Mathematical Biosciences and Engineering

https://www.aimspress.com/journal/mbe

MBE, 21(3): 3668–3694.
DOI: 10.3934/mbe.2024162
Received: 05 December 2023
Revised: 17 January 2024
Accepted: 29 January 2024
Published: 18 February 2024

Research article

Exploring Multiple-Objective Optimization for Efficient and Effective Test Paper Design with Dynamic Programming Guided Genetic Algorithm

Han Wang[1,2], **Qingfeng Zhuge**[1,*], **Edwin Hsing-Mean Sha**[1], **Jianghua Xia**[1] **and Rui Xu**[3]

[1] School of Computer Science and Technology, East China Normal University, Shanghai 200062, China

[2] Shanghai Institute of AI for Education, East China Normal University, Shanghai 200062, China

[3] Department of Computer Science, City University of Hong Kong, Hong Kong, China

* **Correspondence:** Email: qfzhuge@cs.ecnu.edu.cn.

Abstract: Automatic test paper design is critical in education to reduce workloads for educators and facilitate an efficient teaching process. However, current designs fail to satisfy the realistic teaching requirements of educators, including the consideration of both test quality and efficiency. This is the main reason why teachers still manually construct tests in most teaching environments. In this paper, the quality of tests is quantitatively defined while considering multiple objectives, including a flexible coverage of knowledge points, cognitive levels, and question difficulty. Then, a model based on the technique of linear programming is delicately designed to explore the optimal results for this newly defined problem. However, this technique is not efficient enough, which cannot obtain results in polynomial time. With the consideration of both test quality and generation efficiency, this paper proposes a genetic algorithm (GA) based method, named dynamic programming guided genetic algorithm with adaptive selection (DPGA-AS). In this method, a dynamic programming method is proposed in the population initialization part to improve the efficiency of the genetic algorithm. An adaptive selection method for the GA is designed to avoid prematurely falling into the local optimal for better test quality. The question bank used in our experiments is assembled based on college-level calculus questions from well-known textbooks. The experimental results show that the proposed techniques can construct test papers with both high effectiveness and efficiency. The computation time of the test assembly problem is reduced from 3 hours to 2 seconds for a 5000-size question bank as compared to a linear programming model with similar test quality. The test quality of the proposed method is better than the other baselines.

Keywords: automated test paper design; multiple objectives optimization; linear programming; dynamic programming; genetic algorithm

1. Introduction

Test paper design is critical in education because it is the main method for teachers to evaluate the learning status of students to facilitate an effective teaching process. To reduce the workload of educators in designing abundant tests during the teaching process, automated test paper design has been widely studied [1–5], where the core technique is to extract questions from a well-managed question bank to serve teaching objectives. Two forms of tests, namely unified tests [1, 2, 4, 5] and adaptive tests [3], are widely applied. As for unified tests, the questions are fixed before testing students and are used for groups. As for adaptive tests, they recommend different questions in an interactive process for students based on historical testing records, which are used for individual students and usually can only be done online using a computer. When fairness and limited resources are the main considerations in real-world test scenarios, such as university entrance examinations on various subjects, designing unified tests for group students is the ultimate requirement. Therefore, this paper mainly focuses on exploring question extraction techniques for unified tests. To design effective tests in reality, multiple factors should be considered at the same time, including question difficulty, cognitive levels, covered knowledge points, and so on. Thus, automatic test construction is widely considered as a multiple-objective optimization problem [6–8]. This is a challenging problem and has proven to be a combinatorial problem of non-deterministic polynomial-time hardness (NP-hardness) [8].

Previously, some kinds of techniques were presented to solve test construction problems. Some studies use randomization approaches to improve the efficiency of test question selection. For example, Naik et al. [1] applied a shuffling algorithm to select questions for building an online test system. This kind of technique is simple and fast, though the effectiveness of the test design cannot be guaranteed due to the random selection, which means it cannot face a more sophisticated testing scenario. Some works apply optimization algorithms, especially linear programming (LP), to select a subset from a question bank that can most satisfy the defined objectives [2, 3]. However, the efficiency of the test construction cannot be guaranteed. The computation latency of test construction grows exponentially with the increasing size of the question bank and the definition of the test objectives. To balance the test quality and generation efficiency, most works explore meta-heuristic algorithms for test paper construction. The considered techniques include genetic algorithms (GA) [4–7, 9–14], ant colony optimization (ACO) [15], particle swarm optimization (PSO) [16], and so on, where GAs obtain a wide interest in the community for solving the problem of test paper construction [9], because it can be easily applied to quickly optimize combinational problems using the representing of chromosomes and obtain satisfying results. In the previous works, it should be noted that there was no unified form of objective definition. In related works, questions regarding difficulty and cognitive levels are widely considered. However, the coverage of knowledge points is not deeply explored, which is one of the most important factors when designing tests. Very few works consider the coverage of knowledge points, and most of them only consider it in a simple way [10, 14], namely that they only consider the number of covered knowledge points in a test. However, in most real cases, a question would relate to multiple knowledge points, and different knowledge points would have different importance values for a course. Thus, a recent work [7] optimized the distribution of knowledge units, called skills in this work, for test construction to be close to the skill importance of a course, where the skill weights of a course were fixed and manually labeled. However, in the whole teaching process, the preference for knowledge points when designing a test would change for different teaching phases. In this case, the method mentioned in [7] is not flexible enough. Moreover, this research is still in its infancy. Although the efficiency and effectiveness of test construction problems have been preliminarily studied, the definition of multiple objectives still needs to be further explored to satisfy the real requirements of educators to face some sophisticated teaching scenarios, which determine the quality of the results of the designed algorithms.

We believe that, considering multi-dimensional teaching requirements, the quantitative definition of test design objectives is crucial to search for a solution space using GAs. To overcome the above-mentioned problems, this work further explores multiple objectives, including the coverage of knowledge points, the cognitive levels of questions, and the question difficulty. Especially, this work introduces a more flexible way to consider the coverage of knowledge points for a more sophisticated scenario, where each question is related to multiple knowledge points and there exists dependency relationships between knowledge points. Different from previous works, some factors of the questions, such as the question difficulty, are not labeled by experts or by using statistical functions; the question difficulty is automatically calculated using the attributes of the question, and the changing importance of knowledge points is automatically gained based on a method of distance calculation, which determines the emphasis degree of a question (in Section 3.2). This scheme provides a more practical way to face sophisticated test design scenarios. Then, to explore the optimal results facing this scenario, a linear programming method is designed that translates multiple non-linear objective functions and constraints into a linear way. By analyzing the results, especially with a large question bank, this method is proven to be not efficient enough. Thus, to obtain both high test quality and high generation efficiency facing this complex test scenario, this work proposes a scheme based on the GA that shows a new way to improve GA-based methods: (1) initializing the population by optimizing the coverage of knowledge points based on dynamic programming; and (2) involving an adaptive selection method that avoids the premature local optimal for GA methods. In-depth discussions and detailed explanations of algorithm design are provided for various test-design algorithms. Our research work shows that GAs with careful designs achieve both high-quality solutions and time-efficient computations compared with various kinds of algorithms. The experimental results show that the proposed methods can obtain high-quality solutions with significantly improved efficiency. The main contributions in this paper are as follows:

- Multiple objectives of the test design problem are defined when considering real test requirements. A flexible way to involve knowledge coverage while considering the different importance of knowledge points is designed. Specifically, the emphasis degree is proposed for the requirements of teachers for changeable key knowledge points of tests, thus providing more flexibility for users to determine the importance of questions;
- The design details of linear programming (LP) for a unified test design with multiple objectives are explained and analyzed to gain optimal results;
- An enhanced GA algorithm for selecting test questions, named dynamic programming guided genetic algorithm with adaptive selection (DPGA-AS), is proposed to improve the fitness and efficiency of the genetic algorithm. Specifically, a dynamic programming method is designed to initialize the population of the GA by solving two, single objective, optimization problems, and the selection method in the GA is improved using an adaptive selection method, thus improving both the effectiveness and efficiency of the GA algorithm;
- A real question bank is constructed and labeled based on questions from a well-known, college-

level, calculus textbook;

- A large number of experiments are conducted. The experimental results are analyzed in deep, and the results show that the proposed algorithm can be over 4000× faster than the technique of linear programming with similar test quality.

The remainder of this article is organized as follows: In the next section, we describe the background of question selection. Section 3 presents model definitions including the problem definition. Section 4 demonstrates our question selection techniques. Section 5 shows the experiments and analysis of the results. Section 6 concludes this paper.

2. Background and related work

To describe our problems and schemes well, this section demonstrates the background of the question selection techniques in detail.

2.1. *High quality tests*

Tests are widely studied due to their crucial role in education. In the learning process, abundant tests, including quizzes and examinations, are used to facilitate the teaching process. Developing high-quality tests is critical for both instructors and students throughout the whole teaching process. On the one hand, tests can stimulate thinking, redirect reasoning, and test students' retention level and application skills [17, 18]; on the other hand, it helps instructors check the progress of reaching their teaching objectives [19, 20]. To develop high-quality tests, we need to decide the major criteria that evaluate the quality of a test and are acceptable in various teaching situations.

Characteristics of a high-quality test should be well-aligned with the learning outcomes [18, 21]. In this case, questions should focus on a targeted knowledge set. What's more, a high-quality test should cover the different cognitive levels of learning taxonomy to assess a student's diverse skills [5, 21, 22]. **Bloom's taxonomy** is the most well-known learning taxonomy. It was first developed by Bloom in the 1950s [23], and was revised in 2001 [24]; the revised version is most widely used and studied in higher education [25]. The revised one is used in this paper. Bloom defines six levels of learning within the cognitive domain, including remembering, understanding, applying, analyzing, evaluating, and creating. The mastery of a skill at a higher level can imply a sufficient degree of mastery at the lower levels [26]. A generally adopted rule is that good tests should cover different difficulty levels [5] to build a good picture of the learning levels and the skills of students. Additionally, a well-designed test assists teachers in identifying problems in learning outcomes and in making good decisions on adjustments.

In reality, educators usually assign appropriate weights to multiple criteria for a test to align with the teaching objectives. For example, the final examination should have full coverage of a set of to-be-tested knowledge points. Therefore, a meaningful model for the quality of tests can be a combination of various weighted factors appropriate for various teaching stages.

