IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 July 2023, accepted 2 October 2023, date of publication 9 October 2023, date of current version 18 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322720

RESEARCH ARTICLE

Automated Parallel Test Forms Assembly Using Zero-Suppressed Binary Decision Diagrams

KAZUMA FUCHIMOTO[1], SHIN-ICHI MINATO[2], (Senior Member, IEEE), AND MAOMI UENO[1], (Member, IEEE)

[1]Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan
[2]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

Corresponding author: Kazuma Fuchimoto (fuchimoto@ai.lab.uec.ac.jp)

This work was supported by the Japan Society for the Promotion of Science Grants-in-Aid for Scientific Research under Grant JP19H05663 and Grant JP22K19825.

ABSTRACT Recently, through the progress achieved in the study of computer science, automated test assemblies of parallel test forms, for which each form has equivalent measurement accuracy but with a different set of items, have emerged as a new standard tool. An important goal for automated test assembly is to assemble as many parallel test forms as possible. Although many automated test assembly methods exist, maximum clique using the integer programming method is known to be able to assemble the greatest number of assembled test forms with the highest measurement accuracy. Nevertheless, because of the high time complexity of integer programming, the method requires a month or more to assemble 300,000 tests. This study proposes a new automated test assembly using Zero-suppressed Binary Decision Diagrams (ZDD): a graphical representation for a set of item combinations. This representation is derived by reducing a binary decision tree. According to the proposed method, each node in the binary decision tree corresponds to an item of an item pool, which is a test item database. Each node has two edges, each signifying that the corresponding item is included in a test form or not. Furthermore, all equivalent nodes are shared, providing that they have equal measurement accuracy and equal test length. Numerical experiments demonstrate that the proposed method can assemble 1,500,000 test forms within 24 hr, although earlier methods have been capable of assembling only 300,000 test forms within 10 days.

INDEX TERMS Automated test assembly, item response theory, parallel test, zero-suppressed binary decision diagrams.

I. INTRODUCTION

With the rapid development of computer science technologies, automated test assemblies [1], [2], [3], [4], [5] of parallel test forms, for which each form has equivalent measurement accuracy but with a different set of items, have been put to practical use. For instance, parallel test forms are necessary when a testing organization administers tests at different times. To accomplish this mode of testing, parallel 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. Consequently, parallel test forms ensure that their scores are equivalent even if different examinees with the same ability take distinct tests.

Automated test assemblies have resolved difficulties such as high computational costs [1], [2], maximization of the number of tests [3], [5], minimization of item exposure bias [4], and so on. Most recently, automated test assembly has been applied for computerized adaptive testing (such as [6], [7]), which selects and presents the optimal item for individual examiners.

Earlier studies [1], [2], [3], [4], [5], [8], [9], [10], [11], [12], [13], [14], [15] have formulated a test assembly as a combinational optimization problem. The test assembly seeks a combination of items that satisfies given test constraints, such as the test lengths and ability measurement accuracy from an item pool.

The associate editor coordinating the review of this manuscript and approving it for publication was Nuno M. Garcia.

The most widely recognized automated test assembly is the *Big Shadow Test method* (BST), which uses Mixed-Integer Programming (MIP) [8]. This method assembles parallel test forms sequentially by minimizing the difference of measurement accuracies between a current assembled test and a set of items that remain in the item pool. Although the Big Shadow Test method can impose test constraints on parallel test forms effortlessly using MIP, the method does entail two crucially important shortcomings. First, the ability measurement accuracy of the examinee decreases as the assembled number of test forms increases. Second, this method does not guarantee maximization of the number of assembled test forms. Test organizations allocate different test forms to each examinee to secure the items from item leaks and exam cheating by the test-takers. Therefore, the second difficulty is a severe difficulty for automated test assembly. Ideally, the number of assembled test forms should exceed the number of examinees.

To resolve this shortcoming, Ishii et al. [3], [16] formalized an automated test assembly with overlapping conditions as a maximum clique problem (MCP), which is a combinational optimization in graph theory. The automated test assembly constructs a graph in which the vertices and the edges respectively represent tests satisfying the test constraint and represent satisfaction of the overlapping constraint. Then, this method extracts the maximum clique from the graph as parallel test forms. The automated test assembly can guarantee the maximum number of assembled test forms exactly. The maximum clique method assembled 10–1,000 times the number of assembled test forms that the traditional methods did. However, this method limits the number of assembled test forms to a hundred thousand because the maximum clique problem has high space complexity.

To reduce the high space complexity of the maximum clique method, Fuchimoto et al. [5] proposed the Hybrid Maximum Clique Algorithm with Parallel Integer Programming (HMCAPIP), which is known to assemble the greatest number of assembled test forms with the highest measurement accuracy. As the first step, test forms are assembled with the overlapping constraint as a maximum clique problem with low time complexity, but with high space complexity. The second step, using integer programming (IP) with low space complexity but with high time complexity, repeats the parallel search of the tests satisfied with the overlapping constraint for all currently assembled tests. However, HMCAPIP retains the heavy time complexity of IP. For example, it requires one week or more to assemble 300,000 test forms. Moreover, the parallel search effectiveness depends on the computer specifications.

To improve the number of assembled test forms, this study proposes a new automated test assembly using Zero-suppressed Binary Decision Diagrams (ZDD). A ZDD is an efficient graphical representation of a set of item combinations [17]. Actually, ZDD is derived by reducing a binary decision tree using reduction rules [17]. These reduction rules are able to decrease the calculation time and computer memory limitation efficiently. Because of this benefit of ZDD, many studies have solved real-world problems such as grid power loss minimization [18], region partitioning for disaster evacuation [19], architectural floor planning [20], and Stackelberg models of combinatorial congestion games [21].

Using the proposed method, each node in the binary decision tree corresponds to an item of an item pool. Each node has two edges, respectively representing whether the corresponding item is included in a test form or not. The binary decision tree can enumerate all parallel test forms with the satisfied test constraints. Nevertheless, the binary decision tree has high space complexity $O(2^n/n)$, where n represents the number of items. To relax this high space complexity, the proposed method compresses the binary decision tree using ZDD in a breadth-first manner [22]: an efficient construction technique for reducing computer memory and calculation time. Specifically, all equivalent nodes are shared, provided that they have equivalent measurement accuracies and equivalent test lengths. Unfortunately, measurement accuracies of two nodes are rarely equivalent. Therefore, the proposed method shares those nodes when the difference in measurement accuracies between the two nodes is less than the threshold value. Then, the measurement accuracy of the shared node is approximated by the average measurement accuracies of the two nodes. Consequently, the proposed method can assemble parallel test forms without redundantly repeating calculations for measurement accuracies and test lengths.

