ_k for the shadow test. The test quality constraints without the information function (ex: test time limits) can be included in the constraints of the IP.

Actually, variable y represents the minimum difference between the information function of the assembling test and the target value $\mathrm{T}(\theta_k)$ and the difference between information functions of the shadow test and the target value $\mathrm{T}_{\mathit{ST}}(\theta_k)$ simultaneously.

Solving the IP assembles a test form one by one to assemble uniform tests. This greedy algorithm reduces computational costs, but it decreases the ability measurement accuracy (test information) as the number of assembled test forms increases.

Although the proposed method in this article also uses the IP, it is formulated to maximize the number of test forms to satisfy test constraints. Therefore, the proposed method can guarantee the uniformity of the ability measurement accuracies for all the test forms.

B. MCA for Assembling Uniform Tests

As described previously, the MCA [20], [21] based method is known to assemble the greatest number of uniform tests

with the highest measurement accuracy. The clique problem is a combinational optimization problem in graph theory.

Letting V be a finite set of vertices and letting E represent a set of edges, the graph is represented as a pair $G = \{V, E\}$.

The MCP searches the clique with the maximum number of vertices in the given graph. Letting $G = \{V, E\}$ be a finite graph and letting $C \subseteq V$ be a clique, the MCP is formally defined as follows:

$$\begin{aligned} &\text{maximize} && |C| \\ &\textit{subject to} && \\ &&& \forall v\, \forall w \in C, \{v, w\} \in E \\ &&& (\text{clique constraint}). \end{aligned} \tag{12}$$

Ishii *et al.* [20], [21] employ the MCP to search the maximum number of uniform test forms. In general, uniform test forms are defined as a set of test forms that has the following specifications.

1) Any test in uniform tests satisfies all test constraints.
2) Any two tests in uniform tests comprise a different set of items (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 as shown in the following:

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

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

This graph $G = \{V, E\}$ is designated as the *corresponding graph*. 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 an upper bound for the test information function on $\mathrm{I}_{\mathit{Test}}(\theta_k)$, a constraint for the test information function is written as follows:

$$L_{\theta_k} \leq \mathrm{I}_{\mathit{Test}}(\theta_k) \leq U_{\theta_k}. \tag{13}$$

Letting OC be the allowed number in the overlapping constraint and realizing that both s and s' are tests, which are the sets of items, the overlapping constraint is defined as follows:

$$\forall s, \forall s' \in V \tag{14}$$

$$|s \cap s'| \leq \mathrm{OC}. \tag{15}$$

This MCP seeks the maximum set of feasible test forms, in which any two test forms satisfy the overlapping constraint. Therefore, this optimization problem theoretically maximizes the number of uniform test forms.

Fig. 1 presents an example of the uniform test assembly using the MCP. The graph has six vertices (feasible tests) T1–T6 and nine edges (equivalent and satisfaction of overlapping constraint). In this graph, the maximum clique (the maximum uniform test) is $C_{\max} = \{\mathrm{T1}, \mathrm{T2}, \mathrm{T3}, \mathrm{T4}\}$.

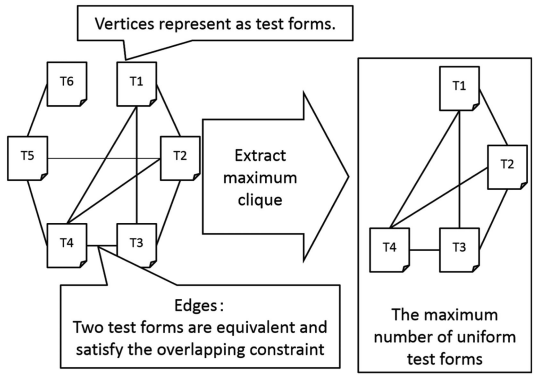


Fig. 1. MCA for uniform test assembly.

This method is applied as follows. An exact method based on maximum clique. problem (ExMCP) consists of the following three steps:

Step 1— *(Assembling feasible test forms):* Step 1 assembles all feasible test forms. They use the branch and bound technique (see, e.g., [36]) to assemble the feasible test forms using test constraints except for the overlapping constraint. Specifically, it prunes edges that exceed the test length constraint or the upper bound of the test information constraints. Finally, Step 1 stores the feasible test forms in system memory.

Step 2— *(Generating corresponding graph):* Step 2 generates the corresponding graph by counting overlapping items among each pair of the feasible test forms constructed by Step 1. The feasible test forms are represented as vertices. Thereby, only if a pair of test forms has fewer common items than the overlapping constraint can one connect the pair.

Step 3— *(Extracting the maximum clique from the graph):* Step 3 extracts the maximum clique from the corresponding graph generated in Step 2. Step 3 returns the result maximum clique as the maximum number of uniform test forms.

The ExMCP guarantees the extraction of the maximum number of uniform test forms with overlapping conditions from all the combinations of feasible test forms from an item pool. However, even when we employ state-of-the-art MCAs (see, e.g., [32] and [37]), the computational time and space costs of the ExMCP are $O(2^F)$ and $O(|F|^2)$, respectively, where F represents the number of feasible test forms. Moreover, the number of feasible test forms F combinatorially increases concomitantly with increasing item pool size. From the explanation represented above, the ExMCP can be understood to require superexponential computational costs. Consequently, the ExMCP is not available for large item pools. In addition, most recently, state-of-the-art MCAs [38], [39] have been proposed. They are efficient only when a graph structure is sparse. However, unfortunately, uniform test assembly problems address dense graphs [20], [21].

Unfortunately, this difficulty for assembling uniform tests cannot ensure a maximum number of assembled uniform tests because the ExMCP has high computational complexity. To resolve this difficulty, an approximate algorithm called RndMCP was proposed in an earlier study [20], [21].

This method has three parameters for computational costs.

$\boldsymbol{L_1}$ is the number of feasible tests assembled in Step 1.

$\boldsymbol{L_2}$ is the time limit of Step 3.

$\boldsymbol{CT}$ is the total calculation time limit of the uniform test assembly.

The RndMCP algorithm is shown as Algorithm 1.

Actually, the RndMCP repeatedly extracts the maximum number of uniform tests from a subgraph of the corresponding global graph. For the case in which L_1 is larger than the size of the maximum clique in the corresponding global graph, the RndMCP asymptotically extracts the maximum clique as the maximum number of uniform tests from the corresponding global graph when the number of subgraphs becomes large [21].

The algorithm has a calculation time limit CT and space complexity $\mathrm{O}(L_1^2)$. Therefore, it can extract uniform tests in a limited computing environment by controlling time complexity and space complexity.