2.2. *Test paper construction*

Traditionally, teachers manually select a set of questions from a question bank based on their knowledge and experience to construct a test. However, there are abundant tests that should be constructed into the whole teaching process. In this case, manually selecting questions is time-consuming, tedious, and easy to make mistakes [1, 4]; even for some experienced instructors, it is hard to know whether the selected questions are good or not in a specific situation. Therefore, how to automate the process of question selection needs to be further studied to reduce workloads for instructors and facilitate the teaching process. Generally, to initiate the automatic process, teachers input some requirements, such as the target chapters and the number of questions; then, the problem selection program outputs a test. The question selection algorithms are critical because they determine the quality of tests.

According to the applied techniques for automatic test paper construction, existing works can be divided into three categories. The first category is randomization-based methods, which randomly select questions from a well-managed question bank [1, 27–30]. For example, Naik et al. [1] proposed a shuffling algorithm to build an online question paper generation system. This kind of method is simple and fast, though the quality of the generated tests cannot be guaranteed [4, 5]. Thus, for a more sophisticated teaching situation, this method is not practical enough. The second category focuses on improving the quality of generated tests based on optimization algorithms. For example, linear programming techniques are designed to address test assembly problems in [2, 3], while considering question difficulty, the number of questions, the test duration, and so on. They can obtain optimal results by linearly defining the objective function and the constraints. However, in most cases, complex test assembly problems cannot be solved in polynomial time, thus leading to a reduction in the efficiency of test generation. However, besides the test quality, the high efficiency of the test assembly is also favored in real teaching situations.

Considering the trade-offs between the test quality and the efficiency of test generation, most of the related works, which represent the third category, focus on designing the meta-heuristic algorithms to construct test papers. Meta-heuristic algorithms are characterized by their ability to efficiently explore and exploit solution spaces, thus making them widely used to solve sophisticated optimization problems in the real world. These algorithms are inspired by nature, such as mimicking the biological processes and swarm movements of living things. For example, GAs [31] mimic human evolution based on the nature selection theory. Ant colony optimization (ACO) draws inspiration from the foraging behavior of ants [32]. Particle swarm optimization (PSO) imitates the behavior of fish and bird swarms [33]. As the field of meta-heuristic algorithms rapidly develops, a host of algorithms inspired by nature are explored [34–36], and some hybrid algorithms are also studied [37, 38]. For the field of automatic test construction, the applied algorithms include GAs [4–7, 9–14], ACO [15], PSO [16], and so on, where GAs are the most widely studied [9] and can obtain satisfying results in related works. **This paper struggles to explore GAs in deep for test paper construction**. With different purposes, the efficiency of GAs and the quality of tests are different. Related works can be divided into single-objective optimization and multiple-objective optimization. As for single-objective optimization, Zhang et al. [4] tried to make the difficulty coefficient of the test paper reasonable, where the test quality was defined based on the difficulty coefficient. Rahim et al. [5] defined the test quality by how many cognitive levels of Bloom's taxonomy were covered: the most effective test covered all six levels, and the quality was the best. In these works, the quality of tests was only measured by one aspect. However, in reality, the requirements of high-quality tests are various and changing in the teaching process. Therefore, these works are not suitable for a progressive teaching or learning process. Multiple objectives are always considered in real teaching scenarios; in this case, single-objective optimization is not effective enough. Thus, multiple-objective optimization based on GAs is widely studied [6, 7, 10–14] for test construction. They improve the GAs for optimizing several objectives, including question difficulty, cognitive levels, time duration, and so on. For example, Yildirim et al. [11] designed the fitness function with consideration of question difficulty and the frequency of selection to provide diversity among feasible test papers. Han et al. [13] considered the question type, difficulty coefficient, and so on, to design the fitness function. Nguyen et al. [6] improved GAs by generating the initial populations based on a simulated annealing algorithm while considering question difficulty and time duration. Wu et al. [7] introduced the distribution of skills into exam paper generation tasks, and considered the question difficulty in the objectives. There are no unified objectives, and whether selected objectives are suitable for test paper construction is still a question. In the above works, a weight sum format to face the multiple-objective problem was widely applied. Additionally, alternative methods to address multiple-objective and many-objective problems based on evolutionary algorithms were studied, where the main concern was to improve the performance of algorithms while considering both convergence and diversity [39] [40], which could be applied within the automatic test design field in the future. Besides automatic test design, we believe the development of the above techniques could also benefit other application domains, such as mountain railway alignment [41] and passenger transportation management [42].

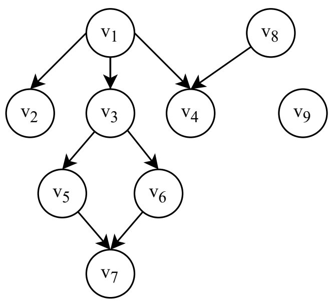
In this paper, the question selection technique is designed based on GA. Different from the previous GA-based works, to meet the various and flexible requirements of instructors, the quality of tests is defined in multiple aspects from the realistic teaching requirements while considering efficiency.

3. Definitions

A well-represented and labeled question bank is of prime importance for automatic question-selection problems. In this section, some terms of the question bank used in this paper are defined and explained.

3.1. Knowledge dependency model

A hierarchical cognitive relationship between knowledge elements is generally adopted in various knowledge areas [43–45]. For example, in logic, multiplication occurs after addition. The hierarchy defines the logical or psychological sequence between knowledge elements. In this work, we use a knowledge dependency graph (KDG) to model the to-be-tested knowledge elements and their interrelationships in a given question bank.



(**a**)KDG

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1	0	1	1	1	0	0	0	0	0
v_2	0	0	0	0	0	0	0	0	0
v_3	0	0	0	0	1	1	0	0	0
v_4	0	0	0	0	0	0	0	0	0
v_5	0	0	0	0	0	0	1	0	0
v_6	0	0	0	0	0	0	1	0	0
v_7	0	0	0	0	0	0	0	0	0
v_8	0	0	0	1	0	0	0	0	0
v_9	0	0	0	0	0	0	0	0	0

(**b**)Adjacent Matrix

Figure 1. An example of a KDG and its adjacent matrix.

Definition 1. *A knowledge dependency graph (KDG) $G = < V, E >$ is a directed acyclic graph (DAG), where $V = < v_1, v_2, ..., v_n >$ is the set of knowledge elements to be tested, and $E \subseteq V \times V$ is the set of edges that represent the dependencies between knowledge elements.*

Figure 1(a) represents an example KDG that involves 9 knowledge elements, where the directed edge $e(v_1, v_2)$ means that element v_1 is a prerequisite for v_2. Usually, a DAG is represented as an adjacent matrix, where the row titles represent outsets and the column titles represent the terminals of the directed edges, as shown in Figure 1(b). For instance, a directed edge from node v_1 to v_2 is represented by a binary value "1" in the cross-section of row v_1 and column v_2 in the adjacent matrix.

3.2. Question attributes

In this section, several attributes are defined for the computation of multiple attributes.

Definition 2. *A question bank $I = < Q, C, T, S, L, Ed >$ is a set that includes the questions and their attributes, where $Q = < q_1, q_2, ..., q_n >$ is the set of the index of questions, $C = < S_{c1}, S_{c2}, ..., S_{cn} >$ is the set of knowledge subset covered in questions, $T = < tb_1, tb_2, ..., tb_n >$ is the set of the cognitive taxonomy of questions, $S = < ns_1, ns_2, ..., ns_n >$ is the set of the number of solving steps in a default answer for questions in Q, $L = < vl_1, vl_2, ..., vl_n >$ is the set of the load of questions, which represents the question difficulty, and $Ed = < ve_1, ve_2, ..., ve_n >$ is the set of the emphasis degree of questions.*

Table 1 shows the attributes of questions and the corresponding explanations. In this table, S_c, *tb*, and *ns* are the basic attributes. *vl* and *ve* are automatically computed based on the basic attributes.

Table 1. The terms and related definitions of Question Attributes.

Notation	Description
S_c	The subset of knowledge elements for each question $q_i \in Q$. Based on a given KDG, there is an assumption that if a question covers a knowledge element v_i, then this question covers all prerequisites of v_i.
tb	The cognitive taxonomy of each question, where the cognitive taxonomy is based on Bloom's taxonomy [24]. The range of *tb* is $< 1, 2, 3, 4, 5, 6 >$ where 1 denotes remembering, 2 denotes understanding, 3 denotes applying, 4 denotes analyzing, 5 denotes evaluating, and 6 denotes creating.
ns	The number of steps in a general solution for each question.
vl	Question load is the objective difficulty/hardness of each question $q_i \in Q$. The computation is with consideration of $\lvert S_c\rvert$, *tb*, and *ns* as shown in Equation 3.1.
ve	The emphasis degree of a question is the relative distance between the knowledge subset of a question and the key knowledge set, where the key knowledge elements are given by instructors. The computation is shown in Equation 3.2.

The question load *vl* is defined as the question difficulty. Three factors, namely $|S_c|$, *tb*, and *ns*, are considered for computation. $|S_c|$ is the number of knowledge elements in a S_c. A question with high $|S_c|$, *tb*, and *ns* is more intuitively challenging. Thus, *vl* is computed as the normalized weighted sum of the above three factors, as shown in Equation 3.1:

$$vl_i = \alpha_1 \frac{|S_{ci}|}{max(|S_c|)} + \alpha_2 \frac{tb_i}{max(tb)} + \alpha_3 \frac{ns_i}{max(ns)}, vl_i \in L, \sum_{j}^{3} \alpha_j = 1, \tag{3.1}$$

where $max(|S_c|)$, $max(tb)$, and $max(ns)$ are the maximum values of these factors among all questions. α_1 to α_3 are the weights for three factors, and the sum of them is 1.

