Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 712752, 22 pages
doi:10.1155/2012/712752

Research Article

Test-Sheet Composition Using Analytic Hierarchy Process and Hybrid Metaheuristic Algorithm TS/BBO

Hong Duan,[1] Wei Zhao,[1] Gaige Wang,[2,3] and Xuehua Feng[4]

[1] *School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China*
[2] *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China*
[3] *Graduate School of Chinese Academy of Sciences, Beijing 100039, China*
[4] *School of Electronic and Information Engineering, Yili Normal University, Yining, Xinjiang 835000, China*

Correspondence should be addressed to Wei Zhao, zhaow577@nenu.edu.cn

Received 10 June 2012; Accepted 27 August 2012

Academic Editor: Jun-Juh Yan

Due to the shortcomings in the traditional methods which dissatisfy the examination requirements in composing test sheet, a new method based on tabu search (TS) and biogeography-based optimization (BBO) is proposed. Firstly, according to the requirements of the test-sheet composition such as the total score, test time, chapter score, knowledge point score, question type score, cognitive level score, difficulty degree, and discrimination degree, a multi constrained multiobjective model of test-sheet composition is constructed. Secondly, analytic hierarchy process (AHP) is used to work out the weights of all the test objectives, and then the multiobjective model is turned into the single objective model by the linear weighted sum. Finally, an improved biogeography-based optimization—TS/BBO is proposed to solve test-sheet composition problem. To prove the performance of TS/BBO, TS/BBO is compared with BBO and other population-based optimization methods such as ACO, DE, ES, GA, PBIL, PSO, and SGA. The experiment illustrates that the proposed approach can effectively improve composition speed and success rate.

1. Introduction

Examination is the process of teaching management which plays an important role in evaluating student achievement, inspiring students' creativity, and improving student learning outcomes. As an important means of testing student learning and teacher teaching, it also provides the necessary information for teachers to improve their teaching methods and the quality of teaching [1]. Also, tests are often used to help instructors assess students' prior knowledge [2]. With the advent of computer based education (CBE), how to take full advantage of computers to replace manual to help the examination organizer deal with a variety of complex repetitive work is an important topic in computer managed instruction, so computer-aided test (CAT) come into being. Computer-aided test can best satisfy the different evaluation requirements; quickly and effectively complete test-sheet composition with randomness, scientificity and rationality.

Test-sheet composition is a hot issue in computer-aided testing. At present, the algorithms to solve test-sheet composition are abductive machine learning [3], harmony search algorithm [4], genetic algorithm [5, 6], and particle swarm optimization [7]. However, these algorithms has many shortcomings, such as, meeting the local constraints, parameter uncertainty, the lack of random selection and too long time required for composing test-sheet, which lead to dissatisfy evaluation requirements.

Firstly proposed by Simon in 2008, biogeography-based optimization (BBO) [8–10] is a novel powerful metaheuristic algorithm, which has good global optimization capability, insensitivity for selection of initial values and parameters, strong robustness, simplicity, and easy implementation. Therefore, it has been widespread concern in the academic research and application and penetrated to a number of fields [11]. However, in the field of test-sheet composition, no application of BBO algorithm exists yet. In this paper, we use an improved BBO algorithm to solve the test-sheet composition problem. To reduce the search space and improve efficiency, an improved BBO algorithm makes use of subencoding based on question type. Moreover, we use tabu search (TS) to avoid duplication of encoding in the migration and mutation operations. Analytic hierarchy process (AHP) is used to determine the weight of the test-sheet goal, and then multiobjective optimization model is converted to a single-objective model by a linear weighted sum. In [12], Hwang et al. use tabu search to solve a set of parallel test sheets with identical test ability and get a good performance. Here, we combine BBO and TS to construct a new algorithm according to AHP and the principle and feature of TS and BBO; that is, we use TS to optimize the mutation and migration operator in BBO and then design the TS/BBO algorithm to get the most optimal question combination to form a testsheet.

The structure of this paper is organized as follows. Section 2 describes the mathematical model in test-sheet composition problem. In Section 3, the weight in test-sheet composition is determined by AHP. Subsequently, preliminary knowledge of TS and BBO algorithm are introduced in Sections 4 and 5, respectively. Then, an improved BBO algorithm for test-sheet composition is presented in Section 6 and the detailed implementation procedure is also described. The simulation experiments are conducted in Section 7. Finally, Section 8 concludes the paper and discusses the future path of our work.

2. Problem Description

Test-sheet composition model is a multiconstraint, multiobjective optimization problem. In this section, we will describe the mathematical model for test-sheet composition, including constraints and objective function.

2.1. Constraints

Automatic test-sheet composition with computer is searching for questions to meet the requirements from item bank. We determine the encoding for the main properties according to objective analyzing and preestablish the goal state matrix **A** for all the questions in the item bank:

$$\mathbf{A}_{M\times N} = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix}, \tag{2.1}$$

where M is the total number of items in the item bank; N is the number of attributes for each item (here, $N = 9$). For goal state matrix **A**, each row represents all properties every item, and each column represents a property of the entire item in the item bank. It has not more than $N - 1$ constraints because a_{i1} $(i = 1, 2, \ldots, M)$ means item number.

The process of composing testsheet is as follows. Firstly, find the row to meet the objectives and requirements of question combination $x = (x_1, x_2, \ldots, x_p)$ in the state matrix **A** according to the default settings or parameters user input, where $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,l_i})$ is a vector composed by the question type $i, x_{i,j} \in [1, M]$ is the number for question j in the type i, $1 \le i \le p$, $1 \le j \le l_i$, where p is the type number of questions in the item bank, l_i is the number of the ith question, and $\sum_{i=1}^{p} l_i = l$, l is the total question number in a testsheet. For simplicity, we use Boolean vector $K = (K_1, K_2, \ldots, K_M)$ instead of $x = (x_1, x_2, \ldots, x_p)$ to represent the required question combination at last. Here, K_m is the state variable for the mth row (i.e., the mth item or question), where, $K_m = 1$ means the mth row selected, that is, there exists $x_{i,j} = m$; $K_m = 0$ means the mth row not selected, that is, there does not exist $x_{i,j} = m$. To simplify, without loss of generality, we use $K(i)$ $(1 \le i \le M)$ to represent the ith question selected or not.

Every objective for test-sheet composition corresponds to a constraint, all the $N-1$ $(9-1 = 8)$ test-sheet objectives corresponding to the following $N - 1$ (eight) constraints. Thus, the mathematical model, that is, the constraints for goal state matrix **A**, can be constructed as follows:

(1) Total score

$$\sum_{i=1}^{M} K(i) a_{i2} = S_P, \tag{2.2}$$

where S_P is total score for the test-sheet composed; a_{i2} is score for the ith question.

(2) Test time

$$\sum_{i=1}^{M} K(i) a_{i3} = T_E, \tag{2.3}$$

where T_E is the total time completing the test-sheet composed; a_{i3} is estimated time for the completion of the ith question.

