Bees Algorithm for Construction of Multiple Test Forms in E-Testing

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

Abstract—The purpose of this research is to automatically construct multiple equivalent test forms that have equivalent qualities indicated by test information functions based on item response theory. There has been a trade-off in previous studies between the computational costs and the equivalent qualities of test forms. To alleviate this problem, we propose an automated system of test construction based on the Bees Algorithm in parallel computing. We demonstrate the effectiveness of the proposed system through various experiments.

Index Terms—E-testing, multiple test forms, test construction, Bees algorithm, parallel computing.

1 Introduction

Educational assessments occasionally need "multiple test forms" in which each form consists of a different set of items but still has qualities that are equivalent (e.g., equivalent amounts of test information based on item response theory (IRT)) to the others. For example, multiple test forms are needed when a testing organization administers a test in different time slots. To achieve this, multiple test forms are constructed in which all forms have equivalent qualities so that examinees, who have taken different test forms, can be objectively evaluated on the same scale.

In order to construct multiple test forms, e-testing, which accomplishes automated test construction, has recently become popular in research areas involving educational measurement [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]. The methods in previous studies have been used to construct all forms of a test to satisfy the same test constraints (e.g., the number of test items and the amount of test information) to ensure that all forms have equivalent qualities. Van der Linden and Boekkooi-Timminga [6] proposed a sequential method of constructing test forms using linear programming to minimize the fitting errors to the test constraints. The items that had been used for constructing the test were removed from the item bank and then the next test forms were constructed from the remaining items. This method was called "sequential construction." However, there was a serious problem in that the fitting errors in these methods increased as the number of constructed test forms increased.

- *P. Songmuang is with the Faculty of Human Sciences, Waseda University, 2-579-15, Mikashima, Tokorozawa-shi, Saitama-ken, 359-1192, Japan. E-mail: pokpong@aoni.waseda.jp.*
- *M. Ueno is with the Department of Social Intelligence and Informatics, Graduate School of Information Systems, The University of Electro-Communications, 1-5-1, Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan. E-mail: ueno@ai.is.uec.ac.jp.*

Manuscript received 23 Dec. 2009; revised 22 Mar. 2010; accepted 3 Aug. 2010; published online 27 Aug. 2010.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2009-12-0205.
Digital Object Identifier no. 10.1109/TLT.2010.29.

To solve this problem, Boekkooi-Timminga [16] and Armstrong et al. [14] proposed methods that simultaneously constructed all test forms to minimize the differences in the fitting errors on the test forms. The former used linear programming and the latter used network-flow programming. Although the differences in the fitting errors on the test forms were minimized, the computational costs of these methods exponentially increased as the size of the item bank or the number of test constraints increased.

To reduce the computational costs, van der Linden [19] proposed a big-shadow-test (BST) method that sequentially constructed test forms by minimizing the difference in fitting errors between a current constructed test form and the remaining set of items in the item bank. BST mitigated the problem with computational costs, it did not fundamentally solve the problem.

Another problem with these methods is that they did not take into consideration the maximum number of possible test forms from an item bank. Namely, none of them guaranteed the maximum number of test forms from an item bank. To solve this, Belov and Armstrong [30] formalized the test constructions to maximize the number of test forms with nonoverlapping (i.e., neither of two test forms had a common item; otherwise, it was called an overlapping item) constraints as maximum set-packing problems. However, nonoverlapping conditions interrupt the generation of a sufficiently large number of test forms from an item bank. That is, nonoverlapping conditions interrupt the effective use of an item bank. To solve this problem, Ishii et al. [20] applied a maximum clique technique to the construction of multiple test forms. This method guaranteed the maximum number of test forms with overlapping items. However, the computational costs also exponentially increased as the item bank increased in size. This meant that it was difficult to apply the method in practice. Thus, although BST cannot guarantee the maximum number of test forms, it is still practically the most useful method. However, it is difficult to use BST with overlapping constraints. That is, the item bank cannot effectively be used by BST.

The proposed method in this paper approximates the optimum search approaches such as [20] and [30] using

random search algorithm. Namely, the method cannot guarantee the maximum number of test forms, but asymptotically or approximately guarantees it. In addition, this method can be utilized for overlapping constraints.

On the other hand, the approximation method still remains the trade-off between the differences in fitting errors and computational costs. Therefore, the proposed method mitigates the trade-off by applying a parallel-computing technique that distributes the computational costs to multiple processors without increasing the differences in fitting errors.

Several studies have used random search algorithms in parallel-computing environments to solve optimization problems. For example, He et al. [33] and Borovska [34] used Genetic Algorithm (GA) and Pirim et al. [35] used a tabu search. Moreover, some studies [36], [37], [38] have compared the efficiencies of random search algorithms using various optimization problems such as engineering optimization problems, a traveling salesman problem, and complex combination problems. The results of these studies revealed that Bees Algorithm (BA) provided the best accuracies for optimal solutions regardless of the lowest computational time compared with simulated annealing (SIM), GA, and Ant Colony algorithm (ANT). Consequently, BA has a strong possibility to alleviate the trade-off in constructing multiple test forms since the problems employed in [36], [37], [38], and the constructing multiple test forms are all combinatorial optimization problems and classified as NP-hard problems.

Therefore, we propose a method of constructing multiple test forms based on BA in parallel computing. The proposed approach provides two main advantages:

1. It alleviates the trade-off between computational costs and differences in fitting errors.
2. It approximately maximizes the number of test forms with overlapping constraints.

Moreover, various experiments were carried out to evaluate how well the new method performed. The results demonstrated the proposed approach could be used to construct multiple test forms with lower differences in fitting errors in less computational time than that with established methods. Another experiment also revealed that the number of test forms constructed with the proposed approach increased as the number of overlapping items increased.

We developed an automated system of test construction and installed it in an actual e-testing system using the new approach.

2 Item Response Theory

Most previous studies on the construction of multiple test forms have employed item response theory to measure the quality of test forms [6], [13], [14], [16], [19], [30].

IRT is a modern test theory that describes the relationship between item characteristics and examinee abilities. IRT measures the abilities of examinees on a fixed scale instead of a fixed test for a fixed population. Therefore, we can measure examinee abilities on the same scale although they have taken test forms with different sets of items.

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

$$u_{ij} = \begin{cases} 1: & \text{Examinee } j \text{ answers item } i \text{ correctly,} \\ 0: & \text{Other cases.} \end{cases}$$

The probability of a correct answer to item i by examinee j with ability $\theta_j \in (-\infty, \infty)$ is assumed to follow the three-parameter logistic model as

$$p(u_{ij} = 1|\theta_j) = c_i + (1 - c_i)\frac{1}{1 + exp[-1.7a_i(\theta_j - b_i)]}, \quad (1)$$

where $a_i \in [0, \infty)$ is the ith item's discrimination parameter, $b_i \in (-\infty, \infty)$ is the ith item's difficulty parameter, and $c_i \in [0, 1]$ is the ith item's guessing probability parameter. The two-parameter logistic model and the Rasch model are obtained from (1) by subsequently setting $c_i = 0$ and $a_i = 1$.

The item information is a measure of how much discrimination an item provides at different ability levels. The Fisher information function based on the two-parameter logistic model is defined as

$$I_i(\theta_j) = a_i{}^2 p_i(\theta_j)[1 - p_i(\theta_j)], \quad (2)$$

where I_i is the information on item i and p_i is the probability of a correct answer to item i with ability θ_j.