The question emphasis degree ve is defined as the emphasis degree for a question based on a key knowledge set given by educators. The emphasis degree is computed by the distance $D(q_i)$ from the knowledge state of a question $q_i \in Q$ to that of the standard question, as shown in Equation 3.2. The knowledge state of a question, also named the Q-matrix in other works [46, 47], is a binary vector. The length of this vector is the total number of knowledge elements $|V|$. Each element in this vector represents whether a question covers the corresponding knowledge point. When the value of an element is "1", then the knowledge point is in the related knowledge subset S_c. The standard question only covers the key knowledge elements and their prerequisites. For example, if the key knowledge set is $< v_4, v_6 >$, as shown in Figure 1, then the knowledge state of the standard question is $[1, 0, 1, 1, 0, 1, 0, 1, 0]$.

$$\begin{aligned} &D(q_i) = \sum_{j=1}^{|V|} \beta_j |y_j - x_j|, \\ &ve_i = 1 - D(q_i), q_i \in Q, \sum_{j=1}^{|V|} \beta_j = 1, ve_i \in Ed, \end{aligned} \tag{3.2}$$

$D(q_i)$ is designed as the weighted Manhattan distance. The weights for all knowledge elements, β_1 to $\beta_{|V|}$, are automatically computed based on the structure of the given KDG. β_j is the normalized value of $v\beta_j$ and is calculated by $\beta_j = \frac{v\beta_j}{\sum_j^{|V|} v\beta_j}$. Given a key knowledge set, $v\beta_j$ is used to evaluate the distance from the ith knowledge point to the key knowledge element in a KDG, the value of $v\beta_j$ is the lower, the ith knowledge point is further. Specifically, the $v\beta$ of the key knowledge elements is 1. Beginning from them, once passing an edge regardless of the direction in the KDG, the $v\beta$ of the corresponding knowledge element is decreased by σ times. If there are no edges between a knowledge point and the key elements, then its $v\beta$ is 0. For example, when the key knowledge element is v_6, then $v\beta_6 = 1$, $v\beta_3 = \sigma$, $v\beta_1 = \sigma^2$, and $v\beta_9 = 0$. In this paper, $\sigma = 1/2$ is set by default. For a question q_i, the smaller $D(q_i)$ is, the closer the distance is, and the emphasis degree of this question is higher. Thus, the ve of each question is computed by the difference between 1 and $D(q_i)$.

As a result, all attributes are gained. Here is an example, for a question q_1: if $S_{c1} =< v_1, v_2, v_3, v_8 >$, $tb_1 = 1, ns_1 = 1$, and the key knowledge set is $< v_4, v_6 >$, then $vl_1 = 1/3\times4/6+1/3\times1/3+1/3\times1/4 = 0.42$ and $ve_1 = 0.57$.

3.3. Problem definition

The question selection problem is defined as a multi-objective optimization problem under the given constraints. This problem format can be found in most engineering fields, such as maximizing the performance and minimizing the cost. Usually, one best solution cannot ensure that all objectives can get the optimal value. Thus, there are always trade-offs between different solutions. In this work, four objectives are considered and formulated. For the selected test $I_s =< Q_s, C_s, T_s, S_s, L_s, Ed_s >$, it is expected to have (1) a high emphasis degree of the key knowledge set, (2) a high coverage of to-be-tested knowledge elements, (3) a high matching degree of different cognitive levels of questions, and (4) a high closeness to the expected difficulty.

The emphasis degree of a test *ED* is the average emphasis degree of the selected questions in this test. The calculation is shown in Equation 3.3, where ve_i is the emphasis degree of the ith question in Q_s. $m = |Q_s|$ is the number of questions in the test. The range of *ED* is [0, 1], the higher the value is, the better.

$$ED = \frac{\sum_{i=1}^{m} ve_i}{m}, ve_i \in Ed_s. \tag{3.3}$$

The coverage ratio of to-be-tested knowledge elements in a test *CR* is the ratio of the number of knowledge elements included in this test to the total number of the to-be-tested knowledge points. Equation 3.4 shows the calculation, where $|V|$ is the total number of knowledge elements in a given KDG, m is the number of selected questions $|Q_s|$. The range of *CR* is [0, 1], and the higher the value, the better.

$$CR = \frac{|\cap_{i=1}^{m} S_{ci}|}{|V|}, S_{ci} \in C_s. \tag{3.4}$$

The matching degree (*MD*) is the degree of similarity between the given ratios for the question taxonomies and that in the selected test. Equation 3.5 shows the calculation, where l_i is the number of questions of the ith question taxonomy. The *MD* is calculated based on the average absolute error of *nt* kinds of questions, where *nt* is the number of question taxonomies. The range is [0, 1], and the higher the value is, the better.

$$MD = 1 - \frac{\sum_{i=1}^{nt} \left|\frac{l_i}{m} - r_i\right|}{nt}. \tag{3.5}$$

The closeness degree (*CD*) is defined as the closeness degree between the expected average difficulty l_{idx} and the average question difficulty of selected questions. The calculation is shown in Equation 3.6, where $m = |Q_s|$ is the number of selected questions. The range of the *CD* is [0, 1], and the higher the value is, the better.

$$CD = 1 - \left|\frac{\sum_{i}^{m} vl_i}{m} - l_{idx}\right|, vl_i \in L_s, \tag{3.6}$$

The quality of a selected test is measured while considering the aforementioned objectives. The objective function is designed as shown in Equation 3.7, where ω_1, ω_2, ω_3, and ω_4 are the weights for the *ED*, *CR*, *MD*, and *CD*, respectively. The sum of the weights is 1. Different weights can be set for different test situations. The range of the objective function is [0, 1], and the higher the value is, the better the quality of the selected test is.

$$F = \omega_1 ED + \omega_2 CR + \omega_3 MD + \omega_4 CD, \sum_{i=1}^{4} \omega_i = 1. \tag{3.7}$$

Overall, the problem in this paper can be defined as follows:

Definition 3. ***Problem statement****: Given a KDG G, a question bank I, a load index l_{idx}, a key knowledge set PC, a set of the ratios of different cognitive taxonomies $R = < r_1, r_2, ..., r_{nt} >$, and the constraint of the number of questions c_{len}, find a subset of I, $I_s = < Q_s, C_s, T_s, S_s, L_s, Ed_s >$, such that maximizing the objective function under the constraint of c_{len}.*

The load index $l_{idx} \in (0, 1]$ is the expected average difficulty of questions in a test; the size of R depends on a specific cognitive taxonomy. For Bloom's taxonomy, $nt = 6$, $r_1, ..., r_6$ are for memorizing, understanding, applying, analyzing, evaluating, and creating, respectively, and the sum of them is 1.

4. Design

In this section, a model based on linear programming (LP) is designed and implemented. This method can get the optimal value; however, it is costly. Thus, the dynamic programming guided GA method is designed to make the technique of the test design effective and efficient.

4.1. Linear programming based question selection

The design of linear programming for this problem is explored to gain optimal results. Two kinds of variables are defined. x_i is a binary variable that represents whether the ith question is selected. y_j is a binary variable that represents whether the jth knowledge element is covered or not. Based on the problem definition in Section 3.3, this problem is described in Equation 4.1:

$$\begin{aligned}
&\textbf{Maximize} \quad \omega_1 \frac{\sum_{i \in Q}(x_i ve_i)}{\sum_{i \in Q} x_i} + \omega_2 \frac{\sum_{j \in V} y_j}{|V|} + \omega_3 \Big(1 - \frac{\sum_{t=1}^{nt} \big|\frac{\sum_{i \in Q_t} x_i}{\sum_{i \in Q} x_i} - r_t\big|}{nt}\Big) + \omega_4 \Big(1 - \big|\frac{\sum_{i \in Q}(x_i vl_i)}{\sum_{i \in Q} x_i} - l_{idx}\big|\Big) \\
&\textbf{S.t.} \\
&0 < \sum_{i \in Q} x_i \le c_{len}, \\
&\sum_{i \in Q} x_i * G(i,j) \ge y_j, for \ j \in V.
\end{aligned} \tag{4.1}$$

where Q is the question set in the question bank, Q_p is the question set for the pth taxonomy, and $G(i, j)$ is a binary number that represents whether the jth knowledge element is covered in the ith question. As to the constraints, the first is the constraint for the number of selected questions, and the second is for the objective of the CR. However, this representation is not linear and cannot be solved by linear programming. In this case, a series of transformations are made.

Theorem 1. *Given n binary variables, $< x_1, x_2, ..., x_n >$, and n positive real numbers $< pr_1, pr_2, ..., pr_n >$, $a = \sum_{i=1}^{n} x_i, b = \sum_{i=1}^{n}(x_i * pr_i), a \in N^*, b \in R^+$. The description "$F = \frac{b}{a}$" $\Longleftrightarrow$ "$a * F = b$".*

$$F \sum_{i \in Q} x_i = \omega_1 \sum_{i \in Q}(x_i ve_i) + \omega_2 \frac{\sum_{i \in Q} \sum_{j \in V} x_i y_j}{|V|} + \omega_3 \Big(\sum_{i \in Q} x_i - \frac{\sum_{t=1}^{nt} |\sum_{i \in Q_t} x_i - r_t * \sum_{i \in Q} x_i|}{nt}\Big) + \omega_4 \Big(\sum_{i \in Q} x_i - \big|\sum_{i \in Q}(x_i vl_i) - l_{idx} \sum_{i \in Q} x_i\big|\Big) \tag{4.2}$$

Based on Theorem 4.1, the objective function F can be converted to Equation 4.2. Then, by finding the maximal value of $F \sum_{i \in Q} x_i$ under different constraints of the number of selected questions, F can be gained by dividing $\sum_{i \in Q} x_i$. However, the $x_i * y_j$ and the absolute value are not linear. To make them in a linear mode, two theorems are presented as follows:

Theorem 2. *Given two binary variables, $a, b \in [0, 1]$, the description "$a * b$" can be linearly modeled as c, subject to $a + b \ge 2c$, and $a + b \le 1 + c$, where c is a binary number.*

Proof. There are four cases of (a, b): when (a, b) is in $< (0, 0), (1, 0), (0, 1) >$, $c = a * b = 0$, in this case, $a + b \ge 0$ and $a + b \le 1$; and when (a, b) is $(1, 1)$, $c = 1$, in this case, $a + b \ge 2$ and $a + b \le 2$. Above all, Theorem 2 is proved to satisfy all cases of (a, b). □

Theorem 3. *Given two variables, $a, b \in R^+$, the description "$|a - b|$" can be linearly modeled as c, subject to $c \geq a - b$, and $c \geq b - a$, where $c \in R^+$ is a positive real number.*

Proof. There are two cases of $|a - b|$: when $a \geq b$, $c = |a - b| = a - b \geq 0$, $b - a \leq 0$, and when $a < b$, $c = |a - b| = b - a > 0$, $a - b < 0$. Thus, for two cases, Theorem 3 holds. □

To linearly describe this problem, three kinds of variables are added: $z_p = x_i * y_j$, $R_t = |\sum_{i \in Q_t} x_i - r_t * \sum_{i \in Q} x_i|$, and $Load = |\sum_{i \in Q}(x_i * vl_i) - l_{idx} * \sum_{i \in Q} x_i|$. The transformation result of Equation 4.1 is shown in Equation 4.3:

$$\begin{aligned}
&\textbf{Maximize} \quad \omega_1 \sum_{i \in Q}(x_i * ve_i) + \omega_2 \frac{\sum_{p=1}^{|V|*|Q|} z_p}{|V|} + \omega_3(\sum_{i \in Q} x_i - \frac{\sum_{t=1}^{nt} R_t}{nt}) + \omega_4(\sum_{i \in Q} x_i - Load) \\
&\textbf{S.t.} \\
&0 < \sum_{i \in Q} x_i \leq c_{len}, \\
&\sum_{i \in Q} x_i * G(i, j) \geq y_j, for\ j \in V, \\
&x_i + y_j \geq 2z_p, p = i * |V| + j\ for\ i \in Q, j \in V, \\
&x_i + y_j \leq 1 + z_p, p = i * |V| + j\ for\ i \in Q, j \in V, \\
&R_t \geq \sum_{i \in Q_t} x_i - r_t * \sum_{i \in Q} x_i, for\ t \in < 1, ..., nt >, \\
&R_t \geq r_t * \sum_{i \in Q} x_i - \sum_{i \in Q_t} x_i, for\ t \in < 1, ..., nt >, \\
&Load \geq \sum_{i \in Q}(x_i * vl_i) - l_{idx} * \sum_{i \in Q} x_i, \\
&Load \geq l_{idx} * \sum_{i \in Q} x_i - \sum_{i \in Q}(x_i * vl_i).
\end{aligned} \tag{4.3}$$

Then, after gaining the maximum value of the above equation, the optimal objective value for F is gained. This process may be repeated several times with different constraints on the number of selected questions. However, this process would be costly, as proven in Section 5, in which the computation latency can be rapidly increased with the size of the question bank. Thus, efficient algorithms for this problem are explored in the following section.