However, when the algorithm assembles $|C|$ uniform tests, its computational cost is at least $\mathrm{O}(|C|^2)$. Therefore, when the number of assembled uniform tests becomes too large, the space complexity disables the test assembly because of computer memory limitations. Actually, it is difficult for the algorithm to assemble more than a hundred thousand uniform tests. Consequently, the RndMCP algorithm has a space complexity problem.

Algorithm 1: RndMCP.

procedure RndMCP(L_1, L_2, CT)
  $C \leftarrow \emptyset, C_{\max} \leftarrow \emptyset$
  $ST \leftarrow current\ time$
  while $(current\ time - ST) < CT$ **do**
    /* Step1 */
    $V \leftarrow$ Assemble feasible L_1 tests randomly.
    /* Step2 */
    $G \leftarrow (V, E)$ Generate a graph that corresponds to a set of feasible tests with overlapping items.
    /* Step3 */
    $C \leftarrow \mathrm{MCA}(G, L_2)$
  $\triangleright$ MCA(G, L_2) extracts the maximum clique from the graph G within calculation time L_2 using Nakanishi and Tomita's maximum clique algorithm [40].
    if $|C_{\max}| < |C|$ **then**
      $C_{\max} \leftarrow C$
    end if
  end while
  Output $C_{\max}$
end procedure

IV. Uniform Test Assembly Using the MCP and Integer Programming

Currently, the RndMCP is known to assemble the greatest number of uniform tests. Nevertheless, the RndMCP is limited to a maximum of 100 000 assembled uniform tests because of the high space complexity $\mathrm{O}(|V|^2)$. To reduce the RndMCP space complexity, we propose a new algorithm: RIPMCP. The main idea is that this method seeks a vertex connected with all vertices of the current clique using the IP. This method repeats the search to expand the maximum clique. The IP problem for assembling uniform tests is shown below. In fact, the IP with no overlapping condition is equivalent to that of Belov's algorithm (2008) [41].

where

$$x_i = \begin{cases} 1, & \text{if the } i\text{th item is selected in the feasible test} \\ 0, & \text{otherwise} \end{cases}$$

maximize

$$\sum_{i=1}^{n} \lambda_i x_i \tag{16}$$

subject to

$$\sum_{i=1}^{n} x_i = \mathrm{M}\ (\text{test length}) \tag{17}$$

$$L_{\theta_k} \le \sum_{i=1}^{n} I_i(\theta_k) x_i \le U_{\theta_k} (\text{test information}) \quad (k = 1, \ldots, K) \tag{18}$$

$$\sum_{i=1}^{n} X_{i,r} x_i \le \mathrm{OC}\ (\text{overlapping constraint}) \quad (r = 1, \ldots, |C|) \tag{19}$$

$$X_{i,r} = \begin{cases} 1, & \text{if the } i\text{th item is selected in the } r\text{th test} \\ & \text{in the current clique } C, \\ 0, & \text{otherwise.} \end{cases}$$

Therein, $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively, denote random variables distributed uniformly on [0,1]. Here, $\{\lambda_i\}(0 \le i \le n)$ are resampled after each problem is solved.

The IP seeks the random tests that are connected to all vertices in a set of already constructed vertices C. The left-hand side of (17) denotes the number of items in assembling test form. Therefore, (17) restricts the total number of items in the assembling test. The term of $\sum_{i=1}^{n} I_i(\theta_k) x_i$ in (18) describes the test information function of the assembling test on the point of θ_k. Consequently, (18) restricts the test information function by setting the upper bound U_{θ_k} and the lower bound L_{θ_k}. The term $\sum_{i=1}^{n} X_{i,r} x_i$ calculates the number of overlapping items between the assembling test x_i and the rth test in C. Consequently, (19) limits the number of overlapping items between any two tests in C. As a result, those constraints (17)–(19) guarantee that an assembling vertex (test) is connected to all vertices (tests) in C.

Algorithm 2: RIPMCP.

procedure RIPMCP(L_1, L_2, CT)
  $C \leftarrow \emptyset, C_{\max} \leftarrow \emptyset$
  $global\ ST \leftarrow current\ time$
  while $(current\ time - ST) < CT$ **do**
    $V \leftarrow \emptyset$
    while $|V| < L_1$ **do**
      $v \leftarrow \mathrm{SolveIP}(C)$
    ▷ Seek a vertex connected with all vertices of the current clique C using the IP.
      if $v \neq \emptyset$ **then**
        $V \leftarrow V \cup v$
      else
        break
      end if
    end while
    if $V \neq \emptyset$ **then**
      $G \leftarrow (V, E)$
      ▷ Generate a graph that corresponds to a set V with overlapping items.
      $C \leftarrow C \cup \mathrm{MCA}(G, L_2)$
  ▷ MCA(G, L_2) extracts the maximum clique from the graph G within calculation time L_2 using Nakanishi and Tomita's maximum clique algorithm [40].
      if $|C_{\max}| < |C|$ **then**
        $C_{\max} \leftarrow C$
      end if
    else
      $C \leftarrow \emptyset$
    end if
  end while
  Output $C_{\max}$
end procedure

This method generates graph structures for the MCP by solving the IP. From this graph generation, this method can assemble uniform tests with lower space complexity $\mathrm{O}(|V|)$ than RndMCP's space complexity $\mathrm{O}(|V|^2)$. However, this method unfortunately has high time complexity $\mathrm{O}(|V| \cdot 2^n)$.

The RIPMCP has the following parameters.

- $\boldsymbol{L_1}$ is the limit of number of vertices using IP.
- $\boldsymbol{L_2}$ is the time limit of the maximum clique extraction of RIPMCP.
- $\boldsymbol{CT}$ is the total calculation time limit of the test assembly.

The RIPMCP algorithm is shown in Algorithm 2.

V. Experiments to Assess RIPMCP

In this section, to demonstrate the advantages of the RIPMCP, we compare the number of assembled uniform tests of our method with those of traditional methods (BST [41] in Section III-A and RndMCP [21] in Section III-B) using simulated and actual item pools. Items in the simulated item pools have discrimination parameters and difficulty parameters of IRT. We generated discrimination parameters as $\log_2 a \sim N(0, 1^2)$ and generated difficulty parameters as $b \sim N(0, 1^2)$. Table I presents the details of the actual item pool. This actual item pool is used in the synthetic personality inventory examination, which is a widely used aptitude test in Japan [42].

