Submodular Memetic Approximation for Multiobjective Parallel Test Paper Generation

Minh Luan Nguyen, *Member, IEEE,* Siu Cheung Hui, and Alvis C. M. Fong, *Senior Member, IEEE*

***Abstract*—Parallel Test Paper Generation is a biobjective distributed resource optimization problem, which aims to generate multiple *similarly optimal* test papers automatically according to multiple user-specified assessment criteria. Generating high-quality parallel test papers is challenging due to its NP-hardness in both of the *collective objective functions*. In this paper, we propose a Submodular Memetic Approximation algorithm for solving this problem. The proposed algorithm is an adaptive memetic algorithm, which exploits the submodular property of the collective objective functions to design greedy-based approximation algorithms for enhancing steps of the multiobjective memetic algorithm. Synergizing the intensification of submodular local search mechanism with the diversification of the population-based submodular crossover operator, our algorithm can jointly optimize the *total quality maximization* objective and the *fairness quality maximization* objective. Our memetic algorithm can achieve provable near-optimal solutions in a huge search space of large datasets in efficient polynomial runtime. Performance results on various datasets have shown that our algorithm has drastically outperformed the current techniques in terms of paper quality and runtime efficiency.**

***Index Terms*—Parallel test paper generation, multiobjective optimization, approximation algorithm, submodular optimization, constraint optimization.**

I. Introduction

With the rapid growth of the Internet and mobile devices, Web-based education has become a ubiquitous learning platform in many institutions to provide students with online learning courses and materials through freely accessible educational websites such as Khan Academy[1], or online classes such as Coursera[2], and Udacity[3]. To make learning effective, it is important to assess the proficiency of the students while they are learning the concepts. Web-based testing has been popularly used for automatic self-assessment especially in Web-based learning environments. The main benefit is that students can take classes at their own pace and get immediate feedback on their proficiency. As there may have many students in an online class [1], to ensure the assessment reliability of large-scale Web-based testing, pedagogical practitioners have suggested that it is necessary to compose multiple tests from a large question pool with equivalent properties.

One promising approach to support large-scale Web-based testing is parallel test paper generation (k-TPG), which generates k *similarly optimal* test papers automatically according to a user specification based on multiple assessment criteria. Specifically, it aims to find k optimal disjoint subsets of questions from a question database to form different test papers according to a user specification based on total time, topic distribution, difficulty degree, discrimination degree, and so on. The generated test papers can then be attempted over the Web by students for assessment purposes.

k-TPG is a challenging problem especially with large number of questions or large number of generated test papers (i.e. large k) due to its NP-hardness [2], [3]. Manually browsing and composing test papers by users is ineffective because of the exponential number of feasible combinations of questions. In the past few years, heuristic-based intelligent techniques such as Tabu Search [4], Particle Swarm Optimization [5] and Ant Colony Optimization [6] have been proposed for k-TPG. However, the quality of generated parallel test papers is often unsatisfactory [4], [5], [6] according to users' test paper specifications. One of the main issues of the current techniques is that they are ineffective to optimize the biobjective functions simultaneously in the very large search space of possible candidates with multicriteria constraints. Although these heuristic-based techniques are straightforward to implement, they suffer from some drawbacks. They are mainly based on traditional vector scalarization method, which uses weighting parameters to reduce multiobjective optimization to single objective setting. Due to the multiobjective optimization [7], [8], [9], [10], of test paper generation, using either pre-determined weighting parameters and fixed number of iterations, tends to get stuck in a local optimal solution and lose convergence [11] especially in a huge search space of large-sized question datasets.

Different from other multiobjective optimization problems, the *collective objective functions* of k-TPG are defined based on the evaluation objectives of k generated test papers instead of a single test paper. In fact, k-TPG is close to the spirit of optimal distributed allocation problems [12] with collective objective functions. As such, it is not easy to apply well-known algorithms for solving k-TPG because they are ineffective for optimizing the collective objective functions. Formally, k-TPG is a biobjective optimal distributed resource allocation problem, which aims to simultaneously maximize two objective functions under a multidimensional Knapsack constraint [13]. The first objective can be formulated as a Welfare Allocation problem [2], which aims to maximize the total quality of the generated test papers. The second objective can be formulated as a Fairness Allocation problem [3], [14], which aims to maximize the fairness quality of the generated test papers. Traditionally, optimizing these two objectives is NP-hard and considered separately. To the best of our knowledge, there has been no attempt at finding an effectively near-optimal solution to both of the objectives simultaneously.

To cope with the NP-hardness of distributed resource allo-

[1]http://www.khanacademy.org/
[2]https://www.coursera.org/
[3]http://www.udacity.com/

cation problems, submodular-based approximation algorithms, which are polynomial time algorithms that have provable guarantees on the solution quality, have been studied for distributed optimization problems [15]. Inspired by this, we propose a Submodular Memetic Approximation (SMA) algorithm for k-TPG. The key idea of SMA is twofold. Firstly, we analyze the properties of the two collective objectives and reformulate the k-TPG problem such that it can be solved effectively by memetic algorithm. Secondly, we employ submodular optimization techniques to design greedy-based approximation algorithm for enhancing steps of multiobjective memetic algorithm. Synergizing the intensification of submodular local search mechanism with the diversification of the population-based submodular crossover operator, our SMA algorithm can jointly optimize the *total quality maximization* objective and the *fairness quality maximization* objective. In particular, SMA is a deterministic greedy-based approximation algorithm, which can achieve near-optimal parallel test papers for large question datasets in efficient polynomial runtime.

II. Related Work

A. Parallel Test Paper Generation

Although there is other research on special-case single test paper generation (1-TPG) [4], [5], [6], [16], this section focuses on work related to k-TPG.