However, this ZDD has a larger measurement error than HMCAPIP has because the shared node is approximated by the average measurement accuracy. Furthermore, this ZDD is incapable of controlling overlapping items. To resolve these shortcomings, the proposed method repeats random sampling [17] from ZDD with low time complexity $O(n)$ to enumerate parallel test forms with measurement errors that are less than a determined value and with a number of overlapping items that is less than a determined value.

Numerical experiments demonstrate that the proposed method can assemble 1,500,000 test forms within 24 hr, although the earlier method can assemble only 300,000 test forms within 10 days.

II. IRT

Most earlier studies of automated test assembly have evaluated the measurement accuracies of parallel test forms (such as [1], [2], [8], [10], [11], [12], [23]) using Item Response Theory (IRT) [24], [25]. It is noteworthy that IRT can measure the ability of examinees on the same scale, even when the examinees have taken different tests. In the two-parameter IRT logistic model, which is employed in most earlier studies, the probability that an examinee $j(= 1, 2, \ldots, m)$ with ability $\theta_j \in (-\infty, \infty)$ answer item

$i(= 1, 2, \ldots, n)$ correctly is defined as

$$p_i(\theta_j) = \frac{1}{1 + exp(-1.7a_i(\theta_j - b_i))}, \tag{1}$$

where $a_i \in [0, \infty)$ and $b_i \in (-\infty, \infty)$ respectively represent the respective discrimination power and difficulty of item i. The asymptotic variance of estimated ability based on IRT approaches the inverse of Fisher Information [24]. Therefore, IRT typically employs Fisher Information as an index of the measurement accuracy for the examinee's ability to estimate. In the two-parameter logistic model, the Fisher Information is defined when item i provides an examinee's ability θ using the following item information function.

$$IIF_i(\theta) = 1.7^2 a_i{}^2 p_i(\theta)(1 - p_i(\theta)). \tag{2}$$

This item information function implies that items with high Fisher Information $IIF_i(\theta)$ are highly discriminatory in the examinee's ability. The test information function $TIF_{Test}(\theta)$ of a test form *Test* is defined as

$$TIF_{Test}(\theta) = \sum_{i \in Test} IIF_i(\theta). \tag{3}$$

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

$$SE_i(\hat{\theta}) = \frac{1}{\sqrt{IIF_i(\hat{\theta})}}, \tag{4}$$

$$SE_{Test}(\hat{\theta}) = \frac{1}{\sqrt{TIF_{Test}(\hat{\theta})}}. \tag{5}$$

Therefore, using the test information function, the testing organization administrators can estimate the measurement accuracy of a test form.

Actually, the test information function is a continuous function of the examinee ability and the item characteristic parameters. Traditional methods (e.g. [1], [2], [8], [10], [11], [12], [23]) treat values of the test information function discretely to simplify computation. For instance, these methods have been evaluated specifically for some points $\Theta = \{\theta_1, \ldots, \theta_k, \ldots, \theta_K\}$ on the ability level θ. According to traditional methods, this study treats the test information function similarly.

III. TRADITIONAL METHODS OF AUTOMATED TEST ASSEMBLY

This section introduces two typical automated test assembly methods from earlier research.

A. BIG SHADOW TEST METHOD

The most widely recognized automated test assembly method, which uses mixed integer programming (MIP), is the *Big Shadow Test* method (BST) devised by van der Linden [8]. Actually, BST assembles parallel test forms sequentially by minimizing the difference of the test information between an assembled test form and a set of items remaining in the item pool. The set of remaining items is called the *shadow test*. Actually, BST optimizes the following MIP problem.

variables

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

$$y \geq 0,$$

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

minimize

$$y$$

subject to

$$\sum_{k=1}^{K} |\sum_{i=1}^{n} IIF_i(\theta_k)x_i - T(\theta_k)| \leq My, \tag{6}$$

$$\sum_{k=1}^{K} |\sum_{i=1}^{n} IIF_i(\theta_k)z_i - T_{ST}(\theta_k)| \leq M_{ST}y, \tag{7}$$

$$\sum_{i=1}^{n} x_i = M, \tag{8}$$

$$\sum_{i=1}^{n} z_i = M_{ST}, \tag{9}$$

$$x_i + z_i \leq 1. \tag{10}$$

$$(i = 1, 2, \ldots, n)$$

where

$$T_{ST}(\theta_k) = \frac{M_{ST}}{M} T(\theta_k), \tag{11}$$

Therein, M and M_{ST} respectively represent the numbers of items in the assembling test form and the shadow test form. Also, $T(\theta_k)$ denotes a target value of the test information function at the ability level θ_k for the assembling test form. For the shadow test, $T_{ST}(\theta_k)$ is a target value of the test information function at ability level θ_k. The test quality constraints without an information function (e.g., test time limits) can be included among the constraints of the MIP.

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

Solving the MIP assembles a test form using items one-by-one to assemble parallel test forms. Although this greedy algorithm of BST reduces computational costs, its measurement accuracy for the examinee's ability decreases as the number of assembled test forms increases.

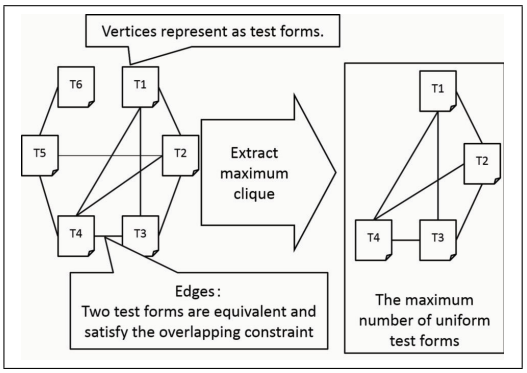

FIGURE 1. Outline of the maximum clique method.