4.2. DPGA-AS based question selection

In this section, the genetic algorithm is deeply explored to efficiently find effective solutions for the presented problem. Figure 2 shows the overview of the proposed method. The question bank contains the given contents. In the question selection part, the proposed algorithm takes the *ED*, *CR*, *MD*, and *CD* into consideration to optimize the quality of question selection. Different from the traditional GA, a GA with adaptive selection (GA-AS) is presented, which improves the flexibility of the GA and gets better results. Then, to improve the efficiency of the question selection method, dynamic programming (DP) methods are designed and implemented for *ED* and *CR* optimization, and an efficiency optimizer is proposed and combined with GA-AS. The combination is named DPGA-AS.

The purpose is to optimize the objective function F with the constraint of the number of selected questions c_{len}. This objective function can be directly used for the fitness function in the GA. To implement the GA-based method, the meaning of a chromosome is important and should be defined. In this paper, a chromosome is one result of the question selection and is represented as a binary vector.

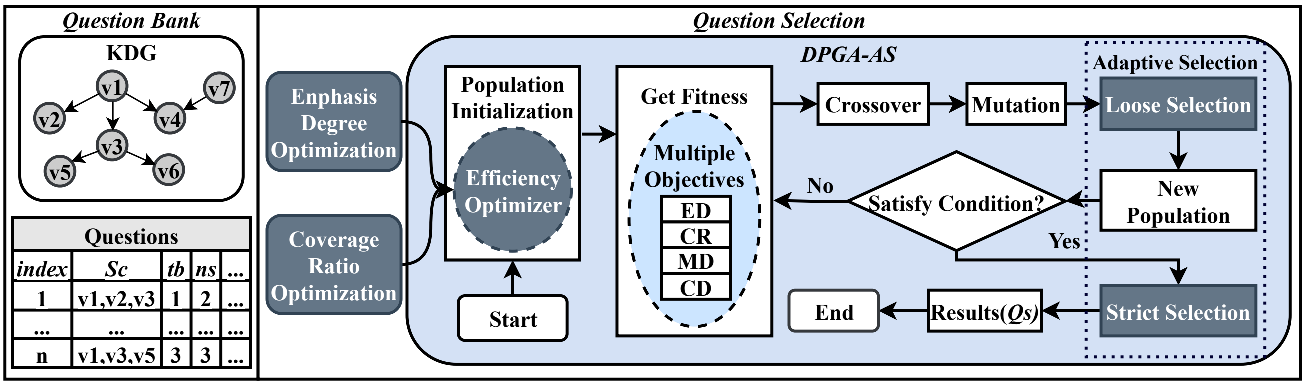


Figure 2. The overview and flow chart of the proposed scheme.

The length of a chromosome is equal to the total number of questions $|Q|$ in the question bank, and the value of each element is 0 or 1.

Index	0	1	2	3	4	5	6	7	8	9
Chromosome	0	1	0	0	1	1	1	0	1	0

Figure 3. An example of a chromosome (result).

If the value is 1, then the related question is selected in this result. An example is displayed in Figure 3. When there are 10 questions in the question bank, then the length of a chromosome is 10. In this figure, the values corresponding to the 1st, 4th, 5th, 6th, and 8th questions are 1; therefore, they are selected.

Figure 2 shows the flow chart of the proposed design. There are mainly 5 steps: population initialization, fitness computation, crossover, mutation, and selection. In this design, population initialization and selection are further explored.

4.2.1. DP based population initialization

In this section, an optimizer is designed in the population initialization to improve the efficiency of the GA-based method. Traditionally, the population is randomly initialized. Inspired by the idea of crossbreeding that the offspring of a hybrid can inherit the advantages of two populations, the initialization method is improved by crossing over the results in single objective optimizations. The details of single objective optimizations are described as follows:

(1) Emphasis Degree Optimization. The problem is to select a Q_s which is the most correlated with the focused knowledge subset PC under the given constraints. This problem is defined as finding a Q_s which has the highest value of $ED \times |Q_s|$ under the constraint of $c_{len} \times l_{idx}$. This problem can be solved by DP, which is important and useful in the optimization area [48].

$$DP_{ik} = \begin{cases} DP_{i-1k}, & vl_i > k, \\ max(DP_{i-1k}, DP_{i-1k-vl_i} + vc_i), & vl_i \le k. \end{cases} \tag{4.4}$$

Equation 4.4 shows the derivation design of the DP method, where DP_{ik} is the maximum sum of the emphasis degree among i questions when c_{bur} is k. In this case, when the question load of the ith question vl_i is higher than k, then the maximum sum is equal to the value for the previous $i-1$ questions. When vl_i is lower than k, then the maximum sum is either the case without the ith question or the case with the ith question.

Algorithm 1 DP based Emphasis Degree Optimization.

```
Input:
    Constraints, $c_{bur} = c_{len} \times l_{idx}$;
    $G = < V, E >, I = < Q, C, T, S, L, Cr >$.
Output:
    A set of selected questions $Q_s$.
1: Let $DP[1..n, 1..c_{bur}]$ be an new table; $n = |Q|$;
2: Let $Q_s$ is a null array;
3: for $i = 1 \to n$ do
4:     for $k = 1 \to c_{bur}$ do
5:         if $vl_i > k$ then
6:             $DP[i, k] \leftarrow DP[i-1, k]$;
7:         end if
8:         if $vl_i \leq k$ then
9:             if $DP[i-1, k] \leq DP[i-1, k - vl_i] + vc_i$ then
10:                $DP[i, k] \leftarrow DP[i-1, k - vl_i] + vc_i$;
11:            end if
12:        end if
13:    end for
14: end for
15: Trace the DP array and get the selected questions $Q_s$;
16: return $Q_s$.
```

Algorithm 1 shows the process of this method. Lines 3-12 get the optimal sum of the ED within the given constraints. After traversing, the optimal sum would be stored in the final element DP_{ik}, where $i = |Q|, k = c_{bur}$. The selected questions are gained by backtracking the questions which constitute the optimal result. Then, Q_s would be gained and returned.

(2) Coverage Ratio Optimization. The purpose of this section is to optimize the CR value under the given constraints. This problem is to find a Q_s that has the highest CR value under the given constraints. This problem can be proven to be an NP-hard problem. Therefore, there are no efficient algorithms that can be designed for this problem to get optimal results.

To handle this problem, a greedy method is proposed. The basic idea is to select the most worthy questions. Here, the worth of a question is defined as a cost value, $\frac{vl_i}{|Sc_i \cap U|}$, where U is a set of uncovered knowledge elements. The size of U would decrease with the execution of the question selection process. This value reflects the cost of a question when this question is selected to increase the CR value. When selecting a question that does not reduce any knowledge element in U, then the cost of this question is infinite. In this case, the question with the low question difficulty vl and a high coverage of uncovered knowledge would have a low cost, and it tends to be selected. Thus, the lower the cost is, the worthier the related question is. As shown in Algorithm 2, lines 1-3 are the initialization, lines 4-12 select questions in order under the constraints, and the selected question is always the one with the smallest cost value (Line 9).

Algorithm 2 Greedy based Coverage Optimization

```
Input:
    Constraints, $c_{len}$, $c_{bur} = c_{len} \times l_{idx}$;
    $G = < V, E >, I = < Q, C, T, S, L, Cr >$.
Output:
    A set of selected questions $Q_s$.
 1: Let $U$ is the set of the uncovered knowledge elements and initialized by $U = C$;
 2: Let $Q_s$ is a null array, $Q^* = Q$;
 3: $j \leftarrow 0$; $k \leftarrow 0$;
 4: while $j \leq c_{len}$ and $k \leq c_{bur}$ do
 5:     Remove the questions from $Q^*$ whose $vl$ is higher than $c_{bur} - k$;
 6:     if No questions left in $Q^*$ or $U$ is null then
 7:         break;
 8:     end if
 9:     $i \leftarrow$ The question index which has smallest $\frac{vl_i}{|Sc_i \cap U|}$;
10:     Add $i$th question to $Q_s$; $U \leftarrow U - Sc_i \cap U$;
11:     $j \leftarrow j + 1$; $k \leftarrow k + vl_i$; $i \leftarrow i + 1$;
12: end while
13: return $Q_s$.
```

The results of the above two methods are used to crossover. The crossover method is the single point method which would be deeply described in Section 4.2.3. It randomly selects the position and performs the crossover. This process would repeat multiple times until the population size reaches the expectation np. As a result, the initial population is optimized in an efficient way. Fewer epochs are needed, thus leading to a reduction of computation latency.

4.2.2. Get Fitness

As shown in Equation 3.7, the fitness calculation is the same as the objective function.

4.2.3. Crossover

Chromosomes are randomly selected to crossover for evolution. The possibility of the crossover pc is defined as the possibility for a chromosome to be selected for crossover. In this paper, the pc is set to 0.9 by default. The method of crossover is a single point crossover. In this process, a cross position is randomly selected, after which two chromosomes crossover based on this position. Figure 4 shows an example, where position 4 is selected for crossover and two offspring are generated. The first offspring is the combination of the first 5 elements in the last original chromosome and the last 5 elements in the first original chromosome. After this step, two chromosomes would generate two offspring. If the fitness values of the new offspring are not the lowest among all chromosomes, then another two chromosomes in the original population will be replaced.