In [4], a Tabu Search (TS) approach was proposed to solve the k-TPG problem, which considers only the fairness maximization objective under multicriteria constraints. To solve the problem, the objective function is formulated to minimize the maximal mutual difference of the average discrimination degree of two arbitrary test papers: minimize $\max_{\forall 1\leq i\leq j\leq k}|f(P_i)-f(P_j)|$, where $f(P_i), i = 1..k$ is the average discrimination degree of test paper P_i. Although this formulation seems reasonable, it does not have any effective method for optimization. Thus, the TS approach is purely heuristic search. As a result, the quality of the generated test papers is poor and the computational cost is very high.

In [5], Ho et al. proposed a Partical Swarm Optimization (PSO) approach to solve the same k-TPG problem as in [4] with a similar objective function. Performance results showed that this approach outperforms the Genetic Algorithm algorithm with different values of $k = 2, 3, 4$.

In [6], Ant Colony Optimization (ACO) was proposed for multiple test paper generation. Different from [5], this work considered both total quality and fairness quality maximization. The objective function is formulated as follows: maxmize $\sum_{l=1}^{k} f(P_l) - \sum_{\forall 1\leq i\leq j\leq k}|f(P_i) - f(P_j)|$, where $f(P_i), i = 1..k$ is the average discrimination degree of test paper P_i with four predetermined weighting parameters for multicriteria constraint satisfaction. There are two possible issues with this formulation. Firstly, the number of weighting parameters is $4k$, which makes it difficult and time consuming to determine. Secondly, similar to [4], [5], this objective function is not easy to optimize as no special property was exploited. ACO also generates quality papers by optimizing an objective function. It optimizes the test paper quality by simulating the foraging behavior of real ants. Test papers are considered as routes which are constructed by m ants. At each iteration, m ants are dispatched for stochastically constructing their own solutions to improve the objective function.

B. Submodular Function Optimization

In this section, we review some fundamental concepts of submodular functions [17] in combinatorial optimization.

Definition 1 (Discrete Derivative)**.** Given a set X, a non-negative function $f : 2^X \to \mathbb{R}^+$, $S \subseteq X$, and $x \in X$. Let $\frac{\partial f(S)}{\partial x} = \triangle_f(x|S) = f(S \cup \{x\}) - f(S)$ be the *discrete derivative* of f at S with respect to x.

The discrete derivative is also called the *marginal value* of the set function $f(S) : \{0, 1\}^n \to \mathbb{R}^+$ at element x.

Definition 2 (Submodularity)**.** Given a set X, a non-negative function $f : 2^X \to \mathbb{R}^+$ is *submodular* if for every set $S, T \subseteq X$,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$$

Equivalently, a function $f : 2^X \to \mathbb{R}^+$ is *submodular* if for every set $A \subseteq B \subseteq X$ and $x \in X \setminus B$,

$$\triangle_f(x|A) \geq \triangle_f(x|B)$$

In addition, a submodular function is monotone if $f(S) \leq f(T), S \subseteq T$. The second condition in Definition 2 can be deduced from the first condition by substituting $S = A \cup \{x\}$ and $T = B$, where $A \subseteq B \subseteq X$ and $x \in X \setminus B$.

Submodular function optimization has been extensively studied over the last 30 years since the preliminary work by L. Lovasz [17]. Generally, there exist efficient polynomial algorithms for minimizing a submodular function [18], [19], [20], [21]. However, maximizing a submodular function is known to be NP-hard [22]. Although NP-hard, the decreasing marginal property has led to the existence of a general Greedy-based Approximation Algorithm [23], [24] for the maximization problem. It can achieve a good approximation ratio of $(1 - \frac{1}{e}) \approx 0.63$. This is known to be the best achievable ratio found in recent literature. Submodular functions have many useful basic properties such as linearity, linear combination and truncation [18].

C. Memetic Algorithms

MAs have been widely used for solving various real-world applications such as scheduling and planning [25], [26], [27]. MAs are classified into three categories: simple hybrid, adaptive hybrid, and memetic automation [28], [29]. 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. Simple and adaptive hybrids focus more on the learning process of MA while memetic automation focuses more on the evolutionary computation, which is designed specifically for complex problem-solving. Recently, there have been increasing interest in investigating simple hybrid and adaptive hybrid for tackling multiobjective optimization problems [30]. In simple hybrid, special population-based methods [31], [32], [33] or individual

improvement methods [33] are designed to deal with multiobjective optimization. In adaptive hybrid, some adaptive coordinations of individual improvement methods [34], [35] are proposed for handling multiobjective optimization.

In [36], we proposed a memetic algorithm for solving 1-TPG problem. However, this algorithm is unable to solve the two collective objectives of k-TPG effectively. In this research, we propose a novel deterministic greedy-based memetic approximation algorithm, called SMA that adopts a hybridization of simple hybrid and adaptive hybrid [7], and submodular optimization for k-TPG. Based on submodular property of objective functions, submodular local search and submodular crossover operator of SMA are able to jointly optimize the two collective objectives effectively and efficiently.

III. Problem Specification for k-TPG

A. Question Dataset

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

- *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 [6] or manually by human experts. Table I shows a sample Math question dataset.

TABLE I
AN EXAMPLE MATH DATASET

(a) Question Table

$\mathcal{Q}$_ID	o	a	e	t	d	c	y
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_3
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_3

(b) Topic Table

$\mathcal{C}$	$name$
c_1	Integration
c_2	Differentiation

(c) Question Type Table

$\mathcal{Y}$	$name$
y_1	Multiple choice
y_2	Fill-in-the-blank
y_2	Long Question

B. 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* $C = \{(c_1, pc_1), (c_2, pc_2), .., (c_M, pc_M)\}$*:* 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* $Y = \{(y_1, py_1), (y_2, py_2), .., (y_K, py_K)\}$*:* 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.

C. Parallel Test Paper Generation

In 1-TPG, it selects an optimal subset of questions to form a test paper P, which maximizes the average discrimination degree $f(P) = \frac{1}{N}\sum_{i=1}^{N} e_i, q_i \in P$ while satisfying multiple user-specified criteria in a test paper specification $\mathcal{S}_P \approx \mathcal{S}$.