B. HYBRID MAXIMUM CLIQUE ALGORITHM USING PARALLEL INTEGER PROGRAMMING METHOD

Hybrid Maximum Clique Algorithm Using Parallel Integer Programming method (HMCAPIP) [5] is known to assemble the greatest number of tests with the highest measurement accuracy.

Many automated test assemblies [1], [2], [8], [10], [11], [12], [23] have solved a problem to minimize differences in ability measurement accuracies based on test information among tests. Nevertheless, they are not guaranteed to maximize the number of assembled test forms. Testing organizations must therefore allocate different tests to each examinee to secure the item contents.

To overcome this difficulty, Fuchimoto et al. [5] proposed a two-step parallel algorithm. The first step of their method assembles parallel test forms with an overlapping constraint as a maximum clique problem [3], [16] that has low time complexity but high space complexity. The clique is a graph structure, which is any two vertices with an edge. Here, the overlapping constraint (OC) restricts the number of overlapping items among tests. Figure 1 presents an outline of the maximum clique method. The method constructs a graph in which the vertices and the edges respectively represent tests satisfying the test constraint and tests satisfying the overlapping constraint. The maximum clique method extracts the maximum clique as parallel test forms from the graph.

The second step, using integer programming (IP) method [26] with low space complexity but high time complexity, repeats the parallel search of the tests satisfied with the overlapping constraint for all currently assembled tests. The IP for assembly of a new test is presented below.

where

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

maximize

$$\sum_{i=1}^{n} \lambda_i x_i, \tag{12}$$

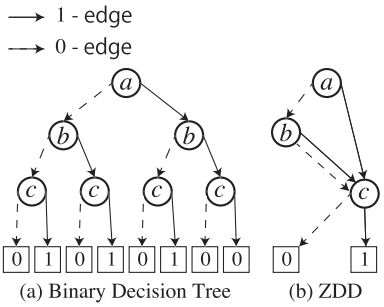

FIGURE 2. Binary decision tree and zero suppressed binary decision diagrams.

subject to

$$\sum_{i=1}^{n} x_i = \mathrm{M}, \tag{13}$$

$$LB_{\theta_k} \leq \sum_{i=1}^{n} IIF_i(\theta_k) x_i \leq UB_{\theta_k}, \tag{14}$$

$$(k = 1, 2, \ldots, K)$$

$$\sum_{i=1}^{n} X_{i,r} x_i \leq OC,$$

$$(r = 1, 2, \ldots, |C|)$$

$$X_{i,r} = \begin{cases} 1 & \text{if the } i\text{–th item is selected in the } r\text{–th test} \\ & \text{in the current clique } C, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Therein, $\lambda_1, \lambda_2, \ldots, \lambda_n$ respectively denote random variables distributed uniformly on [0,1], LB_{θ_k} and UB_{θ_k} denote a lower bound and an upper bound for the test information function on $TIF_{Test}(\theta_k)$. Actually, $\{\lambda_i\}(0 \leq i \leq n)$ are resampled after each problem is solved.

IV. PROPOSED METHOD

Currently, HMCAPIP is known to be capable of assembling the greatest number of test forms. Nevertheless, the improvement of HMCAPIP is constrained because of the high time complexity of IP in the second step. Particularly, HMCAPIP requires one week or more to assemble 300,000 tests because the IP computation time increases along with the number of assembled test forms. Moreover, the HMCAPIP parallel search performance depends on the computer specifications.

To resolve these difficulties, we propose a new automated test assembly: ATA-ZDD (Automated Test Assembly using ZDD). According to the proposed method, each node in the binary decision tree corresponds to an item of an item pool. Each node has two edges, respectively representing that the corresponding item is included in a test form, or not. The binary decision tree can enumerate all parallel test forms with the satisfied test constraints. Nevertheless, the binary decision tree has high space complexity $O(2^n/n)$. To relax this high space complexity, the proposed method

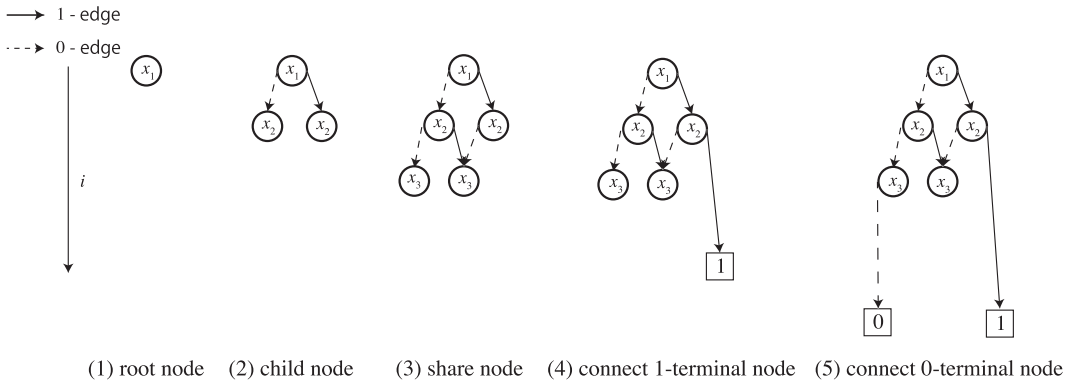

FIGURE 3. Outline of the proposed method.

compresses the binary decision tree using ZDD based on a breadth-first manner [22], [27], which is known as an efficient construction technique for reducing the necessities for computer memory and calculation time. Specifically, all equivalent nodes are shared, provided that they have the same measurement accuracies and the same test lengths. Because of this feature, the proposed method can assemble the parallel test forms without redundantly repeating calculations for the measurement accuracies and the test lengths.

A. ZERO SUPPRESSED BINARY DECISION DIAGRAMS (ZDD)

Actually, Zero Suppressed Binary Decision Diagrams (ZDD) [17] constitute an efficient data structure for a set of item combinations from a finite set.