4.2.4. Mutation

Mutations can improve the diversity of the population, and increase fitness. A possibility of the mutation pm is defined as the possibility for a chromosome to mutate, and it controls whether a chromosome can mutate. In this paper, the pm is set as 0.4 by default. In the mutation process, if a chromosome can mutate, then a mutation position is randomly selected, and the value in that position would be changed from 0 to 1 or 1 to 0.

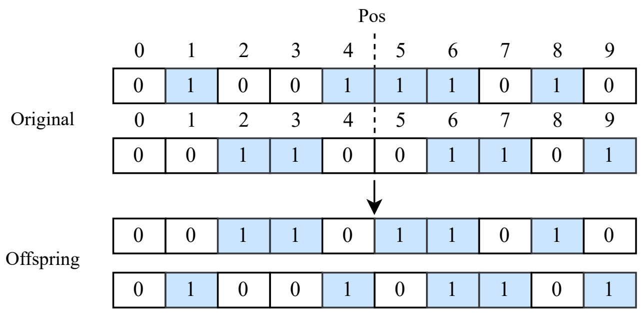


Figure 4. An example of crossover.

4.2.5. Adaptive Selection

Selection is a critical part of the GA. It is used to select chromosomes that will be left for the next population. Generally, chromosomes that do not fit the conditions (cannot satisfy the constraints) would be evicted in the selection phase. However, some good genes may be removed too early in this process, and the result may be not good enough. Different from the traditional GA design, which uses the same selection method for the whole process, this paper designs the selection method in two phases, as shown in Figure 2, loose selection and strict selection, which are adaptively applied in the process of the GA to better get ideal results. When the optimization process is not near the termination, the process executes the loose selection. In this case, it only eliminates the ones whose length is 0 or higher than a loose threshold of c_{len}. The threshold is $c_{len} + 1/3c_{len}$ in this paper. When the process nears the termination condition, it would use strict selection. The chromosomes with the number of selected questions exceeding c_{len} would be removed. In this case, the loose selection is to prevent valuable questions from being removed, and the strict selection is to satisfy the constraint of the number of selected questions. After that, a new population forms.

The termination condition is used to end this process. In this paper, the condition is defined as the number of iterations or epoch thresholds. It is manually given which is analyzed in Section 5. When the number of epochs reaches the given threshold, then a result with the highest fitness value will be selected. If the iteration is not enough, then, the process will proceed to the fitness calculation step.

5. Evaluation

In this section, a series of experiments are conducted to evaluate the quality and efficiency of the proposed schemes. Compared with baseline methods, the quality of the selected questions and the computation latency are evaluated.

5.1. Setup

The experiments are implemented on a commercial computer, which is equipped with 3.6GHz processors and 16 GB memory. All models are implemented in Python 3.8. The Gurobi optimizer [49] is used for the implementation of linear programming. The question bank data are collected and labeled based on calculus problems in a textbook [50], which is widely used for college calculus teaching. The labels include S_c, tb and ns. vl and ve are computed automatically based on S_c, tb, and ns. They are illustrated in Table 1 in Section 3.2.

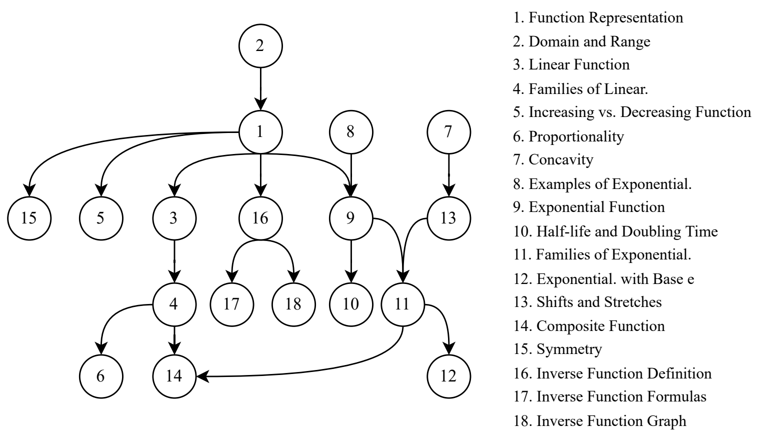

1. Function Representation
2. Domain and Range
3. Linear Function
4. Families of Linear.
5. Increasing vs. Decreasing Function
6. Proportionality
7. Concavity
8. Examples of Exponential.
9. Exponential Function
10. Half-life and Doubling Time
11. Families of Exponential.
12. Exponential. with Base e
13. Shifts and Stretches
14. Composite Function
15. Symmetry
16. Inverse Function Definition
17. Inverse Function Formulas
18. Inverse Function Graph

Figure 5. The knowledge dependency graph (KDG) of the real question bank.

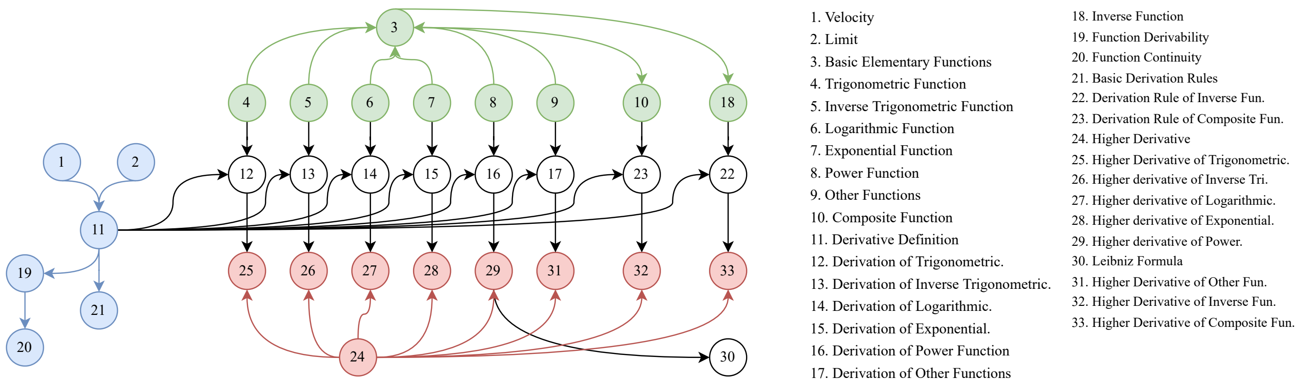

1. Velocity
2. Limit
3. Basic Elementary Functions
4. Trigonometric Function
5. Inverse Trigonometric Function
6. Logarithmic Function
7. Exponential Function
8. Power Function
9. Other Functions
10. Composite Function
11. Derivative Definition
12. Derivation of Trigonometric.
13. Derivation of Inverse Trigonometric.
14. Derivation of Logarithmic.
15. Derivation of Exponential.
16. Derivation of Power Function
17. Derivation of Other Functions
18. Inverse Function
19. Function Derivability
20. Function Continuity
21. Basic Derivation Rules
22. Derivation Rule of Inverse Fun.
23. Derivation Rule of Composite Fun.
24. Higher Derivative
25. Higher Derivative of Trigonometric.
26. Higher derivative of Inverse Tri.
27. Higher derivative of Logarithmic.
28. Higher derivative of Exponential.
29. Higher derivative of Power.
30. Leibniz Formula
31. Higher Derivative of Other Fun.
32. Higher Derivative of Inverse Fun.
33. Higher Derivative of Composite Fun.

Figure 6. The knowledge dependency graph (KDG) of the large question bank.

The experimental question bank consists of two sets, namely a small question bank, and a large question bank, which are named SmallSize and LargeSize in the following descriptions. To implement this work, the KDG and all attributes of the question bank are extracted and analyzed by a group of graduate students within two months. For the small question bank, 18 knowledge elements are extracted and modeled as a KDG, as shown in Figure 5. Based on this KDG, 206 real questions are labeled. The cognitive taxonomy *tb* is labeled using Bloom's taxonomy based on the rules in [51]. The large question bank is set up based on 33 knowledge elements extracted from the chapters about derivation, including the definitions of derivation and continuity. The KDG for the large question bank is shown in Figure 6. An edge in the KDG represents the dependency between two knowledge elements. For example, the function continuity is established upon the definition of the function derivability. The dependency is represented by an edge from node 19, which represents the function derivability, to node 20, which represents the function continuity in the KDG, as shown in Figure 6. Then, 5000 questions are simulated based on this KDG. For simplicity, the number of steps *ns* is set as 1 for all simulated questions. The question load distribution of the large question bank is similar to the small one.

In the experiments, we evaluate the effectiveness of multiple algorithms of question selection by comparing the fitness value to the quality of the selected test similar to other works [6]. In the following, *Fitness* represents the fitness value and is calculated by Equation 3.7. This value is combined by the *ED*, *CR*, *MD*, and *CD* as defined by Equation 3.4, 3.3, 3.5 and 3.6, respectively. The efficiency is evaluated by the total computation latency.

The compared solutions and the proposed schemes to be evaluated in our experiments are listed in the following:

- *RSF* is the baseline method that randomly selects questions to form the test paper, as used in [1]. The best version of the *RSF* is used for comparison.
- *GA* is the question selection method based on the genetic algorithm, which initializes the population using the randomization method; the selection method is the same in the whole process as used in other works [4,5,7,9].
- *ACO* is the question selection method based on ant colony optimization, which is applied in [15].
- *PSO* is the question selection method based on particle swarm optimization, which is applied in [16].
- *LP* is the method based on the designed techniques of linear programming, which can obtain optimal results under the given constraints.
- *GA-AS* is the proposed method that uses loose selection and strict selection.
- ***DPGA-AS*** is the optimized method that updates both population initialization and selection as presented in Section 4.2.

Table 2. The Settings of ω for Four Cases.

	ω_{ED}	ω_{CR}	ω_{MD}	ω_{CD}
Case 1	0.25	0.25	0.25	0.25
Case 2	0.4	0.2	0.2	0.2
Case 3	0.4	0.1	0.1	0.4
Case 4	0.1	0.4	0.25	0.25

The computation model for the quality of the question selection can be tuned by some weights (α,

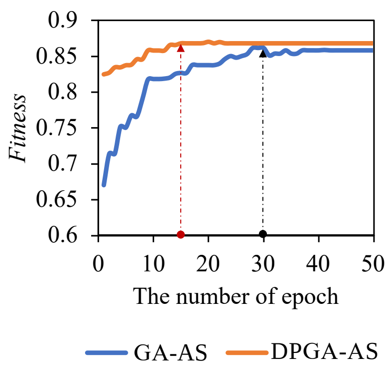
β, ω). The weights reflect the preference for the considered factors in the models. In this work, β is automatically gained based on $\sigma = 1/2$ in Section 3.2. Other weights are averagely distributed among all the related factors by default. $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$, $\omega_1 = \omega_2 = \omega_3 = \omega_4 = 1/4$. We leave the deep exploration of the design space constructed by different weight settings to future work. In this paper, to evaluate the influence of the setting of ω, four cases of the test are evaluated. The detailed settings are shown in Table 2. Case 1 is the default setting. Cases 2 and 3 are for the situations of classroom quizzes that focus on some emphasized knowledge elements. Case 4 is for the conclusion tests, which focus on the coverage of all learned knowledge elements.