In parallel test paper generation (k-TPG), user specification is given as a pair of $\langle k, \mathcal{S} \rangle$, where k is an integer parameter in addition to the test paper specification $\mathcal{S}$. The k-TPG aims to select k disjoint subsets of questions $P_1, P_2, .., P_k$ from a question dataset $\mathcal{Q} = \{q_1, q_2, .., q_n\}$ to form k test papers with specification $\mathcal{S}_{P_i}$, which satisfies the test paper specification such that $\mathcal{S}_{P_i} \approx \mathcal{S}$, $\forall i \in [1, k]$. Apart from maximizing the objective function as in single test paper generation, k-TPG aims to generate the k test papers with similar optimal quality according to the specification. It aims to provide similar quality test papers for different users. As such, there are two important *collective objective functions* that need to be maximized in k-TPG: *total quality maximization* and *fairness quality maximization*. Especially, these objective functions are defined over the set of k generated test papers instead of each individual one as in single test paper generation.

Total Quality Maximization: It aims to maximize the sum of discrimination degrees of k test papers $\sum_{i=1}^{k} f(P_i)$, where $f(P_i)$ is quality objective of test paper $P_i, i \in [1, k]$ based on its question discrimination degrees. Formally, the objective is stated as follows:

$$\underset{P_1,..,P_k}{\text{maximize}} \sum_{i=1}^{k} f(P_i)$$

Fairness Quality Maximization: It aims to maintain the fairness among the generated test papers with equivalent discrimination degrees. However, it is not sufficient to solely

maximize the total quality objective function to achieve this purpose as it is possible that some of the generated test papers may have very poor discrimination degrees. Most of the previous studies have formulated the k-TPG problem in an ineffective manner that makes it difficult to optimize the formulated objective function [4], [5], [6] due to using either pre-determined weighting parameters and fixed number of iterations. Instead, we can optimize for a fair allocation of the discrimination degrees of k generated test papers as follows:

$$\underset{P_1,..,P_k}{\text{maximize}} \min_{1\leq i\leq k} f(P_i)$$

where fairness quality of the k test papers $P_1,..,P_k$ is defined as $\min_{1\leq i\leq k} f(P_i)$. In fact, this way of formulation is more effective for fairness optimization [14], [37].

Joint Total and Fairness Quality Maximization: Both of the two collective objectives have been studied individually in the past. Instead of optimizing these two problems separately, we can jointly optimize them. The k-TPG problem aims to jointly maximize the total quality objective function:

$$\underset{P_1,..,P_k}{\text{maximize}} \quad \sum_{i=1}^{k} f(P_i) \tag{1}$$

and the fairness quality objective function:

$$\underset{P_1,..,P_k}{\text{maximize}} \min_{1\leq i\leq k} f(P_i) \tag{2}$$

and optimizing multi-objective constraint such that each specification on $\langle N, T, D, C, Y\rangle$ of the generated test paper $\mathcal{S}_{P_i}$ is equivalent to that of $\mathcal{S}$ and subject disjoint constraint: $P_i \cap P_j = \emptyset, \forall i \neq j$.

We note that constraint includes: content hard-constraint (topic and question types) and assessment soft-constraint (total time and discrimination degree constraint). Due to practical purpose of k-TPG, we only consider one best returned set of test papers.

IV. Proposed SMA Approach

A. Problem Reformulation and Intuition

In this section, we further analyze the special properties of k-TPG on the two objective functions. As discussed, it is challenging to simultaneously optimize both of the objective functions. By exploiting the relationship between the two objective functions based on the submodular property, we can devise an effective and efficient multiobjective optimization approach. We observe that the quality function $f(P)$ is submodular due to the linearity property [24].

Lemma 1. Given a test paper P generated from a question set $\mathcal{Q}$ ($P \subseteq \mathcal{Q}$), the quality evaluation function of discrimination degree $f(P) : 2^{\mathcal{Q}} \rightarrow \mathbb{R}^+$ is submodular and monotone.

Hence, the total quality objective is also submodular due to a linear combination of k submodular functions [24].

Corollary 1. The total quality objective function of k test papers $\sum_{i=1}^{k} f(P_i), P_i \subseteq \mathcal{Q}$ is submodular and monotone.

Unfortunately, we note that optimizing the total quality objective alone is not sufficiently good to ensure for fairness quality maximization. To overcome, we take the advantage of the submodularity to reformulate the total quality objective such that it also integrates the fairness quality objective.

Let $f_\phi(P) = \min\{f(P), \phi\}$ be a truncated function, where ϕ is a non-negative constant. From Lemma 1, $f_\phi(P)$ is also submodular and monotone due to [24]. For any constant ϕ, we have the following important observation:

$$\min_{l\in\{1,..,k\}} f(P_l) \geq \phi \iff \sum_{l=1}^{k} f_\phi(P_l) = k\phi \tag{3}$$

Note that k is a constant in the user specification. Hence, this means that the fairness quality objective value is larger than or equal to ϕ if the total quality objective value is $k\phi$ and vice versa. More importantly, we have the following result:

Definition 3 (Relaxed k-TPG). Let ρ be the optimal total quality value and α be the approximation ratio of a near-optimal solution for the total quality, i.e., $\sum f(P_i) \geq \alpha\rho$. Assume that the optimal fairness value is ϕ^*, i.e., $\max_{P_1,..,P_k} \min_l f(P_l) = \phi^*$, we define a relaxed problem of the original k-TPG by maximizing the fairness quality under a total quality constraint:

$$\begin{aligned} &\underset{P_1,..,P_k}{\text{maximize}} \min_{1\leq i\leq k} f(P_i) \\ &\text{s.t. } \sum f(P_i) \geq \alpha\rho, P_i \bigcap_{\forall i\neq j} P_j = \emptyset \end{aligned} \tag{4}$$

Theorem 1. Solving the relaxed k-TPG problem is equivalent to solving the following problem:

$$\begin{aligned} &\max_{P_1,..,P_k} \sum_{l=1}^{k} f_{\phi^*}(P_l) \\ &\text{s.t. } \sum f(P_i) \geq \alpha\rho, P_i \bigcap_{\forall i\neq j} P_j = \emptyset \end{aligned} \tag{5}$$

Proof: The fairness quality objective (4) of the relaxed k-TPG problem is equivalent to the problem given in (5) since, by (3), there exists k test papers $P_1,..,P_k$ such that $\sum_{l=1}^{k} f_{\phi^*}(P_l) = k\phi^*$. In addition, because of the property of the truncated function $f_{\phi^*}(P_l)$, we have $\max \sum_{l=1}^{k} f_{\phi^*}(P_l) \leq k\phi^*$. Thus, for any optimal generated papers $P_1,..,P_k$ of the problem in (5), it must satisfy that $\sum_{l=1}^{k} f_{\phi^*}(P_l) = k\phi^*$. Hence, we have $\min_l f(P_l^*) = \phi^*$ due to (3).

