Memetic Comp. (2012) 4:33–47
DOI 10.1007/s12293-012-0077-z

REGULAR RESEARCH PAPER

Divide-and-conquer memetic algorithm for online multi-objective test paper generation

Minh Luan Nguyen · Siu Cheung Hui · Alvis C. M. Fong

Received: 6 September 2011 / Accepted: 2 February 2012 / Published online: 15 February 2012

Abstract Online test paper generation (Online-TPG) generates a test paper automatically online according to user specification based on multiple assessment criteria, and the generated test paper can then be attempted online by user. Online-TPG is challenging as it is a multi-objective optimization problem that is NP-hard, and it is also required to satisfy the online generation requirement. In this paper, we propose an efficient multi-objective optimization approach based on the divide-and-conquer memetic algorithm (DAC-MA) for Online-TPG. Instead of solving the multi-objective constraints simultaneously, the set of constraints is divided into two subsets of relevant constraints, which can then be solved separately and effectively by evolutionary computation and local search of DAC-MA. The empirical performance results have shown that the proposed approach has outperformed other TPG techniques in terms of runtime efficiency and paper quality.

Keywords Memetic algorithms · Constraint satisfaction · Multi-objective optimization · Dimensionality reduction · Online test paper generation

M. L. Nguyen · S. C. Hui
School of Computer Engineering,
Nanyang Technological University,
Singapore 639798, Singapore
e-mail: NGUY0093@ntu.edu.sg

S. C. Hui
e-mail: asschui@ntu.edu.sg

A. C. M. Fong (✉)
School of Computing and Math Sciences,
Auckland University of Technology,
Auckland, New Zealand
e-mail: acmfong@gmail.com

1 Introduction

Test paper generation (TPG) generates test papers automatically based on multiple assessment criteria. It aims to find an optimal subset of questions from a question database to form a test paper according to user specification on total time, topic distribution, difficulty degree, discrimination degree, etc. The generated test paper will then be used for testing purpose as in traditional pen-and-pencil tests. Currently, many techniques such as dynamic programming [15], tabu search [16,18], biologically inspired algorithms [17,25,34] and swarm optimization [13,14,38] have been proposed in the research community for automatic TPG. However, these techniques generally require long runtime for generating good quality test papers.

Online test paper generation (Online-TPG) generates a test paper automatically online according to user specification, and the generated test paper can then be attempted online by user. Online-TPG is a challenging problem. Firstly, TPG is categorized as a multi-objective optimization problem on constraint satisfaction, which is NP-hard [18]. Secondly, Online-TPG is required to be solved online efficiently. However, the current TPG techniques have not taken the online generation requirement into consideration as TPG is traditionally considered as an offline process similar to other multi-objective optimization problems such as timetabling [35] and job-shop scheduling [3]. One of the main issues of Online-TPG is its exhaustive search in a very large search space of possible candidates with multi-objective constraints. This is often known as the curse of high dimensionality [37] because the multi-objective optimization process of TPG, using either weighting parameters or Pareto front, could easily get stuck in a local optimal solution and lose the convergence [19]. Therefore, the TPG process is computationally expensive.

TIn [29], we have proposed an efficient optimization approach based on the divide-and-conquer (DAC) framework to reduce the high dimensionality of multi-objective optimization for Online-TPG. Instead of solving the set of constraints simultaneously, DAC solves its two decomposed subsets of constraints progressively. As such, we can enhance the quality of the solution in multi-objective optimization, and eliminate the need of using a complex objective function with its weighting parameters which are generally not easy to determine. The main drawback of DAC is its heuristic to generate deterministically a unique initial solution, which could lead to a local optimal because the search process focuses only on optimizing the unique initial solution among the several possible ones.

This paper proposes an efficient approach, called divide-and-conquer memetic algorithm (DAC-MA), for Online-TPG. In the proposed approach, we incorporate a memetic algorithm into the DAC framework to further enhance the performance of Online-TPG. The set of constraints is divided into two subsets of relevant constraints, namely *content constraints* and *assessment constraints*, which can then be solved separately and progressively by evolutionary computation and local search of DAC-MA respectively. Specifically, the evolutionary computation of DAC-MA explores different initial solutions that satisfy the *content constraints*; whereas, the local search of DAC-MA exploits these solutions to optimize on the *assessment constraints*.

Although DAC-MA tackles the same Online-TPG problem as DAC, it has three novel contributions. Firstly, different from the local optimal optimization approach of DAC, DAC-MA can achieve global optimal solutions for multi-objective optimization in Online-TPG by constructing stochastically a diversified population of representative solutions with evolutionary computation. Secondly, DAC-MA proposes an efficient approach for a practical multi-objective optimization problem with the curse of high dimensionality of constraints, for which classical MA has not addressed. Thirdly, the performance of the local search in DAC-MA is further improved by using the nearest neighbor search. To show its efficiency, the computational complexity of DAC-MA is also analyzed in this paper.

In this paper, we discuss the proposed DAC-MA approach for online-TPG. The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 gives the problem specification. The proposed DAC-MA approach for Online-TPG is presented in Sect. 4. Section 5 discusses the Online-TPG. Section 6 gives the performance results of the proposed approach and its comparison with other TPG techniques. Finally, the conclusion is given in Sect. 7.

2 Related work

2.1 Test paper generation

In [15], dynamic programming was proposed to construct test papers by optimizing an objective function incrementally based on the recursive optimal relation of the objective function. In tabu search (TS) [18], an objective function is also defined based on multi-criteria constraints and weighting parameters for test paper quality. TS optimizes test paper quality by the evaluation of the objective function. In [17], a genetic algorithm (GA) was proposed to generate quality test papers by optimizing a fitness ranking function based on the principle of population evolution. In [34], differential evolution (DE) was proposed for TPG. DE is similar to the spirit of GA with some modifications on solution representation, fitness ranking function, and the crossover and mutation operations to improve the performance. In [25], an artificial immune system (AIS) was proposed to use the clonal selection principle to deal with the highly similar antibodies for elitist selection in order to maintain the best test papers for different generations.

In addition, swarm intelligence algorithms such as particle swarm optimization and ant colony optimization have also been investigated for TPG. In [13], particle swarm optimization (PSO) was proposed to generate multiple test papers by optimizing a fitness function which is defined based on multi-criteria constraints. In [14], ant colony optimization (ACO) was proposed to generate quality test papers by optimizing an objective function which is based on the simulation of the foraging behavior of real ants.

As observed from the above discussion, to generate good test papers, the current TPG techniques generally define an objective function based on multi-criteria constraints and weighting parameters for test paper quality. Then, the objective function is improved iteratively to optimize simultaneously the multi-objective criteria of test paper quality. As such, these techniques generally require weighting parameters and some other parameters such as population size, tabu length, etc. for each TPG that is not easy to determine. In addition, these TPG techniques also generally take long runtime for generating good quality test papers especially for large datasets of questions.

2.2 Memetic algorithms

Memetic algorithms (MA) [12,20,30,31] represent one of the recent growing research areas in evolutionary computation. Inspired by principles of natural selection, the term "memetic algorithm" was first introduced by Moscato [28] who viewed MA as a form of population-based hybrid genetic algorithm (GA) coupled with an individual learning procedure capable of performing local refinements. As a synergy of the diversification process in a population-based approach with the intensification process in individual improvement mechanism, MA is able to converge to high quality solutions more efficiently than their conventional counterparts such as evolutionary algorithms, simulated annealing, and tabu search.

MAs have been widely used for solving various real-world applications such as scheduling, planning, vehicle routing, non-linear optimization, control system, and aircraft design. In these applications, MAs are classified into three categories: simple hybrid, adaptive hybrid, and memetic automation [6]. Both simple hybrid and adaptive hybrid are commonly used as a hybridization of evolutionary computation and local search. To enhance the performance, simple hybrid incorporates domain-specific knowledge whereas adaptive hybrid uses population diversity management and adaptation strategies. Different from simple and adaptive hybrids which focus more on the learning process of MA, memetic automation focuses more on the evolutionary computation process, which is designed specifically for problem-solving in a complex dynamic environment.

Recently, there have been increasing interest in investigating simple hybrid and adaptive hybrid for tackling multi-objective optimization problems [12,22]. In simple hybrid, special population-based methods [11,20,21] or individual improvement methods [21] are designed to deal with multi-objective optimization. In adaptive hybrid, some adaptive coordinations of individual improvement methods [4,5] are proposed for handling multi-objective optimization. Traditionally, all of these approaches are proposed for *offline* multi-objective optimization by using either weighting parameters [27] or Pareto front [21] methods. However, these approaches are computationally expensive especially when there is a high number of multi-objective constraints [10,19,21,37].