(3) Chapter scores

$$\sum_{i=1}^{M} K(i) a_{i2} C(j) = S_C^j, \tag{2.4}$$

where $C(j) = \begin{cases} 0, & j \neq a_{i4} \\ 1, & j = a_{i4}; \end{cases}$ S_C^j is the score of the jth chapter; a_{i4} is the chapter number for the ith question.

(4) Knowledge point score

$$\sum_{i=1}^{M} K(i) a_{i2} O(j) = S_O^j, \tag{2.5}$$

where $O(j) = \begin{cases} 0, & j \neq a_{i5} \\ 1, & j = a_{i5}; \end{cases}$ S_O^j is the score for the jth knowledge point; a_{i5} is the knowledge point number for the ith question.

(5) Question type score

$$\sum_{i=1}^{M} K(i) a_{i2} T(j) = S_T^j, \tag{2.6}$$

where $T(j) = \begin{cases} 0, & j \neq a_{i6} \\ 1, & j = a_{i6}; \end{cases}$ S_T^j is the score for the jth question type; a_{i6} is the question type number for the ith question. All the questions in item bank are divided into four kinds (i.e., Type 1, Type 2, Type 3, and Type 4);

(6) Cognitive level score

$$\sum_{i=1}^{M} K(i) a_{i2} A(j) = S_A^j, \tag{2.7}$$

where $A(j) = \begin{cases} 0, & j \neq a_{i7} \\ 1, & j = a_{i7}; \end{cases}$ S_A^j is the score for the jth cognitive level; a_{i7} is the cognitive level number for the ith question.

(7) Difficulty degree

$$\frac{\left[\sum_{i=1}^{M} K(i) a_{i2} a_{i8}\right]}{S_P} = \mathrm{DIF}_P, \tag{2.8}$$

where DIF_P is the difficulty degree of the test-sheet composed; a_{i8} is difficult degree for the ith question.

(8) Discrimination degree

$$\frac{\left[\sum_{i=1}^{M} K(i) a_{i2} a_{i9}\right]}{S_P} = \mathrm{DIS}_P, \tag{2.9}$$

where DIS_P is the discrimination degree of the test-sheet composed; a_{i9} is discrimination degree for the ith question.

2.2. Objective Function

Through analysis, eight constraints of the test-sheet composition model (total score, test time, chapter score, knowledge point score, question type score, cognitive level score, difficulty degree, and discrimination degree) should be equal to the evaluation requirements or with the minimum error of the evaluation requirements. Thus, the test-sheet composition is a multiconstraint, multiobjective optimization problem. In practical application, we can consider the error between the above eight constraints and evaluation requirements as objective function f. Therefore, the composed paper is optimal when f reaches minimum.

The mathematical model (i.e., objective function) for the test-sheet composition is as follows:

$$\begin{aligned}
\min \quad & f = \sum_{j=1}^{2}\left(d_j^+ + d_j^-\right) \times \omega_j + \sum_{i=1}^{h}\left(d_{3i}^+ + d_{3i}^-\right) \times \omega_3 + \sum_{i=1}^{l}\left(d_{4i}^+ + d_{4i}^-\right) \times \omega_4 + \sum_{i=1}^{p}\left(d_{5i}^+ + d_{5i}^-\right) \times \omega_5 \\
& \quad + \sum_{i=1}^{q}\left(d_{6i}^+ + d_{6i}^-\right) \times \omega_6 + \sum_{j=7}^{8}\left(d_j^+ + d_j^-\right) \times \omega_j \\
\text{s.t.} \quad & \sum_{j=1}^{M} K(j) a_{j2} - d_1^+ + d_1^- = S_P \\
& \sum_{j=1}^{M} K(j) a_{j3} - d_2^+ + d_2^- = T_E \\
& \sum_{i=1}^{h}\left(\sum_{j=1}^{M} K(j) a_{j2} C(i) - d_{3i}^+ + d_{3i}^-\right) = \sum_{i=1}^{h} S_C^i \\
& \sum_{i=1}^{l}\left(\sum_{j=1}^{M} K(j) a_{j2} O(i) - d_{4i}^+ + d_{4i}^-\right) = \sum_{i=1}^{l} S_O^i \\
& \sum_{i=1}^{p}\left(\sum_{j=1}^{M} K(j) a_{j2} T(i) - d_{5i}^+ + d_{5i}^-\right) = \sum_{i=1}^{p} S_T^i
\end{aligned}$$

$$\begin{aligned}
&\sum_{i=1}^{q}\left(\sum_{j=1}^{M}K(j)a_{j2}A(i)-d_{6i}^{+}+d_{6i}^{-}\right)=\sum_{i=1}^{q}S_A^i\\
&\frac{\left[\sum_{j=1}^{M}K(j)a_{j2}a_{j8}\right]}{S_P}-d_7^{+}+d_7^{-}=\mathrm{DIF}_P\\
&\frac{\left[\sum_{j=1}^{M}K(j)a_{j2}a_{j9}\right]}{S_P}-d_8^{+}+d_8^{-}=\mathrm{DIS}_P\\
&d_i^{+},d_i^{-}\ge 0,\quad d_i^{+}\times d_i^{-}=0\quad (i=1,2,7,8)\\
&d_{ij}^{+},d_{ij}^{-}\ge 0,\quad d_{ij}^{+}\times d_{ij}^{-}=0\quad (i=3,j=h)\\
&d_{ij}^{+},d_{ij}^{-}\ge 0,\quad d_{ij}^{+}\times d_{ij}^{-}=0\quad (i=4,j=l)\\
&d_{ij}^{+},d_{ij}^{-}\ge 0,\quad d_{ij}^{+}\times d_{ij}^{-}=0\quad (i=5,j=p)\\
&d_{ij}^{+},d_{ij}^{-}\ge 0,\quad d_{ij}^{+}\times d_{ij}^{-}=0\quad (i=6,j=q)\\
&\omega_j\ge 0\quad (j=1,2,\ldots,8),\quad \sum_{j=1}^{8}\omega_j=1.
\end{aligned}\tag{2.10}$$

In the above objective function f, d_i^{+} $(i=1,2,7,8)$, d_{ij}^{+} $(i=3,j=1,2,\ldots,h)$, d_{ij}^{+} $(i=4,j=1,2,\ldots,l)$, d_{ij}^{+} $(i=5,j=1,2,\ldots,p)$, and d_{ij}^{+} $(i=6,j=1,2,\ldots,q)$ are the positive deviation between the test-sheet property and evaluation requirements, that is, the part beyond the evaluation requirements; d_i^{-} $(i=1,2,7,8)$, d_{ij}^{-} $(i=3,j=1,2,\ldots,h)$, d_{ij}^{-} $(i=4,j=1,2,\ldots,l)$, d_{ij}^{-} $(i=5,j=1,2,\ldots,p)$, and d_{ij}^{-} $(i=6,j=1,2,\ldots,q)$ are the negative deviation between the test-sheet property and evaluation requirements, that is, the part less than the evaluation requirements. The positive deviation and negative deviation cannot exist at the same time; therefore, their product will be 0, that is, $d^{-}\times d^{+}=0$; ω_j $(j=1,2,\ldots,8)$ is weight for test-sheet composition, whose sum is 1.