To construct a test, a test author monitors the information functions of all items in the test using a test information function, which is the sum of the information functions of the test items. The test information function of a test including g items is defined as

$$I(\theta_j) = \sum_{i=1}^{g} I_i(\theta_j). \quad (3)$$

The traditional methods of constructing multiple test forms, which are described in the next section, construct all test forms so that they have equal qualities by minimizing the fitting errors indicated by the differences between the test information functions of the constructed test forms and the target values of the expected test information function at a set of the examinee's ability levels, $\Theta = \{\theta_1, \ldots, \theta_k, \ldots, \theta_K\}$. The values of the information function of item i at ability level θ_k are denoted as $I_i(\theta_k)$, $(i = 1, \ldots, m)$, and the target values are denoted as $T(\theta_k)$. We must note that the construction methods of multiple test forms were supposed to be implemented after each item's IRT parameters have been collected in the item bank.

3 Traditional Methods of Constructing Multiple Test Forms

3.1 Sequential Method of Constructing Test Forms

Van der Linden and Boekkooi-Timminga [6] proposed a method that sequentially constructs test forms using linear programming to minimize the following fitting errors:

$$\text{minimize} \quad \sum_{k=1}^{K} \left| \sum_{i=1}^{m} I_i(\theta_k) x_i - T(\theta_k) \right|, \quad (4)$$

where

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

Items that have been employed once are removed from the item bank. Therefore, the fitting errors increase as the number of constructed test forms rises.

3.2 Simultaneous Method of Constructing Test Forms

To reduce the differences in fitting errors between test forms, Boekkooi-Timminga [16] proposed a method using linear programming that simultaneously constructed multiple test forms and minimized the differences in fitting errors.

Let $f, (f = 1, \ldots, F)$ be the fth test form. The problem can be formalized as

$$\text{minimize } y \tag{5}$$

subject to

$$\sum_{k=1}^{K} \left| \sum_{i=1}^{m} I_i(\theta_k) x_{if} - T(\theta_k) \right| \leq y, \quad f = 1, \ldots, F, \tag{6}$$

where

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

However, it is known that the computational costs of this method exponentially increase as the data size increases.

3.3 Big-Shadow-Test Method

To mitigate the problem with computational costs in the simultaneous methods of constructing test forms, van der Linden [19] proposed a big-shadow-test method using linear programming that sequentially constructs test forms by minimizing the differences in fitting errors between a currently constructed test form and the set of items remaining in the item bank. They called their "shadow test form" the remaining items set.

The model for currently constructed test form and shadow test form is

$$\text{minimize } y \tag{7}$$

subject to

$$\sum_{k=1}^{K} \left| \sum_{i=1}^{m} I_i(\theta_k) x_i - T(\theta_k) \right| \leq y, \tag{8}$$

$$\sum_{k=1}^{K} \left| \sum_{i=1}^{m} I_i(\theta_k) z_i - T_s(\theta_k) \right| \leq y, \tag{9}$$

where

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

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

and T_s denotes the target value for the shadow test form. The combination of (7-9) minimizes the differences in fitting errors between the currently constructed test form and the items remaining in the item bank.

This method eases the computational costs and reduces the differences in fitting errors between the test forms. However, this does not fundamentally solve the problem with computational costs, which remains when large data sizes are used.

On the other hand, another problem with these methods is that they have not taken into consideration a maximum number of possible test forms from an item bank. This means that none of them can guarantee the maximum number of test forms.

3.4 Methods of Maximizing Number of Test Forms

Belov and Armstrong [30] proposed a method that formalizes the construction of multiple test forms to maximize the number of test forms from an item bank as maximum set-packing problems. Although this method guaranteed the maximum number of test forms from an item bank, no items were allowed to overlap in the test forms. This interrupted the generation of a sufficiently large of number of test forms from the item bank. Consequently, nonoverlapping conditions interrupted the item bank from being effectively used.

To solve this problem, Ishii et al. [20] applied the maximum clique technique to the construction of multiple test forms. Nevertheless, the computational costs exponentially increased as the data size increased. Namely, this method is difficult to implement in practice.

Therefore, although BST cannot guarantee the maximum number of test forms, it is still practically the most useful method. However, it is difficult to use BST with overlapping constraints.

3.5 Problems with Traditional Methods

The two main problems with traditional methods of constructing multiple test forms can be summarized below:

1. In order to construct equivalent test forms, the traditional methods enable test forms to be constructed that minimize the differences in fitting errors between all forms. However, the differences in fitting errors decrease as the computational costs increase. That is, there is a trade-off between the differences in fitting errors between the test forms and the computational costs.
2. A maximum number of test forms from an item bank that cannot be guaranteed and overlapping constraints are difficult to be implemented. That is, the item bank cannot effectively be used in practice.

The main purpose of the research discussed in this paper is to solve these two problems.

4 Methods of Constructing Multiple Test Forms Based on Bees Algorithm in Parallel Computing

4.1 Algorithm for Constructing Multiple Test Forms in Parallel Computing

In this paper, we propose a method which approximates the optimum search approaches such as [20] and [30] using random search algorithm. Namely, the proposed method cannot guarantee the maximum number of test forms, but asymptotically or approximately guarantees it. In addition, the proposed method can be utilized for overlapping constraints.

On the other hand, the approximation method still remains the trade-off between the differences in fitting errors and computational costs. Therefore, the proposed method mitigates the trade-off by applying a parallel-computing technique that distributes the computational costs to multiple processors without increasing the differences in fitting errors.

Several studies have used random search algorithms in parallel-computing environments to solve optimization problems. For example, He et al. [33] and Borovska [34] used GA and Pirim et al. [35] used a tabu search.

Moreover, some studies have compared the efficiencies of random search algorithms using various optimization problems. For example, Yang [36] compared GA and BA using engineering optimization problems. Wong et al. [37] compared ANT, GA, and BA using a traveling salesman problem. While Pham et al. [38] used complex combination problems to compare ANT, SIM, GA, and BA. The results of these studies revealed that BA provided the best accuracies for optimal solutions with the lowest computational time. Accordingly, BA has a strong possibility to alleviate the trade-off in constructing multiple test forms since the problems employed in [36], [37], [38], and the construction of multiple test forms are all combinatorial optimization problems and classified as NP-hard problems.

Consequently, we employ BA to construct the multiple test forms discussed in this paper.

4.2 Bees Algorithm

BA is an optimization algorithm inspired by the natural foraging behavior of honey bees to find the optimal solution [39]. Honey bees live in a hive where they store honey that they have foraged. Honey bees can communicate the locations of food sources to their hive mates by performing a so-called "waggle dance." The durations of this dance are proportional to the quantities of food at the sources. By engaging in this behavior, large groups of bees are recruited to forage sources that contain large quantities of food. This reduces the individual time required to forage for food.

To introduce the main idea behind BA, we will briefly describe the study by Wong et al. [37], which proposed BA for solving the traveling salesman problem.

The well-known traveling salesman problem is defined as follows: given n cities, find the shortest route that starts in a specific city, visit all other cities once, and finish in the starting node. This problem is the well-known NP-hard problem.