In this research, we propose a new memetic algorithm, called DAC-MA that adopts a hybridization of simple hybrid and adaptive hybrid, and the principle of dimensionality reduction for Online-TPG. In addition, domain-specific knowledge, Pareto front and population diversity management strategy are also incorporated to enhance the performance of Online-TPG.

3 Problem specification for Online-TPG

3.1 Question dataset

Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ be a dataset consisting of n questions, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ be a set of m different topics, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ be a set of k different question types. Each question $q_i \in \mathcal{Q}$, where $i \in \{1, 2, \ldots, n\}$, has 8 attributes $\mathcal{A} = \{q, o, a, e, t, d, c, y\}$ defined as follows:

Table 1 An example math dataset

$\mathcal{Q}$_ID	o	a	e	t	d	c	y
(a) Question table							
q_1	...	...	4	9	1	c_1	y_1
q_2	...	...	7	10	2	c_1	y_1
q_3	...	...	5	7	6	c_1	y_1
q_4	...	...	7	10	9	c_1	y_1
q_5	...	...	6	8	4	c_1	y_1
q_6	...	...	4	6	5	c_2	y_1
q_7	...	...	5	2	3	c_2	y_1
q_8	...	...	3	2	6	c_2	y_1
q_9	...	...	4	3	8	c_1	y_2
q_{10}	...	...	3	5	7	c_1	y_2
q_{11}	...	...	6	3	4	c_1	y_2
q_{12}	...	...	7	1	9	c_2	y_2
q_{13}	...	...	6	3	10	c_2	y_2

$\mathcal{C}$	Name
(b) Topic table	
c_1	Integration
c_2	Differentiation

$\mathcal{Y}$	Name
(c) Question type table	
y_1	Multiple choice
y_2	Fill-in-the-blank

- *Question q*: It is used to store the question identity.
- *Content o*: It is used to store the content of a question.
- *Answer a*: It is used to store the answer of a question.
- *Discrimination degree e*: It is used to indicate how good the question is in order to distinguish user proficiency. It is an integer value ranging from 1 to 7.
- *Question time t*: It is used to indicate the average time needed to answer a question. It is an integer value in minutes.
- *Difficulty degree d*: It is used to indicate how difficult the question is to be answered correctly. It is an integer number ranging from 1 to 10.
- *Related topic c*: It is used to store a set of related topics of a question.
- *Question type y*: It is used to indicate the type of a question. There are mainly three question types, namely fill-in-the-blank, multiple choice and long question.

Question attributes can be labeled semi-automatically [14] or manually by human experts. Table 1 shows a sample Math question dataset.

3.2 Test paper specification

A *test paper specification* $\mathcal{S} = \langle N, T, D, C, Y\rangle$ is a tuple of five attributes which are defined based on the attributes of the selected questions as follows:

- *Number of questions* N: It is an optional input for the number of questions specified for the test paper.
- *Total time* T: It is the total time specified for the test paper.
- *Average difficulty degree* D: It specifies the average difficulty degree for all the questions in the test paper.
- *Topic distribution* $C = \{(c_1, pc_1), (c_2, pc_2), \ldots, (c_M, pc_M)\}$: It specifies the proportion of topics. The user can enter either the proportion or the number of questions for each topic. If the number of questions is entered, then it will be converted into the corresponding proportion.
- *Question type distribution* $Y = \{(y_1, py_1), (y_2, py_2), \ldots, (y_K, py_K)\}$: It specifies the proportion of question types. The user can enter either the proportion or the number of questions for each question type. Similarly, if the number of questions is entered, then it will be converted into the corresponding proportion.

3.3 Problem specification

Given a test paper specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y}\rangle$, the TPG process aims to find a subset of questions from a question dataset $\mathcal{Q} = \{\amalg_\infty, \amalg_\in, \ldots, \amalg_\backslash\}$ to form a test paper P with specification $\mathcal{S}_P$ that maximizes the average discrimination degree and satisfies the test paper specification such that $\mathcal{S}_P = \mathcal{S}$.

By rewriting the topic distribution pc_l for each specified topic c_l and the question type distribution py_j for each specified question type y_j in terms of irreducible rational numbers [36], we have $pc_l = a_l/A_l, l = 1\ldots M$, and $py_j = b_j/B_j, j = 1\ldots K$, where a_l, A_l, b_j and B_j are integer numbers. Based on the question attributes and test paper specification $\mathcal{S}$, the TPG problem can be formulated as a standard 0–1 fractional integer linear programming (ILP) problem [36] as shown in Fig. 1.

In Fig. 1, constraint (1) is the constraint on the number of questions, where $x_i \in \{0, 1\}$ is a binary variable associated with question $q_i, i = 1\ldots n$, in the dataset. Constraint (2) is the total time constraint. Constraint (3) is the average difficulty degree constraint. Constraint (4) is the topic distribution constraint. The relationship of a question $q_i, i = 1\ldots n$, and a topic $c_l, l = 1\ldots M$, is represented as r_{il} such that $r_{il} = 1$ if question q_i is related to topic c_l and $r_{il} = 0$ if otherwise. Constraint (5) is the question type distribution constraint. The relationship of a question $q_i, i = 1\ldots n$, and a question type $y_j, j = 1\ldots K$, is represented as s_{ij} such that $s_{ij} = 1$ if question q_i is related to question type y_j and $s_{ij} = 0$ if otherwise.

$$
\begin{array}{lll}
Maximize & \sum_{i=1}^n e_i x_i / \sum_{i=1}^n x_i & \\
s.t. & \sum_{i=1}^n x_i = N & (1)\\
 & \sum_{i=1}^n t_i x_i = T & (2)\\
 & \sum_{i=1}^n (d_i - D) x_i = 0 & (3)\\
 & \sum_{i=1}^n (A_l r_{il} - a_l) x_i = 0 \quad \forall l = 1..M & (4)\\
 & \sum_{i=1}^n (B_j s_{ij} - b_j) x_i = 0 \quad \forall j = 1..K & (5)\\
 & x \in \{0,1\}^n & (6)
\end{array}
$$

Fig. 1 Standard 0–1 fractional ILP problem for Online-TPG

It is important to note that the TPG process occurs online where user expects to generate a test paper within an acceptable response time. Hence, the Online-TPG problem has another implicit constraint for the online requirement, i.e., the TPG process has to be completed within τ minutes. We refer the Online-TPG problem as an *online multi-objective optimization* problem. Therefore, Online-TPG is as hard as other optimization problems due to its computational NP-hardness, and it is also required to be solved efficiently in runtime.

4 The DAC-MA approach for Online-TPG

In this section, we propose the DAC-MA approach for Online-TPG. As shown in Fig. 1, most of the constraints of the TPG formulation are in the form of linear equality constraints. Each constraint is often considered as a dimension in a search space of multi-objective constraints. As such, the number of dimensions or objectives in Online-TPG could be high especially when the test paper is specified with many topics and question types. However, we observe that content and question type constraints can be easily satisfied if we can find an appropriate fixed number of questions in a test paper. After satisfying the content constraints, the Online-TPG becomes a simpler problem with just two remaining constraints on total time and average difficulty degree. Therefore, to reduce the curse of high dimensionality for multi-objective optimization of Online-TPG, we can divide the set of constraints into two subsets of relevant constraints, namely *content constraints* and *assessment constraints*, which can then be solved separately and effectively by evolutionary computation and local search of DAC-MA respectively. In the test paper specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y}\rangle$ shown in Fig. 1, the content constraints include constraint (4) on topic distribution C and constraint (5) on question type distribution Y, whereas the assessment constraints include constraint (2) on total time T and constraint (3) on average difficulty degree D.