Figures 2(a) and 2(b) respectively depict examples of binary decision tree and zero-suppressed binary decision diagrams. These data structures have terminal nodes of two kinds represented as rectangles in Fig. 2: 0-terminal and 1-terminal. Paths from the root node to the 1-terminal node in these data structures correspond to a set of item combinations. Then, every non-terminal node shown as a circle in Fig. 2 is labeled by an element of a finite set with order. Moreover, each has two edges: the 0-edge and 1-edge. The 1-edge (0-edge) signifies that the parent node is included (not included) in the item combinations. Therefore, the data structures of Fig. 2 correspond to the same set of item combinations, which is $\{\{a, c\}, \{b, c\}, \{c\}\}$.

The salient difference between data structures of Figs. 2(a) and 2(b) is their degree of compression efficiency. In fact, ZDD can enumerate only three nodes for the set of item combinations in Fig. 2(b). To construct efficient data structures, Minato [17] applies the following two reduction rules to the binary decision tree:

- Share all equivalent nodes having the same item and the same pair of children.
- Delete all nodes for which 1-edge points directly to the 0-terminal node.

By the two reduction rules, ZDD can enumerate all feasible solutions efficiently with a reasonable calculation time and memory limitation.

B. AUTOMATED TEST ASSEMBLY USING ZDD (ATA-ZDD)

This section presents a new automated test assembly: ATA-ZDD. For automated test assembly using ZDD, the analyses used for this study define a finite set $I = \{x_1, x_2, \ldots, x_n\}$ with ordered items, where n is the item pool size. Then $S \subseteq I$ is designated as an item combination. For a given test constraint function $f : 2^S \to \{0, 1\}$, the families of sets $\mathcal{T}$ for f are denoted as $\mathcal{T} = \{S \subseteq I \mid f(S) = 1\}$, which are called the parallel test forms. Here, the test constraints are defined formally as presented below.

$$|S| = \mathrm{M} \text{ (test lengths)}, \tag{16}$$

$$LB_{\theta_k} \leq \sum_{x_i \in S} IIF_{x_i}(\theta_k) \leq UB_{\theta_k}.$$
$$(k = 1, 2, \ldots, K)$$
$$\text{(test information constraints)} \tag{17}$$

The main idea of ATA-ZDD is to enumerate the item combinations satisfying these test constraints using ZDD. Here, we describe the use a test length variable as TL and a test information array as $TI = [ti_1, \ldots, ti_k, \ldots, ti_K]$ for using the breadth-first technique [22], [27] and branch-and-bound. The test length variable and each element of the test information array respectively correspond to $|S|$ (eq. 16) and $\sum_{x_i \in S} IIF_{x_i}(\theta_k)$ (eq. 17) at ability level θ_k. The main algorithm of ATA-ZDD consists of five steps, as presented in Fig. 3.

1) Step 1 generates a root node x_1 and sets zero for the test length variable TL and each element $TI[k]$ of the test information array.
2) Step 2 generates a 0-child node and a 1-child node with a 0-edge and a 1-edge. Then, step 2 adds one to the test length variable TL of 1-child nodes. Furthermore, step 2 adds the item information $I_i(\theta_k)$ of i to each element $TI[k]$ of the test information array of 1-child nodes.
3) Step 3 shares all equivalent nodes (having the equivalent test length variable TL and the equivalent test information array TI). Nevertheless, ATA-ZDD shares those nodes when the difference in test information between two nodes is less than the threshold value

Algorithm 1 ATA-ZDD

```
procedure ATA-ZDD(n, M)
    create a new node v_root                                   ▷ root node
    v_root.state.TL ← 0
    for k ← 1 to K do
        v_root.state.TI[k] ← 0
    end for
    N_1 ← {v_root}                                             ▷ N_i is node sets of depth i
    for i ← 2 to n do
        N_i ← ∅
    end for
    N_{n+1} ← {0-terminal node, 1-terminal node}
    for i ← 1 to n do
        for each v ∈ N_i do
            for each x ∈ {0, 1} do                             ▷ 0-edge, 1-edge
                {i', state'} ← Child(i, M, v.state, x)
                                  ▷ i' is the depth of the child node. state' is TL and TI of the child node.
                v' ←create a new node                          ▷ child node
                if {i', state'} is {n + 1, 0} then
                    v' ← 0-terminal node
                else if {i', state'} is {n + 1, 1} then
                    v' ← 1-terminal node
                else
                    v'.state ← state'
                    share_node ← False
                    for each w ∈ N_{i+1} do
                        if v'.state.TL = w.state.TL then
                            for k ← 1 to K do
                                if I_th ≤ |v'.state.TI[k] − w.state.TI[k]| then
                                    next w
                                end if
                            end for
                            UpdateState(v', w)
                            v' ← w                             ▷ share node
                            share_node ← True
                            break
                        end if
                    end for each
                    if share_node is False then
                        N_{i+1} ← N_{i+1} ∪ v'
                    end if
                end if
                v.child[x]← v'
            end for each
        end for each
    end for
    return Reduce(v_root)
end procedure
procedure Child(i, M, state, x)
    if x = 1 then
        state'.TL ← state.TL + 1
        for k ← 1 to K do
            state'.TI[k] ← state.TI[k] + IIF_i(θ_k)             ▷ IIF_i(θ_k) in eq(2)
        end for
    end if
    if state'.TL = M then
        for k ← 1 to K do
            if not LB_{θ_k} < state'.TI[k] < UB_{θ_k} then
                return {n + 1, 0}                              ▷ 0-terminal node
            end if
        end for
        return {n + 1, 1}                                      ▷ 1-terminal node
    end if
    if state'.TL + n − i < M then
        for k ← 1 to K do
            if UB_{θ_k} < state'.TI[k] then
                return {n + 1, 0}                              ▷ 0-terminal node
            end if
        end for
    end if
    return {i + 1, state'}
end procedure
procedure UpdateState(v', w)
    for k ← 1 to K do
        w.state.TI[k] ← (v'.state.TI[k] + w.state.TI[k])/2
    end for
end procedure
```

I_{th} because the test information arrays of two nodes are rarely equivalent. Then, each element of the test information array is approximated by the average of the test information elements of two nodes when the nodes are shared.