In practice, the weight ω_j $(j=1,2,\ldots,8)$ has an important impact on the test-sheet composition whether the result satisfies the requirements. Therefore, how to scientifically and reasonably determine the weight of the test-sheet goal is worthy of further study. In the next section, we will use AHP to determine the weights.

3. Determining the Weight in Test-Sheet Composition Using AHP

AHP [13] is an effective combination of qualitative and quantitative analysis of multiobjective evaluation method, proposed by the University of Pittsburgh professor Saaty in the 1970s. Advantages of AHP are easy implementation, simple structure, speed, and robustness. Due to these advantages, it has many real-world applications, such as evaluating course website quality, improving the service quality of e-learning, and evaluating the web-based e-learning system.

```
Begin
    Step 1: Setting up the decision hierarchy by breaking down the decision problem into a
            hierarchy of interrelated decision elements, as a tree containing the overall goal at
            the top with many levels of criteria and subcriteria in between and the alternatives
            at the bottom
    Step 2: Collecting input data by pairwise comparisons of decision elements
    Step 3: Using the eigenvalue method to estimate the relative weights of decision elements
    Step 4: Aggregating the relative weights of decision elements to arrive at a set of ratings for
            the decision alternatives.
End.
```

Algorithm 1: The algorithm of analytic hierarchy process.

3.1. AHP Algorithm

Proposed by Zahedi [14], four main steps are identified in using the AHP to solve a decision-making problem. The detailed algorithm AHP is described in Algorithm 1.

3.2. Determining the Weights Using AHP

AHP can be used to determine goal weight for the test-sheet composition. Integrating the feature of test-sheet composition into the AHP, specific solution process is shown in Algorithm 2 (Figure 1).

We calculate consistency ratio $\mathrm{CR} = 0.0822$ for the compassion matrix C with Algorithm 2 and get the weights as shown in Table 2, so we can draw a conclusion that the compassion matrix C has satisfactory consistency.

In the actual exam, test-sheet goal weight is various. AHP can determine different test-sheet goal weight responding to different requirements according to the different compassion matrix C.

4. Biogeography-Based Optimization (BBO)

Biogeography is the study of the migration, speciation, mutation, and extinction of species [15]. Biogeography has frequently been thought of as a process that compels equilibrium in the number of species in islands. However, equilibrium in a system can also be viewed as a minimum-energy configuration, so we see that biogeography can be considered as an optimization process. This idea is further discussed in [16].

Firstly proposed by Simon in 2008, biogeography-based optimization (BBO) is a new evolution algorithm developed for the global optimization [8]. It is inspired by the immigration and emigration of species between islands (or habitats) in search for more compatible islands. Each solution is called a “habitat” (or “island”) with an HSI (habitat suitability index) and represented by an n-dimension real vector. An initial individual of the habitat vector is generated at random. The habitat with a high HSI is considered to be good solution, while the habitat with a low HSI is considered to be bad solution. Low HSI can take in many new good features form the high HSI, so that these low HSI solutions have

```
Begin
    Step 1: Define test-sheet problem and determine test-sheet composition goal
    Step 2: Structure the hierarchy from the top through the intermediate levels (criteria on which
            subsequent levels depend) to the lowest level (Figure 1).
    Step 3: Construct a set of pair-wise comparison matrix $C$ (size $n \times n$, here $n = 8$, shown in
            (Table 1). The pair-wise comparisons are done in terms of which element dominates
            the other.
    Step 4: Calculate the sorted weights. Calculate maximum eigenvalue $\lambda_{\max}$ and its eigenvector
            $\omega'$ for comparison matrix $C$ through $Cx = \lambda x$, and then normalize $\omega'$ to get the
            final weight vector $\omega$ (Table 2).
    Step 5: Test consistency
        Step 5.1: Calculate consistency index CI through $\mathrm{CI} = (\lambda_{\max} - n)/(n - 1)$
        Step 5.2: Find the average Random Index RI through Table 3.
        Step 5.3: Calculate consistency ratio CR through $\mathrm{CR} = \mathrm{CI}/\mathrm{RI}$
    Step 6: If $\mathrm{CR} < 0.1$, the   comparison matrix $C$ is consistent and return; if $\mathrm{CR} \geq 0.1$, the
            comparison matrix $C$ should be modified, and return Step 4.
End.
```

Algorithm 2: The algorithm of AHP for test-sheet composition.

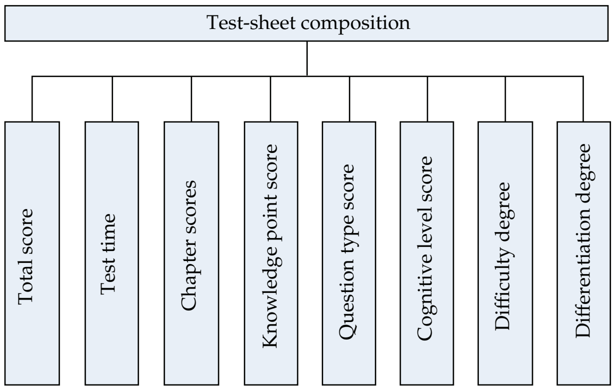

Figure 1: Level tree structure of test-sheet composition.

a relatively high possibility that become high HSI solutions. In BBO, habitat H is a vector of n (suitable index vectors (SIVs)) initialized randomly and then implements migration and mutation operator to achieve the optimal solution. The new candidate solution is generated from the entire habitat in population by using the migration and mutation operators.

In BBO, migration operator can change existing habitat and modify existing solution. Migration is a probabilistic operator that adjusts habitat X_i. The probability X_i modified is proportional to its immigration rate λ_i, and the source of the modified probability from X_j is proportional to the emigration rate μ_j. Migration operator is shown in Algorithm 3.

Mutation is also a probabilistic operator that randomly modifies habitat SIVs based on the habitat a priori probability of existence. Very high HSI solutions and very low

Table 1: Comparison matrix C for test-sheet composition.

	C1	C2	C3	C4	C5	C6	C7	C8
$C1$	1	3	5	6	3	6	5	6
$C2$	1/3	1	2	5	1	3	2	3
$C3$	1/5	1/2	1	1	1/4	3	1	4
$C4$	1/6	1/5	1	1	1/6	1/3	1/5	1/3
$C5$	1/3	1	4	6	1	6	5	6
$C6$	1/6	1/3	1/3	3	1/6	1	1/5	1
$C7$	1/5	1/2	1	5	1/5	5	1	3
$C8$	1/6	1/3	1/4	3	1/6	1	1/3	1

$^*C1, C2, \ldots, C8$ present 8 constraints, respectively, that is, total score, test time, chapter score, knowledge point score, question type score, cognitive level score, difficulty degree, and discrimination degree.

Table 2: Weights ω.