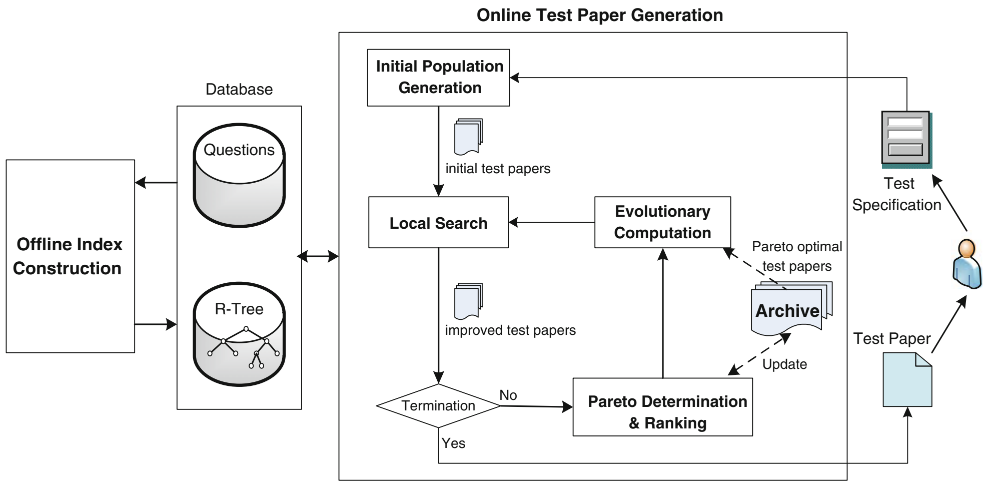

Fig. 2 The proposed DAC-MA approach

The proposed DAC-MA approach is shown in Fig. 2 which consists of the following two main processes:

– *Offline Index Construction*: It constructs an effective indexing structure for supporting local search for improving the quality of the generated paper.
– *Online Test Paper Generation*: It generates an optimal test paper which satisfies the specified content constraints and assessment constraints using the memetic algorithm.

In the Offline Index Construction process, we use an effective 2-dimensional data structure, called R-Tree [2,26], to store questions based on the time and difficulty degree attributes. R-Tree has been widely used for processing queries on 2-dimensional or 3-dimensional spatial databases. As there is no specified rule on grouping of data into nodes in R-Tree, different versions of R-Trees have been proposed. The R-Tree used here is similar to the R-Tree version discussed in [2], with some modifications on index construction in order to enhance efficiency. Some of the modified operations include insertion, subtree selection, overflow handling and node splitting. Each leaf node in R-Tree is a minimum bounding rectangle (MBR) which is the smallest rectangle in the 2-dimensional representation that tightly encloses all question data based on the time and difficulty degree attributes.

5 Online test paper generation

In the Online-TPG process, it first generates an initial population of test papers by satisfying the content constraints. Next, it performs local search to improve the quality of test papers by minimizing the assessment constraint violations. The test papers are then ranked based on a fitness function. Evolutionary computation is then performed based on genetic operators to further diversify and improve the quality of the test papers. The improvement process is repeated until an optimal test paper with high quality is generated. As illustrated in Fig. 2, the Online-TPG process consists of 4 major steps: Initial Population Generation, Local Search, Pareto Determination & Ranking and Evolutionary Computation. Algorithm 8 presents the overall DAC-MA approach for Online-TPG.

5.1 Initial population generation

This step generates an initial population of test papers by satisfying the content constraints. As previously mentioned, the content constraints can be easily satisfied by maintaining an appropriate fixed number of questions in a test paper. However, a user may not need to specify the number of questions for the test paper during test paper specification. In this case, the minimal number of questions for a test paper needs to be estimated. It can be done by using the content constraints on topic distribution and question type distribution specified by the user. From the problem formulation given in Fig. 1, the lowest common denominator A for all $A_l, l = 1 \dots M$, and lowest common denominator B for all $B_j, j = 1 \dots K$, can be computed. Let N be the number of questions for the test paper. By some simple arithmetic, we can find that $N = k * N_b, k \in \mathbb{N}, k \geq 1$, where N_b is computed as:

$$N_b = \frac{A * B}{gcd(A, B)}$$

where $gcd(A, B)$ is the greatest common divisor of A and B. To compute $gcd(A, B)$, we use the well-known Euclidean algorithm [36] which is based on the modular arithmetic operator. To find an appropriate number of questions such that $N = k * N_b, k \geq 1$, we try with $k = 1, 2, \dots$ and check

Algorithm 1: Divide_And_Conquer_Memetic_Algorithm

```
Input: $\mathcal{S} = (\mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y})$ - test paper specification; $\mathcal{Q}$ - question dataset; $\mathcal{R}$ - R-Tree
Output: $P$ - test paper
begin
1   Initialize the Archive $\mathcal{A}_r \leftarrow \emptyset$;
2   Generate initial population $\mathcal{I} \leftarrow \{P_1, P_2, \ldots, P_{K_{pop}}\}$;          /* content constraint satisfaction */
3   while Termination condition is satisfied do
4       foreach $P_i$ in $\mathcal{I}$ do                                           /* assessment constraint optimization */
5           $P_i' \leftarrow$ Local_Search($\mathcal{S}, \mathcal{P}_\rangle, \mathcal{R}$);
            $\mathcal{A}_r \leftarrow \mathcal{A}_r \bigcup P_i'$;                  /* insert new test papers into Archive */
6       Update the Archive $\mathcal{A}_r$ by Pareto Optimal Determination and Ranking;
7       Generate new population $\mathcal{I}$ by Evolutionary Computation;       /* content constraint exploration */
8   return $P \leftarrow \underset{P_i \in \mathcal{A}_r}{\text{argmin}}\ f(P_i)$
```

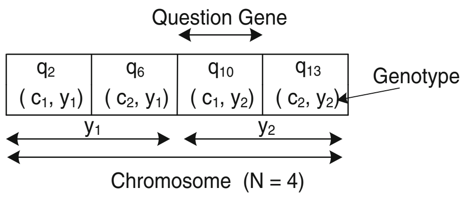

Fig. 3 Test paper chromosome and question gene

whether N is suitable for the test paper which satisfies the total time constraint specified.

Next, we discuss how to represent a test paper for evolutionary computation. It is natural to represent a test paper as a *chromosome* in terms of a vector. The chromosome representation of a test paper with N questions is a list of N *genes*. Each question gene $\langle q, (c, y)\rangle$ consists of 3 components: question, related topic and question type. The collection of all topic-question type pairs (c, y) of the question genes in a chromosome is called *genotype* of that chromosome. The question genes in a chromosome are also organized firstly according to the question type, and then the related topic. Figure 3 illustrates an example of the test paper chromosome.

To generate an initial population of test papers, we need to determine the population size. The population size K_{pop} is very important in evolutionary computation because it determines the diversification of the population in order to achieve global optimal solution. For diversity, we try to keep as many distinguishing genotypes as possible in the population. It is important to enumerate the number of all possible genotypes with respect to the content constraints in order to determine the appropriate size of the initial population. Therefore, given the number of N question genes and the content constraints, we need to enumerate all possible topic-question type assignments so that the content constraints are satisfied. This enumeration problem, in fact, is a variant of the Polya's Theory of Counting [9], which provides a solution to count the number of all possible genotypes with duplications. By eliminating the duplications, the number of all distinguishing genotypes

Table 2 Population size determination

N	≤ 4	5–8	9–15	16–30	31–40	>41
K_{pop}	3	6	10	30	50	100

can be counted. As the number of genotypes increases when N increases, it is more effective to allow a variable-size population according to the number of genotypes. Table 2 gives the population size K_{pop} according to the number of questions N based on [9].

To generate the initial population of K_{pop} test papers in which each test paper will have the same number of N question genes, we use a randomization technique to ensure that all the generated test papers will satisfy the content constraints and have as many different genotypes as possible. Assume that we have already fixed the question types of N questions, we use N randomizers which generate integer numbers in uniform distribution to assign different topics into each of the question genes in the test paper for generating the population. After that, N random questions are chosen based on the genotypes. Figure 4 shows an example of the initial population generated from the test paper specification $\mathcal{S} = \langle 4, 30, 5, \{(c_1, 0.5), (c_2, 0.5)\}, \{(y_1, 0.5), (y_2, 0.5)\}\rangle$ based on the Math dataset.

5.2 Local search

After generating the initial population of test papers, we conduct local search by minimizing assessment constraint violations to improve the quality of the test papers. Before discussing the local search algorithm, we need to define assessment constraint violation and the fitness function for the evaluation of the quality of a test paper.

Definition 1 (*Assessment constraint violation*) Given a test paper specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y}\rangle$. Let P be a generated test paper with specification $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P\rangle$.