4) Step 4 connects 1-edge to 1-terminal node if the test length variable TL and the test information array TI satisfy the test constraints (eq. 16, eq. 17) after Step 3.
5) Step 5 prunes a 1-edge or 0-edge that does not satisfy the test constraint by connecting the 0-terminal node. The pruning has the three conditions presented below.
   1. M $< TL$.
   2. $UB_{\theta_k} < TI[k] \quad (k = 1, 2, \ldots, K)$.
   3. $M = TL \quad and \quad TI[k] < LB_{\theta_k} \quad (k = 1, 2, \ldots, K)$.
6) ATA-ZDD repeats Step 2 through Step 5 for all items of finite set I according to the order. Then, ATA-ZDD applies the reduction rules to the constructed ZDD.

The ATA-ZDD algorithm is presented as Algorithm 1. Algorithm 1 is unable to enumerate exactly parallel test forms to satisfy the test information constraints (eq. 17) because each element of the test information array for shared nodes is approximated by the average of the test information of two nodes in Step 3. Furthermore, this ZDD cannot control overlapping items.

To resolve these shortcomings, ATA-ZDD enumerates test forms that satisfy these constraints exactly using random sampling [17] with low time complexity O(n) from ZDD.

For this discussion, we define the families of sets with the test constraints and the overlapping constraints as parallel test forms with overlapping items $\mathcal{T}_{OC}$.

The algorithm consists of the following three steps.

1) Step 1 sets $\mathcal{T}_{OC}$ to ϕ.
2) Step 2 searches for a test form using random sampling [17] from ZDD ($T \in \mathcal{T}$). Here, the random sampling has low time complexity $O(n)$.
3) Step 3 checks that the random sampling test form T satisfies the test information constraints (eq. 17) and the overlapping constraints with all parallel test forms with overlapping items $\mathcal{T}_{OC}$ ($\forall T_{OC} \in \mathcal{T}_{OC} \mid (|T_{OC} \cap T| \leq OC)$). When the random sampling test form T satisfies these constraints, it is added to the parallel test forms with overlapping items $\mathcal{T}_{OC}$ ($\mathcal{T}_{OC} \leftarrow \mathcal{T}_{OC} \cup T$).
4) The proposed method repeats Step 1 through Step 3 until a determined calculation time is reached.

This algorithm is shown in Algorithm 2 as ATA-ZDD with OC.

Thus, ATA-ZDD with OC can enumerate parallel test forms with the equivalent test information and with the number of overlapping items that is less than a determined value via repeating random sampling [17] from ZDD with low time complexity $O(n)$.

V. EXPERIMENTS

This section presents experiments demonstrating the effectiveness of the proposed method: ATA-ZDD.

A. EFFECTIVENESS OF THE THRESHOLD PARAMETER

The proposed method has a tradeoff by threshold parameter I_{th} between the number of assembled test forms and that of nodes for ZDD construction. Therefore, we evaluate the tradeoff by changing the value of the threshold parameter to obtain the optimal value maximizing the number of assembled test forms. We compare the performances by changing the values of the threshold parameter I_{th} using both simulated and actual item pools. Items in the simulated item pools have discrimination parameters and difficulty

Algorithm 2 ATA-ZDD with OC

procedure ATA-ZDD with OC(OC)
  $\mathcal{T}_{OC} \leftarrow \emptyset$
  while within a time limitation **do**
    for each $T \in \mathcal{T}$ **do** ▷ random sampling
      for each $k \in K$ **do**
        $TIF_{\theta_k} \leftarrow CalculateTestInfo(T, \theta_k)$ ▷ eq(3)
        if $TIF_{\theta_k} < LB_{\theta_k}$ *or* $UB_{\theta_k} < TIF_{\theta_k}$ **then**
          ▷ Test information constraint
          Next T
        end if
      end for
      for each $T_{OC} \in \mathcal{T}_{OC}$ **do**
        if OC $< |T \cap T_{OC}|$ **then**
          ▷ Overlap constraint
          Next T
        end if
      end for
      $\mathcal{T}_{OC} \leftarrow \mathcal{T}_{OC} \cup T$
    end for
  end while
  return $\mathcal{T}_{OC}$
end procedure

TABLE 1. Details of the actual item pool.

Item Pool Size	Parameter a			Parameter b		
	Range	Mean	SD	Range	Mean	SD
87	0.15 – −0.67	0.35	0.13	−2.09 – −4.55	0.73	1.62
93	0.19 – −0.69	0.43	0.12	−3.92 – −3.61	−0.79	1.19
104	0.13 – −1.10	0.59	0.21	−0.18 – −4.55	1.50	1.18
141	0.24 – −1.09	0.64	0.15	−1.41 – −3.91	0.60	0.85
158	0.15 – −3.08	0.44	0.25	−4.00 – −4.00	−1.12	1.43
175	0.12 – −0.93	0.39	0.13	−2.93 – −3.12	−0.25	1.11
220	0.16 – −0.92	0.46	0.15	−4.00 – −2.82	−1.28	1.09
Total 978	0.12 – −3.08	0.46	0.19	−4.00 – −4.55	−0.22	1.57

TABLE 2. Test information constraints for automated test assembly.

TIF(θ)(Lower bound / Upper bound)				
$\theta = -2.0$	$\theta = -1.0$	$\theta = 0.0$	$\theta = 1.0$	$\theta = 2.0$
8.0/9.6	12.8/14.4	12.8/14.4	12.8/14.4	8.0/9.6

parameters according to the two-parameter logistic model of IRT. We generated discrimination parameters as $\log_2 a \sim N(0, 1^2)$ and difficulty parameters as $b \sim N(0, 1^2)$. A detailed description of the actual item pool is presented in Table 1. This actual item pool is used for the synthetic personality inventory (SPI) examination: a popular Japanese aptitude test [28].

For this study, the constraints applied for the test are presented below.

1) The test contains 100 items.
2) The test information constraints are described by the lower and upper bounds of the test information function in Table 2.

We determined these constraints according to the actual test setting [28]. The threshold parameter of ATA-ZDD is changed from 0.01 to 0.45 in 0.01 steps. Overlapping items are not controlled for this experiment.

Here, we conduct experiments using a computer (Ryzen 9 5950X 3.40 GHz CPU; AMD Corp.) with 128 GB main memory and operating system (64bit Ubuntu; Linux).

Table 3 presents the number of assembled test forms by changing the value of I_{th}. In the table, No. tests signifies the number of assembled test forms, Compression Ratio expresses the number of nodes in ZDD divided by the number of nodes in the binary decision tree, and Time denotes the calculation time (minute) for ZDD construction.