There is an outline of BA [37] for solving the traveling salesman problem in Fig. 1. First, artificial bees generate the initial population of routes (solutions) using a random search technique [40] to find cities they will visit next. Then, the initial population is evaluated to measure the total length of each route (fitness value). Second, the artificial bees iteratively improve the initial population. That is, the routes from the initial population are selected according to selection probabilities that are inversely proportional to the total lengths of the routes. After that, artificial bees are recruited to improve the selected routes. This method of recruiting is applied by observing the waggle dance of honey bees. The numbers of recruited artificial bees are inversely proportional to the total lengths of the selected routes. Then, the artificial bees generate a new population

```
Initialize a population of solutions with a random search
while The stopping criterion is not met do
  Evaluate the fitness of the population
  Select the solutions to a neighborhood search
  Recruit artificial bees to improve the selected solutions
  Generate a new population of solutions
end while
```

Fig. 1. Outline of Bees algorithm.

using a neighborhood-search technique [41] in which the artificial bees find shorter routes being influenced by the selected routes. Namely, the artificial bees select cities using selection probabilities that are inversely proportional to the distances between cities and the selection probabilities of cities that are included in the selected routes are higher than that of the other cities. The process in the second step is iterated until the stopping criterion is met.

4.3 Bees Algorithm for Constructing Multiple Test Forms

In this section, we propose a method of constructing test forms based on BA that constructs multiple equivalent test forms by minimizing the difference in fitting errors between test forms and maximizing the number of test forms. This construction has an approximate time complexity of $O(c \cdot m! \cdot 2^f)$, where c is the number of test constraints, m is the number of items in an item bank, and f is the number of constructed test forms that satisfy all test constraints. Therefore, the construction of multiple test forms is classified as an NP-hard problem. To reduce the computational time, we divided the construction of test forms into two steps (two-step test construction):

Step A (Satisfying Test Constraints). Construct test forms only to minimize the fitting errors of each form to test constraints without taking into consideration the equivalence of test forms. Therefore, the approximate time complexity of this step is $O(c \cdot m!)$. Here, the constructed test forms are still not equivalent.

Step B (Equating Test Forms with Maximizing the Number of Them). Extract the most equivalent set of test forms from the constructed test forms in Step A that minimizes the difference in fitting errors among test forms and maximizes the number of test forms. The approximate time complexity in this step is $O(2^f)$.

The time complexity for constructing test forms is reduced from $O(c \cdot m! \cdot 2^f)$ to $O(c \cdot m! + 2^f)$. The BA in the proposed method is applied as a search algorithm to both steps.

Test constraints can be divided into the following two types: 1) test constraints about each test form (e.g., the number of total test items and the number of items from each subject) and 2) test constraints about the relationships among test forms (e.g., the number of overlapping items).

The details on both steps are described as follows:

Step A. The BA for Step A is outlined in Fig. 2. In this step, each artificial bee constructs one test form by sequentially selecting items that satisfy test constraints and minimize fitting errors until the construction of the test

```
Initialize test forms with a random search
while The stopping criterion is not met do
  Evaluate the fitting errors of the test forms
  Select the test forms for a neighborhood search
  Recruit artificial bees to improve the selected test forms
  Construct new test forms
end while
```

Fig. 2. Outline of Bees algorithm for construction of multiple test forms: Step A.

form is completed. As mentioned above, the test constraints in this step are only constraints about each test form. The first group of artificial bees constructs test forms using a random search and the later groups of artificial bees improve the constructed test forms using a neighborhood search. For more details, Step A is divided into the following five steps:

In Step A-1, the first group of artificial bees is generated and it constructs test forms. The artificial bees select the first items according to the uniform distributions of "item-selection probabilities." Next, the artificial bees iteratively update the item-selection probability distributions according to two rules:

1. The item-selection probability of each remaining item is inversely proportional to the fitting error of the constructed test form if this form includes the remaining item.
2. The item-selection probability of each remaining item becomes zero if no test constraints are satisfied.

The item-selection probability, p, for item i after t items are selected is extended from Luecht's [22] method as

$$p_{i,t} = \frac{(d_{i,t}/q_{i,t})}{\sum_{i \in A_t}(d_{i,t}/q_{i,t})}, \tag{10}$$

$$q_{i,t} = \sum_{k=1}^{K} \left| \left(\frac{T(\theta_k) - \sum_{i=1}^{m} I_i(\theta_k) x_i}{g - t + 1} - I_i(\theta_k) \right) \right|, \tag{11}$$

where $q_{i,t}$ is an item-selection coefficient and $d_{i,t}$ is a binary variable that equals zero if item i does not satisfy any test constraints or is one otherwise. Here, A_t is the set of indexes of remaining items, g is the number of total items for the test, and m is the total of items in the item bank. The expression in (11),

$$\frac{T(\theta_k) - \sum_{i=1}^{m} I_i(\theta_k) x_i}{g - t + 1},$$

means the fitting error at θ_k after t items have been selected. The artificial bees iteratively select the next items according to the updated item-selection probability distributions until the test-form constructions are completed. After all artificial bees in the first group have completed the test-form constructions, all constructed test forms are stored in a system memory.

Next, the test forms in the system memory are selected so that they can be improved by the next group of artificial bees using a neighborhood search.

In Step A-2, the fitting errors for the test forms in the system memory are evaluated and the form-selection probability distribution is calculated. The selection probability, p, of test form f can be calculated as

$$p_f = \frac{1/e_f}{\sum_{f=1}^{N}(1/e_f)}, \tag{12}$$

$$e_f = \sum_{k=1}^{K} \left| \sum_{i=1}^{m} I_i(\theta_k) x_{if} - T(\theta_k) \right|, \tag{13}$$

where e_f denotes the fitting errors of the test form, and N denotes the number of total test forms in the system memory. According to (12-13), the selection probabilities of test forms are inversely proportional to the fitting errors.

In Step A-3, the test forms in the system memory, which will be improved by the next group of artificial bees, are selected according to the form-selection probability distribution in Step A-2.

In Step A-4, artificial bees in the second group are generated and recruited to improve the selected test forms from Step A-3 according to the probability distribution calculated using (12-13) but here, e_{f} denotes the fitting errors of selected test forms and N denotes the number of total selected test forms. The number of recruited artificial bees, N_{bee}, for selected test form f can be calculated as

$$N_{bee,f} = N_{\text{All bees}} \cdot p_f, \tag{14}$$

where $N_{\text{All bees}}$ is the number of total artificial bees in the second group.

In Step A-5, the artificial bees construct new test forms using a neighborhood search in which the artificial bees sequentially select items to minimize the fitting errors being influenced by the selected test forms in Step A-3. For more details, the artificial bees iteratively update the item-selection probability distributions according to two rules:

1. The rules in Step A-1.
2. If each remaining item is included in the selected test form, the selection probability of this item is higher than the selection probabilities of the other items.

In these steps, the item-selection probability, $p_{i,t}$, in (10) is combined with BA [37] as

$$p_{i,t} = \frac{(\rho_{i,t})^{\alpha} \cdot (d_{i,t}/q_{i,t})^{\beta}}{\sum_{i \in A_t}(\rho_{i,t})^{\alpha} \cdot (d_{i,t}/q_{i,t})^{\beta}}, \tag{15}$$

where $\rho_{i,t}$ is the selection fitness of the item that increases the item-selection probability if item i is included in a selected test form or otherwise, it decreases the item-selection probability. Here, α is a binary variable that turns the influence of selection fitness on and off and $\beta \in [0, \infty)$ controls the significance level for adjusting the proportion between the selection fitness $\rho_{i,t}$ and the term $(d_{i,t}/\rho_{i,t})$ that refers to fitting errors of the currently constructed test form. That is, if β is small, bees select next items to follow the selected test forms. If β is large, bees select next items to minimize the fitting errors and ignore the selected test forms. If α is zero, (15) becomes (10).