The input data of the experiments include the key or emphasized knowledge set $KSet$ for a test, the load index l_{idx} representing the expected average degree of the hardness of questions, and the constraint of the total number of questions in a test c_{len} by the instructors. The parameters used by the small and large question banks are shown in Table 3.

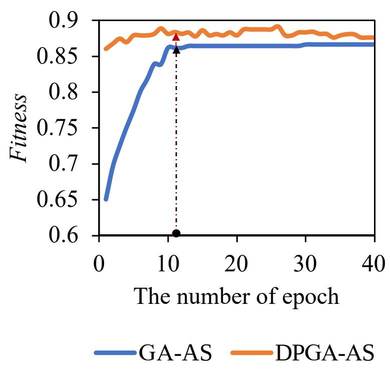
Table 3. An example of input.

	$KSet$	l_{idx}	c_{len}
SmallSize	5,13	0.5	10
LargeSize	19,21	0.8	10

For the GA-based models, the setting of the following parameters impacts the quality of the models: the size of population np, the possibility of crossover pc, the possibility of mutation pm, and the number of epochs. In this paper, the default setting of the np, pc, and pm is $np = 200$, $pc = 0.9$, $pm = 0.4$, which are set based on many experiments. For the number of epochs or iterations, enough iterations can produce high-quality results, while reducing epochs would improve the efficiency of the selection algorithm because more epochs always lead to a higher latency. To select the proper number of epochs, the change of *Fitness* with the increasing number of epochs is analyzed to find the turning point of the curves in Figure 7. As shown in Figure 7, the curves of *Fitness* vs. the number of epochs are drawn for two question banks. In Figure 7(a), the *Fitness* of *GA-AS* tends to be stable after 15 epochs and that of *DPGA-AS* becomes stable after 30. Thus, for the small question bank, 15 and 30 are selected for *GA-AS* and *DPGA-AS*, respectively. With the same logic, for the large question bank, 12 is chosen for both *GA-AS* and *DPGA-AS*, as shown in Figure 7(b).



(**a**)SmallSize



(**b**)LargeSize

Figure 7. The selection of the number of epochs.

5.2. *Test quality*

Table 5 shows the results in terms of the *ED*, *CR*, *MD*, *CD*, and *Fitness* of the algorithms for all evaluated methods of two question banks in different cases. From the results, several conclusions can be drawn. First, the quality of the *RSF* is not stable. There is one instance that shows a relatively high fitness value. However, the performance of the instance is inferior in almost all evaluation metrics. This indicates that the *RSF* is not practical to achieve multiple optimization goals.

Table 4. Test quality for evaluated schemes in the small dataset.

	Case 1					Case 2					Case 3					Case 4				
	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**
RSF	0.37	0.67	0.76	0.97	0.69	0.37	0.67	0.76	0.97	0.63	0.37	0.67	0.76	0.97	0.67	0.37	0.67	0.76	0.97	0.73
GA	0.47	0.83	0.93	0.98	0.80	0.52	0.78	0.93	0.98	0.75	0.46	0.56	0.81	0.99	0.73	0.37	0.89	0.93	0.94	0.86
ACO	0.50	0.72	1	0.98	0.80	0.59	0.78	0.89	0.98	0.77	0.69	0.61	0.78	0.98	0.81	0.31	0.89	1	0.91	0.86
PSO	0.52	0.78	1	0.99	0.82	0.57	0.78	1	0.99	0.78	0.64	0.67	1	0.98	0.82	0.42	0.94	0.93	0.97	0.89
LP	0.52	1	1	0.97	**0.87**	0.67	0.89	1	0.91	**0.83**	0.74	0.67	0.65	0.99	**0.86**	0.47	1	1	1	**0.95**
GA-AS	0.52	0.94	1	0.98	0.86	0.67	0.89	1	0.86	0.82	0.74	0.67	0.66	0.97	0.85	0.42	1	0.93	0.99	0.92
DPGA-AS	0.52	0.94	1	0.99	**0.87**	0.64	0.83	1	0.95	0.82	0.74	0.67	0.66	0.98	**0.86**	0.47	1	1	1	**0.95**

Table 5. Test quality for evaluated schemes in the large dataset.

	Case 1					Case 2					Case 3					Case 4				
	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**	ED	CR	MD	CD	**Fitness**
RSF	0.18	0.67	0.68	0.92	0.61	0.18	0.67	0.68	0.92	0.53	0.18	0.67	0.68	0.92	0.55	0.18	0.67	0.68	0.92	0.69
GA	0.44	0.82	1	0.93	0.79	0.43	0.82	1	0.93	0.72	0.54	0.81	1	0.97	0.75	0.44	0.82	1	0.93	0.86
ACO	0.75	0.82	0.89	0.96	0.85	0.83	0.70	0.93	0.97	0.85	0.89	0.70	0.86	0.94	0.89	0.61	0.85	0.95	0.99	0.89
PSO	0.54	0.82	1	0.98	0.83	0.63	0.82	1	0.99	0.82	0.87	0.67	0.94	0.94	0.88	0.54	0.85	1	0.98	0.88
LP	0.81	0.82	1	0.93	**0.89**	0.92	0.70	1	0.91	**0.89**	0.92	0.70	0.70	0.99	**0.95**	0.41	0.94	1	0.99	**0.91**
GA-AS	0.71	0.82	1	0.97	0.87	0.85	0.70	1	0.98	0.87	0.85	0.70	0.78	0.99	0.92	0.50	0.88	1	0.99	0.90
DPGA-AS	0.71	0.81	1	0.96	0.87	0.92	0.70	0.78	0.98	0.86	0.92	0.70	0.78	0.98	**0.95**	0.51	0.91	1	1	**0.91**

The *LP* can always achieve the highest fitness values in all cases. However, the computation efficiency of this method is much lower than other methods, as shown in Section 5.3. Compared with other methods, the performance of the proposed algorithms, *GA-AS* and *DPGA-AS*, are near the best in all cases; in some cases, the *DPGA-AS* can achieve the best performance. For example, in Case 1, the *Fitness* of *GA-AS* and *DPGA-AS* are 24% and 27% higher compared with *RSF* in two question banks. The *Fitness* values of *GA-AS* and *DPGA-AS* in two question banks are only 0.5%-2% lower than that of the *LP*. In Case 3, the *Fitness* of *DPGA-AS* is the highest, which is 73%, 27%, 7%, and 8% higher than *Random*, *GA*, *ACO*, and *PSO*, respectively, in the large question bank. Similar results can be gained in Case 4. In other cases, the *DPGA-AS* algorithm achieves satisfying *Fitness* values, which are close to the *Fitness* value of *GA-AS* with a little improvement. Thus, the proposed methods can obtain satisfying results for different settings of ω, which show that they are suitable for different teaching situations. These results reflect the high effectiveness of the proposed algorithms.

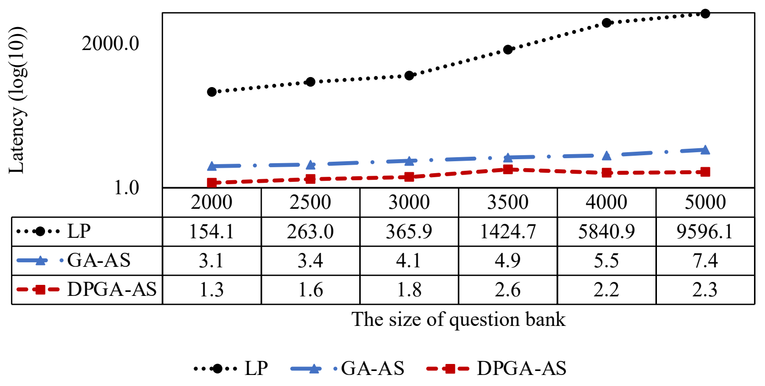
5.3. *Computation performance*

To evaluate the efficiency of the proposed algorithms, the computation latency is recorded for all experiments. Table 6 shows the results of the *LP*, *GA-AS*, and *DPGA-AS* in all cases for two question banks. Additionally, to show the efficiency of the proposed method *DPGA-AS*, the relative speed-up ratio is presented in this table.

Table 6. Computation latency.

	Case 1				Case 2				Case 3				Case 4			
	SmallSize		LargeSize		SmallSize		LargeSize		SmallSize		LargeSize		SmallSize		LargeSize	
	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up	Lat.(s)	Spd.up
LP	0.6	**1x**	9596.1	**4171x**	0.7	**1x**	9660.8	**4024x**	0.6	**1x**	9284.3	**4420x**	0.5	**2x**	9432.2	**4491x**
GA-AS	1.1	**3x**	7.4	**2x**	1.1	**3x**	7.7	**3x**	1.2	**3x**	6.8	**2x**	1	**4x**	7.0	**2x**
DPGA-AS	0.3		2.3		0.3		2.4		0.2		2.1		0.2		2.1	

Spd.up is the speed up by DPGA-AS compared with LP and GA-AS, respectively.



	2000	2500	3000	3500	4000	5000
LP	154.1	263.0	365.9	1424.7	5840.9	9596.1
GA-AS	3.1	3.4	4.1	4.9	5.5	7.4
DPGA-AS	1.3	1.6	1.8	2.6	2.2	2.3

Figure 8. The change of latency with the increasing of the size of question bank (The data are from simulated 5000-size question bank).

For all cases, *DPGA-AS* is much faster than the other two methods. In Case 1, the latency of *DPGA-AS* is reduced by 78.2% and 68.9% in two question banks, respectively, as compared with *GA-AS*. Moreover, similar results can be gained in other cases. The reasons why *DPGA-AS* can speedup are as follows: as to small question bank, *DPGA-AS* can help the GA converge faster. Thus, with fewer epochs, the latency can be significantly decreased. As to the large question bank, the number of epochs for two question banks is the same; however, the number of epochs can be set as fewer for *DPGA-AS*, where the latency is also noticeably decreased. This is because the size of population np reduces more rapidly with the increasing iteration in *DPGA-AS*. Therefore, with a smaller population, the latency can also be reduced.