TABLE I
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	-4.00–4.55	-0.22	1.16

TABLE II
Constraints for Test Assembly

I(θ)(Lower bound / Upper bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0.0$	$\theta = 1.0$	$\theta = 2.0$
2.0/2.4	3.2/3.6	3.2/3.6	3.2/3.6	2.0/2.4

We set the test constraints as follows.

1) The test includes 25 items.
2) The allowed maximum number of overlapping items is changed from zero to ten by one.

The test information constraints are described by the lower and upper bounds of the test information function $I(\theta_k)$. They are shown in Table II. We determined these constraints according to the actual test setting [42].

We used 24 h as a time limitation for all methods. For the RndMCP and the RIPMCP, we found the computational cost constraints L_1 as 100 000, L_2 as 3 h, and CT as 24 h. We determined the parameter values $L_1, L_2,$ and CT according to an explanation by Ishii *et al.* [21]. We employed the same parameter values as those presented by Ishii *et al.* [21]. The RndMCP and the RIPMCP were implemented in Java (the source code is available[1]). For the BST, we determine the target value of the information function $T(\theta_k)$ as follows:

$$T(\theta_k) = \{(\text{lower bounds of the information function}) + (\text{upper bounds of the information function})\}2.$$

Here, we conduct experiments on a machine with an Intel (R) Core i9-9900X 3.50-GHz CPU, and 128-GB main memory running a Linux (64-bit Ubuntu) operating system. This article will propose a parallel computing algorithm for uniform test assembly in the next section. This machine specification allows parallel computing (ten processor cores). Note that parameter L_1 depends on the main memory capacity. Therefore, we determined the same main memory capacity as that of Ishii *et al.* [21]. For the BST [25] and the RIPMCP, we apply CPLEX [43] for the IP problem.

Table III shows the quantities of assembled uniform tests using our method and using the traditional methods by changing the item pool sizes and the overlapping constraints. It is noteworthy that the "HMCAPIP" will be proposed in the next section. We discuss the HMCAPIP results later.

In traditional methods, for all cases except OC = 0, the RndMCP assembles a greater number of tests than the BST does because the aim of the BST is not to maximize the number of assembled tests. When OC = 0, the BST assembles a

[1] http://www.ai.lab.uec.ac.jp/software-e/.

TABLE III
NUMBER OF ASSEMBLED UNIFORM TESTS FOR EACH METHOD IN LARGE-SCALE ITEM POOLS

Item Pool Size	OC	BST [41]	RndMCP [21]	RIPMCP	HMCAPIP
500	0	15	10	**17**	**17**
	1	19	22	**47**	41
	2	19	65	**267**	240
	3	19	223	**1144**	730
	4	19	936	**5032**	3348
	5	19	4324	12550	**14331**
	6	19	19817	29207	**49837**
	7	19	61740	67969	**97792**
	8	19	93678	98406	**122378**
	9	19	99469	104991	**127229**
	10	19	99979	105002	**127149**
1000	0	25	17	**34**	**34**
	1	39	61	**318**	272
	2	39	282	**1892**	1431
	3	39	1585	7557	**9000**
	4	39	9793	20653	**39970**
	5	39	46162	55024	**106881**
	6	39	90127	96527	**134050**
	7	39	99396	106834	**139172**
	8	39	99979	107942	**139757**
	9	39	99998	107735	**140059**
	10	39	100000	107672	**140067**
2000	0	61	32	**70**	**70**
	1	79	186	**1531**	988
	2	79	1463	6963	**7569**
	3	79	12456	25364	**51401**
	4	79	62424	72520	**108165**
	5	79	96859	103354	**129257**
	6	79	99891	106362	**131791**
	7	79	99993	107434	**132273**
	8	79	100000	107774	**132090**
	9	79	100000	107998	**133550**
	10	79	100000	107783	**140700**
978(actual)	0	31	18	**35**	**35**
	1	38	63	**348**	286
	2	38	297	**1844**	1334
	3	38	1717	6960	**7050**
	4	38	10252	14866	**31724**
	5	38	45746	52126	**73693**
	6	38	88947	93704	**108935**
	7	38	99993	104339	**118165**
	8	38	100000	105823	**119797**
	9	38	100000	105805	**119758**
	10	38	100000	105956	**124200**

The bold numbers in the table signify the best performances.

greater number of tests than the RndMCP does because the random subgraph is too small for this test assembly setting.

Actually, the RIPMCP assembles a greater number of tests than the traditional methods do. The reason is that the RIPMCP has lower space complexity than the RndMCP has. However, the difference of the numbers of tests between the RIPMCP and the RndMCP becomes slight as the number of tests becomes large because the RIPMCP performance is limited as a result of its high time complexity. In the next section, we address this limitation.

VI. HYBRID MAXIMUM CLIQUE ALGORITHM WITH PARALLEL INTEGER PROGRAMMING

The RIPMCP has lower space complexity $\mathrm{O}(|V|)$ than that of RndMCP. This advantage is expected to increase the number of assembled uniform tests for limited memory. Nevertheless, the effect of RIPMCP compared to RndMCP remains limited because of the high time complexity $\mathrm{O}(|V| \cdot 2^n)$. Therefore, to relax the high time complexity, we propose a new two-step parallel algorithm: HMCAPIP. The first step seeks a maximum clique that is as large as possible up to the limitations imposed by computer memory using RndMCP with low time complexity, but with high space complexity. The second step is a repeated search for a vertex that is connected with all vertices of the current clique from the remaining vertices using IP with low space complexity but with high time complexity.

The first step seeks a maximum clique that is as large as possible up to the limitations of computer memory using the RndMCP with the low calculation time limit CT, but with high space complexity $\mathrm{O}(|V|^2)$. It costs only polynomial order but disables search cliques with more than a hundred thousand vertices because of computer memory limitations. Therefore, the algorithm switches from the RndMCP to the IP method, which has low space complexity $\mathrm{O}(|V|)$.

The second step repeatedly seeks a vertex connected with all vertices of the current clique from the remaining vertices using IP with low space complexity $\mathrm{O}(|V|)$, but with high time complexity $\mathrm{O}(|V| \cdot 2^n)$. The number of additionally assembled uniform tests using the IP is limited because of its high time complexity. To relax this difficulty, our method parallelizes the second step. Sequentially, finding a vertex using IP is difficult to parallelize efficiently. Therefore, our method seeks vertices, as shown in Fig. 2.