To describe selection fitness, $\rho_{i,t}$, F_t denotes the set of indexes of items in the selected test form after t items have been selected. The fitness parameter is

Initialize sets of test forms
while The stopping criterion is not met **do**
  Evaluate the differences in fitting errors between test forms in each set
  Select the sets of test forms for a neighborhood search
  Recruit artificial bees to improve the selected sets of test forms
  Select test forms for new sets
end while

Fig. 3. Outline of Bees algorithm for construction of multiple test forms: Step B.

$$\rho_{i,t} = \begin{cases} \frac{\lambda}{|F_t|}, & i \in F_t, \quad |A_t| > 1 \\ \frac{1-\lambda}{|A_t - F_t|}, & i \notin F_t, \quad |A_t| > 1 \\ 1, & |A_t| = 1 \end{cases}$$
$$\forall i \in A_t, 0 \le \lambda \le 1,$$

where λ is a fitness value. If λ equals 1, the next item is selected according to the set of indexes of items F_t, else the next item is selected from items that are not included in F_t. $|F_t|$ and $|A_t|$ are the numbers of elements in sets F_t and A_t, respectively.

After all artificial bees in the second group have completed the test-form constructions, all constructed test forms are evaluated and stored in the system memory if the test forms satisfy two conditions:

1. The fitting errors of the test forms are smaller than the smallest fitting error in the system memory.
2. The test forms are not the same as the stored test forms in the system memory.

If the stopping criterion is not met, the process returns to Step A-2; otherwise, it goes to Step B.

A collection of test forms with small fitting errors is created in this step. However, at this stage, the constructed test forms are still not equivalent because the test constraints describing the relations between test forms are not satisfied.

Step B. The BA for Step B is outlined in Fig. 3. In Step B, the largest and most equivalent set of test forms, which minimizes the difference in fitting errors between test forms and maximizes the number of test forms, is extracted from the collection of test forms from Step A. The difference in fitting errors between test forms is indicated by a standard deviation, σ, of fitting errors. The test constraints satisfied in this step concern the relationships among test forms. Each artificial bee in the first group extracts a set of test forms by sequentially selecting them to minimize the standard deviation of fitting errors using a random search until this artificial bee cannot find any more available test forms. The later groups of artificial bees improve the extracted sets of test forms using a neighborhood search. To provide more detail, Step B is divided into additional five steps.

In Step B-1, the artificial bees in the first group are generated and they load the collection of test forms from the system memory. They select the first test forms according to the uniform distributions of "form-selection probabilities." Next, the artificial bees iteratively calculate the form-selection probability distributions according to the rule that the selection probability of each remaining test form is inversely proportional to the standard deviation of fitting errors of the currently extracted set of test forms if this set includes the remaining test form. The form-selection probability, p, for the test form, f, after l test forms are selected is defined as

$$p_{f,l} = \frac{(d_{f,l}/\sigma_{f,l})}{\sum_{f \in A_l}(d_{f,l}/\sigma_{f,l})}, \tag{16}$$

$$\sigma_{f,l} = \sqrt{\frac{1}{l+1}\sum_{r \in V_{f,l}}(e_r - \mu_{V_{f,l}})^2}, \tag{17}$$

$$\mu_{V_{f,l}} = \frac{1}{l+1}\sum_{r \in V_{f,l}} e_r, \tag{18}$$

where $\sigma_{f,l}$ is the standard deviation and $d_{f,l}$ is a binary variable that equals zero if form f does not satisfy any test constraints and is one otherwise. Here, A_l is the set of indexes of remaining test forms, $V_{f,l}$ is the set of indexes of selected test forms including the remaining test form, f, e_r is the fitting error of test form r defined in (13), and $\mu_{V_{f,l}}$ is the average of fitting errors of test forms in $V_{f,l}$. To maximize the number of test forms, the artificial bees iteratively select the next test forms according to the calculated form-selection probability distributions until all available test forms have been selected. After all artificial bees finish extracting the test forms, all sets of test forms are stored in the system memory.

Next, the sets of test forms in the system memory are selected so that they can be improved by the next group of artificial bees using a neighborhood search.

In Step B-2, the standard deviations of fitting errors of the sets of test forms in the system memory are evaluated and the selection-probability distribution of the sets of test forms is calculated. The selection probability, p, of the set of test forms s can be calculated as

$$p_s = \frac{(1/\sigma_s)}{\sum_{s=1}^{M}(1/\sigma_s)}, \tag{19}$$

$$\sigma_s = \sqrt{\frac{1}{l}\sum_{r \in V_s}(e_r - \mu_{V_s})^2}, \tag{20}$$

$$\mu_{V_s} = \frac{1}{l}\sum_{r \in V_s} e_r, \tag{21}$$

where σ_s denotes the standard deviation of the set of test forms, M denotes the number of total sets of test forms in the system memory, V_s is a set of indexes of selected test forms in the set of test forms, and μ_{V_s} is the average of fitting errors of test forms in V_s. According to (19-21), the selection probabilities of test forms are inversely proportional to the standard deviations.

In Step B-3, the sets of test forms in the system memory, which will be improved by the next group of artificial bees, are selected according to the generated selection-probability distribution in Step B-2.

In Step B-4, artificial bees in the second group are generated and recruited to improve the selected sets of test forms in Step B-3 according to the probability distribution

calculated using (19-21), but here, σ_s denotes the standard deviation of a selected set of test forms s and M denotes the number of total selected sets of test forms. The number of recruited artificial bees, N_{bee}, for the selected set of test forms s can be calculated as

$$N_{bee,s} = N_{\text{All bees}} \cdot p_s, \tag{22}$$

where $N_{\text{All bees}}$ is the number of total bees in the second group.

In Step B-5, the artificial bees sequentially select the test forms using a neighborhood search to minimize the difference in fitting errors being influenced by the selected sets of test forms in Step B-3. To provide more detail, the artificial bees iteratively calculate the form-selection probability distributions according to two rules:

1. The rule in Step B-1.
2. If each remaining test form is included in the selected set of test forms, the form-selection probability of this form is higher than the form-selection probabilities of the other forms.

In these steps, the form-selection probability, $p_{f,l}$, in (16) is combined with BA [37] as

$$p_{f,l} = \frac{\left(\rho_{f,l}\right)^{\alpha} \cdot \left(d_{f,l}/\sigma_{f,l}\right)^{\beta}}{\sum_{r \in A_l} \left(\rho_{r,l}\right)^{\alpha} \cdot \left(d_{r,l}/\sigma_{r,l}\right)^{\beta}}, \tag{23}$$

where $\rho_{f,l}$ is the selection fitness of the test form that increases the selection probability if test form f is included in a selected set of test forms or otherwise, it decreases the selection probability. Here, α is a binary variable that turns the influence of selection fitness on and off and $\beta \in [0, \infty)$ controls the significance level for adjusting the proportion between the selection fitness $\rho_{f,l}$ and the term $(d_{f,l}/\sigma_{f,l})$ that refers to fitting errors of the currently extracted set of test forms. That is, if β is small, bees select next test forms to follow the selected set of test forms. If β is large, bees select next test forms to minimize the difference in fitting errors and ignore the selected set of test forms. If α is zero, (23) becomes (16). The artificial bees iteratively select the next test forms according to the calculated form-selection probability distributions until all available test forms have been selected to maximize the number of test forms.