Weights	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8
ω'	0.7511	0.3098	0.1738	0.0700	0.4924	0.0914	0.2130	0.0930
ω	0.3423	0.1412	0.0792	0.0319	0.2244	0.0417	0.0970	0.0424

HSI solutions are equally improbable. Medium HSI solutions are relatively probable. The mutation rate m is expressed as

$$m = m_{\max}\left(\frac{1 - P_s}{P_{\max}}\right), \tag{4.1}$$

where $m_{\max}$ is a user-defined parameter.

Additionally, the mutation operator tends to increase the population diversity. Mutation can be described in Algorithm 4. The basic framework of BBO algorithm can be simply described in Algorithm 5. More details about the migration operator, mutation operator, and BBO algorithm can be found in [8] and in the Matlab code [17].

5. Tabu Search (TS)

Tabu search (TS) [18] is a metaheuristic framework which takes advantage of its use of adaptive memory strategies to manage simultaneously the variety and intensification searches in the solution space of the optimization problem under way. The TS framework is composed by the following major components.

(1) *Solution Configuration*

The candidate solution for the optimization problem under way is encoded by a configuration. Relying on the characteristic of the problem, the solution configuration $x = [x_1, x_2, \ldots, x_n]$, which contains n decision variables, could be a vector taking real numbers, binary values, or mixed combinations of them. Initially, a configuration is set randomly within the reasonable ranges of the decision variables.

Table 3: Random index RI for different n.

n	3	4	5	6	7	8	9	10	11	12	13	14
RI	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.54	1.56	1.57

```
Begin
    for $i = 1$ to NP
        Select $X_i$ with probability based on $\lambda_i$
        if rand$(0, 1) < \lambda_i$ then
            for $j = 1$ to NP
            Select $X_j$ with probability based on $\mu_j$
            if rand$(0, 1) < \mu_j$ then
                Randomly select an SIV $\sigma$ from $X_j$
                Replace a random SIV in $X_i$ with $\sigma$
            end if
            end for
        end if
    end for
End.
```

Algorithm 3: The algorithm of habitat migration operator.

(2) *Move Function*

A move function $S(x)$ is used to guide the solution configuration x to explore its local solution space. The step size of the move should be limited in an appropriate range according to the practical problem. An oversized step makes the search to step over a promising neighbor, while an undersized step causes the algorithm to be inefficient. By performing the move function, the solution configuration is turned into a neighboring configuration, that is, $x' = S(x)$, where x' is a neighboring configuration of x.

(3) *Neighborhood*

The move function defines a neighborhood which bounds the neighboring configurations which are reachable by implementing a move operation to the current solution configuration using $S(x)$. Generally, the neighborhood of solution x is defined by $\Omega(x) = \{x' \mid x' = S(x)\}$.

```
Begin
    for $i = 1$ to NP
        Compute the probability $P_i$
        Select SIV $X_i(j)$ with probability based on $P_i$
        if rand$(0, 1) < m_i$ then
            Replace $X_i(j)$ with a randomly generated SIV
        end if
    end for
End.
```

Algorithm 4: The algorithm of mutation operator.

```
Begin
    1: Initialization. Set the generation counter $G = 1$; Initialize the population $P$ randomly
        and each habitat corresponding to a potential solution to the given problem.
    Step 2: Evaluate the fitness for each individual in $P$
    Step 3: While the termination criteria is not satisfied do
        Sort the population from best to worst.
        For each habitat, map the HSI to the number of species $S$,
        Calculate the immigration rate $\lambda_i$ and the emigration rate $\mu_i$ for each individual $X_i$.
        Modify the population with the migration operator shown in Algorithm 3.
        Update the probability for each individual.
        Mutate the population with the mutation operation shown in Algorithm 4.
        Evaluate the fitness for each individual in $P$.
        Sort the population from best to worst.
        $G = G + 1$;
    Step 4: end while
End.
```

Algorithm 5: The algorithm of biogeography based optimization.

(4) *Tabu List*

A tabu list which stops a recent move to be reversed is kept and modified in the TS framework. Once a move (say, swapping the ith and the jth entries of x) is carried out, the inverse of this move (swapping the two entries again) is recorded in the tabu list and is labeled tabu-active. All the tabu moves are excluded from the neighborhood $\Omega(x)$, and thus become inaccessible.

(5) *Aspiration Criterion*

To allow the solution configuration to move to an attractive but tabu neighboring configuration, an aspiration criterion is strategically designed to overrule the tabu status of such a desired move. Aspiration criterion provides a restricted degree of freedom in accepting a tabu move that achieves a threshold of attractiveness.

(6) *Stopping Criterion*

The stopping criterion depends on the purpose of the problem. There are a number of alternatives such as a minimal solution quality level, a given CPU time limit, a maximal number of iterations between two improvements of the best solution found.

Basically, the TS approach can be summarized as shown in Algorithm 6.

6. Tabu Search/Biogeography-Based Optimization (TS/BBO)

As we all know, the standard TS algorithm is good at exploring the search space and locating the region of global minimum, but it is relatively slow at exploitation of the solution. On the other hand, standard BBO algorithm is usually quick at the exploitation of the solution, though its exploration ability is relatively poor. Therefore, in our work, a hybrid metaheuristic algorithm by integrating tabu search into biogeography-based optimization,

```
Begin
    Step 1: Randomly generate an initial configuration, $x = [x_1, x_2, \ldots, x_n]$, with an
            empty tabu list
    Step 2: Repeat the following statements until the stopping criterion is met
        Step 2.1: Identify the neighborhood $\Omega(x) = \{x' \mid x' = S(x)\}$ of current
                  configuration $x$, and consider a subset $H(x)$ from $\Omega(x)$
        Step 2.2: Choose the best (in terms of a moving evaluation score)
                  configuration $x^* \in H(x)$ which is either nontabu or is tabu
                  but satisfying the aspiration criterion
        Step 2.3: Update the tabu list and let $x = x^*$
    Step 3: Output the best configuration found
End.
```

Algorithm 6: Algorithm of tabu search.

so-called TS/BBO, is used to solve the problem of test-sheet composition. The difference between TS/BBO and BBO is that the hybrid migration operator is used to replace the original BBO mutation operator. In this way, this method can explore the new search space by the mutation of the TS algorithm and exploit the population information with the migration of BBO, and therefore it can overcome the lack of the exploitation of the TS algorithm. In the following, we will show the algorithm TS/BBO which is a variety of TS and BBO. Firstly, we describe tabu search migration and mutation operation, and then a mainframe of TS/BBO is shown.

6.1. Tabu Search Migration and Mutation Operation

The critical operator of TS/BBO is the tabu search migration operator, which composes the tabu search with the migration of BBO. In this algorithm, we can find that the migration between the population X_i and X_j has no repeat because of the use of tabu search. The core idea of the proposed hybrid migration operator is based on two considerations. First, poor solutions can receive many new features from good solutions. Second, the final solution cannot include any repeat SIV. Pseudocode of tabu search migration operation can be described in Algorithm 7.

In the same way, we compose the tabu search and original mutation operator which modifies the habitat with low HSI in order to avoid the repeat to improve the search efficiency. Pseudocode of tabu search mutation operation can be described in Algorithm 8.