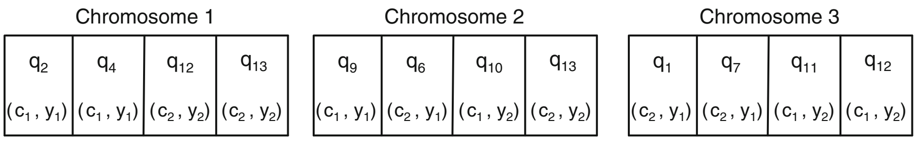
Chromosome 1
q2 q4 q12 q13
(c1 , y1) (c1 , y1) (c2 , y2) (c2 , y2)
Chromosome 2
q9 q6 q10 q13
(c1 , y1) (c2 , y1) (c1 , y2) (c2 , y2)
Chromosome 3
q1 q7 q11 q12
(c2 , y1) (c2 , y1) (c1 , y2) (c1 , y2)

Fig. 4 Initial population generation

Assessment constraint violation indicates the differences between the test paper specification and the generated test paper according to the total time constraint and the average difficulty degree constraint. Therefore, assessment constraint violation consists of total time constraint violation and average difficulty degree constraint violation which are defined as follows:

Total time constraint violation:

$$\triangle T(\mathcal{S}_P, \mathcal{S}) = \frac{T_P - T}{T}$$

Average difficulty degree constraint violation:

$$\triangle D(\mathcal{S}_P, \mathcal{S}) = \frac{D_P - D}{D}$$

A generated test paper P with specification $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P\rangle$ is said to satisfy the assessment constraints in $\mathcal{S}$ if $|\triangle T(\mathcal{S}_P, \mathcal{S})| \le \alpha$ and $|\triangle D(\mathcal{S}_P, \mathcal{S})| \le \beta$, where α and β are two predefined thresholds which indicate acceptable quality satisfaction on total time and average difficulty degree respectively.

Typically, a fitness function can be defined to compare test papers based on assessment constraint violation. Intuitively, a good test paper should have small values of $\triangle T$ and $\triangle D$.

Definition 2 (*Fitness*) Given a test paper specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y}\rangle$. The fitness $f(P)$ of a test paper P is defined in terms of assessment constraint violation as follows:

$$f(P) = \triangle T(\mathcal{S}_P, \mathcal{S})^2 + \triangle D(\mathcal{S}_P, \mathcal{S})^2$$

Local search aims to find better questions to substitute the existing questions in the test paper in order to minimize assessment constraint violations. Algorithm 10 presents the local search algorithm. It starts from an original test paper P_0 and then iteratively moves to its neighboring solution $P_1 \in N(P_0)$, where $N(P_0)$ is a neighborhood region. Here, the neighborhood region of P_0 is defined so that its genotype is preserved. More specifically, the neighborhood region of P_0 is any test paper that has the same genotype as P_0. As such, the neighborhood region of P_0 could be very large. To form a new test paper, each question q_k in the original test paper P_0 is substituted by another better question q_m which has the same topic and question type such that the assessment constraint violations are minimized. Specifically, the choice of the neighboring solution is determined by minimizing the fitness function $f(P_1)$, $P_1 \in N(P_0)$. To achieve this efficiently, we need to prune the search space and find the best question for substitution. The termination conditions for the local search are based on the criteria for quality satisfaction and the number of iterations.

Algorithm 2: Local_Search

```
Input: $\mathcal{S} = (N, T, D, C, Y)$ - test paper specification;
       $P_0 = \{q_1, q_2, .., q_N\}$ - original test paper;
       $\mathcal{R}$ - R-Tree
Output: $P_1$ - Improved test paper
begin
1   $\mathcal{P} \leftarrow \{P_0\}$;
2   while Termination condition is not satisfied do
3       foreach $q_i$ in $P_0$ do
4           Compute 2-dimensional region $W$ ;
5           $q_m \leftarrow$ Best_First_Search($q_i$, $W$, $\mathcal{R}$);
6           $P_1 \leftarrow \{P_0 - \{q_i\}\} \cup \{q_m\}$ ;
7           Compute fitness $f(P_1)$;
8           Insert new test paper $P_1$ into $\mathcal{P}$;
9       $\mathcal{P} \leftarrow \{P_0\} \leftarrow \operatorname{argmin}_{P_1 \in \mathcal{P}} f(P_1)$ /* best move*/;
10  return $P_1 \leftarrow P_0$
```

5.2.1 Pruning search space

In the local search process, it needs to scan through the entire question list several times to find the most suitable question for improvement. Hence, exhaustive searching on a large question list is computational expensive as it requires $O(N)$ time. To accelerate this step, we focus only on substitution questions that help to improve both aspects of the assessment constraint violation because it helps the search process converge faster towards Pareto Optimal. To do that, we need to prune the search space to find a 2-dimensional region W that contains possible questions for substitution.

Let $\mathcal{S}_{P_0} = \langle N, T_0, D_0, C_0, Y_0\rangle$ be the specification of a test paper P_0 generated from a specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y}\rangle$. Let P_1 be the test paper created after substituting a question q_k of P_0 by another question $q_m \in \mathcal{Q}$ with $\mathcal{S}_{P_1} = \langle N, T_1, D_1, C_1, Y_1\rangle$. The relations of total time and average difficulty degree between P_1 and P_0 can be expressed as follows:

$$T_1 = T_0 + t_m - t_k \tag{7}$$

$$D_1 = D_0 + \frac{d_m}{N} - \frac{d_k}{N} \tag{8}$$

where t_k and t_m are the question time of q_k and q_m respectively, and d_k and d_m are the difficulty degree of q_k and q_m respectively.

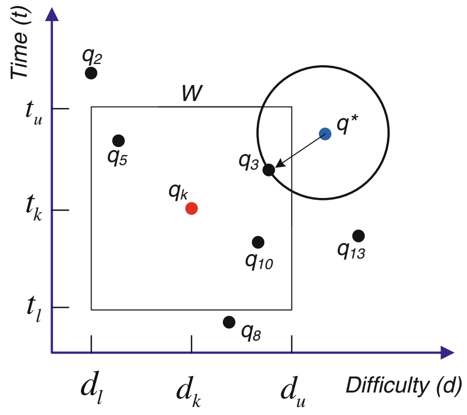

Fig. 5 The 2-dimensional region W and best question selection

Let's consider the total time violation of P_0. If $|\triangle T(\mathcal{S}_{P_0}, \mathcal{S})| = \frac{|\mathcal{T}_l - \mathcal{T}|}{\mathcal{T}} \geq \alpha$ and $T_0 \leq T$, where α is the predefined threshold for constraint satisfaction on total time, we can select q_m to improve the total time satisfaction by increasing the total time from T_0 to T_1 such that $|\triangle T(\mathcal{S}_{P_1}, \mathcal{S})| \leq \alpha$. There are two possibilities:

- $T_0 \leq T_1 \leq T$: By substituting the right hand side of equation (7) into the inequality $| \triangle T(\mathcal{S}_{P_1}, \mathcal{S})| \leq \alpha$. We can derive $t_m \in [t_k + T - T_0 - \alpha T, t_k + T - T_0]$.
- $T_0 \leq T \leq T_1$: Similarly, we can derive $t_m \in [t_k + T - T_0, t_k + T - T_0 + \alpha T]$.

Therefore, we have $t_m \in [t_l, t_u]$, where $t_l = t_k + T - T_0 - \alpha T$, $t_u = t_k + T - T_0$. If $| \triangle T(\mathcal{S}_{P_0}, \mathcal{S})| = \frac{|\mathcal{T}_l - \mathcal{T}|}{\mathcal{T}} \geq \alpha$ and $T_0 > T$, we can derive the same result.

Similarly, we can also derive the result for the difficulty degree of q_m: $d_m \in [d_l, d_u]$, where $d_l = d_k + N(D - D_0) - \beta N D$, $d_u = d_k + N(D - D_0) + \beta N D$, where D_0, D and β are the average difficulty degree of P_0 and $\mathcal{S}$, and the predefined threshold for constraint satisfaction on average difficulty degree respectively. Figure 5 shows the region W of questions for substitution.

5.2.2 Finding best question for substitution

Among all the questions located in the 2-dimensional region W, this step finds the best question that minimizes the fitness function in order to enhance the test paper quality.

Consider question q_m as a pair of variables on its question time t and difficulty degree d. The fitness function $f(P_1)$ can be expressed as a multivariate function $f(t, d)$:

$$f(P_1) = \triangle T(\mathcal{S}_{P_1}, \mathcal{S})^2 + \triangle D(\mathcal{S}_{P_1}, \mathcal{S})^2$$
$$f(t, d) = \left(\frac{T_1 - T}{T}\right)^2 + \left(\frac{D_1 - D}{D}\right)^2$$

From Eqs. (7) and (8), we have:

$$T_1 - T_0 = t - (T - T_0 + t_k) = t - t^*$$
$$D_1 - D_0 = d - (ND - ND_0 + d_k) = d - d^*$$

where $t^* = T - T_0 + t_k$ and $d^* = ND - ND_0 + d_k$. Therefore,

$$\begin{aligned} f(t, d) &= \frac{(t - t^*)^2}{T^2} + \frac{(d - d^*)^2}{D^2} \\ &\geq \frac{(t - t^*)^2 + (d - d^*)^2}{T^2 + D^2} \\ &= \frac{distance^2(q_m, q^*)}{T^2 + D^2} \end{aligned}$$

where q^* is a question having question time t^* and difficulty degree d^*.