In step (a), using multiple processors, the second step seeks P vertices connected with all vertices of the current clique using IP in parallel, where P represents the number of parallelizations for seeking a vertex using the IP. Here, the optimal value of the objective function differs for each search because λ_i is resampled. Then, the found vertices are added to a set S (where S represents a set of vertex connected with all vertices of the current clique). This procedure is repeated until $S_{\mathrm{UB}} \leq |S|$ (where S_{UB} represents the limit of the number of vertices using IP). Moreover, the solution of IP is used as the initial lower bound LB in the next search for the branch and bound algorithm. Subsequently, in step (b), the second step seeks a maximum clique MC from the found vertices in S. In step (c), the found maximum clique combines with the current clique. Results show that, by dividing the computational cost, the HMCAPIP can assemble a greater number of uniform tests than the RIPMCP can. If the IP has no solution, then the search might fall into a local solution. In such a case, to avoid the local solution, the algorithm removes D_1 vertices randomly from the current clique.

The following seven parameters are used for the HMCAPIP.

- L_1 is the number of feasible tests of the RndMCP stored in computer memory.
- L_2 is the time limit of the maximum clique extraction of the RndMCP.
- CT' is the total calculation time limit of the RndMCP.
- S_{UB} is the limit of the number of vertices using IP.
- D_1 is the number of removed vertices from the current clique when the IP has no solution.

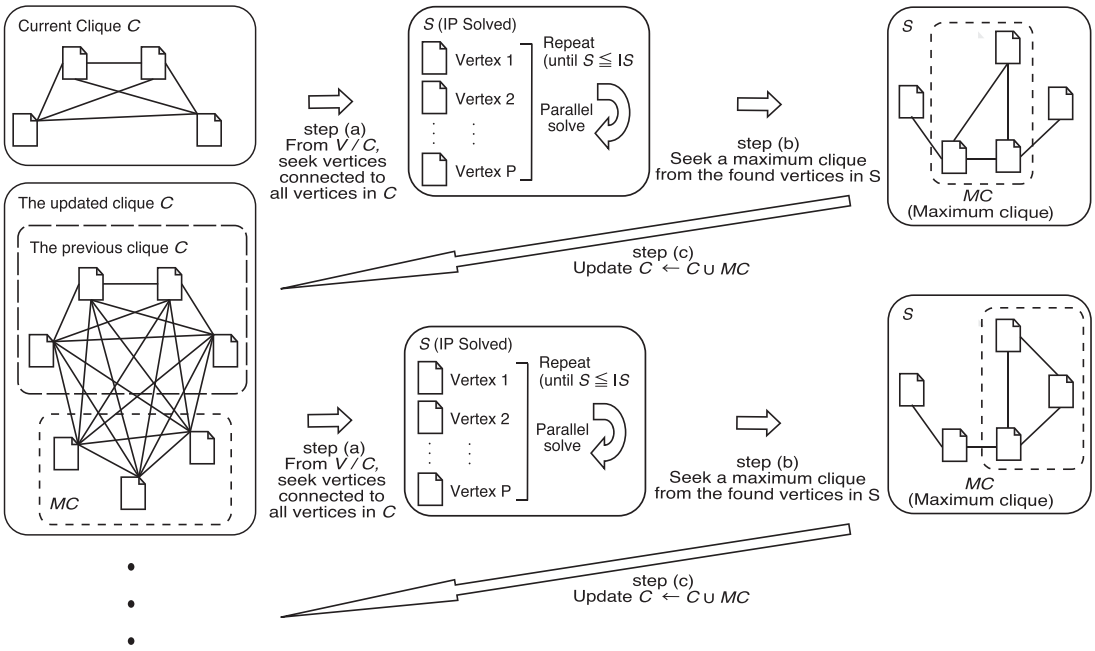


Fig. 2. Outline of the second step.

P is the number of parallelizations for seeking a vertex by IP.

CT is the total calculation time limit of the test assembly.

The HMCAPIP algorithm is shown in Algorithm 3.

VII. Experiments Used to Assess HMCAPIP

As described in this section, we estimate the individual parameters for the HMCAPIP, which has some tradeoff among the values of parameters.

Therefore, we evaluate the tradeoff by changing the values of parameters to ascertain optimal values to maximize the number of assembled uniform tests. Subsequently, we compare the performance of the proposed method with earlier reported methods and the RIPMCP using simulated and actual item pools.

We set the test constraints as follows.

1) The test includes 25 items.
2) Allowed maximum numbers of overlapping items are changed from zero to ten by increments of one.

The test information constraints are presented in Table II.

The HMCAPIP was implemented in Java (the source code is available[2]).

A. Optimization of Parameter S_{UB}

Our method has a tradeoff by the value of S_{UB} between the search time to seek a maximum clique from the vertices in S and the search time to seek a vertex connected with all vertices of the current clique using IP. Therefore, we evaluate the tradeoff by changing the value of S_{UB} to infer the optimal value to maximize the number of assembled uniform tests.

[2] http://www.ai.lab.uec.ac.jp/software-e/.

We compare the performances of the values of S_{UB} using the simulated item pool with 1000 items. Items in the simulated item pools have discrimination parameters and difficulty parameters of IRT. We generated discrimination parameters as $\log_2 a \sim N(0, 1^2)$ and difficulty parameters as $b \sim N(0, 1^2)$.

We have 24 h as a time limitation for uniform test assembly. For our method, we set the computational cost constraints L_1 as 100 000, L_2 as 3 h, CT' as 3 h, D_1 as 100, and CT as 24 h. We determined the parameter values L_1, L_2, D_1, CT, and CT' according to an explanation by Ishii *et al.* [21], [44]. We employed the same parameter values as those presented by Ishii *et al.* [21], [44]. The experiments compare the number of tests for $S_{\mathrm{UB}} \in \{10, 100, 1000\}$.

Table IV presents the quantities of assembled uniform tests by changing the value of S_{UB}. In the table, the following information is presented. No. tests denotes the quantities of assembled uniform tests. Avg. search time (MCA) [s] denotes the average search time of the maximum clique in S. Avg. search time (IP) [s] represents the average of the search time of IP for uniform test assembly.

When $\mathrm{OC} \leq 3$, our method entails a tradeoff between Avg. search time (MCA) and Avg. search time (IP). Specifically, when the S_{UB} value becomes large, Avg. search time (MCA) increases but Avg. search time (IP) decreases. However, when $\mathrm{OC} > 3$, the magnitude of the tradeoff becomes small because Avg. search time (MCA) and Avg. search time (IP) do not change greatly for the S_{UB} values. In most cases, $S_{\mathrm{UB}} = 100$ assembles the greatest number of tests. Therefore, in this article, the S_{UB} value is determined as 100.