The relaxed k-TPG problem scarifies somewhat on the optimality of the total quality objective in order to achieve a near-optimal fairness quality objective. In addition, the total quality constraint $\sum f(P_i) \geq \alpha\rho$ is satisfied implicitly (in Section IV-D) by modifying an algorithm, which always produces near-optimal total quality value of at least $\alpha\rho$, in order to also maximize the reformulated objective in (5). Generally we do not know the optimal value ϕ^*. Therefore, we need to find it using the binary search strategy, starting with the interval $[\phi_{min}, \phi_{max}]$.

B. Overview of SMA

Based on above intuition, we observe that reformulation problem (4) can be solved effectively by the local search and evolutionary computation of memetic algorithm and submodular optimization technique. Thus, we propose novel Submodular Memetic Approximation (SMA) algorithm for k-TPG.

The proposed approach will generate k test papers progressively by using the approximation algorithm for 1-TPG and adjusting the fairness value ϕ. Due to the NP-hardness of 1-TPG, when ϕ approaches its optimal value, we can only

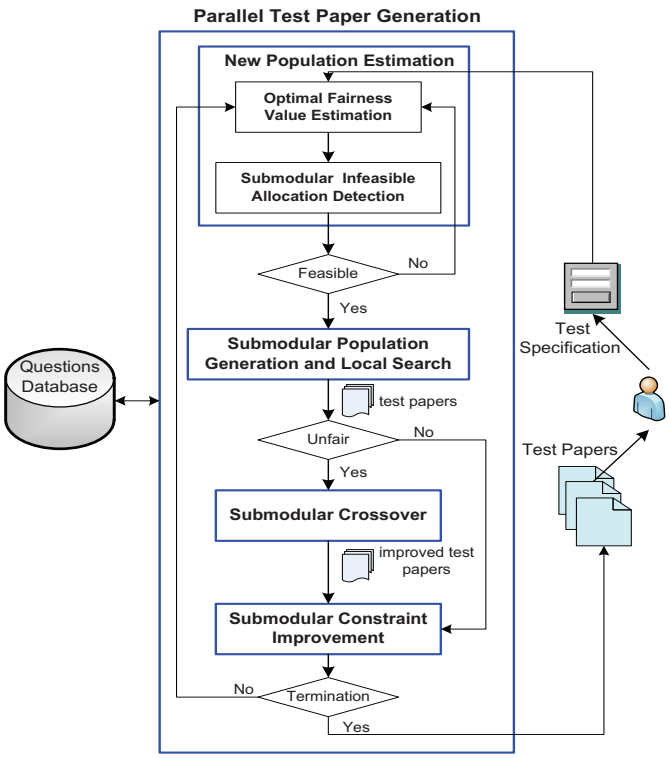


Fig. 1. Proposed SMA Approach

achieve a fraction $\beta \leq 1$ of the optimal objective value of ϕ, where β is a constant that will be determined in Section IV-E. As such, our goal is to allocate questions into k test papers such that for all k papers, we have $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$. Figure 1 shows 4 main steps of the proposed SMA approach.

- *New Population Estimation:* It uses binary search to estimate the optimal fairness value ϕ of the next evolutionary optimization process. Then, it detects whether it is possible to generate k papers such that $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$ for early infeasible allocation detection.
- *Submodular Population Generation and Local Search:* It progressively generates k papers for the current evolutionary generation by using a greedy-based algorithm.
- *Submodular Crossover Operator:* It swaps questions of the k generated test papers by a novel crossover operator for improving fairness allocation if there exists a test paper P_l such that $f_\phi(P_l) \leq \beta\phi$.
- *Submodular Constraint Improvement:* It optimizes individually the local constraint satisfaction of all test papers.

C. New Population Estimation

1) Optimal Fairness Value Estimation: In each evolutionary generation, a new set of k papers are generated according to a new fairness value ϕ. This step aims to find a new optimal value of ϕ to continue the co-optimization process. We use binary search strategy with the interval $[\phi_{min}, \phi_{max}] = [0, N \times \max_{q \in \mathcal{Q}} f(q)]$. In each step, we test the center $\phi = (\phi_{min} + \phi_{max})/2$ of the current interval $[\phi_{min}, \phi_{max}]$ of possible fairness value. Specifically, it goes to the Collective Optimization of Total Quality step to check whether it is possible to generate k satisfied papers such that $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$. If it is possible, the optimization process goes to the next Competitive Optimization of Test Paper Quality step. The value of ϕ is increased later on. Otherwise, the value of ϕ is decreased. This search process will terminate after at most $\lceil \log_2(N \times \max_{q \in \mathcal{Q}} f(q)) \rceil$ steps.

2) Submodular Infeasible Allocation Detection: This step aims to detect whether it is possible to generate a new population of k papers such that $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$. It checks whether the estimated value of ϕ is appropriate or not. We define a test paper instance in the population.

Definition 4 (Test Paper Representation). Given a specification $\mathcal{S}$, a *single test paper* P is encoded by a subset of question items $P = (q_1, q_2, ..., q_N), q_i \in \mathcal{Q}; 1 \leq i \leq N$. The quality of P is evaluated by an objective function $f(P)$, which is the average of the discrimination degree of its questions. A *parallel test paper* solution is a set of k disjoint single test paper $P_1, P_2, ..., P_k$.