As T and D are predefined constants and q^* is a fixed point in the 2-dimensional space, the best question q_m to replace question q_k in P_0 is the question point that is the nearest neighbor to the point q^* (i.e., the minimum value of the function $f(P_1)$) and located in the region W. Figure 5 shows an example in which q_3 is the best question to replace q_k because it is the nearest neighbor to q^* and located in the region W.

To find the best question q_m for substitution efficiently, we perform the best first search (BFS) [33] with the R-Tree. BFS recursively visits the nearest question whose MBR is close to q^*. For efficiency, BFS uses a memory-resident min-heap $\mathcal{H}$ [7] to manage all the questions in the R-Tree that have been accessed. The search continues until a question de-heaped from $\mathcal{H}$ is located in W. As the time complexity of BFS is $O(\lg N)$, we can improve the time complexity of the scanning step to $O(\lg N)$.

5.3 Pareto determination and ranking

After local search, we have the improved test papers in the population. The next step is to determine Pareto optimal test papers that satisfy the Pareto optimal property [11,39] based on the assessment constraint violations. Pareto optimal determination is performed based on the improved test papers from local search and also the current Pareto optimal test papers in the shared memory Archive.

Definition 3 (*Domination relationship*) Given a test paper specification $\mathcal{S}$, a test paper P_i is said to dominate another test paper P_j w.r.t $\mathcal{S}$, denoted as $P_i \succ_{\mathcal{S}} P_j$, if and only if:

- $\triangle T(P_i, \mathcal{S}) \leq \triangle T(P_j, \mathcal{S})$; and
- $\triangle D(P_i, \mathcal{S}) \leq \triangle D(P_j, \mathcal{S})$; and
- $\triangle T(P_i, \mathcal{S}) < \triangle T(P_j, \mathcal{S})$ or $\triangle D(P_i, \mathcal{S}) < \triangle D(P_j, \mathcal{S})$.

Definition 4 (*Pareto optimal*) Given a test paper specification $\mathcal{S}$, a test paper P_i is said to be a Pareto optimal solution w.r.t $\mathcal{S}$ if and only if there does not exist any test paper $P_j \neq P_i$ such that $P_j \succ_{\mathcal{S}} P_i$.

Pareto optimal determination has been studied for multi-objective optimization for a long time. Although several methods have been proposed, they are not efficiently scalable due to high runtime complexity such as $O(dn^2)$ [11] and $O(dn^3)$ [39], where n is the number of data points in a d-dimensional space of attributes or objectives. However, we found that Pareto optimal determination is in fact a *maximum vector* problem [24], which has an efficient scalable *divide-and-conquer* algorithm of $O(n(\log n)^{d-2} + n \log n)$. For Pareto optimal determination of test papers, we modify the algorithm given in [24] with $d = 2$.

All Pareto optimal test papers will be ranked in ascending order according to the fitness value of $f(P)$. Then, these test papers are stored in the Archive that is a memory-resident table storing all test papers gathered so far in ascending order of the fitness value. For efficiency, the Archive is implemented according to the following ways. Firstly, it is implemented as a variable-sized Archive to avoid loosing good test paper solutions of the stochastic optimization process. The size of the Archive can be adjusted according to the number of distinguishing genotypes of test paper solutions. Secondly, if a candidate test paper solution is not dominated by any solutions in the Archive, it will be added to the Archive. Similarly, if any test paper solutions in the Archive are dominated by the candidate solution, it will be removed from the Archive. Thirdly, the Archive only keeps at most one test paper for each distinguishing genotype to preserve the diversification of the population. Hence, if there are two test papers with the same genotype, they will be compared according to the domination relationship, fitness value and average discrimination degree before deciding which one is added into the Archive. The best Pareto optimal test paper has the minimum fitness function value.

5.4 Evolutionary computation

In evolutionary computation, it aims to diversify the test papers stored in the Archive based on the content constraints. There are two main steps in evolutionary computation: selection and reproduction.

5.4.1 Selection

Test paper chromosomes are selected from the Archive to breed a new generation of population. Individual test paper solutions in the Archive are selected based on the fitness values. The fitness function is designed stochastically so that a small proportion of less fit test paper solutions are also selected to keep the diversity of the new generation. Here, the well-studied method, called roulette wheel selection [1], is used for selecting potentially useful test paper solutions. The roulette wheel selection procedure is repeated until there are enough selected individuals of K_{pop} in the population. The probability for being selected is based on the fitness value associated with each individual test paper chromosome. If $f(P_i)$ is the fitness value of individual P_i in the population, its probability of being selected is:

$$p_i = \frac{1/f(P_i)}{\Sigma_{j=1}^{K_{pop}}(1/f(P_j))}$$

5.4.2 Reproduction

It aims to produce the next population of test paper solutions with new genotypes from the selected test papers. In doing so, the two genetic operators: *mutation* and *crossover* are designed such that new genotypes are explored based on the content constraint satisfaction.

Mutation is a genetic operator used to maintain genetic diversity of a population of chromosomes towards global optimization by preventing the population of chromosomes from becoming too similar to each other. Here, the mutation operation aims to explore different genotypes. We use a typical mutation operator, called two-points gene exchange of a chromosome [1], to select $K_{pop}/4$ random individuals from the parent population for mutation. In each mutation, two random question genes with different topics and question types of a chromosome are selected. For example, in the two question genes, $\langle q_1, (c_1, y_1)\rangle$ and $\langle q_{12}, (c_2, y_2)\rangle$, shown in Fig. 6, the corresponding topics are swapped to yield a chromosome that has two new topic-question types at the same positions of the two original question genes. Therefore, the new chromosome will potentially have a new genotype while content constraint satisfaction is preserved. As the questions q_1 and q_{12} are no longer valid for the two new topic-question types (c_2, y_1) and (c_1, y_2), the questions q_1 and q_{12} are replaced by the questions q_6 and q_{10}.

Crossover is a genetic operator used to vary the chromosomes from one generation to the next for reproduction. Here, the crossover operation aims to improve the test paper quality towards assessment constraint satisfaction while preserving content constraint satisfaction. We use the gene exchange crossover operator [1] based on two chromosomes. The crossover operator exchanges a pair of question genes that have a common topic-question type of

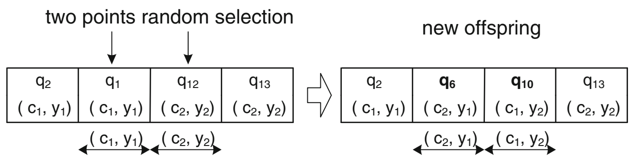

Fig. 6 Mutation operation

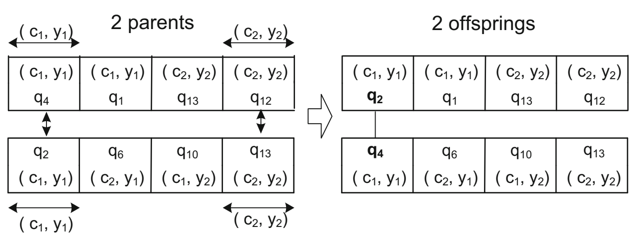

Fig. 7 Crossover operation

two parent chromosomes to render two child chromosomes. Figure 7 shows an example of chromosome crossover. The two chromosomes have two common topic-question type pairs (c_1, y_1) and (c_2, y_2). The questions q_2 and q_4 are exchanged. However, the questions q_{12} and q_{13} cannot be exchanged because q_{13} has already existed in the test paper.