B. Effectiveness of the Lower Bound

In the proposed method, the IP solution is used as the initial lower bound LB for the next search. In this section, we evaluate the effectiveness of the lower bound LB using the same simulated item pool as that used in Section VII-A.

Algorithm 3: HMCAPIP.

procedure HMCAPIP$(L_1, L_2, CT', S_{UB}, D_1, P, CT)$
  $global\ ST \leftarrow current\ time$
  $C \leftarrow \text{Step1}(L_1, L_2, CT')$
  $C_{\max} \leftarrow \text{Step2}(C, S_{UB}, D_1, P, CT)$
  Output $C_{\max}$
end procedure
function Step1(L_1, L_2, CT')
  return RndMCP(L_1, L_2, CT') ▷ Algorithm 1
end function
function Step2(C, S_{UB}, D_1, P, CT)
  $C_{\max} \leftarrow C$
  while $(current\ time - ST) < CT$ **do**
    $S \leftarrow \emptyset$
    repeat
      $Sol \leftarrow \emptyset$
      parallel for $p \leftarrow 1 \ldots P$ **do**
        ▷ Seek P vertices in parallel
        $Sol_p \leftarrow \text{SearchVertex}C, S$
        ▷ Seek a vertex with all vertices of C connected
      end parallel for
      if $Sol \neq \emptyset$ **then** ▷IP has solutions
        $S \leftarrow S \cup Sol$
      else
      $C \leftarrow \text{Remove}(C, D_1)$ ▷ Remove D_1 vertices from C
      $break$
      end if
    until $S_{\text{UB}} \leq |S|$
    if $S \neq \emptyset$ **then**
      $G \leftarrow (S, E)$
      ▷ Generate a graph that corresponds to a set S with overlapping items.
  $MC \leftarrow \text{MCA}\ (G, L_2)$
▷ MCA(G, L_2) extracts the maximum clique from the graph G within calculation time L_2 using Nakanishi and Tomita's maximum clique algorithm [40].
      $C \leftarrow C \cup MC$
        ▷ Combine MC with the current clique
      if $|C_{\max}| < |C|$ **then** $C_{\max} \leftarrow C$ **end if**
    end if
  end while
  return $C_{\max}$
end function
function SearchVertex C, S
  LB $\leftarrow$ max $\{\sum_{i=1}^{n} \lambda_i x_i : x \in S\}$
    ▷ Objective function eq:16
  return SolveIP C, LB
    ▷ Solve IP with initial lower bound LB
end function

We have 24 h as a time limitation for uniform test assembly. In addition, we set the computational cost constraints L_1 as 100 000, L_2 as 3 h, CT' as 3 h, S_{UB} as 100, D_1 as 100, and CT as 24 h. Using these conditions, we compare the quantities of test configurations with LB and those without LB.

Table V presents the quantities of assembled uniform tests with and without a lower bound. The following are presented in the table. No. tests represents the quantities of assembled uniform tests, Avg. search nodes denotes the quantities of search nodes in IP, and Avg. search time [s] stands for the average of search time of IP for the uniform test assembly.

IP with LB assembles a greater number of tests than that without LB does because LB reduces the search space of IP. In fact, both Avg. search nodes and Avg. search time of with LB are lower than those without LB.

C. Optimization of the Number of Parallelizations

To relax the high time complexity of IP in the RIPMCP, our method repeats the parallel search of the vertices connected with all vertices of the current clique using IP. The efficiency of the parallel search depends on the number of tests and the value of P, where P represents the number of parallelizations for seeking a vertex by the IP. In this section, we evaluate the number of assembled tests by changing the value of P using the same simulated item pool used in Section VII-A.

We have 24 h as a time limitation for uniform test assembly. In addition, we set the computational cost constraints L_1 as 100 000, L_2 as 3 h, CT' as 3 h, S_{UB} as 100, D_1 as 100, and CT as 24 h. The experiments compare the quantities of tests by $P = 1, 2, 5, \text{and } 10$.

Table VI presents the numbers of assembled uniform tests by changing P. No. tests denotes the quantities of assembled uniform tests. In addition, Avg. search time [s] represents the average of search time to find one vertex using IP in the second step.

Actually, when OC becomes small, all the values of P assemble almost identically to the number of tests. The numbers converge to the maximum quantities of assembled uniform tests because the extract maximum quantities of uniform tests are not large as a result of the tight OC. However, when OC becomes large, $P = 10$ tends to assemble the greatest number of tests. That result indicates that the efficiency of parallelization increases as the number of tests increases. Therefore, the value of P is determined as 10 in these analyses.

In the next section, we discuss comparison of our method with the traditional methods by increasing a time limitation.

D. Performances for Large-Scale Item Pool

To demonstrate the benefits of the HMCAPIP, we compare the number of assembled uniform tests of the HMCAPIP with those of traditional methods (BST [41] in Section III-A and RndMCP [21] in Section III-B) and the RIPMCP using simulated and actual item pools. Items in the simulated item pools have discrimination parameters and difficulty parameters of IRT. We generated discrimination parameters as $\log_2 a \sim N(0, 1^2)$ and difficulty parameters as $b \sim N(0, 1^2)$. Details of the actual item pool are listed in Table I.

We used 24 h as a time limitation for all methods. For the HMCAPIP, we set the computational cost constraints L_1 as 100 000, L_2 as 3 h, CT' as 3 h, S_{UB} as 100, D_1 as 100, P as 10, and CT as 24 h. The values of parameters were ascertained

TABLE IV
SEARCH PERFORMANCES BY CHANGING THE PARAMETER S_{UB}

Item Pool Size	OC	Proposal								
		$S_{\mathrm{UB}} = 10$			$S_{\mathrm{UB}} = 100$			$S_{\mathrm{UB}} = 1000$		
		No.tests	Avg. search time (MCA) [s]	Avg. search time (IP) [s]	No.tests	Avg. search time (MCA) [s]	Avg. search time (IP) [s]	No.tests	Avg. search time (MCA) [s]	Avg. search time (IP) [s]
1000	0	**34**	**0.0001**	55.24	**34**	0.8632	47.52	**34**	457.4781	**37.23**
	1	243	**0.0001**	24.81	**261**	0.2375	20.24	253	180.0563	**10.07**
	2	1595	**0.0001**	33.29	**1639**	0.1338	25.63	1349	177.5153	**9.14**
	3	**7522**	**0.0001**	12.41	7355	0.0010	11.05	6936	180.0590	**6.36**
	4	33150	**0.0001**	3.11	**33160**	0.0008	3.04	31012	0.0583	**2.96**
	5	63706	**0.0001**	4.01	**65458**	0.0006	**3.58**	63719	0.0535	4.00
	6	105367	**0.0001**	**4.46**	**105391**	0.0007	**4.46**	105239	0.0463	4.51
	7	108286	**0.0001**	7.97	**108765**	0.0007	**7.56**	108279	0.0443	7.99
	8	108759	**0.0001**	8.08	**109245**	0.0006	**7.66**	108761	0.0455	8.10
	9	108698	**0.0001**	8.12	**109202**	0.0007	**7.66**	108746	0.0475	8.10
	10	108699	**0.0001**	8.10	**109203**	0.0005	**7.68**	108744	0.0472	8.12

The bold numbers in the table signify the best performances.