In Case 1, *DPGA-AS* is 1x and 4171x faster in the two question banks, respectively, as compared with *LP*. Moreover, similar results can be gained in other cases. While the *LP* can always get the optimal *Fitness* value, the cost is exponentially increased with the increase in the size of the question bank, as shown in Figure 8. The *Fitness* values of *GA-AS* and *DPGA-AS* are close to that of the *LP* with 1-2% reduction; however, the latency is much slower than the *LP*, especially with a large question bank. It shows that questions can be selected with both high quality and high efficiency using the proposed *DPGA-AS* algorithm with good scalability.

5.4. Parameter analysis

In this section, some critical parameters and their impacts on the fitness value and computation latency are analyzed and discussed. As to the GA-based methods, the size of population np, the possibility of crossover pc, the possibility of mutation pm, and the number of epochs would impact the quality of models.

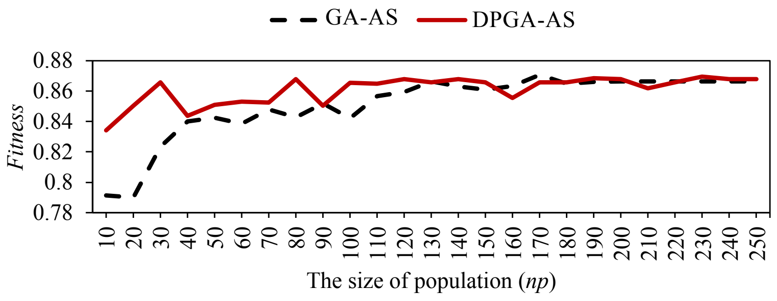


Figure 9. The change of *Fitness* of two proposed schemes for the small question bank in the case one with the increasing of *np* when *pc* = 0.9, *pm* = 0.4.

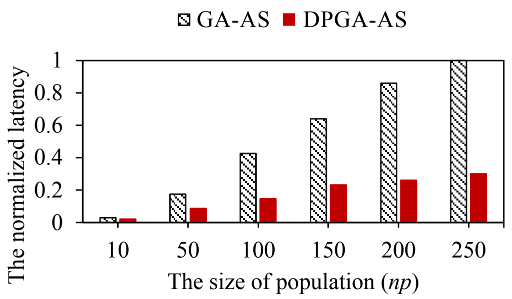


Figure 10. The normalized latency of two proposed schemes for the small question bank with the increasing of *np* when *pc* = 0.9, *pm* = 0.4.

Figure 9 presents the change of the *Fitness* with the increase in the size of the population *np*. Here, we only show the result when *pc* = 0.9 and *pm* = 0.4, the trends are similar in most situations. In this figure, the overview trend is that the quality grows as the size of the population *np* increases. The quality of tests generated by the *GA-AS* algorithm is worse than *DPGA-AS* until *np* = 150; after *np* = 150, its quality value turns out to be stable. As to *DPGA-AS*, the trend is unstable before *np* = 100, but the quality is always better than *GA-AS*; after that, the trend tends to be stable. The *Fitness* is almost unchanged after *np* = 180 for both *GA-AS* and *DPGA-AS* algorithms. Therefore, *np* = 200 is set as a reasonable population size in the experiments. Figure 10 shows the normalized latency along an increase of the *np*: the latency of *GA-AS* grows significantly as *np* increases, while the computation latency of *DPGA-AS* grows slowly. When the population grows, the gap of latency between *GA-AS* and *DPGA-AS* expands significantly. It concludes that the quality of the question selection model improves with an increasing population with a cost of computation latency. Therefore, the population parameter *np* needs to be carefully selected to balance the requirements of effectiveness and efficiency.

The change of the *Fitness* of two schemes with various settings of the possibility of crossover *pc* and the possibility of mutation *pm* is shown in Figure 11(a) and Figure 11(b), respectively. The red line represents the Fitness value across various *pc* with *pm* set as 0.4. Several observations can be gained from Figure 11(a) and Figure 11(b). First, in most cases, the quality of generated tests improves when the *pc* increases. The *Fitness* value of the *GA-AS* algorithm tends to be stable after *pc* = 0.7 for *pm* ≤ 0.8; the *Fitness* value of the *DPGA-GA* algorithm tends to be stable after *pc* = 0.8 for *pm* ≤ 0.6. Second, when the *pm* is too large, the quality drops. In Figure 11(a), the quality value drops when

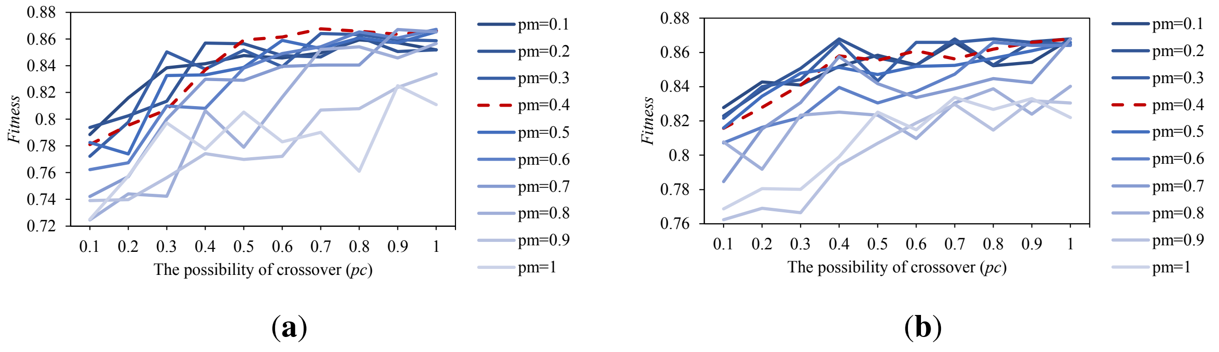


Figure 11. The change of *Fitness* with the increasing of *pc* and *pm* when $np = 200$.(a) *GA-AS*; (b) *DPGA-AS*.

$pm \geq 0.8$. And similarly in Figure 11(b), the quality drops when $pm \geq 0.7$. Therefore, we choose the default setting as $pc = 0.9$, $pm = 0.4$, which is in a reasonable range.

The settings of the *pc* and *pm* parameters also have impacts on the computation latency. Figure 12(a) shows the change of latency with the increasing of *pc* when $pm = 0.4$ and $np = 200$. The experimental data show that the latency of the *GA-AS* algorithm is sensitive to the value of *pc*. The latency rises faster for the *GA-AS* algorithm as compared to *DPGA-AS*. In other words, the parameter *pc* has to be carefully chosen for the *GA-AS* algorithm, while it is not a major factor that impacts the efficiency of the *DPGA-AS* algorithm.

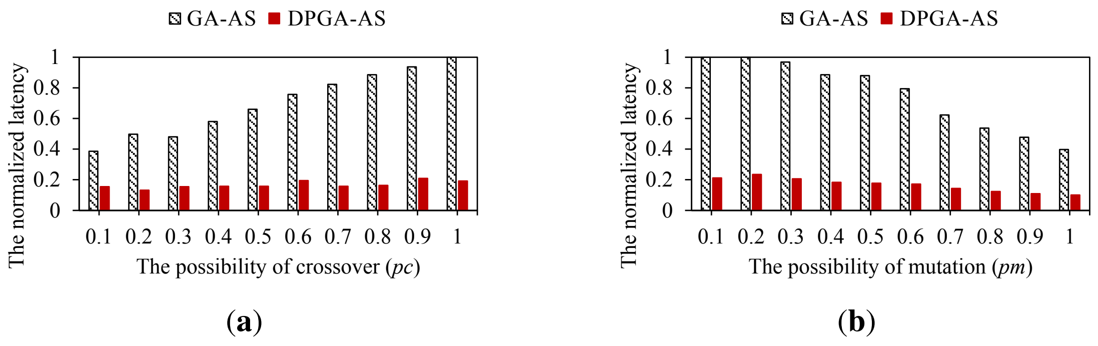


Figure 12. The normalized latency of two proposed schemes for the small question bank (a) with the increasing of *pc* when $pm = 0.4, np = 200$; (b) with the increasing of *pm* when $pc = 0.4, np = 200$.

As a result, we can mainly focus on the quality improvement when choosing the *pc* for the *DPGA-AS* method. Figure 12(b) shows the change of latency with an increase in the *pm* when $pc = 0.9$ and $np = 200$. The overall trend for both *GA-AS* and *DPGA-AS* algorithms is that the latency drops as the *pm* grows. The decline of latency for the *DPGA-AS* method is slower than that of *GA-AS*. For the *GA-AS* algorithm, the latency rapidly drops for the $pm > 0.6$. As to the *DPGA-AS* algorithm, the latency gradually decreases. As a result, the impact of the *pm* on efficiency is much smaller than that on the effectiveness of the *DPGA-AS* algorithm. The experimental results indicate that the *DPGA-AS* is stable and easy to tune when both the effectiveness and the efficiency are considered.

6. Conclusion and future work

This paper has addressed several problems, including quantifying the quality of tests with multiple objectives, designing the technique of linear programming to gain the optimal results of this problem, improving the test quality based on GA-based question selection, and enhancing the efficiency of the GA-based question selection technique. First, by analyzing real requirements from educators, the quality of tests is quantified based on four aspects: the emphasis degree on key knowledge elements, the coverage of the to-be-tested knowledge set, the matching degree to the expected question taxonomies, and the closeness to the expected difficulty of generated tests. Then, the technique of linear programming is designed for this objective function to get the optimal test quality. However, this method is not efficient enough. Thus, we propose a dynamic programming guided GA method, DPGA-AS, to optimize the quality of tests with high efficiency. Some single-objective optimization problems are explored and analyzed based on a dynamic programming method to improve the population initialization of GA, and an adaptive selection method is designed to improve the performance of the GA. In our experiments, a real question bank is analyzed and used for the evaluation. Compared with other methods, the proposed methods can get the tests with high quality, high efficiency, and good scalability. The quality is near optimal compared with the technique of linear programming. The proposed method, DPGA-AS, is over 4000× faster than the technique of linear programming for a 5000-size question bank.