The crossover operation is conducted on a predefined number of $K_{pop}/4$ random pairs of genes. For effective crossover, we partition the population of test papers into 4 groups according to the positive and negative values of assessment constraint violations as follows:

1. $\triangle T(\mathcal{S}_P, \mathcal{S}) < 0$ and $\triangle D(\mathcal{S}_P, \mathcal{S}) < 0$
2. $\triangle T(\mathcal{S}_P, \mathcal{S}) < 0$ and $\triangle D(\mathcal{S}_P, \mathcal{S}) > 0$
3. $\triangle T(\mathcal{S}_P, \mathcal{S}) > 0$ and $\triangle D(\mathcal{S}_P, \mathcal{S}) < 0$
4. $\triangle T(\mathcal{S}_P, \mathcal{S}) > 0$ and $\triangle D(\mathcal{S}_P, \mathcal{S}) > 0$

The crossover operation is performed between 2 individuals from group (1) and group (4), or between individuals from group (2) and group (3). It is possible that both individuals are improved in terms of assessment constraint satisfaction after gene exchange by the crossover operator.

5.5 Termination

After evolutionary computation, a new generation of test paper population is generated. The TPG process is repeated with the local search again until the termination conditions are reached. As setting a hard termination condition based on the online runtime requirement may affect badly the algorithm's performance, we use the following two typical termination conditions in our experimental study:

– *Quality satisfaction*: The algorithm will terminate if a high quality test paper is generated.
– *Maximum number of iterations in which no better test paper is found*: This parameter is generally set to 500 iterations for the online runtime requirement.

In practice, we can set a suitable online runtime requirement (for example, 2 minutes) as a termination condition by considering the tradeoff between practical experimental results and the user's expectation.

5.6 Computational complexity

In this section, we analyze the computational complexity of the proposed DAC-MA approach for the Online-TPG process.

5.6.1 Space complexity

It estimates the number of all possible candidate solutions from a given test paper specification. There is at most $O(2^n)$ candidates, where n is the number of questions in the dataset. Recall that we can determinate the number of questions N of a test paper based on the content constraints. Thus, the search space in DAC-MA is reduced to at most $\binom{n}{N} \approx O(n^N)$ candidates. This is approximated by using the well-known Binomial Bound Inequality [7]. The reduced search space in DAC-MA is much smaller as compared with the entire search space $O(2^n)$. In fact, as there are so many invalid test papers in the $O(n^N)$ candidates that violate the content constraints, the actual number of valid candidates considered in DAC-MA is much smaller than $O(n^N)$.

5.6.2 Time complexity

We summarize the time complexity involved in the various steps of our proposed DAC-MA approach as follows:

1. *Determining the number of questions* N: $O(\log N)$
2. *Initial population generation*: $O(K_{pop} \times N)$
3. *Local search*: $O(K_{pop} \times N \times \log n \times \tau_{eval})$
4. *Pareto determination and ranking*: $O(n^N \times \log n^N) = O(n^N \times N \times \log n)$
5. *Evolutionary computation*: $O(K_{pop} \times N)$

where n is the number of questions in the dataset, N is the estimated number of questions in a test paper from a given user specification, $O(\log N)$ is the time complexity of the Euclidian algorithm, $O(\log n)$ is the time complexity of finding the best question for substitution in the R-Tree. We also define the time complexity of a fitness evaluation as τ_{eval} in our analysis.

Let g be the number of generations needed for optimization in the DAC-MA approach. Hence, the total time complexity of the DAC-MA is in the order of:

$$O(\log N + K_{pop} \times N + g \times (K_{pop} \times N \times \log n \times \tau_{eval} + n^N \times N \times \log n + K_{pop} \times N))$$
$$\approx O(g \times (K_{pop} \times N \times \log n \times \tau_{eval} + n^N \times N \times \log n))$$
$$\approx O(g \times \log n \times N \times (K_{pop} \times \tau_{eval} + n^N))$$

5.6.3 Comparison

To show the computational advantages of the proposed DAC-MA approach, we compare DAC-MA with a conventional

Table 3 Comparison on computational complexity ($N \ll n$)

	Space	Time
MA	$O(2^n)$	$O(g \times n^2(K_{pop} \times \tau_{eval} + (nc)^{d-4}2^n))$
DAC-MA	$O(n^N)$	$O(g \times \log n \times N(K_{pop} \times \tau_{eval} + n^N))$

Table 4 Test datasets

	D_1	D_2	D_3	D_4
#Questions	20,000	30,000	40,000	50,000
#Topics	40	50	55	60
#Question types	3	3	3	3

memetic algorithm [21] with Pareto optimal determination [24] for multi-objective optimization of Online-TPG. Similar to DAC-MA, we can analyze the conventional memetic algorithm whose complexity is $O(g \times n^2 \times (K_{pop} \times \tau_{eval} + (nc)^{d-4}2^n))$, where d is the number of specified constraints in the test paper specification, and c is a constant.

Table 3 compares the computational complexity of the proposed DAC-MA with the conventional memetic algorithm. As can be seen, DAC-MA is capable of reducing time complexity quite significantly. More importantly, its time complexity is independent of the number of specified constraints as compared with that of the conventional memetic algorithm. This shows that the proposed DAC-MA is an effective multi-objective optimization approach for Online-TPG.

6 Performance evaluation

In this section, we evaluate the performance of the proposed DAC-MA approach for Online-TPG. The experiments are conducted on a Windows XP environment, using an Intel Core 2 Quad 2.66 GHz CPU with 3.37 GB of memory. The performance of DAC-MA is measured and compared with other techniques including genetic algorithm (GA) [15], particle swarm optimization (PSO) [13], differential evolution (DE) [34], ant colony optimization (ACO) [14], tabu search (TS) [18], and classical MA (MA) [21]. Here, we adapt the classical memetic algorithm [21] with Pareto optimal determination [24] for multi-objective optimization of Online-TPG.

6.1 Datasets

As there is no benchmark datasets available, we generate 4 large-sized synthetic datasets, namely D_1, D_2, D_3 and D_4 with number of questions of 20,000, 30,000, 40,000 and 50,000 respectively for performance evaluation. In these four datasets, the values of each attribute are generated according to a normal distribution. Table 4 shows the summary of the four datasets.

6.2 Experiments

To evaluate the performance of the DAC-MA approach, we have designed 12 test specifications in the experiments. We vary the parameters in order to have different test criteria in the test specifications. The number of topics is specified between 2 and 40. The total time is set between 20 and 240 minutes, and it is also set proportional to the number of selected topics for each specification. The average difficulty degree is specified randomly between 3 and 9.

We perform the experiments according to the 12 test specifications for each of the following 7 algorithms: GA, PSO, DE, ACO, TS, MA, and DAC-MA. We measure the runtime and quality of the generated test papers for each experiment.

6.3 Quality measures

The performance of the proposed DAC-MA approach is evaluated based on paper quality and runtime. To evaluate the quality, we define mean discrimination degree and mean constraint violation.

Definition 5 (*Mean discrimination degree*) Let $P_1, P_2, \ldots, P_k$ be the generated test papers on a question dataset $\mathcal{D}$ w.r.t different test paper specifications $\mathcal{S}_i, i = 1 \ldots k$. The mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ is defined as:

$$\mathcal{M}_d^{\mathcal{D}} = \frac{\sum_{i=1}^{k} E_{P_i}}{k}$$

where E_{P_i} is the average discrimination degree of P_i.

The constraint violations of a generated test paper is computed based on the differences between each of its attributes in the generated paper and the corresponding attributes in the test specification. As such, the mean constraint violation consists of two components:

- *Assessment constraint violation*: It is defined in Definition 1 that consists of the total time constraint violation and average difficulty degree constraint violation.
- *Content constraint violation*: Kullback–Leibler (KL) divergence [23] is a commonly used measure for evaluating the statistical difference between two distributions. In content constraint violation, the KL divergence is used to measure the difference of the topic and question type distributions between the specification $\mathcal{S}_P$ of the test paper P and the test paper specification $\mathcal{S}$.

Definition 6 (*Content constraint violation*) Given a test paper specification $\mathcal{S} = \langle \mathcal{N}, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{Y} \rangle$. Let P be a generated test paper with specification $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P \rangle$.

Content constraint violation consists of topic distribution violation $\triangle C(\mathcal{S}_P, \mathcal{S})$ and question type distribution violation $\triangle Y(\mathcal{S}_P, \mathcal{S})$, which are the differences between the generated test paper and the test paper specification according to the topic distribution and question type distribution respectively. These violations are defined as follows:

Topic distribution Violation:

$$\triangle C(\mathcal{S}_P, \mathcal{S}) = D_{KL}(pc_p || pc) = \sum_{i=1}^{M} pc_{pi} \log \frac{pc_{pi}}{pc_i}$$

Question type distribution violation:

$$\triangle Y(\mathcal{S}_P, \mathcal{S}) = D_{KL}(py_p || py) = \sum_{j=1}^{K} py_{pj} \log \frac{py_{pj}}{py_j}$$

where pc_p and pc are the topic distributions of P and the test paper specification $\mathcal{S}$ respectively, and py_p and py are the question type distributions of P and the test paper specification $\mathcal{S}$ respectively.