As described previously, the proposed method entails a tradeoff between the number of assembled test forms and that of nodes for ZDD construction. When the threshold parameter I_{th} is small, the proposed method tends to assemble the greatest number of assembled test forms, but it causes memory overflow.

When the threshold parameter I_{th} becomes large, the calculation time and the number of nodes for ZDD construction decreases because the number of shared nodes is increased.

Therefore, to decrease the calculation time and memory limitations, the threshold parameter should be increased. As described herein, the threshold is determined by the value which can assemble the greatest number of assembled test forms for each item pool.

Regarded in greater detail, Table 4 shows the number of prunings in the Algorithm. 1 and the compression ratio obtained using the reduction rules. When the threshold parameter I_{th} becomes sufficiently small, the number of prunings increases because the number of shared nodes decreases and the number of branches increases. On the other hand, differences in the compression ratio with reduction rules are small, irrespective of the threshold parameter I_{th}.

B. COMPARISON OF ATA-ZDD TO CONVENTIONALLY USED METHODS

To demonstrate the ATA-ZDD benefits, we compare the number of assembled test forms of ATA-ZDD with those produced using the traditional methods of (Big Shadow Test Method [8] (designated for comparison as BST) and Hybrid Maximum Clique Algorithm with Parallel Integer Programming (designated for comparison as s HMCAPIP) [5]) using simulated and actual item pools described in the preceding subsection.

We set the test constraints as presented below.

1) The test contains 100 items.
2) The allowed maximum numbers of overlapping items are changed from two to thirty by increments of two.
3) The time limitation of all methods is 24 hr.
4) The test information constraints are the same as those presented in the preceding subsection.

We determined the parameter values of HMCAPIP according to an explanation given in the literature by Fuchimoto et al. [5]. For HMCAPIP and BST, we apply CPLEX [29] for the integer programming problem.

Table 5 presents the numbers of assembled test forms produced using the proposed method and using the traditional

TABLE 3. Number of tests, number of vertices of ZDD, and calculation times by changing the l_{th} parameter

l_{th}	Item Pool Size = 1000			Item Pool Size = 2000			Item Pool Size = 978		
	No. tests	Compression ratio	Time	No. tests	Compression ratio	Time	No. tests	Compression ratio	Time
under 0.27		Memory Over			Memory Over			Memory Over	
0.28		Memory Over			Memory Over		4.65×10^{127}	3.77×10^{-288}	1202
0.29		Memory Over			Memory Over		4.21×10^{126}	2.37×10^{-288}	220
0.30		Memory Over			Memory Over		3.01×10^{128}	1.57×10^{-288}	193
0.31		Memory Over			Memory Over		1.34×10^{130}	1.18×10^{-288}	142
0.32		Memory Over			Memory Over		9.11×10^{123}	8.04×10^{-289}	99
0.33		Memory Over			Memory Over		1.56×10^{123}	8.22×10^{-289}	78
0.34	1.67×10^{136}	1.20×10^{-293}	1122		Memory Over		5.29×10^{120}	5.50×10^{-289}	61
0.35	1.43×10^{136}	8.33×10^{-294}	654		Memory Over		7.82×10^{124}	3.73×10^{-289}	40
0.36	1.38×10^{136}	6.84×10^{-294}	338		Memory Over		3.11×10^{117}	2.05×10^{-289}	39
0.37	8.12×10^{135}	6.00×10^{-294}	258		Memory Over		5.33×10^{120}	3.24×10^{-289}	24
0.38	3.20×10^{136}	4.41×10^{-294}	132	2.11×10^{167}	2.16×10^{-594}	1361	1.17×10^{129}	3.08×10^{-289}	11
0.39	1.03×10^{136}	3.25×10^{-294}	78	6.34×10^{166}	1.85×10^{-594}	959	1.07×10^{128}	2.65×10^{-289}	5
0.40	9.10×10^{135}	2.51×10^{-294}	49	3.66×10^{167}	1.44×10^{-594}	550	9.87×10^{123}	1.57×10^{-289}	5
0.41	1.51×10^{136}	1.73×10^{-294}	25	2.56×10^{167}	1.21×10^{-594}	330	2.73×10^{119}	1.19×10^{-289}	5
0.42	1.93×10^{135}	1.37×10^{-294}	17	2.11×10^{167}	8.95×10^{-595}	260	1.75×10^{121}	9.59×10^{-290}	3
0.43	1.51×10^{136}	1.08×10^{-294}	12	1.05×10^{167}	7.99×10^{-595}	210	1.31×10^{120}	8.53×10^{-290}	3
0.44	9.54×10^{135}	8.54×10^{-295}	8	4.18×10^{167}	6.13×10^{-595}	133	8.62×10^{116}	8.21×10^{-290}	3
0.45	3.48×10^{135}	7.73×10^{-295}	7	1.13×10^{167}	4.77×10^{-595}	85	2.19×10^{119}	4.15×10^{-290}	2

TABLE 4. Number of pruning and number of share nodes of ZDD by changing the parameter l_{th}.

l_{th}	Item Pool Size = 1000			Item Pool Size = 2000			Item Pool Size = 978		
	No. pruning	Reduction rule		No. pruning	Reduction rule		No. pruning	Reduction rule	
under 0.27		Memory Over			Memory Over			Memory Over	
0.28		Memory Over			Memory Over		5892801	0.08	
0.29		Memory Over			Memory Over		5389210	0.01	
0.30		Memory Over			Memory Over		4868817	0.06	
0.31		Memory Over			Memory Over		4100665	0.06	
0.32		Memory Over			Memory Over		3413287	0.04	
0.33		Memory Over			Memory Over		3034305	0.05	
0.34		Memory Over			Memory Over		2692426	0.04	
0.35	8234729	0.78		15892039	Memory Over		2070770	0.03	
0.36	7537993	0.79		14591090	Memory Over		1939610	0.02	
0.37	6636419	0.78		14023900	Memory Over		1865923	0.04	
0.38	4852511	0.80		17023890	Memory Over		1179711	0.06	
0.39	3634472	0.79		15892039	0.63		719357	0.00	
0.40	2794332	0.79		14591090	0.63		857468	0.04	
0.41	1921650	0.79		14023900	0.63		827739	0.03	
0.42	1523517	0.78		13716345	0.63		612405	0.03	
0.43	1196380	0.78		12961675	0.63		547942	0.00	
0.44	956115	0.78		10889604	0.64		547434	0.03	
0.45	901037	0.77		8359699	0.64		435407	0.02	

methods, by changing item pool sizes and overlapping constraints.