Our proposed algorithm for this step is a greedy-based approximation algorithm that generalizes the basic greedy-based approximation algorithm for submodular function. In this step, we consider only the linear constraints (4) to simplify the checking process. As such, the result obtained is the upper bound of the actual approximate solution. However, it is sufficient for the early infeasible allocation detection purpose. This algorithm is outlined in Algorithm 1.

Recall that each of the k papers $P_1, P_2, .., P_k$ has N questions. Hence, the main loop consists of $k * N$ iterations (in Line 2). In each iteration, we first check that whether the question q is possibly allocate to a paper P_t. If yes, we then compute the marginal improvement value $\Psi_{t,q}$ of allocating question q into P_t. Next, we greedily choose the best allocation of test paper and question (P_{t^*}, q^*) with the maximum $\Psi_{t,q}$ value. To verify whether q is possibly allocate into test paper P_t, we first check whether P_t is full, i.e., $|P_t| = N$. Then, we check whether allocating q to P_t will violate the constraints.

We provide a theoretical result of this step. For general cases, we prove that it achieves an approximation ratio of $\frac{1}{2}$.

Theorem 2. Consider only the content constraints for the general cases, i.e., $k \geq 2$, the Collective Optimization of Total Quality step can obtain a set of generated test papers $P_1, .., P_k$ with the total quality value (in Line 7) such that:

$$\sum_{i=1}^{k} f_\phi(P_i) \geq \frac{1}{2} \max_{P'_1,..,P'_k} \sum_{i=1}^{k} f_\phi(P'_i) = \frac{1}{2} OPT$$

Proof: Let n be the number of questions in the dataset $\mathcal{Q}$. Let H be the original problem of allocating $k * N$ out of n questions to k papers $P_1, .., P_k$ such that it maximizes the total quality objective $\sum_{i=1}^{k} f_\phi(P_i)$, where N is the number of questions in each paper. Let H' be the problem on the $n-1$ remaining questions after the first question q_t is selected for paper P_j, i.e., the first question is unavailable for further selection. On the problem H', the quality evaluation function $f_j(P_j) = f_\phi(P_j)$ is replaced by $f'_j(P_j) = f_j(P_j \cup \{q_t\}) - f_j(\{q_t\})$. All other quality evaluation function $f_i(P_i)$, $i \neq j$, are unchanged. Note that Algorithm 1 can be considered as first allocating question q_t to paper P_j and then allocating

Algorithm 1: Infeasible_Allocation_Detection

Input: k - number of test papers; $\mathcal{S} = (N, T, D, C, Y)$ - test paper specification; ϕ - fairness
Output: **Yes** - if $\sum_{l=1}^{k} f_\phi(P_l) \leq \frac{1}{2}k\phi$; **No** - if otherwise
begin

1 $P_l \leftarrow \emptyset$ for all $l = 1..k$;
2 **for** *i=1 to k*N* **do**
  foreach *question* $q \in \mathcal{Q} \setminus (P_1 \bigcup ... \bigcup P_k)$ ***and*** $l \in \{1..k\}$ **do**
3 **if** q *satisfies the cardinality constraint and content constraints* P_l **then**
4 Compute the marginal improvement value: $\Psi_{l,q} \leftarrow w(P_l + \{q\}) - w(P_l)$;
5 $(l^*, q^*) \leftarrow \text{argmax}_{(l,q)} \Psi_{l,q}$;
6 $P_{l^*} \leftarrow P_{t^*} \bigcup \{q^*\}$;
7 $\xi = \sum_{l=1}^{k} f_\phi(P_l)$;
8 **if** $\xi < \frac{1}{2}k\phi$ **then return** ***Yes*** ;
  else return ***No*** ;

the other questions using a recursive call on H' until $k * N$ questions have been allocated.

Let $VAL(H)$ be the value of the allocation produced by Algorithm 1, $OPT(H)$ be the value of the actual optimal allocation. Let $p = f_j(\{q_t\})$. By definition of H', it is clear that $VAL(H) = VAL(H') + p$. We will show that $OPT(H) \leq OPT(H')+2p$. Let $\mathcal{P} = P_1, .., P_k$ be the optimal allocation for H and assume that $q_t \in P_i$, i.e., question q_t is allocated to paper P_i. Let $\mathcal{P}' = P'_1, .., P'_k$ be the allocation of questions $2^{th}, .., k * N^{th}$ that is similar to $\mathcal{P}$. This is a possible solution to H'. Let's compute its value by comparing it to $OPT(H)$. All test papers except P_i get the same allocation and all test papers except P_j have the same quality evaluation function. Without loss of generality, we may assume that $i \neq j$. We see that paper P_i loses at most $f_i(\{q_t\})$ since f_i is submodular. But $f_i(\{q_t\}) \leq f_j(\{q_t\}) = p$ and test paper P_i loses at most p. Test paper P_j loses at most p since by monotonicity of f_j, $f'_j = f_j(P_j \cup \{q_t\}) - f_j(\{q_t\}) \geq f_j(P_j) - p$. Therefore, $OPT(H') \geq OPT(H) - 2p$. This proof is completed by induction on H' since H' is also a submodular function:

$$OPT(H) \leq OPT(H) + 2p \leq 2VAL(H') + 2p = 2VAL(H)$$

Theorem 2 implicates that Algorithm 1 achieves a 1/2-approximation ratio for the total quality maximization objective based on a given fairness value ϕ. In other words, the obtained total quality value ξ is greater than or equal to 0.5 times of the actual optimal value. Due to the property of the truncated function, if the current fairness value ϕ is smaller than the near-optimal obtainable value $\phi*$, the obtained total quality value must be at least $\frac{1}{2}k\phi$. In addition, let ξ' be the obtained total quality value of Algorithm 1 by considering both content constraints and assessment constraints. As discussed before, we have $\xi' \leq \xi$. Therefore, if we know that $\xi < \frac{1}{2}k\phi$, then we have $\xi' < \frac{1}{2}k\phi$. This means that the current fairness value ϕ is larger than the optimal fairness $\phi*$. Therefore, it is impossible to allocate k generated test papers such that the fairness requirement $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$, is satisfied.