To describe selection fitness, $\rho_{f,l}$, F_l denotes the set of indexes of test forms in the selected set that is expected to be improved. The selection fitness is

$$\rho_{f,l} = \left\{ \begin{array}{lll} \frac{\lambda}{|F_l|}, & f \in F_l, & |A_l| > 1 \\ \frac{1-\lambda}{|A_l - F_l|}, & f \notin F_l, & |A_l| > 1 \\ 1, & |A_l| = 1 & \end{array} \right\}$$
$$\forall f \in A_l, 0 \leq \lambda \leq 1,$$

where λ is a fitness value. If λ equals 1, the next test form is selected according to the set of indexes of test forms F_l, else the next test form is selected from the test forms that are not included in F_l. $|F_l|$ and $|A_l|$ are the numbers of elements in sets F_l and A_l, respectively.

After all artificial bees have finished extracting the test forms, the sets of test forms are evaluated and stored in the system memory if these sets have standard deviations of fitting errors that are smaller than the smallest standard deviation in the system memory.

If the stopping criterion is not satisfied, the process returns to Step B-2; otherwise, this method selects the set of test forms that has the smallest standard deviation of fitting errors, as the final result.

However, there is a trade-off between the differences in fitting errors between test forms and the computational time. Therefore, in the next section, we explain the application of a parallel-computing technique to the construction of multiple test forms based on BA to minimize the trade-off.

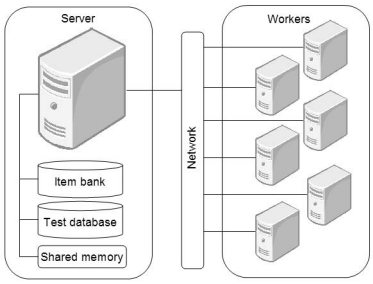

Fig. 4. Structure of parallel-computing environment.

4.4 Parallel Computing

The proposed construction of multiple test forms based on BA is implemented in a parallel-computing environment that includes one server and several workers as shown in Fig. 4. The server has an item bank, a test database, and a system memory. The server and workers are connected via a network.

According to Fig. 1, the processes that can be divided and distributed to be performed by the workers are: 1) initialize the population of solutions and 2) generate a new population of solutions. These divisible processes in the proposed approach are Steps A-1, A-5, B-1, and B-5. Using a parallel-computing technique, the computational cost of constructing the test forms for each processor core is calculated by dividing the computational cost by the number of processor cores. Therefore, we can decrease the computational time by increasing the number of total processor cores of workers. As a result, we can relax the trade-off by using the proposed method and the parallel-computing technique.

The new method was developed using Java as the development tool and implemented in a parallel-computing environment, which is the Java Parallel Processing Framework [42]. The system had six units of computer nodes including one server and five workers and each unit was equipped with a 2.5-GHz Quad-Core Intel processor. The workers have a total of 20 processor cores.

Since the five workers had the same performance, the divisible processes in the proposed approach were divided into five equal parts before they were distributed to the workers.

5 Experiments and Results

We carried out four experiments to evaluate the proposed method for constructing multiple test forms. We used

TABLE 1
Distributions of Item Parameters

Parameter	Attribute Information	Simulated Item Bank			Actual Item Bank	
		I=5000	I=10000	I=20000	I=517	I=2385
a	Range	0.1∼0.9	0.1∼0.9	0.1∼0.9	0.3∼1.167	0.3∼1.424
	Mean	0.503	0.498	0.501	0.540	0.541
	SD	0.285	0.289	0.289	0.160	0.177
b	Range	-3.0∼3.0	-3.0∼3.0	-3.0∼3.0	-3.233∼1.307	-4.202∼1.300
	Mean	-0.270	-0.298	-0.297	-1.116	-1.208
	SD	1.237	1.219	1.219	0.686	0.703

TABLE 2
Details on Test Constraints

Constraint	Simulated Item Bank		Actual Item Bank			
	$I \in \{5000, 10000, 20000\}$		I=517		I=2385	
	Number of constraints	Value	Number of constraints	Value	Number of constraints	Value
Maximum average rate of correct answers	1	65	1	70	1	70
Minimum average rate of correct answers	1	40	1	50	1	50
Number of items from each subject	2	20∼32	1	20	2	20∼32
Maximum number of items from each area of subject	53	1∼3	14	1∼2	53	1∼3
Minimum number of items from each area of subject	53	0∼2	14	0∼1	53	0∼2
Number of total items	1	80	1	20	1	80

simulated item banks in the first experiment and actual item banks from the Japan Information Technology Engineers' Examination [43] in the remaining experiments.

5.1 Accuracy and Speed of Construction of Multiple Test Forms

We compared the proposed method (BA) with BA in a parallel-computing environment, the big-shadows-test method [19], the GA for constructing multiple test forms proposed by Sun et al. [32] (GA_S), and a GA based on a two-step test construction (GA_2) to demonstrate its accuracy and speed in constructing multiple test forms. GA_S simultaneously constructed multiple test forms to minimize the fitting errors and the difference in the fitting errors. Although some experiments in [32] proved that GA_S could construct multiple equivalent test forms quite well, the implemented test constraints and the implemented item banks were too simple for actual application. Here, we compared GA_S with the proposed method in this experiment. Moreover, we developed GA_2 based on the two-step test construction described in Section 4.3 in which BA is replaced by GA to compare the performances of BA and general GA under the same conditions. BA, BST, GA_S, and GA_2 were used to construct multiple test forms to minimize the fitting errors indicated by the sum of the absolute differences (SADs) between the expected test information function and the test information functions of the constructed test forms at five levels of ability, $\Theta = \{-2, -1, 0, 1, 2\}$, and to minimize the difference in fitting errors indicated by the standard deviation of SADs in the constructed test forms. The test information function described in this paper is based on the two-parameter logistic model of IRT.

We used three simulated item banks that had a total number of items I of 5,000, 10,000, and 20,000. Each item in the item banks belonged to one area of a subject and each area belonged to one subject. The distributions of item parameters a and b in the item bank are given in Table 1. The set of test constraints for all the item banks was the same. The details on the test constraints are listed in Table 2. Each construction method was used to construct five test forms without overlapping the items between the test forms. The target values of the expected test information function, $T(\theta_k)$, at each ability level of Θ were assigned as follows: $\{1, 5, 12, 15, 2\}$.

We defined the stopping criteria of BA and GA_2 as follows:

For Step A:

The fitting errors of constructed test forms are not lower than the smallest fitting error of the stored test forms in the system memory.

For Step B:

1. The differences in fitting errors of extracted sets of test forms are not lower than the smallest difference of fitting errors of stored set of test forms in the system memory.
2. The fitting errors of extracted sets of test forms are not lower than the smallest fitting error of stored set of test forms in the system memory.

The β parameters in Steps A and B of BA for item banks $I = 5{,}000, 10{,}000, \text{ and } 20{,}000$ were 10, 15, and 20, respectively. The other BA parameters are defined as follows: $\alpha = 1$ and $\lambda = 0.95$ for Steps A and B.