When OC becomes large, the proposed method can assemble a greater number of assembled test forms than the traditional methods do. It is noteworthy that the proposed method can assemble the 1,500,000 test forms within 24 hr, which would take one month or more if using conventional methods. These findings suggest that the proposed method has important advantages in large-scale testing, which necessitates the number of test forms.

Even when OC becomes small, HMCAPIP can assemble a slightly greater number of tests than other methods do. The results demonstrate that the numbers converge to the maximum number of assembled test forms because, as a result of the tight OC, the exact maximum number of test forms is not large. Nevertheless, because the test information is approximated by the mean when the nodes of ZDD are shared, the proposed method does not guarantee exactly the maximum number of assembled test forms. Therefore, the performance of the proposed method is limited with a tight OC.

When $OC \geq 24$, because of the time complexity $O(n)$ of random sampling, the proposed method for item pool size = 1000 can assemble a greater number of test forms than the proposed method for item pool size = 2000 does. In fact, when $OC = 24$, the proposed method for item pool size = 1000 samples 1,640,000,000 test forms, but the proposed method for item pool size = 2000 samples only 630,000,000 test forms.

Nevertheless, because each element of the test information array is approximated by the mean when nodes are shared, ATA-ZDD does not guarantee exactly all enumerations with satisfaction of the test constraints. To evaluate the effectiveness of this approximation, we simulate the percentage of paths which satisfy the test information constraint when using random sampling. Specifically, when the random sampling is repeated one billion times, this experiment calculates the percentage of paths which strictly satisfy the constraints of the test information.

TABLE 5. Numbers of tests assembled in 24 hours.

Item Pool Size	OC	BST	HMCAPIP	Proposal
1000	2	5	**18**	2
	4	6	**19**	4
	6	7	**20**	5
	8	7	**24**	7
	10	10	**30**	13
	12	17	**43**	28
	14	23	**81**	70
	16	56	200	**220**
	18	61	650	**800**
	20	67	2198	**3654**
	22	72	2264	**16500**
	24	75	4240	**57996**
	26	81	4931	**108690**
	28	87	5658	**121255**
	30	100	6174	**124367**
	32	102	6189	**124400**
	34	104	6221	**124423**
	36	109	6235	**124449**
	38	111	6372	**125123**
	40	117	6514	**125192**
2000	2	5	**22**	4
	4	6	**28**	10
	6	9	**45**	26
	8	17	**103**	77
	10	23	**379**	342
	12	46	1802	**1879**
	14	76	6779	**9991**
	16	102	10163	**34422**
	18	109	10163	**52232**
	20	113	11122	**52271**
	22	126	11219	**53223**
	24	127	11798	**53311**
	26	128	11976	**53480**
	28	130	12121	**54126**
	30	135	12273	**54212**
	32	141	12333	**54392**
	34	144	12342	**54423**
	36	144	12365	**54446**
	38	149	12383	**54876**
	40	154	12412	**54928**
978 (actual)	2	3	**18**	2
	4	4	**20**	3
	6	5	**22**	4
	8	5	**26**	6
	10	7	**35**	10
	12	9	**59**	21
	14	12	127	**137**
	16	33	403	**473**
	18	37	1537	**1895**
	20	44	7038	**8783**
	22	160	30808	**45243**
	24	256	97423	**147086**
	26	259	97823	**853324**
	28	261	99852	**1545602**
	30	265	99999	**1546212**
	32	268	100021	**1548327**
	34	273	100112	**1548452**
	36	275	100222	**1548498**
	38	281	100258	**1548675**
	40	286	100314	**1548902**

TABLE 6. Percentage of paths satisfying the test information constraint.

Item Pool Size	I_{th}	Percentage of paths
1000	0.17	0.07
2000	0.20	0.07
978	0.16	0.51

Table 6 shows that the percentage of paths that satisfy the constraints of the test information. The results suggest that the proposed method still has room for improvement in the threshold parameter because the number of strictly satisfied tests decreases as the amount of node-sharing increases.

VI. CONCLUSION

We proposed a new automated test assembly: ATA-ZDD. According to this proposed method, each node in the binary decision tree corresponds to an item of an item pool. Each node has two edges, respectively signifying that the corresponding item is included in a test form, or not. Furthermore, all equivalent nodes are shared, providing that they have the same measurement accuracy and the same test length.

Numerical experiments demonstrated that the proposed method assembled a greater number of test forms than the conventional methods did. It is noteworthy that the proposed method was able to assemble 1,500,000 test forms within 24 hr, whereas a currently widely used method assembled only 300,000 test forms within 10 days. Moreover, these results suggest that the drastically different quantities of assembled parallel test forms created using these methods would differ increasingly with extended calculation time.

Nevertheless, ATA-ZDD does not guarantee exactly all enumerations with the satisfaction of test constraints: each element of the test information array is approximated by the mean when nodes are shared. Therefore, the proposed method still has room for improvement of its threshold parameter optimization because the number of strictly satisfied tests is reduced as the number of node-sharing increases.

Furthermore, this study specifically examines automated test assembly using only those constraints related to the test length and test information. In fact, actual examinations require other test constraints. For instance, the proposed method does not regulate the number of times each item has been used in the assembled test forms. Therefore, the distribution of item usage counts is not uniform, which is designated as an item exposure bias problem [30]. The exposure bias problem is known to impair the reliability of items and tests [30]. To resolve the item exposure problem, we expect to implement the proposed method using a probabilistic approach of item exposure such as probabilistic eligibility by Linden [31] in computerized adaptive testing (CAT).