D. Submodular Population Generation and Local Search

This step aims to solve progressively k problem instances of 1-TPG to generate a new population of k papers $P_1, P_2, .., P_k$. Based on submodular property of $f_\phi(P)$, we greedily select questions that maximize the objective functions while paying attention to satisfy the multiple knapsack constraints. Our motivation is based on some applications, which maximize a submodular function under a knapsack constraint as follows:

$$\max_S f(S) \text{ s.t. } \sum cx \leq b, x \in S$$

where $c(x) \geq 0$ is a cost function and $b \in \mathbb{R}$ is a budget constraint. Sviridenko [38] proposed a greedy-based algorithm for solving this problem with an approximation ratio of $1 - \frac{1}{e}$. It defines the marginal gain $\frac{\Delta_f(x|S)}{c(x)}$ when adding an item x into the current solution S as a ratio between the discrete derivative $\Delta_f(x|S)$ and the cost $c(x)$. This algorithm starts with $S = \emptyset$, and then iteratively adds the element x that maximizes the marginal gain ratio among all elements that satisfy the remaining budget constraint:

$$S_{i+1} = S_i \cup \{ \underset{x \in V \setminus S_i : c(x) \leq b - c(S_i)}{\arg\max} \frac{\Delta(x|S_i)}{c(x)} \}$$

We extend the algorithm of submodular function maximization under a knapsack constraint for solving the case of multiple knapsack constraints. In [36], the 1-TPG problem can be reformulated as follows:

$$\max_P f(P) \text{ s.t. } \sum Aq \leq b, q \in P \subseteq \mathcal{Q}$$

where A is a matrix and b is the vector of multiple knapsack constraints. Algorithm 2 gives the greedy-based algorithm for this step. This algorithm maintains a set of weights that are incrementally adjusted in a multiplicative manner. These weights capture the possibility that each constraint is closed to be violated when maximizing the objective function of a paper. In each iteration, the algorithm selects an available question that maximizes the sum of marginal gain ratio normalized by the weights as follows:

$$P_l^{i+1} = P_l^i \cup \{ \underset{q_j \in \mathcal{Q}}{\arg\max} \sum_{i=1}^{m} \frac{\Delta_{f_\phi}(q_j|P_l)}{A_{ij}h_i} \}$$

The above algorithm returns a set of generated papers, which are feasible according to the multiple knapsack constraints. Recall from Definition 1 that $\Delta_{f_\phi}(q_j|P_l)$ is the incremental marginal value of question q_j to the paper P_l.

Lemma 2. Algorithm 2 can attain a set of k generated test papers, which are feasible according to multiple knapsack constraints after exactly $k * N * n$ iterations, where N is the number of questions of each paper in the user specification.

Proof: Let $P_l, l = 1..k$, be a generated paper when the loop in Line 4 terminates. It is obvious that the loop in Line 4 will terminate after exactly N main iterations due to the monotone submodular property of $f_\phi(P_l)$.

Without loss of generality, we assume that the matrix $A \in [0, 1]^{m \times n}$ and the vector $b \in [1, \infty)^m$. Let q_v be the first

Algorithm 2: Submodular_Population_Generation

Input: $\mathbf{A}q \leq \mathbf{b}, q \in \{0,1\}^n$ - 0-1 ILP problem;
$A = [a_{ij}]_{m \times n}$; $b \in \mathbb{R}^m$; μ - a parameter
Output: $P_1, P_2, .., P_k$ - test papers
begin
1 **for** *l=1 to k* **do**
2 $P_l \leftarrow \emptyset$;
3 **for** *i=1 to m* **do** $h_i = 1/b_i$;
4 **while** $\sum_{i=1}^{m} b_i h_i \leq \mu$ *and* $|P_l| \leq N$ **do**
5 $q_j \leftarrow \arg\max_{q_j \in \mathcal{Q}} \sum_{i=1}^{m} \frac{\Delta_{f_\phi}(q_j|P_l)}{A_{ij} h_i}$;
6 **if** *q satisfies the cardinality constraint and content constraints* P_l **then**
7 $P_l \leftarrow P_l \cup \{q_j\}$;
8 **for** *i=1 to m* **do** $h_i \leftarrow h_i \mu^{\frac{A_{ij}}{b_i}}$;
9 $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{q_j\}$;
10 **return** $P_1, P_2, .., P_k$

Algorithm 3: Submodular_Crossover_Operator

Input: $P_1, P_2, .., P_k$ - papers with unfair allocation
Output: $P'_1, P'_2, .., P'_k$ - papers with fair allocation
begin
1 Sort $P_i, i = 1..k$ according to $f(P_i)$ decreasingly;
while *true* **do**
2 Select best test paper P_i ;
3 Select worst test paper P_j;
4 **if** $f_\phi(P_j) < \beta\phi$ *and* $f_\phi(P_i) \geq 3\beta\phi$ **then**
5 Select swapping questions from P_i: $\Lambda_i = \{q_1^i, q_2^i, ...q_t^i\}$ and P_j: $\Lambda_j = \{q_1^j, q_2^j, ...q_t^j\}$;
6 **foreach** $q^i \in \Lambda_i$ **do**
7 $P_j \leftarrow \{P_j \setminus \{q_j\}\} \cup \{q_i\}$;
8 $P_i \leftarrow \{P_i \setminus \{q_i\}\} \cup \{q_j\}$;
9 Re-sort P_i according to $f(P_i)$ decreasingly;
10 **return** $P_1, P_2, .., P_k$

question that induces a violation in some constraints. Suppose q_v induces a violation in constraint i at iteration $t < N$. In other words, we have $\sum_{q \in P_t} > b_i$, and therefore,

$$b_i h_{it} = b_i h_{i0} \prod_{q \in P_l} \mu^{\frac{A_{ij}}{b_i}} = \mu^{\sum_{q \in P_l} \frac{A_{ij}}{b_i}} > \mu$$

where the last inequality is formed because $h_{i0} = 1/b_i$. As a result, we have $\sum_{i=1}^{m} b_i h_i > \mu$. This implies a contradiction with the loop condition specified in Line 4.