TABLE V
EFFECTIVENESS OF THE LOWER BOUND

Item Pool Size	OC	Proposal					
		without LB			with LB		
		No. tests	Avg. search nodes	Avg. search time [s]	No. tests	Avg. search nodes	Avg. search time [s]
1000	0	**34**	254232.3	47.52	**34**	**234232.3**	**45.82**
	1	261	144763.9	22.72	**293**	**134352.1**	**20.24**
	2	1639	146458.8	26.05	**1670**	**135961.9**	**25.63**
	3	7355	18974.5	11.05	**7383**	**17793.0**	**11.04**
	4	33160	1307.7	3.04	**36325**	**965.4**	**2.66**
	5	65458	233.2	3.58	**74654**	**219.8**	**2.33**
	6	105391	154.7	4.46	**107527**	**149.6**	**3.84**
	7	108765	133.0	7.56	**114296**	**128.1**	**4.52**
	8	109245	128.4	7.66	**114835**	**126.8**	**4.55**
	9	109202	129.5	7.66	**114796**	**126.1**	**4.54**
	10	109203	129.7	7.68	**114781**	**126.6**	**4.57**

The bold numbers in the table signify the best performances.

TABLE VI
PERFORMANCE OF PARALLEL SOLUTION OF IP

Item Pool Size	OC	Proposal							
		$P = 1$		$P = 2$		$P = 5$		$P = 10$	
		No. tests	Avg. search time [s]	No. tests	Avg. search time [s]	No. tests	Avg. search time [s]	No. tests	Avg. search time [s]
1000	0	**34**	45.82	**34**	24.32	**34**	10.31	**34**	**6.24**
	1	**293**	20.24	290	14.36	236	5.44	272	**2.97**
	2	**1670**	25.63	1584	18.29	1605	9.63	1657	**5.33**
	3	7383	11.04	7972	8.74	**9462**	6.35	9000	**4.09**
	4	36325	2.66	36057	1.99	37333	1.44	**39970**	**1.28**
	5	74654	2.33	76997	2.02	94127	1.08	**106881**	**0.66**
	6	107527	3.84	106967	3.78	119624	1.82	**134050**	**0.98**
	7	114296	4.52	114668	4.18	126496	2.00	**139172**	**1.07**
	8	114835	4.55	115203	4.20	126759	2.01	**139757**	**1.09**
	9	114796	4.54	115090	4.21	126703	2.02	**140059**	**1.07**
	10	114781	4.57	115092	4.20	126702	2.04	**140067**	**1.04**

The bold numbers in the table signify the best performances.

by comparing the numbers of assembled tests in Sections VII-A– VII-C. For RndMCP, RIPMCP, and BST, we set the same computational cost constraints used in Section V.

We present the results in Table III. When OC = 0 and 1, the HMCAPIP assembles almost identical numbers of uniform tests as the RIPMCP does. The numbers converge to the maximum number of assembled uniform tests because the exact maximum number of uniform tests is not large as a result of the tight OC.

The table shows that the HMCAPIP relaxes the high time complexity problem of the RIPMCP by IP in parallel. Therefore, the difference of the quantities of uniform tests between the RndMCP and the HMCAPIP is still large even when the number of assembled tests becomes greater than 100 000. The

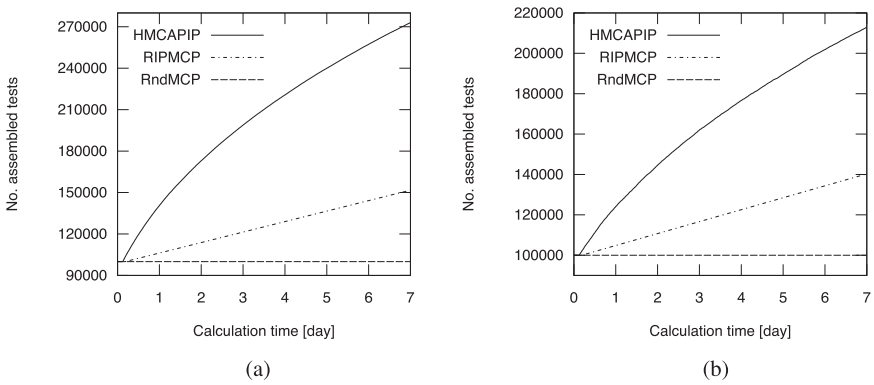


Fig. 3. Numbers of assembled uniform tests in 168 h. (a) Simulated item pools. (b) Actual item pools.

TABLE VII
NUMBERS OF ASSEMBLED UNIFORM TESTS IN 168 H

item pool size	OC	RndMCP	Proposal	
			RIPMCP	HMCAPIP
2000	0	32	**70**	**70**
	5	96859	149403	**254418**
	10	100000	151592	**274900**
978 (actual)	0	18	**35**	**35**
	5	45746	103763	**139048**
	10	100000	140185	**214350**

The bold numbers in the table signify the best performances.

main reason is that the HMCAPIP divides the computational cost of the second step by parallel search using IP. Consequently, the results demonstrated the effectiveness of the proposed method.

E. Performance of Extended Calculation Time

The RIPMCP and the HMCAPIP have lower space complexity than the RndMCP has. This advantage increases the number of assembled uniform tests for a limited amount of memory. However, the improvement of the RIPMCP remains limited because of its high time complexity. The HMCAPIP divides computational costs into multiple processors. Therefore, the difference of the quantities of tests between the RIPMCP and the HMCAPIP might increase if the calculation time increases. To confirm this point, we compare the quantities of assembled uniform tests using RndMCP, RIPMCP, and HMCAPIP by extending the time limit to 168 h (seven days). We use the simulation item pool with 2000 items, which generates the greatest number of tests in the simulation item pools and actual item pool with 978 items.