6.2. Mainframe of TS/BBO

By incorporating above-mentioned tabu search migration and mutation operator into original BBO algorithm, the TS/BBO has been developed as a new algorithm. TS/BBO algorithm is given in Algorithm 9.

```
Begin
    for $i$ = 1 to NP
        Select $X_i$ with probability based on $\lambda_i$
        if rand(0, 1) < $\lambda_i$ then
          for $j$ = 1 to NP
          Select $X_j$ with probability based on $\mu_j$
          ALLOW = $X_j$
          TS = $\Phi$
          for $k$ = 1 to $D$
            if rand(0, 1) < $\mu_j$ then
              Randomly select an SIV $\sigma$ from ALLOW
              Replace a random SIV in $X_i$ with $\sigma$
              ALLOW = ALLOW$\setminus\{\sigma\}$
              TS = TS $\cup$ $\{\sigma\}$
            end if
          end for
          end for
        end if
    end for
End.
```

Algorithm 7: Tabu search migration operator of BBO.

```
Begin
    for $i$ = 1 to NP
        Compute the probability $P_i$
        Select SIV $X_i(j)$ with probability based on $P_i$
        ALLOW = {1 to $M$}
        TS = $\Phi$
        if rand(0, 1) < $m_i$ then
          Replace $X_i(j)$ with a randomly generated SIV from ALLOW
          ALLOW = ALLOW$\setminus${SIV}
          TS = TS $\cup$ {SIV}
        end if
    end for
End.
```

Algorithm 8: Tabu search mutation operator of BBO.

6.3. Algorithm TS/BBO for Test-Sheet Composition

In TS/BBO, the standard continuous encoding of TS/BBO cannot be used to solve test-sheet composition directly. In order to apply TS/BBO to test-sheet composition, one of the key issues is to construct a direct relationship between the test-sheet sequences and the vector of individuals in TS/BBO.

```
Begin
    Step 1: Initialization. Set the generation counter $G = 1$; and randomly initialize
            a population of NP individuals $P$, the parameter TS, $M$, ALLOW
    Step 2: Evaluate the fitness(HSI) for each individual in $P$ according to (4.1).
    Step 3: while The halting criteria is not satisfied do
            Sort the population from worst to best according to HSI
            For each individual, map the HSI to the number of species
            Calculate the immigration rate $\lambda_i$ and the emigration rate $\mu_i$ for each
            individual $X_i$
            Modify the population with the tabu search migration operator shown in
            Algorithm 7
            Update the probability for each individual
            Mutate the population with the tabu search mutation operator shown in
            Algorithm 8
            Evaluate the fitness for each individual in $P$
            Memorize the best solution achieved so far
              $G = G + 1$
    Step 4: end while
End.
```

Algorithm 9: Algorithm description of TS/BBO.

6.3.1. Preprocessing

We will do the following preprocessing before the design of TS/BBO algorithm for test-sheet composition

(a) We regroup all the questions in item bank according to question type, and then the candidates take the same question type together to form a subset. So all the questions in item bank can be divided into several different subsets, and then all questions will be renumbered.

(b) Because question type in the requirements and item bank has the same score, we can calculate the number of questions required for every question type, and then we get the total number of questions in a testsheet.

6.3.2. Encoding

When using TS/BBO to solve test-sheet composition problem, the status code for each habitat represents a candidate solution, that is, a test-sheet composition scheme. Therefore, how to determine the effective habitat status code is a key issue.

Traditional encoding method is as follows: the status code of each habitat is represented by a binary string whose length is the number of questions in total in item bank, and the number “1” indicates that the question corresponding to the number is selected, while number “0” indicates that the question corresponding to the number is not selected; the length of the number “1” indicates the number of questions contained in total in a testsheet. This encoding method is simple and maximizes a random search at most, but it has increased the search space, reducing the search efficiency.

Therefore, an alternative encoding method is proposed in this paper. This method rearranges questions into different subsets according to the different question type.

The status code of each habitat is represented by an l-dimensional vector $x = (x_1, x_2, \dots, x_p)$. The dimension l for vector x depends on total questions in a testsheet, where the vector $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,l_i})$ is composed by the ith question type, the integer $x_{i,j} \in [1, M]$ means the jth question in the ith question type; $1 \le i \le p$, $1 \le j \le l_i$, p is the total question in item bank; l_i is the number of the ith question type, and $\sum_{i=1}^{p} l_i = l$, l is the total question in some test-sheet composed.

For example, we have seven questions $(1.a, 2.b, 3.c, 4.d, 5.e, 6.f,$ and $7.g)$ in item bank; questions a and g belong to question type 1, 1 score for each question; questions b, e, and f belong to question type 2, 2 scores for each question; questions c and d belong to question type 3, 3 scores for each question. Questions in item bank will be rearranged and renumbered according to question type and the result is $(1.a, 2.g, 3.b, 4.e, 5.f, 6.c,$ and $7.d)$. According to the question type requirement $\{$question type 2 : question type 3$\} = \{4:3\}$, we get a total of 3 questions in the testsheet, that is, $M = 7$. $p = 3$, $l_1 = 0$, $l_2 = 2$, $l_3 = 1$, and $l = l_1 + l_2 + l_3 = 3$. So (4, 5, 7) represents a test-sheet composition scheme; that is, it represents a test-sheet containing three questions which are questions e, f, and d.

6.3.3. The Algorithm TS/BBO for Test-Sheet Composition

Improved BBO can adapt to the needs of testsheet, while optimization algorithms can improve the BBO fast search capabilities and increase the search to the global possible optimum solution. HSI in Habitat i is represented by the objective function $f(x_i)$ in test-sheet composition model the smaller the value f, then the higher HSI in Habitat i.

Based on the above analysis, the pseudo code of improved BBO-TS/BBO for test-sheet composition is described as shown in Algorithm 10.

7. Simulation Experiments

In this section, we look at the performance of TS/BBO as compared with BBO and other population-based optimization methods. Firstly, we compare performances between BBO and TS/BBO, and then we compare performances between TS/BBO and other population-based optimization methods such as ACO, DE, ES, GA, PBIL, PSO, and SGA.

To allow a fair comparison of running times, all the experiments were performed on a PC with a Pentium IV processor running at 3.0 GHz, 1 GB of RAM and a hard drive of 160 Gbytes. Our implementation was compiled using MATLAB R2012a (7.14) running under Windows XP. In the following, we will describe the problem we use to test the performance of the TS/BBO.