The stopping criterion for GA_S was when the average and standard deviations of the fitting errors of the set of constructed test forms (a new generation) were not lower than that of the set of previously constructed test forms (parent).

BST, GA_S, and GA_2 were developed using Java as the development tool. The linear programming problems in

TABLE 3
Results for Accuracy and Speed of Constructing Multiple Test Forms

Method	Number of Processor Cores	Size of Item Bank	Average of SADs of Test Information Functions	SD of SADs of Test Information Functions	Computational Time (minutes)
BST	1	5000	1.856	0.728	547
		10000	1.821	0.618	1158
		20000	1.530	0.827	2685
GA_S	1	5000	0.924	0.814	1358
		10000	1.511	0.952	1787
		20000	1.345	0.891	2984
GA_2	1	5000	1.452	0.954	2874
		10000	1.984	0.687	3507
		20000	1.775	0.721	5810
BA	1	5000	0.132	0.017	484
		10000	0.107	0.021	950
		20000	0.099	0.015	1760
BA	32	5000	0.123	0.014	32
		10000	0.097	0.016	55
		20000	0.106	0.010	104

TABLE 4
Results for Parameter Tuning for Accuracy of Constructing Multiple Test Forms

Method	β in Step A	β in Step B	Average of SADs of Test Information Functions	SD of SADs of Test Information Functions	Computational Time (minutes)
BA	1	1	0.0389	0.00604	40
	1	10	0.0377	0.00135	412
	1	100	0.0370	0.00134	489
	10	1	0.0071	0.01307	363
	10	10	0.0021	0.00091	1063
	10	100	0.0023	0.00095	1240
	100	1	0.0092	0.00252	43
	100	10	0.0064	0.00162	320
	100	100	0.0066	0.00160	448

BST were solved by using CPLEX [44]. The implemented parallel-computing environment for BST consisted of nine units of computer nodes including one server and eight workers. Each unit was equipped with a 2.5-GHz Quad-core Intel processor. The workers have a total of 32 processor cores.

Table 3 lists the method, the number of processor cores, the size of item bank, average and standard deviation for the SADs of the constructed test forms, and computational time.

The averages and standard deviations for the SADs of the constructed test forms using BA are smaller for item banks $I = 5{,}000$, $10{,}000$, and $20{,}000$ with a single processor core than the results from BST, GA_2, and GA_S.

According to the results, BA can be used to construct test forms with fitting errors and with differences in these errors that are smaller than the results from the traditional methods. However, BA required a higher computational time.

When the parallel-computing technique is used, the averages and standard deviations of the test forms constructed using BA in the parallel-computing environment are equivalent to the results of BA using a single processor, but BA in the parallel-computing environment required a lower computational time than the other methods using a single processor.

The results obtained from this experiment indicated that the proposed method improves the traditional construction of multiple test forms.

In the next three experiments, the actual item banks were implemented to show the effectiveness of the proposed method in the context of actual situations.

5.2 Parameter Tuning for Accuracy of Constructing Multiple Test Forms

In this experiment, we changed the β parameters in (15) and (23) for Steps A and B to show how BA controls the fitting errors and the difference in the fitting errors. We used the actual item bank $I = 519$ with 32 constraints. The distributions of item parameters a and b in the item bank are given in Table 1. Each item in the item bank belongs to one area of a subject and each area belongs to one subject. The details of the test constraints are listed in Table 2. The expected number of constructed test forms was four and no overlapping items between the test forms were allowed. Before constructing the test forms, we defined the target values of the expected test information function, $T(\theta_k)$, at each Θ ability level as follows: {1.1, 1.3, 1.1, 0.5, 0.25}. The proposed method was implemented in the parallel-computing environment described in Section 4.4.

Table 4 lists the method, the value of β in Steps A and B, the average and standard deviation for the SADs of the constructed test forms, and the computational time. Here, the BA subscriptions, respectively, indicate the values of β in Steps A and B and x indicates any value of β where $\beta \in \{1, 10, 100\}$. The results show that the average and standard deviations for the SADs of the constructed test forms using $BA_{10,10}$ are the lowest. We can see that,

TABLE 5
Results for Processor Cores Related Performance

Method	Number of Processor Cores	Number of Test Forms	Average of SADs of Test Information Functions	SD of SADs of Test Information Functions	Computational Time (minutes)
BA	4	9	0.0912	0.0103	50
	8	9	0.0893	0.0089	45
	12	9	0.0902	0.0074	38
	16	9	0.0898	0.0084	31
	20	9	0.0915	0.0075	27

although $\mathrm{BA}_{1,x}$ or $\mathrm{BA}_{x,1}$ requires low computational time, the averages and standard deviations for the SADs of the constructed test forms using them are large.

When the β parameters in Step A are changed, the averages of the fitting errors for the SADs of the constructed test forms using $\mathrm{BA}_{10,x}$ and $\mathrm{BA}_{100,x}$ are lower than those when using $\mathrm{BA}_{1,x}$. However, the averages and standard deviations of the fitting errors for the SADs of the constructed test forms using $\mathrm{BA}_{100,x}$ are not lower than those when using $\mathrm{BA}_{10,x}$. For more details, in Step A, the numbers of constructed test forms using $\mathrm{BA}_{100,x}$ are smaller than those when using $\mathrm{BA}_{10,x}$, since $\mathrm{BA}_{100,x}$ satisfies the stopping criterion faster than $\mathrm{BA}_{10,x}$. Therefore, the possibility of finding equivalent test forms in Step B when the β parameters in Step A are equal to 100 becomes lower than when the β parameters in Step A are equal to 10.

When the β parameters in Step B are changed, the averages and standard deviations for the SADs of the constructed test forms using $\mathrm{BA}_{x,100}$ are close to that of the constructed test forms when using $\mathrm{BA}_{x,10}$, but $\mathrm{BA}_{x,100}$ requires a higher computational time. This shows the difference in computational time between a BA with a high β parameter in Step B that ignores the selected set of test forms as described in Section 4.3 and a BA with an appropriate β parameter in Step B.

5.3 Number of Processor Cores Related Performance

This experiment was used to demonstrate the computational time for the proposed method when the number of processor cores in the parallel-computing environment increased. We employed the actual item bank I of 2,385 with 112 test constraints. The distributions of item parameters a and b in the item bank and the details of the test constraints are given in Tables 1 and 2. The target values of the expected test information function, $T(\theta_k)$, at each ability level of Θ were {2, 5, 4, 2, 1}. To find the differences in the computational time, we changed the number of processor cores (4, 8, 12, 16, and 20) and overlapping items between test forms were not allowed. The proposed method was used to construct multiple test forms to minimize the fitting errors that were described in Section 5.1.

We defined the stopping criteria as described in Section 5.1. The parameters of BA are defined as follows: $\alpha = 1, \beta = 1, \lambda = 0.95$ for Steps A and B.

Table 5 lists the number of constructed test forms, average and standard deviation of the SADs for the constructed test forms, and computational time. The results indicate that there are no significant differences among SADs due to the different numbers of processor cores. The computational time, on the other hand, decreases when the number of processor cores increases.

Fig. 5 plots the relation between the computational time and the number of processor cores. The horizontal axis indicates the number of processor cores and the vertical axis indicates the computational time. This figure shows that the computational time decreases in inverse proportion to the number of processor cores.