This work exhibits certain limitations that can be addressed in the future. The application of a weight sum format to transform multiple objective problems into a single objective format introduces certain constraints. The manual setting of weights for each objective imposes a burden on practical applications. Consequently, our future work will explore approaches for automatically adjusting the weights of fitness functions, thereby taking multiple objectives into consideration. Moreover, this format may compromise optimal results when the result space is non-convex. Therefore, alternative methods for solving multiple objective problems will be thoroughly investigated. Additionally, this paper tries to explore the effective computation models and methods that capture the real requirements of educators to generate high-quality tests automatically. We believe that the computation model and methods are open to new ideas and improvements when we acquire new understandings of the teaching and learning process.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is partially supported by Shanghai Science and Technology Commission Project 20511101600, NSFC 62372183, NSFC 62372182, and NSFC 61972154.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. K. Naik, S. Sule, S. Jadhav, S. Pandey, Automatic question paper generation system using randomization algorithm, *Int. J. Eng. Tech. Res. (IJETR)*, **2** (2014), 192–194.
2. W. J. Linden, *Linear models for optimal test design*, Springer, (2005). https://doi.org/10.1007/0-387-29054-0_3
3. W. J. Linden, Review of the shadow-test approach to adaptive testing, *Behaviormetrika*, (2021), 1–22. https://doi.org/10.1007/s41237-021-00150-y
4. K. Zhang, L. Zhu, Application of improved genetic algorithm in automatic test paper generation, in *2015 Chinese Automation Congress (CAC)*, (2015), 495–499. https://doi.org/10.1109/cac.2015.7382551
5. T. N. T. A. Rahim, Z. A. Aziz, R. H. A. Rauf, N. Shamsudin, Automated exam question generator using genetic algorithm, in *2017 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, (2017), 12–17. https://doi.org/10.1109/ic3e.2017.8409231
6. T. Nguyen, T. Bui, H. Fujita, T.-P. Hong, H. D. Loc, V. Snasel, et al., Multiple-objective optimization applied in extracting multiple-choice tests, *Eng. Appl. Artif. Intell.*, **105** (2021), 104439. https://doi.org/10.1016/j.engappai.2021.104439
7. Z. Wu, T. He, C. Mao, C. Huang, Exam paper generation based on performance prediction of student group, *Inform. Sci.*, **532** (2020), 72–90. https://doi.org/10.1016/j.ins.2020.04.043
8. M. Aktaş, Z. Yetgin, F. Kılıç, Ö. Sünbül, Automated test design using swarm and evolutionary intelligence algorithms, *Expert Syst.*, **39** (2022). https://doi.org/10.1111/exsy.12918
9. N. H. I. Teo, N. A. Bakar, M. R. A. Rashid, Representing examination question knowledge into genetic algorithm, in *2014 IEEE Global Engineering Education Conference (EDUCON)*, (2014), 900–904. https://doi.org/10.1109/educon.2014.6826203
10. Z. Jia, C. Zhang, H. Fang, The research and application of general item bank automatic test paper generation based on improved genetic algorithms, in *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, **1** (2011), 14–18. https://doi.org/10.1109/ccieng.2011.6007945
11. M. Yildirim, A genetic algorithm for generating test from a question bank, *Computer Appl. Eng. Educ.*, **18** (2010), 298–305. https://doi.org/10.1002/cae.20260
12. M. Shao, W. Li, J. Du, The research and implementation of technology of generating test paper based on genetic algorithm, in *Intelligence Computation and Evolutionary Computation: Results of 2012 International Conference of Intelligence Computation and Evolutionary Computation (ICEC)*, (2013), 657–663. https://doi.org/10.1109/mines.2012.232
13. L. Han, X. Li, The analysis of exam paper component based on genetic algorithm, in *2014 Fourth International Conference on Communication Systems and Network Technologies*, (2014), 561–564. https://doi.org/10.1109/csnt.2014.118
14. Y. Zhang, J. Zhang, P. Wang, Research and implementation of intelligent test paper composition based on genetic algorithm, in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, (2018), 552–556. https://doi.org/10.1109/itme.2018.00128

15. D. Liu, J. Wang, L. Zheng, Automatic test paper generation based on ant colony algorithm, *J. Softw.*, **8** (2013), 2600–2606. https://doi.org/10.4304/jsw.8.10.2600-2606
16. T. Nguyen, T. Bui, B. Vo, Multi-swarm single-objective particle swarm optimization to extract multiple-choice tests, *Vietnam J. Computer Sci.*, **6** (2019), 147–161. https://doi.org/10.1142/s219688881950009x
17. K. O. Jones, J. Harland, J. M. Reid, R. Bartlett, Relationship between examination questions and bloom's taxonomy, in *2009 39th IEEE frontiers in education conference*, (2009), 1–6. https://doi.org/10.1109/fie.2009.5350598
18. S. Bloxham, P. Boyd, Developing effective assessment in higher education: A practical guide, *McGraw-Hill Education (UK)*, (2007).
19. N. Utaberta, B. Hassanpour, Aligning assessment with learning outcomes, *Procedia-Social Behav. Sci.*, **60** (2012), 228–235. https://doi.org/10.1016/j.sbspro.2012.09.372
20. A. Smith, S. L.-Munk, A. Shelton, B. Mott, E. Wiebe, J. Lester, A multimodal assessment framework for integrating student writing and drawing in elementary science learning, *IEEE Transact. Learn. Technol.*, **12** (2018), 3–15. https://doi.org/10.1109/tlt.2018.2799871
21. A. Alammary, LOsMonitor: A machine learning tool for analyzing and monitoring cognitive levels of assessment questions, *IEEE Transact. Learn. Technol.*, **14** (2021), 64–652. https://doi.org/10.1109/tlt.2021.3116952
22. A. J. Swart, Evaluation of final examination papers in engineering: A case study using bloom's taxonomy, *IEEE Transact. Educ.*, **53** (2009), 257–264. https://doi.org/10.1109/te.2009.2014221
23. B. S. Bloom, M. D. Englehard, Committee of College and University Examiners, *Taxonomy Educ. Object.*, Longmans, **2** (1964).
24. L. W. Anderson, D. R. Krathwohl, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*, Addison Wesley Longman, (2001).
25. L. Smith, J. King, A dynamic systems approach to wait time in the second language classroom, *System*, **68** (2017), 1–14. https://doi.org/10.1016/j.system.2017.05.005
26. M. Riojas, S. Lysecky, J. Rozenblit, Educational technologies for precollege engineering education, *IEEE Transact. Learn. Technol.*, **5** (2011), 20–37. https://doi.org/10.1109/tlt.2011.16
27. F. K. Gangar, H. G. Gori, A. Dalvi, Automatic question paper generator system, *Int. J. Computer Appl.*, **166** (2017), 42–47. https://doi.org/10.5120/ijca2017914138
28. V. S. Rao, V. C. Sai, S. S. Sandeep, MV Jayadeep, Automated exam paper process based on schedule and authenticity, in *International Conference of Advance Research & Innovation(ICARI)*, (2020). https://doi.org/10.2139/ssrn.3643880
29. M. S. R. Chim, G. V. Kale, Automatic question paper generation using parametric randomization, *J. Gujarat Res. Soc.*, **21** (2019), 444-451.
30. S. A. El-Rahman, A. H. Zolait, Automated test paper generation using utility based agent and shuffling algorithm, *Int. J. Web-based Learn. Teach. Technol. (IJWLTT)*, **14** (2019), 69–83. https://doi.org/10.4018/ijwltt.2019010105
31. J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, (1992).

32. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Transact. Syste. Man Cybernet.*, **26** (1996), 29–41. https://doi.org/10.1109/3477.484436

33. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95-international conference on neural networks*, **4** (1995), 1942-1948.

34. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

35. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028

36. B. Abdollahzadeh, F. S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems, *Comput. Indust. Eng.*, **158** (2021), 107408. https://doi.org/10.1016/j.cie.2021.107408

37. X. Bao, H. Jia, C. Lang, A novel hybrid harris hawks optimization for color image multilevel thresholding segmentation, *IEEE Access*, **7** (2019), 76529–76546. https://doi.org/10.1109/access.2019.2921545

38. Y. Xiao, Y. Guo, H. Cui, Y. Wang, J. Li, Y. Zhang, IHAOAVOA: An improved hybrid aquila optimizer and african vultures optimization algorithm for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 10963–11017. https://doi.org/10.3934/mbe.2022512

39. J. Liu, Y. Wang, N. Fan, S. Wei, W. Tong, A convergence-diversity balanced fitness evaluation mechanism for decomposition-based many-objective optimization algorithm, *Integr. Computer-Aided Eng.*, **26** (2019), 159–184. https://doi.org/10.3233/ica-180594

40. J. Liu, Y. Wang, Y. Cheung, A $c\alpha$-dominance-based solution estimation evolutionary algorithm for many-objective optimization, *Knowledge-based Syst.*, **248** (2022), 108738. https://doi.org/10.1016/j.knosys.2022.108738

41. W. Li, H. Pu, P. Schonfeld, J. Yang, H. Zhang, L. Wang, et al., Mountain railway alignment optimization with bidirectional distance transform and genetic algorithm, *Computer-Aided Civil Infrastruct. Eng.*, **32** (2017), 691–709. https://doi.org/10.1111/mice.12280

42. M. Wei, S. Zhang, T. Liu, B. Sun, The adjusted passenger transportation efficiency of nine airports in China with consideration of the impact of high-speed rail network development: A two-step DEA-OLS method, *J. Air Transport Manag.*, **109** (2023), 102395. https://doi.org/10.1016/j.jairtraman.2023.102395

43. J. L. Gordon, Creating knowledge maps by exploiting dependent relationships, in *Appl. Innovat. Intell. Syst. VII*, (2000), 64–78. https://doi.org/10.1007/978-1-4471-0465-0_5

44. N. Henze, W. Nejdl, Logically characterizing adaptive educational hypermedia systems, in *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, (2003), 20–24. https://doi.org/10.1080/13614560410001728128

45. M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proceed. IEEE*, **104** (2015), 11–33. https://doi.org/10.1109/jproc.2015.2483592

46. Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, et al., Ekt: Exercise-aware knowledge tracing for student performance prediction, *IEEE Transact. Knowledge Data Eng.*, **33** (2019), 100–115. https://doi.org/10.1109/tkde.2019.2924374

47. J. D. L. Torre, Dina model and parameter estimation: A didactic, *J. Educ. Behav. Statist.*, **34** (2009), 115–130. https://doi.org/10.3102/1076998607309474

48. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms*, MIT press, (2009).

49. Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, (2023).

50. D. H. Hallett, A. M. Gleason, W. G. McCallum, *Calculus: Single and multivariable*, John Wiley and Sons Inc, (1998).

51. D. C. Webb, Bloom's Taxonomy in Mathematics Education, *Encyclopedia of Mathematics Education*, Springer Netherlands, (2014), 63–68. https://doi.org/10.1007/978-94-007-4978-8_17