REFERENCES

[1] 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. 21, no. 2, pp. 141–161, Apr. 2008.
[2] 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. 2011.
[3] 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. 2014.
[4] T. Ishii and M. Ueno, "Clique algorithm to minimize item exposure for uniform test forms assembly," in *Proc. Int. Conf. Artif. Intell. Educ.* Cham, Switzerland: Springer, 2015, pp. 638–641.
[5] K. Fuchimoto, T. Ishii, and M. Ueno, "Hybrid maximum clique algorithm using parallel integer programming for uniform test assembly," *IEEE Trans. Learn. Technol.*, vol. 15, no. 2, pp. 252–264, Apr. 2022.

[6] W. J. van der Linden and M. Ueno, "Shadow-test approach to adaptive testing," *Behaviormetrika*, vol. 49, no. 2, pp. 165–167, 2022.
[7] M. Ueno and Y. Miyazawa, "Two-stage uniform adaptive testing to balance measurement accuracy and item exposure," in *Artificial Intelligence in Education*. Durham, U.K.: Springer, Jul. 2022, pp. 626–632.
[8] W. J. Van der Linden, *Linear Models for Optimal Test Design*. New York, NY, USA: Springer, 2005.
[9] W. J. Linden and J. J. Adema, "Simultaneous assembly of multiple test forms," *J. Educ. Meas.*, vol. 35, no. 3, pp. 185–198, Sep. 1998.
[10] E. Boekkooi-Timminga, "The construction of parallel tests from IRT-based item banks," *J. Educ. Statist.*, vol. 15, no. 2, p. 129, 1990.
[11] 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, Mar. 1994.
[12] 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, Sep. 1998.
[13] 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.
[14] J. Pereira and M. Vilà, "Variable neighborhood search heuristics for a test assembly design problem," *Expert Syst. Appl.*, vol. 42, no. 10, pp. 4805–4817, Jun. 2015.
[15] 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, nos. 2–3, pp. 319–332, Mar. 2006.
[16] T. Ishii, P. Songmuang, and M. Ueno, "Maximum clique algorithm for uniform test forms assembly," in *Proc. 16th Int. Conf. Artif. Intell. Educ.*, 2013, pp. 451–462.
[17] S.-I. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," in *Proc. 30th Int. Design Autom. Conf.*, 1993, pp. 272–277.
[18] T. Inoue, K. Takano, T. Watanabe, J. Kawahara, R. Yoshinaka, A. Kishimoto, K. Tsuda, S.-I. Minato, and Y. Hayashi, "Distribution loss minimization with guaranteed error bound," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 102–111, Jan. 2014.
[19] A. Takizawa, Y. Takechi, A. Ohta, N. Katoh, T. Inoue, T. Horiyama, J. Kawahara, and S.-I. Minato, "Enumeration of region partitioning for evacuation planning based on ZDD," in *Proc. 11th Int. Symp. Oper. Res. Appl. Eng.* Huangshan, China: IET, 2013, pp. 1–8.
[20] A. Takizawa, Y. Miyata, and N. Katoh, "Enumeration of floor plans based on a zero-suppressed binary decision diagram," *Int. J. Architectural Comput.*, vol. 13, no. 1, pp. 25–44, Mar. 2015.
[21] S. Sakaue and K. Nakamura, "Differentiable equilibrium computation with decision diagrams for Stackelberg models of combinatorial congestion games," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 9416–9428.
[22] H. Iwashita and S.-I. Minato, "Efficient top-down ZDD construction techniques using recursive specifications," Hokkaido Univ., Sapporo, Japan, Tech. Rep. TCS-TR-A-13-69, 2013.
[23] W. J. van ver 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.
[24] F. M. Lord and M. R. Novick, *Statistical Theories of Mental Test Scores*. Reading, MA, USA: Addison-Wesley, 1968.
[25] F. B. Baker and S. H. Kim, *Item Response Theory: Parameter Estimation Techniques* (Statistics: A Series of Textbooks and Monographs), 2nd ed. New York, NY, USA: Taylor & Francis, 2004.
[26] T. Ishii and M. Ueno, "Algorithm for uniform test assembly using a maximum clique problem and integer programming," in *Artificial Intelligence in Education*. Cham, Switzerland: Springer, 2017, pp. 102–112.
[27] D. E. Knuth, "The art of computer programming: Bitwise tricks & techniques," *Binary Decis. Diagrams*, vol. 4, pp. 47–135, Mar. 2009.
[28] *Synthetic Personality Inventory (SPI)*, Recruit., Tokyo, Japan, 2023.
[29] *ILOG CPLEX Optimization Studio CPLEX User's Manual 12.9*, IBM, Armonk, NY, USA, 2019.
[30] H. Wainer, "Rescuing computerized testing by breaking Zipf's law," *J. Educ. Behav. Statist.*, vol. 25, no. 2, pp. 203–224, Jun. 2000.
[31] W. J. van der Linden and S. W. Choi, "Improving item-exposure control in adaptive testing," *J. Educ. Meas.*, vol. 57, no. 3, pp. 405–422, Sep. 2020.

KAZUMA FUCHIMOTO received the B.E. and M.E. degrees from The University of Electro-Communications, in 2020 and 2022, respectively, where he is currently pursuing the D.E. degree. His research interests include educational technology and computer science.

SHIN-ICHI MINATO (Senior Member, IEEE) received the B.E., M.E., and D.E. degrees from Kyoto University, in 1988, 1990, and 1995, respectively. He was with the NTT Laboratories, from 1990 to 2004. He was a Visiting Scholar with Stanford University, in 1997. He joined Hokkaido University as an Associate Professor, in 2004, where he has been a Professor, since October 2010. Since April 2018, he has been a Professor with Kyoto University (present position), where he is currently a Professor with the Graduate School of Informatics. His research interests include efficient representations and manipulation algorithms for large-scale discrete structures, such as Boolean functions, sets of combinations, sequences, and permutations. In addition to being a Senior Member of IEICE and IPSJ. He is a member of JSAI and JSCS.

MAOMI UENO (Member, IEEE) received the Ph.D. degree in computer science from the Tokyo Institute of Technology, in 1994. He has been a Professor with the Graduate School of Information Systems, The University of Electro-Communications, since 2013. His research interests include machine learning, data mining, Bayesian statistics, Bayesian networks, and educational technology. He was conferred the Best Paper Award at IEEE ICTAI2008.