This means that the computational time for the test constructions using the proposed method can be decreased while keeping the fitting errors approximately constant. Namely, the proposed method relaxes the trade-off between the fitting errors and the computational time.

5.4 Overlapping Construction of Test Forms

The proposed method allows the overlapping items; therefore, it is expected to increase the number of constructed test forms. This experiment revealed how the number of constructed test forms increased when the number of overlapping items increased. The proposed method was used to construct multiple test forms not only to minimize the fitting errors that were described in Section 5.1 but also to maximize the number of test forms from an item bank. We used the smallest item bank, $I = 517$, and 32 test constraints. The number of total test items was 20. To find the different numbers of constructed test forms when the number of overlapping items increased, we changed the number of overlapping items (0, 1, 2, 3, and 4). The proposed method was implemented in the parallel-computing environment described in Section 4.4.

We defined the stopping criteria as described in Section 5.1 and added one more criterion that stops the calculation of Step B when the sizes of extracted sets of test forms are not larger than the largest size of stored set of test forms in the system memory. The parameters of BA are defined as follows: $\alpha = 1, \beta = 1, \lambda = 0.95$ for Steps A and B.

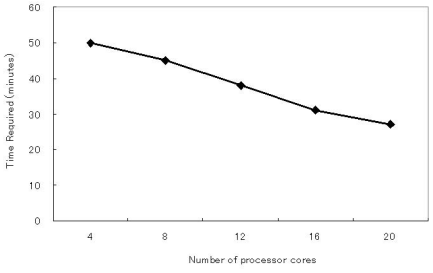

Fig. 5. Computational time related to number of processor cores.

TABLE 6
Results for Overlapping Test Construction

Method	Number of Overlaps Items	Number of Test Forms	Average of SADs of Test Information Functions	SD of SADs of Test Information Functions	Computational Time (minutes)
BA	0	8.00	0.0415	0.0152	9
	1	18.67	0.0423	0.0177	12
	2	54.00	0.0437	0.0194	19
	3	192.00	0.0440	0.0185	44
	4	733.00	0.0442	0.0198	1148

As mentioned before, Ishii et al. [20] proposed a method which guarantees the maximum number of possible constructed test forms with allowing the overlapping items. However, the computational time of the method increases exponentially as the size of the item bank increases. In this experiment, the method cannot provide the guaranteed maximum number of test forms from the item bank in reasonable time. On the other hand, BST requires a huge computational time when overlapping constraints are permitted. Therefore, we performed this experiment using only the proposed method.

Table 6 lists the number of overlapping items, the number of constructed test forms, the average and standard deviation of SADs for the constructed test forms, and the computational time.

The number of constructed test forms exponentially increases with the rise in the number of overlapping items. Although the number of test forms increases, the standard deviation for SADs does not tend to increase. This means that the differences in fitting errors in the constructed test forms are constant for the numbers of test forms.

However, the computational time with the proposed method increased in proportion to the number of overlapping items. This problem might be solved by increasing the number of processor cores in the system.

6 Automated Test Construction System (ATCS)

Using the proposed method of constructing multiple test forms, we developed an automated test construction system and installed it into an e-testing system [45].

The e-testing system was developed to support test authors in creating items, analyzing test data, and in constructing and delivering large-scale assessments, such as for university entrance examinations, the Japan Information Technology Engineers' Examination, or for Tests of English as a Foreign Language. Since the proposed method can construct multiple equivalent test forms using large item banks without a trade-off between the difference in fitting errors and the computational time, it appropriates for an e-testing system.

Although ATCS using the proposed method was developed for constructing high-stake tests from large item banks, this system also supports the construction of low-stake tests from small item banks such as online assessments in classes and self-assessments in e-learning.

When a test author constructs multiple test forms using ATCS, he/she first defines the test constraints for the expected multiple test forms through an interface, as shown in Fig. 6. For more details, the maximum and minimum numbers of items from each knowledge domain are entered into the right table of the interface. The interface in Fig. 6 shows the details of constraints for constructing tests from the item bank of Japanese Language Proficiency Test. The target values for the expected test information function, the maximum and minimum rates for correct answers, the maximum and minimum response times, and the number of overlapping items are entered into the left table of this interface. After the test author has clicked onto the "Start construction" button, ATCS automatically receives the defined test constraints from the interface, and constructs test forms using the defined test constraints and the items from the item bank. After ATCS has finished constructing the multiple test forms, it stores the constructed test forms in the test database. ATCS is implemented in the parallel-computing environment described in Section 4.4.

7 Conclusions

We proposed a method of constructing multiple test forms based on the Bees Algorithm in parallel computing. The proposed method distributes the computational costs over multiple processors to mitigate the trade-off between computational costs and the differences in fitting errors on the test forms. We compared the proposed method and traditional methods of constructing multiple test forms using actual item banks and test constraints. The results revealed that the proposed method required lower computational time than the traditional methods while the differences in fitting errors for the constructed test forms

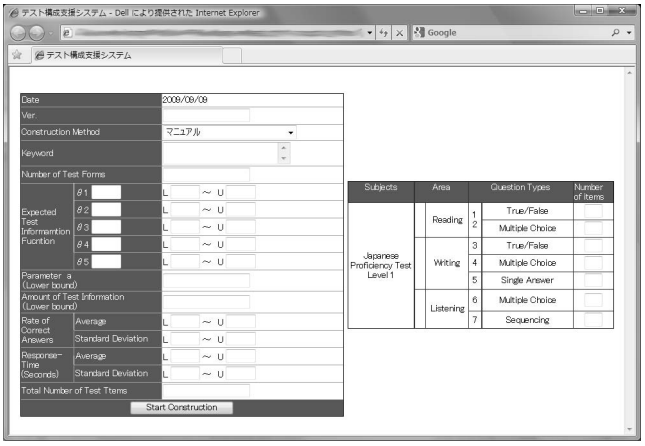

Fig. 6. Test constraint interface.

were lower or close to that of the traditional methods. These results confirm that the trade-off was mitigated using the new method.

Moreover, the proposed method approximately guarantees the maximum number of test forms from an item bank with overlapping constraints. We demonstrated the construction of multiple test forms with overlapping constraints using the proposed methods. The results indicated that the number of constructed test forms exponentially increased with the rise in the number of overlapping items. That is, the item bank could be used more effectively by permitting overlapping constraints. Moreover, only the new approach could construct the multiple test forms in reasonable time. This meant that the method we propose requires less computational time than the traditional methods and it is possible to implement it in practice.

In addition, we developed an automated test construction system using the proposed method and installed it in an actual e-testing system.

To be exact, we should have determined that all the constructed test forms would satisfy the expected constraints to prove the effectiveness of the proposed method. However, it is impossible to provide a huge number of actual test forms to examinees. It should be noted that this paper implicitly assumes that the constructed test forms ideally satisfy the expected test constraints. Moreover, some limitations from the experiments still remain, such as each item provides evidence of only one area or only a few types of test constraints were used. To obtain more generalized results, there should be more types of constraints (e.g., avoiding a related item in the same test form, the number of words in the items, the response times, the sequence of items order, and the distribution of the item-selection frequencies) and more types of item banks (e.g., item banks that have different distributions of the item parameters, an item belongs to one or more areas).