7.1. TS/BBO versus BBO

We randomly generate an item bank with M (e.g., $M = 10000$) questions, and then choose some questions to form a testsheet. Its constraints are shown as follows:

(1) the total score is 100;

(2) the test time is 120 minutes;

(3) chapter score $\{1:2:3:4:5:6:7:8:9:10\} = \{6:6:8:12:9:12:12:15:10:10\}$;

```
Begin
    Step 1: Preprocessing. Preprocess the questions in item bank described in
            subsection 6.3.1, and determine $L$ which is the total number of questions
            contained in a test-sheet
    Step 2: Initializing. Set the generation counter $G = 1$; set the status code of habitat $i$
            (i.e., SIV) $x_i$ according to subsection 6.3.2; randomly initialize a
            population of NP individuals $P$; set the maximum variation rate $m_{\max}$ and
            migration rate $p_{\text{mod}}$; set dimension for the optimization problem $D$; set the
            maximum capacity of habitat species $S_{\max}$; set maximum of immigration
            function $I$ and the maximum of emigration function $E$ and the maximum of
            elite individuals retained $z$; set the parameter TS, $M$, ALLOW related to
            tabu search
    Step 3: Calculating the immigration and emigration rate. Calculating the species
            number $S_i$, the immigration $\lambda(S_i)$ and emigration rate
            $\mu(S_i)$ $(i = 1, 2, \ldots, n)$ corresponding to habitat $i$ based on its SIV $f(x_i)$
            according to (2.10)
    Step 4: while The halting criteria is not satisfied do
            Sort the population from worst to best according to HSI
            For each individual, map the HSI to the number of species
            Calculate the immigration rate $\lambda_i$ and the emigration rate $\mu_i$ for each
            individual $X_i$
            Modify the population with the taboo search migration operator shown in
            Algorithm 7
            Update the probability for each individual
            Mutate the population with the taboo search mutation operator shown in
            Algorithm 8
            Evaluate the fitness for each individual in $P$
            Memorize the best solution achieved so far
            $G = G + 1$
    Step 5: end while
End.
```

Algorithm 10: Algorithm of TS/BBO for test-sheet composition.

(4) knowledge point score {1 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10:11:12:13:14:15:16:17:18:19:20} = {3 : 3 : 3 : 3 : 4 : 4 : 6 : 6 : 4 : 5 : 6 : 6 : 6 : 6 : 6 : 9 : 5 : 5 : 5 : 5} (a chapter contains two knowledge points);

(5) question type score {Type 1 : Type 2 : Type 3 : Type 4} = {10 : 16 : 24: 50};

(6) cognitive level score {Memorizing : Understanding : Applying : Analyzing : Comprehending : Evaluating} = {10 : 15 : 20 : 30 : 15 : 10};

(7) difficulty degree is 0.4900;

(8) discrimination degree is 0.4050.

We get the goal weights for test-sheet composition {total score : test time : chapter score : knowledge point score : questions type score : cognitive level score : difficulty degree : discrimination degree} = {0.3423 : 0.1412 : 0.0792 : 0.0319 : 0.2244 : 0.0417 : 0.0970 : 0.0424} according to the Algorithm 2 as shown in Section 3.2.

For BBO and TS/BBO, we used the same following parameters: habitat modification probability = 1, immigration probability bounds per gene = [0, 1], step size for numerical integration of probabilities = 1, maximum immigration and migration rates for each

Table 4: Best normalized optimization results on test-sheet composition problem. The numbers shown are the best results found after 100 Monte Carlo simulations of BBO and TS/BBO algorithm.

Parameter			Algorithm	
Popsize	*Maxgen*	*Keep*	BBO	TS/BBO
50	50	10	3.2243	**3.1316**
50	50	2	3.8055	**3.7929**
100	50	2	**3.7608**	3.8957
100	100	2	3.4048	**3.1300**
100	100	10	3.5727	**2.6118**

island = 1, and mutation probability = 0.005. We must point out that TS/BBO needs to set tabu search table except for the above parameters. To compare the different effects among the parameters *Popsize*, *Maxgen*, and *Keep* (the number of elitisms), we ran 100 Monte Carlo simulations of BBO and TS/BBO algorithm on the above problem to get representative performances. Tables 4, 5, 6, and 7 show the results of the simulations. In other words, Tables 4, 5, and 6 shows the best, worst, and average performance of BBO and TS/BBO algorithm found by BBO and TS/BBO algorithm over 100 Monte Carlo runs, respectively, while Table 7 shows the average CPU time consumed by BBO and TS/BBO algorithm averaged over 100 Monte Carlo runs.

From Table 4, we see that TS/BBO performed the best (on average) on four of the 5 groups. Table 5 shows that BBO was the worst at finding objective function minima when multiple runs are made. Table 6 shows that TS/BBO performed the best on average on all the five groups. Table 7 shows that TS/BBO was a little more effective at finding objective function minima when multiple runs are made, performing the best on 3 of the 5 groups, while BBO performed the best effective on two of the groups. In sum, from Tables 4, 5, 6, and 7, we can draw the conclusion that the more generations, the more populations, and the more elitisms, the smaller objective function value, while the CPU time consumes more.

7.2. *TS/BBO versus Population-Based Optimization Method*

In order to explore the benefits of TS/BBO, in this section we compared its performance on test-sheet composition problem with seven other population-based optimization methods. ACO (ant colony optimization) [19–21] is an algorithm that is based on the pheromone deposition of ants. DE (differential evolution) [22–25] is a simple method that uses the difference between two solutions to probabilistically adapt a third solution. An ES (evolutionary strategy) [26, 27] is an algorithm that allows more than two parents to reproduce an offspring. A GA (genetic algorithm) [28–32] is a method that is based on natural selection in the theory of biological evolution. PBIL (probability-based incremental learning) [33] is a type of GA that maintains statistics about the population instead of maintaining the population directly. PSO (particle swarm optimization) [34] is based on the swarming behavior of birds, fish, and other creatures. A stud genetic algorithm (SGA) [35] is a GA that uses the best individual at each generation for crossover.

We did some fine tuning on each of the optimization algorithms to get optimal performance, to get the optima for every algorithm. For ACO, we used the following parameters: initial pheromone value $\tau_0 = 1E - 6$, pheromone update constant $Q = 20$, exploration constant $q_0 = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay

Table 5: Worst normalized optimization results on test-sheet composition problem. The numbers shown are the best results found after 100 Monte Carlo simulations of BBO and TS/BBO algorithm.

Parameter			Algorithm	
Popsize	*Maxgen*	*Keep*	BBO	TS/BBO
50	50	10	**6.0553**	5.3541
50	50	2	**6.1344**	5.6907
100	50	2	**5.7704**	5.5479
100	100	2	**5.4701**	4.8595
100	100	10	**6.1830**	4.5499

Table 6: Mean normalized optimization results on test-sheet composition problem. The numbers shown are the minimum objective function values found by BBO and TS/BBO algorithm, averaged over 100 Monte Carlo simulations.