Table VII presents the numbers of assembled uniform tests given the time limitation of 168 h for RndMCP, RIPMCP, and HMCAPIP. Results show that the proposed methods can assemble a greater number of tests than the RndMCP can because the proposed methods have lower space complexity than the RndMCP has. Because of the space complexity limitation, the number of tests by the RndMCP does not increase as the calculation time increases. As a result, the HMCAPIP assembles 1.5–2.7 times greater numbers of uniform tests than the RndCMP does, except for the case of OC = 0. When OC = 0, the quantities of assembled uniform tests of the proposed methods are equal because they converge to a maximal number of uniform tests. When OC becomes large, the differences of the numbers of tests between the RIPMCP and the HMCAPIP become large because the HMCAPIP divides the high time complexity of IP.

Fig. 3 shows the quantities of assembled uniform tests of the HMCAPIP and the RIPMCP with OC = 10 for (a) simulated item pool size 2000 and for (b) actual item pool size 978, which assembled the greatest numbers of tests in Table VII. The results demonstrate that the difference of the numbers of assembled uniform tests between the HMCAPIP and the RIPMCP becomes large as the calculation time increases. Results suggest that the proposed method assembles a greater number of uniform tests given a longer calculation time.

VIII. CONCLUSION

In this article, we proposed a new algorithm for assembling uniform tests. The proposed method assembled a 1.5–2.7 times greater number of uniform tests than traditional methods did. To achieve this result, we applied IP to the MCP for improving results obtained by Ishii *et al.* [20], [21]. Ishii *et al.* [20], [21] described a need for storing a huge graph structure for assembling a greater number of uniform tests. The computational environment limits the number of assembled uniform tests. Specifically, the proposed method can assemble a greater number of tests by repeatedly searching a vertex connected with all vertices of the current clique using IP. By storing only connected vertices with the current clique, the proposed method improves the efficiency of space complexity usage and the number of assembled uniform tests. Moreover, to relax the computational complexity, we proposed a new two-step parallel algorithm: HMCAPIP. The first step seeks a maximum clique using the RndMCP with low constant time but high space complexity. The second step repeats the parallel search of the vertices connected with all vertices of the current clique using IP with low space complexity but with high time complexity.

To demonstrate the performance of the proposed method, we conducted three experiments using simulated and actual data. Results indicate that the proposed method assembled a greater number of uniform test forms than the traditional methods did. Moreover, results suggest that the drastically

different numbers of assembled uniform tests between the proposal and the traditional methods would increase by extending the calculation time.

However, the IP of the proposed methods still includes high time complexity. Therefore, the improvement of the proposed method might be limited. In fact, the HMCAPIP requires one week or more to assemble 300 000 tests. Moreover, the performance of the HMCAPIP depends on the number of processors in a computer. Therefore, when the number of processors becomes small, the effectiveness of the HMCPIP also becomes small. In addition, the proposed method has room for the improvement of parameter optimization to maximize the number of assembled uniform tests because this article used the parameter values presented by Ishii *et al.* [21], [44].

Furthermore, this article focuses on uniform tests assembly only with constraints respect to test length and test information. However, in practice, actual examinations require other constraints. For example, the proposed methods do not control how many times each item has been used in the assembled uniform test forms. Therefore, the distribution of the item use counts does not become uniform, which is called an item exposure bias problem [45]. This difficulty is known to decrease the reliability of items and tests [45]. Ideally, the item exposure distribution would be uniform. For this purpose, Ishii and Ueno [46] proposed a clique algorithm, which is a test assembly method, by searching the clique with minimum item exposure using IP. An important future task is to resolve the item exposure bias problem using our methods in the same way as in [46].

REFERENCES

[1] F. M. Lord, *Applications of Item Response Theory to Practical Testing Problems*, 1st ed. Evanston, IL, USA: Routledge, Jul. 1980.

[2] T. J. J. M. Theunissen, "Binary programming and test design," *Psychometrika*, vol. 50, no. 4, pp. 411–420, Dec. 1985.

[3] T. J. J. M. Theunissen, "Some applications of optimization algorithms in test design and adaptive testing," *Appl. Psychol. Meas.*, vol. 10, no. 4, pp. 381–389, 1986.

[4] 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)*. Enschede, The Netherlands: Dept. Educ. Sci. Technol., Univ. Twente, 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," *Appl. Psychol. Meas.*, 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. Educ. Statist.*, vol. 14, pp. 279–290, 1989.

[8] T. A. Ackerman, "An alternative methodology for creating parallel test forms using the IRT information function," in *Proc. Annu. Meeting Nat. Council Meas. Educ.*, Mar. 30, 1989, pp. 1–25.

[9] J. J. Ameda, "Models and algorithms for the construction of achievement tests," Ph.D. dissertation, Dept. Educ., Univ. Twente, Enschede, The Netherlands, 1990.

[10] J. J. Adema, E. Boekkooi-Timminga, and W. J. van der Linden, "Achievement test construction using 0-1 linear programming," *Eur. J. Oper. Res.*, vol. 55, no. 1, pp. 103–111, 1991.

[11] J. J. Adema, "Methods and models for the construction of weakly parallel tests," *Appl. Psychol. Meas.*, 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," *Appl. Psychol. Meas.*, 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," *Psychol. Testing*, vol. 44, no. 2, pp. 195–210, 1997.

[14] R. M. Luecht, "Computer-assisted test assembly using optimization heuristics," *Appl. Psychol. Meas.*, 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. Educ. Meas.*, vol. 35, no. 3, pp. 185–198, Sep. 1998.

[16] R. B. Fletcher, "A review of linear programming and its application to the assessment tools for teaching and learning (as TTLE) projects," Univ. Auckland, New Zealand, Tech. Rep. 5, 2000.

[17] G.-J. Hwang, P.-Y. Yin, and S.-H. Yeh, "A tabu search approach to generating test sheets for multiple assessment criteria," *IEEE Trans. Educ.*, vol. 49, no. 1, pp. 88–97, Feb. 2006.

[18] K.-T. Sun, Y.-J. Chen, S.-Y. Tsai, and C.-F. Cheng, "Creating IRT-based parallel test forms using the genetic algorithm method," *Appl. Meas. Educ.*, vol. 2, no. 21, pp. 141–161, 2008.

[19] P. Songmuang and M. Ueno, "Bees algorithm for construction of multiple test forms in E-testing," *IEEE Trans. Learn. Technol.*, vol. 4, no. 3, pp. 209–221, Jul.–Sep. 2011.

[20] T. Ishii, P. Songmuang, and M. Ueno, "Maximum clique algorithm for uniform test forms," in *Proc. 16th Int. Conf. Artif. Intell. Educ.*, 2013, pp. 451–462.