The parameter μ in Algorithm 2 is important as it ensures that multiple knapsack constraints will not be violated while maximizing the submodular objective function. Let $H = \min\{b_i/A_{ij} : A_{ij} > 0\}$ be the width of the knapsack constraints. By experiments, we found that Algorithm 2 can achieve near-optimal results by setting the parameter $\mu = e^H m$, where e is the natural exponential and m is the number of knapsack constraints.

E. Submodular Crossover Operator

This step devises a novel crossover operator to optimize the fairness quality objective for the entire population. Recall that our goal is to allocate questions into k test papers such that for all papers, we have $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$. However, this is not easy to achieve especially when ϕ approaches the optimal value. This is because in the small and less abundant pool of questions, Algorithm 2 greedily selects all questions with high value $f_\phi(q)$ for some of the generated papers $P_1, P_2, .., P_s$ to satisfy $f_\phi(P_l) \geq \beta\phi, \forall l = 1..s$. As a result, subsequent paper $P_{s+1}, .., P_k$ might not have enough good questions to satisfy this requirement. In this case, we need to swap some of the questions from the satisfied papers with questions from unsatisfied papers to achieve a fair allocation.

We will move questions from satisfied papers to unsatisfied papers and vice versa, until all papers satisfy the requirement $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$. To do that, we sort all test paper P_i according to their paper quality $f(P_i)$ in decreasing order. Let's define the *swapping operation* as follows. Select a best satisfied paper $P_i = \{q_1^i, q_2^i, ...q_N^i\}$ for which $f_\phi(P_i) \geq 3\beta\phi$. Such a paper is always ensured through appropriate choice of α and β as shown in Lemma 4. Then, we select a worst unsatisfied paper P_j, i.e., $f_\phi(P_j) \leq \beta\phi$. In the paper P_i, choose $t < N$ such that $f_\phi(\{q_1^i, q_2^i, ...q_{t-1}^i\}) < \beta\phi$, $f_\phi(\{q_1^i, q_2^i, ...q_t^i\}) \geq \beta\phi$, and $f_\phi(\{q_t^i\}) < \beta\phi$. Let $\Lambda_i = \{q_1^i, q_2^i, ...q_t^i\}$. As $f_\phi(\emptyset) = 0$, the set Λ_i is not empty. In the reversed direction, let $\Lambda_j = \{q_1^j, q_2^j, ...q_t^j\}$ be a set of questions in P_j such that each pair of questions $q_l^i, l = 1..t$, and $q_l^j, l = 1..t$, has the same topic and question type. This is done because both test papers P_i and P_j satisfy the hard constraints on topic and question types. Therefore, in crossover operator, we can always find such matching parts, which have the same topic and question type, to do the swapping We reallocate the questions of papers P_i and P_j by swapping questions of the two sets Λ_i and Λ_j. We note that the swapping operation does not violate the multiple knapsack constraints. Thus, all newly generated papers remain feasible. Finally, we re-sort the entire test papers after swapping. More importantly, this operation improves the fairness allocation among all papers. The swapping operation is given in Algorithm 3.

Lemma 3. The swapping operation improves the fairness allocation of test papers P_i and P_j as follows:
$f_\phi((P_j \setminus \Lambda_j) \cup \Lambda_i) \geq \beta\phi$, and $f_\phi((P_i \setminus \Lambda_i) \cup \Lambda_j) \geq f_\phi(P_i) - 2\beta\phi$

Proof: Due to the monotone property of f_ϕ, we have $f_\phi(P_j \cup \Lambda_i) \geq f_\phi(\Lambda_i) \geq \beta\phi$. It remains to prove that $f_\phi((P_i \setminus \Lambda_i) \cup \Lambda_j) \geq f_\phi(P_i) - 2\beta\phi$. Suppose that $f_\phi(P_i - \Lambda_i) < f_\phi(P_i) - 2\beta\phi$. Let $P'_i = P_i - \Lambda_i$. Then, we have

$$f_\phi(P'_i \cup \Lambda_i) - f_\phi(P'_i) > 2\beta\phi$$

But $f_\phi(\Lambda_i) \leq f_\phi(\{q_1^i, q_2^i, ...q_{t-1}^i\}) + f_\phi(\{q_t^i\}) < 2\beta\phi$
due to submodularity of f_ϕ and the fact that $f_\phi(\{q_t^i\}) < \beta\phi$. Hence, adding Λ_i to P'_i increases more value than adding Λ_i to an empty set. This contradicts the submodularity of f_ϕ.

Therefore, swapping questions will not reduce the value of $f_\phi(P_i)$ greater than $2\beta\phi$. Hence, P_i is still satisfied. Moreover, the unsatisfied test paper now becomes satisfied by this operation. We also guarantee that we can always perform these swapping operations, until all test papers are satisfied. This issue is solved by choosing $\beta = \alpha/3$, where $\alpha = 1/2$ is the approximation ratio of the Algorithm 1 in Section IV-C2 for the total quality maximization.

Lemma 4. If we choose $\beta = \alpha/3 = 1/6$, it is guaranteed that after at most k swapping operations, all test papers will be satisfied, i.e., $f_\phi(P_l) \geq \beta\phi, \forall l = 1..k$.

Proof: To simplify the notation, without loss of generality, let's assume that $\phi = 1$. Since the optimal fairness quality for $f_\phi = f_1$ is 1, the optimal total quality for f_1 is about k. Algorithm 2 obtains an allocation $\mathbf{P}$ of k test papers that is a fraction α of the optimal. Hence, we have $\sum_{i=1}^{k} f_1(P_i) \geq \alpha k$. Recall that the sum $\sum_{i=1}^{k} f_1(P_i)$ is the total quality of $\mathbf{P}$. We need to estimate the maximal number of unsatisfied papers that can have. Let θ be the fraction of unsatisfied papers.