Parameter			Algorithm	
Popsize	*Maxgen*	*Keep*	BBO	TS/BBO
50	50	10	4.5200	**4.2490**
50	50	2	4.9518	**4.7521**
100	50	2	4.7500	**4.7138**
100	100	2	4.2129	**4.0310**
100	100	10	4.4600	**3.4253**

rate $\rho_l = 0.5$, pheromone sensitivity $\alpha = 1$, and visibility sensitivity $\beta = 5$. For TS/BBO, we used the parameters presented in Section 7.1. For DE, we used a weighting factor $F = 0.5$ and a crossover constant CR = 0.5. For the ES, we produced $\lambda = 10$ offspring for each generation and standard deviation $\sigma = 1$ for changing solutions. For the GA, we used roulette wheel selection, single-point crossover with a crossover probability of 1, and a mutation probability of 0.01. For PBIL, we used a learning rate of 0.05, 1 good population member, 0 bad population members to use to update the probability vector each generation, and a 0 probability vector mutation rate. For PSO, we used only global learning (no local neighborhoods), an inertial constant = 0.3, a cognitive constant = 1, and a social constant = 1 for swarm interaction. For the SGA, we used single-point crossover with a crossover probability of 1, and a mutation probability of 0.01. Note that the algorithm DE has not the parameter *Keep*, so we cannot compare *Keep* between TS/BBO and DE.

To compare the different effects among the parameters *Popsize*, *Maxgen*, and *Keep* (the number of elitism), we ran 100 Monte Carlo simulations of each algorithm on the above problem to get representative performances. Tables 8, 9, 10, and 11 show the results of the simulations. Tables 8, 9, and 10 show the best, worst, and average performance of each algorithm over 100 Monte Carlo runs, respectively, while Table 11 shows the average CPU time consumed by each algorithm over 100 Monte Carlo runs.

From Table 8, we see that TS/BBO performed the best (on average) on the five groups. Table 9 shows that PBIL was the worst at finding objective function minima on all the five groups when multiple runs are made, while the SGA was the best on three of the groups and TS/BBO was the best on two of the groups in the worst values. Table 10 shows that TS/BBO was the most effective at finding objective function minima when multiple runs are made, performing the best on 3 of the 5 groups, while SGA was the second most effective, performing the best on 2 of the 5 groups. Table 11 shows that TS/BBO was the most effective at finding objective function minima when multiple runs are made, performing the best on

Table 7: Average CPU time on test-sheet composition problem. The numbers shown are the minimum average CPU time (Sec) consumed by BBO and TS/BBO algorithm.

Parameter			Algorithm	
Popsize	*Maxgen*	*Keep*	BBO	TS/BBO
50	50	10	5.80	**5.52**
50	50	2	4.24	**3.60**
100	50	2	9.19	**8.31**
100	100	2	**17.56**	17.78
100	100	10	**25.77**	26.01

Table 8: Best normalized optimization results on test-sheet composition problem. The numbers shown are the best results found after 100 Monte Carlo simulations of each algorithm.

Parameter			Algorithm							
Popsize	*Maxgen*	*Keep*	ACO	DE	ES	GA	PBIL	PSO	SGA	TS/BBO
50	50	10	5.4134	5.1592	5.5487	4.1308	6.7791	5.3212	3.5948	**3.1316**
50	50	2	6.3357	5.8272	6.1941	5.3657	8.1954	6.1651	4.4489	**3.7929**
100	50	2	5.9669	5.6499	5.9860	5.1919	7.9242	5.9516	3.9447	**3.8957**
100	100	2	5.6849	5.2775	5.6952	4.6971	7.8740	5.6846	3.4552	**3.1300**
100	100	10	5.7449	5.2911	5.7152	4.0889	7.6467	5.6881	3.5391	**2.6118**

Table 9: Worst normalized optimization results on test-sheet composition problem. The numbers shown are the best results found after 100 Monte Carlo simulations of each algorithm.

Parameter			Algorithm							
Popsize	*Maxgen*	*Keep*	ACO	DE	ES	GA	PBIL	PSO	SGA	TS/BBO
50	50	10	7.1331	6.7607	7.2079	6.2192	**9.8729**	7.0046	6.0518	**5.3541**
50	50	2	7.0307	6.5340	6.9859	6.6226	**9.8666**	6.8968	**5.6263**	5.6907
100	50	2	6.8203	6.4536	6.9457	6.3000	**9.2658**	7.0567	**5.1012**	5.5479
100	100	2	6.6193	6.1777	6.5664	5.9887	**9.2589**	6.5046	**4.6660**	4.8595
100	100	10	6.7001	6.1475	6.5471	5.0184	**9.2101**	6.5030	4.7275	**4.5499**

all the 5 groups. In sum, from Tables 8, 9, 10, and 11 we can draw the conclusion that the more generations, the more populations, and the more elitisms, the smaller objective function value, while the CPU time consumes more. From Table 8 groups 2 and 3 (i.e., row 2 and row 3), we can arrive at a conclusion that the objective function value is not always better when *Popsize* is increasing. From Table 8 groups 1 and 2 (i.e., row 2 and row 3) or groups 4 and 5 (i.e., row 4 and row 5), we can come to a conclusion that the more *Keep*, the better results when other parameters are the same. Also, From Table 8 we can reach a decision that increasing *Maxgen* can get more effective than increasing *Popsize*.

The simulation implemented in this section shows that the algorithm TS/BBO that we proposed performed the best and most effectively, and it can solve the test-sheet problem perfectly.

8. Conclusion

To improve performance for test-sheet composition, we combined the advantage of tabu search and biogeography-based optimization and proposed a new algorithm TS/BBO.

Table 10: Mean normalized optimization results on test-sheet composition problem. The numbers shown are the minimum objective function values found by each algorithm, averaged over 100 Monte Carlo simulations.

Parameter			Algorithm							
Popsize	*Maxgen*	*Keep*	ACO	DE	ES	GA	PBIL	PSO	SGA	TS/BBO
50	50	10	6.4966	6.0329	6.4322	5.1371	8.7371	6.4414	4.7929	**4.2490**
50	50	2	6.5258	6.0020	6.3800	5.5267	8.4413	6.3501	**4.5824**	4.7521
100	50	2	6.3249	5.6499	6.3452	5.5034	8.3996	6.3087	4.1814	**4.7138**
100	100	2	6.0259	5.2775	6.0369	4.9790	8.3465	6.0256	**3.6625**	4.0310
100	100	10	6.0896	5.2911	6.0581	4.3342	8.1055	6.0294	3.7515	**3.4253**

Table 11: Average CPU time on test-sheet composition problem. The numbers shown are the minimum average CPU time (Sec) consumed by each algorithm.

Parameter			Algorithm							
Popsize	*Maxgen*	*Keep*	ACO	DE	ES	GA	PBIL	PSO	SGA	TS/BBO
50	50	10	6.37	6.37	4.52	3.87	3.55	4.27	4.94	**3.21**
50	50	2	6.36	6.36	3.56	3.90	4.56	4.17	4.38	**2.21**
100	50	2	13.51	26.09	7.83	8.54	7.96	10.47	10.36	**7.21**
100	100	2	26.09	26.09	15.64	17.35	15.40	20.52	22.69	**13.20**
100	100	10	26.03	26.03	15.51	16.79	18.34	20.41	23.61	**13.01**

Simulation experiment demonstrates that AHP and TS/BBO that we proposed for test-sheet composition optimization problem have the following advantages.