[21] T. Ishii, P. Songmuang, and M. Ueno, "Maximum clique algorithm and its approximation for uniform test form assembly," *IEEE Trans. Learn. Technol.*, vol. 7, no. 1, pp. 83–95, Jan.–Mar. 2014.

[22] M. L. Nguyen, S. C. Hui, and A. C. Fong, "Large-scale multiobjective static test generation for web-based testing with integer programming," *IEEE Trans. Learn. Technol.*, vol. 6, no. 1, pp. 46–59, Jan.–Mar. 2013.

[23] M. Ueno, K. Fuchimoto, and E. Tsutsumi, "E-testing from artificial intelligence approach," *Behaviormetrika*, vol. 48, no. 2, pp. 409–424, 2021.

[24] M. Ueno, "AI based E-testing as a common yardstick for measuring human abilities," in *Proc. 18th Int. Joint Conf. Comput. Sci. Softw. Eng.*, 2021, pp. 1–5.

[25] W. J. van der Linden, *Liner Models for Optimal Test Design*. Berlin, Germany: Springer, 2005.

[26] E. Boekkooi-Timminga, "The construction of parallel tests from IRT-based item banks," *J. Educ. Statist.*, vol. 15, pp. 129–145, 1990.

[27] R. D. Armstrong, D. H. Jones, and Z. Wang, "Automated parallel test construction using classical test theory," *J. Educ. Statist.*, vol. 19, no. 1, pp. 73–90, 1994.

[28] R. D. Armstrong, D. H. Jones, and C. S. Kunce, "IRT test assembly using network-flow programming," *Appl. Psychol. Meas.*, vol. 22, no. 3, pp. 237–247, 1998.

[29] T.-Y. Chang and Y.-F. Shiu, "Simultaneously construct IRT-based parallel tests based on an adapted CLONALG algorithm," *Appl. Intell.*, vol. 36, no. 4, pp. 979–994, Jun. 2012.

[30] J. Pereira and M. Vila, "Variable neighborhood search heuristics for a test assembly design problem," *Expert Syst. Appl.*, vol. 42, no. 10, pp. 4805–4817, 2015.

[31] D. I. Belov and R. D. Armstrong, "A constraint programming approach to extract the maximum number of non-overlapping test forms," *Comput. Optim. Appl.*, vol. 33, pp. 319–332, 2006.

[32] E. Tomita, K. Yoshida, T. Hatta, A. Nagao, H. Ito, and M. Wakatsuki, "A much faster branch-and-bound algorithm for finding a maximum clique," in *Proc. Int. Workshop Front. Algorithmics*, 2016, pp. 215–226.

[33] F. Lord and M. Novick, *Statistical Theories of Mental Test Scores*. Reading, MA, USA: Addison-Wesley, 1968.

[34] F. Baker and S. Kim, *Item Response Theory: Parameter Estimation Techniques (Statistics: A Series of Textbooks and Monographs)*. 2nd ed. New York, NY, USA: Taylor & Francis, 2004.

[35] 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, Jun. 1989.

[36] J. J. Ameda, "Implementations of the branch-and-bound method for test construction problems," Project Psychometric Aspects of Item Banking, Dept. Educ. Sci. Technol., Univ. Twente, Enschede, The Netherlands, Res. Rep. 89-6, 1989.

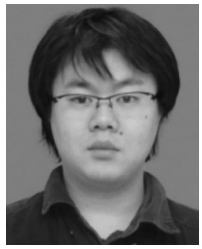
[37] C.-M. Li, H. Jiang, and F. Manyà, "On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem," *Comput. Oper. Res.*, vol. 84, pp. 1–15, 2017.

[38] P. San Segundo, A. Lopez, and P. M. Pardalos, "A new exact maximum clique algorithm for large and massive sparse graphs," *Comput. Oper. Res.*, vol. 66, pp. 81–94, 2016.

[39] L. Chang, "Efficient maximum clique computation and enumeration over large sparse graphs," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 529–538.
[40] H. Nakanishi and E. Tomita, "An $o(2^{0.19171n})$-time and polynomial-space algorithm for finding a maximum clique," *Information Processing Society* Japan, Tokyo, Japan, SIG Tech. Rep. 6, 2008, pp. 15–22.
[41] D. I. Belov, "Uniform test assembly," *Psychometrika*, vol. 73, no. 1, pp. 21–38, 2008.
[42] Synthetic Personality Inventory (SPI), Recruit, Tokyo, Japan, 2022. [Online]. Available: http://www.spi.recruit.co.jp/
[43] ILOG CPLEX Optimization Studio CPLEX User's Manual 12.9, IBM, Armonk, NY, USA, 2019.
[44] T. Ishii, T. Akakura, and M. Ueno, "An approximate maximum clique algorithm using integer programming," (in Japanese), *IEICE Trans. Inf. Syst.*, vol. 100, no. 1, pp. 47–59, 2017.
[45] H. Wainer, "Rescuing computerized testing by breaking Zipf's law," *J. Educ. Behav. Statist.*, vol. 25, pp. 203–224, 2000.
[46] T. Ishii and M. Ueno, "Clique algorithm to minimize item exposure for uniform test forms assembly," in *Proc. Int. Conf. Artif. Intell. Educ.*, 2015, pp. 638–641.
[47] T. Ishii and M. Ueno, "Algorithm for uniform test assembly using a maximum clique problem and integer programming," in *Proc. Int. Conf. Artif. Intell. Edu.*, 2017, pp. 102–112.

Kazuma Fuchimoto received the B.Eng. degree in computer science from the University of Electro-Communications, Tokyo, Japan, in 2020, where he is currently working toward the M.Eng. degree.

His research interests include educational technology and computer science.



Takatoshi Ishii received the Ph.D. degree in computer science from the University of Electro-Communications, Tokyo, Japan, in 2014.

He has been a Researcher with the University of Electro-Communications since 2018. He is also with the Sundai Institute of AI for Education, SATT Company, Ltd., Tokyo. He was with the Tokyo University of Science, Tokyo, from 2016 to 2018. His research interests include educational technology, data mining, and computer science.

Maomi Ueno (Member, IEEE) received the Ph.D. degree in computer science from the Tokyo Institute of Technology, Tokyo, Japan, in 1994.

He has been a Professor with the Graduate School of Information Systems, University of Electro-Communications, Tokyo, since 2013. His research interests include machine learning, data mining, Bayesian statistics, Bayesian networks, and educational technology.

Dr. Ueno received best paper awards at the 2008 IEEE International Conference on Tools with Artificial Intelligence.