REFERENCES

[1] F. Lord, *Application of Item Response Theory to Practical Testing Problems.* Lawrence Erlbaum Associates, 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," *Applied Psychological Measurement,* vol. 10, no. 4, pp. 381-389, 1986.

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

[5] W.J. van der Linden and E. Boekkooi-Timminga, "A Zero-One Programming Approach to Gulliksen's Matched Random Subtests Method," *Applied Psychological Measurement,* vol. 12, no. 2, pp. 201-209, 1988.

[6] W.J. van der Linden and E. Boekkooi-Timminga, "A Maximin Model for Test Design with Practical Constraints," *Psychometrika,* vol. 54, pp. 237-247, 1989.

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

[8] J. Adema, "Implementations of the Branch-and-Bound Method for Test Construction Problems," Technical Report 89-6, Project Psychometric Aspects of Item Banking, Dept. of Education, Univ. of Twente, 1989.

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

[10] J. Adema, "The Construction of Customized Two-Stage Tests," *J. Educational Measurement,* vol. 27, no. 3, pp. 241-253, http://doc.utwente.nl/58414, Sept. 1990.

[11] J. Adema, "Methods and Models for the Construction of Weakly Parallel Tests," *Applied Psychological Measurement,* vol. 16, pp. 53-63, 1992.

[12] J.J. Adema, E. Boekkooi-Timminga, and W.J. van der Linden, "Achievement Test Construction Using 0-1 Linear Programming," *European J. Operations Research,* vol. 55, pp. 103-111, 1998.

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

[14] R.D. Armstrong, D. Jones, and I.-L. Wu, "An Automated Test Development of Parallel Tests from a Seed Test," *Psychometrika,* vol. 57, no. 2, pp. 271-288, June 1992.

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

[16] E. Boekkooi-Timminga, "The Construction of Parallel Tests from Irt-Based Item Banks," *J. Educational Statistics,* vol. 15, no. 2, pp. 129-145, 1990.

[17] R.B. Fletcher, "A Review of Linear Programming and Its Application to the Assessment Tools for Teaching and Learning (aaTTle) Projects," Technical Report 5, Univ. of Auckland, www.tki.org.nz/r/asttle/pdf/technical-reports/techreport05.pdf, 2000.

[18] W.J. van der Linden, "Automated Test Construction Using Minimax Programming," *IRT-Based Test Construction,* W.J. van der Linden, ed., pp. 1-16, Dept. of Education, Univ. of Twente 1987.

[19] W.J. van der Linden, *Linear Models for Optimal Test Design,* first ed. Springer, 2005.

[20] T. Ishii, P. Songmuang, and M. Ueno, "A Method to Extract the Maximum Number of Test Forms Using Maxclique," *Proc. 23rd Ann. Conf. Japanese Soc. for Artificial Intelligence,* 2009.

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

[22] R. Luecht, "Computer-Assisted Test Assembly Using Optimization Heuristics," *Applied Psychological Measurement,* vol. 12, no. 3, pp. 224-236, 1998.

[23] J. Adema, "A Revised Simplex Method for Test Construction Problems," Technical Report 90-5, Dept. of Education, Univ. of Twente, 1989.

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

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

[26] R.D. Armstrong, D.H. Jones, and Z. Wang, "Automated Parallel Test Construction Using Classical Test Theory," *J. Educational Statistics,* vol. 19, pp. 73-90, 1992.

[27] R.D. Armstrong, D.H. Jones, and Z. Wang, "Optimization of Classical Reliability in Test Construction," *J. Educational and Behavioral Statistics,* vol. 23, no. 1, pp. 1-17, 1998.

[28] K.-T. Sun, "A Greedy Approach to Test Construction Problems," *Proc. Nat'l Science Council,* vol. ROC-Part D 11, no. 2, pp. 78-87, 2001.

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

[30] D. Belov and R.D. Armstrong, "Monte Carlo Test Assembly for Item Pool Analysis and Extension," *Applied Psychological Measurement,* vol. 29, no. 4, pp. 239-261, 2005.

[31] A.J. Verschoor, "Genetic Algorithms for Automated Test Assembly," PhD dissertation, Univ. of Twente, http://doc.utwente.nl/60710, Apr. 2007.

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

[33] K. He, L. Zheng, S. Dong, L. Tang, J. Wu, and C. Zheng, "PGO: A Parallel Computing Platform for Global Optimization Based on Genetic Algorithm," *Computers & Geosciences,* vol. 33, no. 3, pp. 357-366, 2007.

[34] P. Borovska, "Solving the Travelling Salesman Problem in Parallel by Genetic Algorithm on Multicomputer Cluster," *Proc. Int'l Conf. Computer Systems and Technologies,* pp. II.11-1-II.11-6, 2006.
[35] H. Pirim, E. Bayraktar, and B. Eksioglu, *Tabu Search: A Comparative Study,* pp. 1-27, http://intechweb.org/book.php?id=35, 2008.
[36] X.-S. Yang, *Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms,* vol. 3562. Springer, 2005.
[37] L.P. Wong, M.Y.H. Low, and C.S. Chong, "A Bee Colony Optimization Algorithm for Traveling Salesman Problem," *Proc. Second Asia Int'l Conf. Modelling & Simulation (AMS '08),* pp. 818-823, 2008.
[38] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm, a Novel Tool for Complex Optimisation Problem," *Proc. Second Int'l Virtual Conf. Intelligent Production Machines and Systems (IPROMS '06),* pp. 454-461, 2006.
[39] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms,* first ed. Luniver, 2008.
[40] F. Solis and R. Wets, "Minimization by Random Search Techniques," *Math. of Operations Research,* vol. 6, no. 1, pp. 19-30, 1981.
[41] P. Hansen and N. Mladenovi, "Variable Neighborhood Search: Principles and Applications," *European J. Operational Research,* vol. 130, no. 3, pp. 449-467, May 2001.
[42] *Java Parallel Processing Framework,* http://www.jppf.org, 2009.
[43] *The Information Technology Engineers Examination (ITEE),* Japan Information-Technology Engineers Examination Center (JITEC), 16th Floor, Bunkyo Green Court, 2-28-8, Hon-Komagome, Bunkyo-ku, Tokyo, Japan, pp. 113-6591, http://www.ipa.go.jp, 2008.
[44] *IBM ILOG CPLEX Optimizer*, 2010.
[45] P. Songmuang and M. Ueno, "Development of Practice of and Integrative e-Testing System," *Japanese Test Soc.,* vol. 4, no. 1, pp. 53-64, 2008.

Pokpong Songmuang received the BEng degree from Thammasat University in 2003, the MEng degree from Nagaoka University of Technology in 2006, and the PhD degree in computer science from the University of Electro-Communications in 2010. He is currently an assistant professor at Waseda University. His research interests include e-testing, data mining, and web technologies.

Maomi Ueno received the PhD degree in computer science from the Tokyo Institute of Technology in 1994. He has been an associate professor in the Graduate School of Information Systems at the University of Electro-Communications since 2007. He has also worked at the Tokyo Institute of Technology (1994-1996), Chiba University (1996-2000), and the Nagaoka University of Technology (2000-2007). He received best paper awards from the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), ED-MEDIA 2008, e-Learn2004, e-Learn2005, and e-Learn2007. His interests are in e-learning, e-testing, e-portfolio, machine learning, data mining, Bayesian statistics, Bayesian networks, and so on. He is a member of the IEEE.