(a) Speed up extracting questions and compute the objective function after preprocessing the item bank according to question type.

(b) Habitat encoding length is equal to the total number of testsheet, saving storage space and reducing the optimization space, to ensure that the constraint for total score is met and improve the solution accuracy.

(c) We use subencoding according to the question type to ensure that constraint for question type score is satisfied, reducing the optimization space and improving the solution accuracy.

(d) AHP determines the test-sheet composition weights, comprehensively considering the objective and subjective factors, in line with the actual test environment, fitting real test needs.

(e) Tabu search optimizes mutation and migration operator to create the algorithm TS/BBO, improving solution efficiency and solution accuracy.

However, the algorithm TS/BBO that we proposed in this paper has the following disadvantages: the preprocessing time increases when the number of questions in item bank is getting bigger, which will affect the entire test-sheet composition speed; need further optimization to improve accuracy; need for further ease the conflict between expanding population diversity and reducing the optimization space. The above problems are worth further study.

References

[1] G. J. Hwang, B. M. T. Lin, and T. L. Lin, “An effective approach for test-sheet composition with large-scale item banks,” *Computers & Education*, vol. 46, no. 2, pp. 122–139, 2006.

[2] Y. C. Lin, Y. T. Lin, and Y. M. Huang, “Development of a diagnostic system using a testing-based approach for strengthening student prior knowledge,” *Computers & Education*, vol. 57, no. 2, pp. 1557–1570, 2011.

[3] E. S. M. El-Alfy and R. E. Abdel-Aal, “Construction and analysis of educational tests using abductive machine learning,” *Computers & Education*, vol. 51, no. 1, pp. 1–16, 2008.

[4] F. Wang, W. H. Wang, Q. K. Pan, and G. Cheng, “Intelligent test-sheet composition research based on harmony search algorithm,” *Computer Simulation*, vol. 27, pp. 298–301, 2010.

[5] Y. Liu, Y. Wang, Y. Du, and J. Zhang, “Multi-object intellectual test paper assembling based on adaptive operator genetic algorithm,” *Computer Applications*, no. S1, pp. 22–24, 2008.

[6] H. Y. Lin, J. M. Su, and S. S. Tseng, “An adaptive test sheet generation mechanism using genetic algorithm,” *Mathematical Problems in Engineering*, vol. 2012, Article ID 820190, 18 pages, 2012.

[7] T. F. Ho, P. Y. Yin, G. J. Hwang, S. J. Shyu, and Y. N. Yean, “Multi-objective parallel test-sheet composition using enhanced particle swarm optimization,” *Educational Technology & Society*, vol. 12, no. 4, pp. 193–206, 2009.

[8] D. Simon, “Biogeography-based optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[9] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, “Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm,” *Journal of Sensor and Actuator Networks*, vol. 1, no. 2, pp. 86–96, 2012.

[10] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, and M. Shao, “Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm,” *Advanced Science, Engineering and Medicine*, vol. 4, no. 6, pp. 550–564, 2012.

[11] X. Li, J. Wang, J. Zhou, and M. Yin, “A perturb biogeography based optimization with mutation for global numerical optimization,” *Applied Mathematics and Computation*, vol. 218, no. 2, pp. 598–609, 2011.

[12] G. J. Hwang, H. C. Chu, P. Y. Yin, and J. Y. Lin, “An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests,” *Computers & Education*, vol. 51, no. 3, pp. 1058–1072, 2008.

[13] T. L. Saaty, *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*, McGraw-Hill, New York, NY, USA, 1980.

[14] F. Zahedi, “The analytic hierarchy process-a survey of the method and its applications,” *Interfaces*, vol. 16, no. 4, pp. 96–108, 1986.

[15] M. V. Lomolino, B. R. Riddle, R. J. Whittaker, and J. H. Brown, *Biogeography*, Sinauer Associates, Sunderland, Mass, USA, 4th edition, 2010.

[16] H. Ma, “An analysis of the equilibrium of migration models for biogeography-based optimization,” *Information Sciences*, vol. 180, no. 18, pp. 3444–3464, 2010.

[17] D. Simon, “The Matlab code of biogeography-based optimization,” http://academic.csuohio.edu/simond/bbo/.

[18] F. Glover, “Tabu search-part I,” *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[19] M. Dorigo, M. Birattari, and T. Stützle, “Ant colony optimization—artificial ants as a computational intelligence technique,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[20] M. Dorigo, L. M. Gambardella, M. Middendorf, and T. Stützle, “Guest editorial: special section on ant colony optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 317–320, 2002.

[21] A. Ketabi and R. Feuillet, “Ant colony search algorithm for optimal generators startup during power system restoration,” *Mathematical Problems in Engineering*, vol. 2010, Article ID 906935, 11 pages, 2010.

[22] Z. Rui and W. Cheng, “A hybrid differential evolution and tree search algorithm for the job shop scheduling problem,” *Mathematical Problems in Engineering*, vol. 2011, Article ID 390593, 20 pages, 2011.

[23] A. Ketabi and M. J. Navardi, “Optimization shape of variable capacitance micromotor using differential evolution algorithm,” *Mathematical Problems in Engineering*, vol. 2010, Article ID 909240, 15 pages, 2010.

[24] W.-H. Ho and A. L.-F. Chan, “Hybrid Taguchi-differential evolution algorithm for parameter estimation of differential equation models with application to HIV dynamics,” *Mathematical Problems in Engineering*, vol. 2011, Article ID 514756, 14 pages, 2011.

[25] A. H. Gandomi, X.-S. Yang, S. Talatahari, and S. Deb, "Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization," *Computers & Mathematics with Applications*, vol. 63, no. 1, pp. 191–200, 2012.

[26] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford, Miss, USA, 1996.

[27] H.-G. Beyer, *The Theory of Evolution Strategies*, Springer, New York, NY, USA, 2001.

[28] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[29] P. C. Chen, C. W. Chen, and W. L. Chiang, "GA-based fuzzy sliding mode controller for nonlinear systems," *Mathematical Problems in Engineering*, vol. 2008, Article ID 325859, 16 pages, 2008.

[30] S. T. Pan, "CSD-coded genetic algorithm on robustly stable multiplierless IIR filter design," *Mathematical Problems in Engineering*, vol. 2012, Article ID 560650, 15 pages, 2012.

[31] M. Shahsavar, A. A. Najafi, and S. T. A. Niaki, "Statistical design of genetic algorithms for combinatorial optimization problems," *Mathematical Problems in Engineering*, vol. 2011, Article ID 872415, 17 pages, 2011.

[32] A. H. Gandomi and A. H. Alavi, "Multi-stage genetic programming: a new strategy to nonlinear system modeling," *Information Sciences*, vol. 181, no. 23, pp. 5227–5239, 2011.

[33] I. C. Parmee, *Evolutionary and Adaptive Computing in Engineering Design*, Springer, Berlin, Germany, 2001.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1945, Perth, Australia, 1995.

[35] W. Khatib and P. Fleming, "The stud GA: a mini revolution?" in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pp. 683–691, Springer, 1998.