The constraint violation (CV) of a generated test paper P with respect to $\mathcal{S}$ is defined as:

$$CV(P, \mathcal{S}) = \frac{\lambda * \triangle T + \lambda * \triangle D + \log \triangle C + \log \triangle Y}{4}$$

As KL divergence may have very large value, the logarithm scale of $\triangle C$ and $\triangle Y$ is used to scale the values to a range between 0 and 100. λ is a constant which is set equal to 100.

Definition 7 (*Mean constraint violation*) The mean constraint violation $\mathcal{M}_c^{\mathcal{D}}$ of k generated test papers $P_1, P_2, \ldots, P_k$ on a question dataset $\mathcal{D}$ w.r.t different test paper specifications $\mathcal{S}_i$, $i = 1 \ldots k$, is defined as:

$$\mathcal{M}_c^{\mathcal{D}} = \frac{\sum_{i=1}^{k} CV(P_i, \mathcal{S}_i)}{k}$$

where $CV(P_i, \mathcal{S}_i)$ is the Constraint Violation of P_i w.r.t. $\mathcal{S}_i$.

To determine the usefulness of a generated test paper, the value of constraint violation should be in a certain range. Specifically, we set the following 4 thresholds for a high quality test paper: $\triangle T(\mathcal{S}_P, \mathcal{S}) \le 0.15$, $\triangle D(\mathcal{S}_P, \mathcal{S}) \le 0.15$, $\log \triangle C(\mathcal{S}_P, \mathcal{S}) \le 5$ and $\log \triangle Y(\mathcal{S}_P, \mathcal{S}) \le 5$. These threshold values are obtained experimentally according to the average of the best quality performance of all algorithms. Based on these thresholds, we have $\mathcal{M}_c^{\mathcal{D}} \le 10$ for high quality test papers, $10 < \mathcal{M}_c^{\mathcal{D}} \le 30$ for medium quality test papers and $\mathcal{M}_c^{\mathcal{D}} > 30$ for low quality test papers.

6.4 Performance results

Figure 8 compares the runtime performance of the seven algorithms based on the four datasets. The results have clearly shown that DAC-MA consistently outperforms other techniques in runtime for the different datasets. It generally requires less than 2 min to complete the paper generation process. Moreover, the proposed DAC-MA approach is scalable in runtime on different dataset sizes. In contrast, other techniques are not efficient to satisfy the online runtime requirement. Specifically, the runtime performance of other techniques degrades quite badly as the dataset size or the number of specified constraints gets larger.

Figure 9 shows the performance results based on the mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ and mean constraint violation $\mathcal{M}_c^{\mathcal{D}}$ of the seven algorithms based on the four datasets. As can be seen from Fig. 9a, DAC-MA has consistently achieved higher mean discrimination degree $\mathcal{M}_d^{\mathcal{D}}$ than other techniques for the generated test papers. In addition, we also observe that DAC-MA has consistently outperformed other techniques on mean constraint violation $\mathcal{M}_c^{\mathcal{D}}$ based on the four datasets. The average constraint violations of DAC-MA tends to decrease whereas the average constraint violations of other approaches increase quite fast when the dataset size or the number of specified constraints gets larger. In particular, DAC-MA can generate high quality test papers with $\mathcal{M}_c^{\mathcal{D}} \le 6$ for all datasets. As such, DAC-MA tends to generate higher quality test papers on larger datasets while other techniques tend to generate lower quality test papers.

The runtime of DAC-MA is more efficient for Online-TPG due to three main reasons. Firstly, the number of constraints is reducible into two subsets of relevant constraints which can be solved effectively by appropriate techniques in DAC-MA. Secondly, DAC-MA utilizes the content constraints to generate initial solutions, thereby pruning the search space significantly to $O(n^N)$ as compared with $O(2^n)$ of other approaches. Thirdly, DAC-MA uses a much simpler objective function without any weighting parameter and it only takes $O(\log n)$ to find suitable questions to improve test paper quality based on R-Tree whereas other approaches need to take $O(n)$. Thus, DAC-MA can improve the computational time. Moreover, as there are more questions with different attribute values on larger datasets and the R-Tree is an effective data structure, DAC-MA is able to generate higher quality test papers. In contrast, the quality performance of other techniques drops quite considerably when many constraints are specified or larger datasets are used. It is because these techniques are easier to get stuck in local optimal.

Table 5 gives the performance comparison between DAC [29] and DAC-MA for each dataset based on the 12 test specifications. As can be seen, the average performance of DAC-MA is consistently better than DAC based on the four datasets. However, the runtime performance of DAC and DAC-MA depends on the test paper specification. If the specified test papers contain many topics or have high total time, DAC-MA outperforms DAC in runtime. Otherwise, DAC achieves better runtime performance. The main reason is that

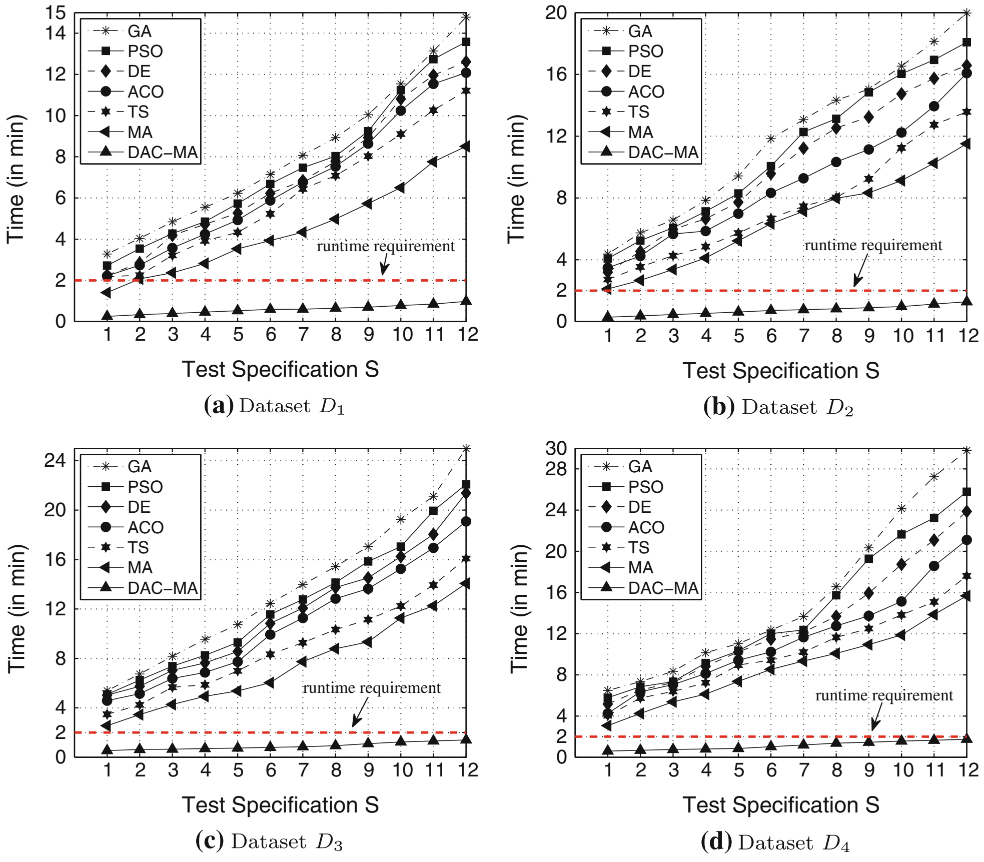

Fig. 8 Performance results based on runtime

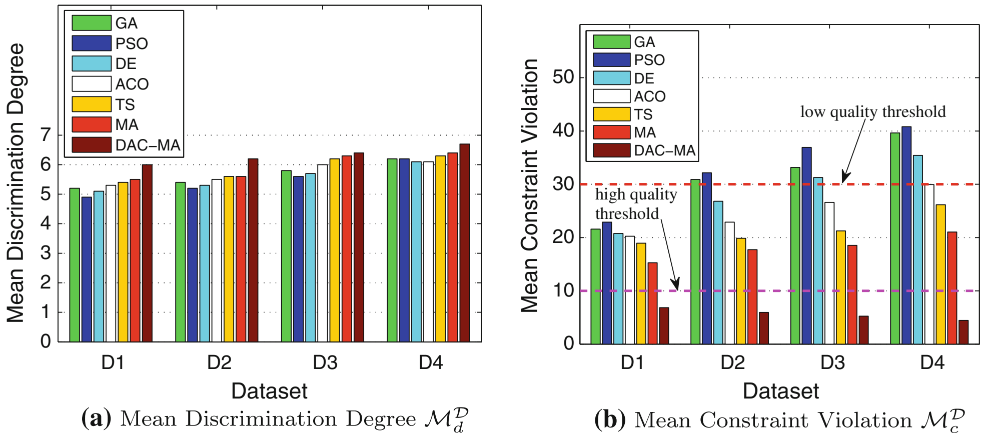

Fig. 9 Performance results based on quality

DAC optimizes a unique initial solution whereas DAC-MA optimizes a diverse initial population of representative solutions. When the number of specified topics or total time is small, the number of distinguishing genotypes is small. There may have enough relevant questions for DAC to optimize the unique solution for generating good quality test paper. The runtime of DAC outperforms DAC-MA in this situation as it is a single-based approach while DAC-MA is a population-based approach. However, when the number of specified topics or total time is high, and the number of distinguishing genotypes is also high. There may not have enough relevant questions for DAC to optimize the unique initial solution. Thus, the best solution achieved by DAC is not as good as that of DAC-MA. In addition, DAC has to spend more time to achieve its best solution than DAC-MA, eventhough it is a single-based approach.

Table 5 Performance comparison between DAC and DAC-MA

	Algorithm	D_1	D_2	D_3	D_4
Average runtime (s)	DAC	38.4	48.0	59.4	72.6
	DAC-MA	35.4	44.4	54.0	68.4
Average discrimination degree $\mathcal{M}_d^{\mathcal{D}}$	DAC	5.50	5.80	6.25	6.40
	DAC-MA	6.00	6.20	6.40	6.7
Mean constraint violation $\mathcal{M}_c^{\mathcal{D}}$	DAC	6.85	5.94	5.25	4.45
	DAC-MA	5.35	4.68	4.24	3.05

7 Conclusion

In this paper, we have proposed an efficient DAC-MA approach for online-TPG. The proposed DAC-MA approach is based on constraint decomposition and memetic algorithm for multi-objective optimization. The underlying idea is that DAC-MA is able to divide the set of constraints into two subsets of relevant constraints, which can then be optimized effectively. In this paper, we have also evaluated the performance of the proposed approach. The performance results have shown that the proposed DAC-MA approach has not only achieved good quality test papers, but also satisfied the online runtime requirement as compared with other heuristic techniques. As such, the proposed approach is effective for Online-TPG in terms of runtime efficiency and paper quality. It is particularly useful for generating test papers online for Web-based testing and intelligent tutoring.

There are two possible directions for further enhancing the performance of the proposed DAC-MA approach. First, approximate local search [32] which has optimal and efficient performance guarantee property can be investigated for maximizing the average discrimination degree while satisfying the constraints during the online test paper optimization process. Second, the online runtime performance of DAC-MA can be improved by using a better population selection method in the evolutionary computation. Instead of using the fitness value, genotypes could be selected for local search improvement based on the ranking of their potential in generating high quality test papers. As there are similar statistical ranking techniques [8] proposed in the machine learning research community, these techniques could be investigated for the population selection method.

References

1. Back T, Fogel DB, Michalewicz Z (1999) Evolutionary computation 1: basic algorithms and operators. IOP Publishing Ltd., UK
2. Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The r*-tree: an efficient and robust access method for points and rectangles. ACM SIGMOD Record 19(2):322–331
3. Bodin L, Golden B, Assad A, Ball M (1983) Routing and scheduling of vehicles and crews: the state of the art. Comput Oper Res 10(2):63–211
4. Bosman PAN, de Jong ED (2006) Combining gradient techniques for numerical multi-objective evolutionary optimization. In: Proceedings of the 8th annual conference on genetic and evolutionary computation, ACM, pp 627–634
5. Caponio A, Neri F (2009) Integrating cross-dominance adaptation in multi-objective memetic algorithms. In: Goh CK, Ong YS, Tan KC (eds) Multi-objective memetic algorithms, vol 171. Springer, New York, pp 325–351
6. Chen XS, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607
7. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, 2nd Edn. McGraw-Hill Science, New York 127–144
8. Das A, Kempe D (2011) Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In: International conference on machine learning ICML
9. de Bruijn NG (1964) Polya's theory of counting. In: Beckenbach EF, Polya G (eds) Applied combinatorial mathematics. Wiley, New York, pp 144–184
10. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
11. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197
12. Goh CK, Ong YS, Tan KC (2009) Multi-objective memetic algorithms, vol 171. Springer, New York
13. Ho TF, Yin PY, Hwang GJ, Shyu SJ, Yean YN (2008) Multi-objective parallel test-sheet composition using enhanced particle swarm optimization. J ETS 12(4):193–206
14. Hu XM, Zhang J, Chung HSH, Liu O, Xiao J (2009) An intelligent testing system embedded with an ant-colony-optimization-based test composition method. IEEE Trans Syst Man Cybern 39(6):659–669
15. Hwang GJ (2003) A test-sheet-generating algorithm for multiple assessment requirements. IEEE Trans Educ 46(3):329–337
16. Hwang GJ, Chu HC, Yin PY, Lin JY (2008) An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests. Comput Educ 51(3):1058–1072
17. Hwang GJ, Lin B, Tseng HH, Lin TL (2005) On the development of a computer-assisted testing system with genetic test sheet-generating approach. IEEE Trans Syst Man Cybern 35(4):590–594
18. Hwang GJ, Yin PY, Yeh SH (2006) A tabu search approach to generating test sheets for multiple assessment criteria. IEEE Trans Educ 49(1):88–97
19. Ishibuchi H, Tsukamoto N, Nojima Y (2008) Evolutionary many-objective optimization: a short review. In: IEEE world congress on evolutionary computation, IEEE, pp 2419–2426
20. Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2):204–223
21. Knowles J, Corne D (2005) Memetic algorithms for multiobjective optimization: issues, methods and prospects. In: Recent advances in memetic algorithms. Springer, Berlin, pp 313–352
22. Knowles J, Corne D, Deb K (2008) Multiobjective problem solving from nature: from concepts to applications. Springer, Berlin
23. Kullback S (1997) Information theory and statistics. Dover Publisher, New York
24. Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors. J ACM 22(4):469–476
25. Lee CL, Huang CH, Li CJ (2007) Test-sheet composition using immune algorithm for e-learning application. New Trends Appl Artif Intell 4570:823–833

26. Manolopoulos Y, Nanopoulos A, Theodoridis Y (2006) R-trees: theory and applications. Springer, Berlin
27. Mei Y, Tang K, Yao X (2011) Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. IEEE Trans Evol Comput 15(2):151–165
28. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program, C3P Report, 826
29. Nguyen ML, Hui SC, Fong ACM (2011) An efficient multi-objective optimization approach for online test paper generation. In: IEEE symposium on computational intelligence in multicriteria decision-making (MDCM), pp 182–189
30. Ong YS, Keane AJ (2004) Meta-lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110
31. Ong YS, Lim M, Chen X (2010) Research frontier: memetic computation—past, present & future. IEEE Comput Intell Mag 5(2):24–31
32. Orlin JB, Punnen AP, Schulz AS (2004) Approximate local search in combinatorial optimization. In: Proceedings of ACM-SIAM SODA, pp 587–596
33. Roussopoulos N, Kelley S, Vincent F (1995) Nearest neighbor queries. In: Proceedings of the ACM SIGMOD, pp 71–79
34. Rui WF, Hong WW, Ke PQ, Chao ZF, Liang JJ (2009) A novel online test-sheet composition approach for web-based testing. In: Symposium on IT in medicine & education, pp 700–705
35. Schaerf A (1999) A survey of automated timetabling. In: Artificial intelligence review, vol 13, issue 2. Springer, Berlin, pp 87–127
36. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
37. Schutze O, Lara A, Coello CAC (2011) On the influence of the number of objectives on the hardness of a multiobjective optimization problem. IEEE Trans Evol Comput 15(4):444–455
38. Tsai KH, Wang TI, Hsieh TC, Chiu TK, Lee MC (2009) Dynamic computerized testlet-based test generation system by discrete pso with partial course ontology. Expert Syst Appl 37(1):774–786
39. Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: International conference on parallel problem solving from nature. Springer, Berlin, pp 292–301