We have: $k\theta\beta + k(1-\theta) \geq \alpha k$. Since the maximum value θ is obtained if all the papers are satisfied, i.e., having the total quality of about $k(1-\theta)$, and the unsatisfied papers are obtained without being satisfied, i.e., having the total quality of about $k\theta\beta$. Hence, we have: $\theta \leq \frac{1-\alpha}{1-\beta}$.

Let's consider the total quality Φ, which is allocated over the satisfied papers. We know that Φ is at least:

$$\Phi \geq \alpha k - \theta k\beta \geq k\frac{\alpha(1-\beta) - \beta(1-\alpha)}{1-\beta} \qquad (7)$$

The first question swapping is possible if: $\frac{\Phi}{k(1-\Phi)} \geq 3\beta$. since if the average remaining total quality over all $(1-\theta)k$ satisfied papers is 3β, then there must be at least one paper such that question swapping can be executed. Since each swap decreases the total quality Φ by no more 2β as previously proved, and since θk swaps is needed to satisfy all unsatisfied papers, it is sufficient to constrain that

$$\frac{\Phi - 2\theta k\beta}{k(1-\theta)} \geq 3\beta \Rightarrow \quad \Phi \geq 3\beta k - \beta\theta k \qquad (8)$$

From (7) and (8), a sufficient condition for β such that enough moves can be carried out to satisfy all unsatisfied papers is

$$\frac{\alpha(1-\beta) - \beta(1-\alpha)}{1-\beta} \geq 3\beta - \beta\frac{1-\alpha}{1-\beta} \Rightarrow \quad \beta \leq \alpha/3$$

by solving this inequality for β and removing the infeasible solution $\beta \geq 1$. Because β is the approximation ratio, we want to maximize β with respect to the above constraint. Hence, we choose $\beta = \alpha/3 = 1/6$.

F. Submodular Constraint Improvement

So far, we have achieved a set of k test papers, having guarantee on the collective objective values and satisfying the content hard-constraints on topics and question types. However, these test papers have not yet been optimized based on the assessment constraints. To tackle this, we propose an efficient local search for assessment constraint optimization to improve the generated paper quality while preserving the obtained objective values and other constraints. Here, we propose an effective method for assessment constraint optimization based on the submodular property.

Algorithm 4: Submodular_Constraint_Improvement

Input: $\mathcal{S} = (N, T, D, C, Y)$ - test paper specification; $P_1, P_2, .., P_k$ - test papers; ϵ - a parameter
Output: $P'_1, P'_2, .., P'_k$ - Improved test papers
begin
1 **for** *l=1 to k* **do**
2 $\mathcal{Q}' \leftarrow \mathcal{Q} \setminus \{P_1 \cup ... \cup P_k\}$;
3 **while** $\exists q_i, q_j : q_i \in P_l, q_j \in \mathcal{Q}'$ *such that* $f_\xi(\{P_l - \{q_i\}\} \cup \{q_j\}) > (1 + \frac{\epsilon}{|\mathcal{N}_{q_i}|^2}) f_\xi(P_l)$ **do**
/* $\mathcal{N}_{q_i}$ neighborhood of q_i */
4 $P_l \leftarrow \{P_l - \{q_i\}\} \cup \{q_j\}$;
5 **return** $P_1, P_2, .., P_k$

Consider a test paper $P^0 = P_l = \{q_1, q_2, .., q_N\}, l = 1..k$, which is generated with feasible satisfaction of multiple knapsack constraints. Let $q = \{0,1\}^n, |q| = N$ be the binary representation of the initial solution P. From the previous step, we have $\mathbf{A}q^0 \leq \mathbf{b}$, which is the 0-1 ILP formulation of the corresponding 1-TPG problem. This step aims to turn the existing solution q^0 to q^1 such that $\mathbf{A}q^1 = \mathbf{b}$. This is the Subset Vector Sum problem, which is known to be NP-hard in general. However, as shown in Section IV-D, the existing test paper P_l has completely satisfied the proportion constraints and cardinality constraint. Hence, we only need to optimize the total time and difficulty degree constraints. Let $\sum_{l=1}^{n} a_{il}q \leq b_i$ be the total time constraint and $\sum_{l=1}^{n} a_{jl}q \leq b_j$ be the difficulty degree constraint. Here, we would like to optimize the assessment constraints while ensuring the objective function value $f(P_l)$ will not decrease. Note that in this step, we use the quality function f discussed in Lemma 4 instead of f_ϕ because we are dealing with the assessment constraints and the objective function has already satisfied the fairness objective. To optimize the assessment constraints, we reformulate the 1-TPG problem to unconstrained submodular maximization by introducing Lagrange multipliers λ_1 and λ_2 as follows:

$$f_\xi(P) = f(P) - \lambda_1|\sum_{l=1}^{n} a_{il}q - b_i| - \lambda_2|\sum_{l=1}^{n} a_{jl}q - b_j|$$

It is not difficult to show that the problem $\max f_\xi(P_l)$ is an unconstrained non-monotone submodular maximization [39]. We adapt an efficient deterministic local search proposed by Feige et al. [39] for unconstrained non-monotone submodular maximization. Algorithm 4 presents the local search algorithm. As shown in [39], this local search algorithm will produce a good solution with approximation ratio of $\frac{1}{3}$ and runtime complexity of $O(n^2 \times \log\frac{1}{\epsilon} \times \log n)$, where $0 < \epsilon < 1$ is an algorithm parameter that could be set to any small value.

G. Termination

The paper generation process is repeated until the following termination condition is reached: $\phi_{max} - \phi_{min} \leq \delta$, where δ is a predefined threshold. This is because the existing fairness

value ϕ is near-optimal. The parameter δ represents a tradeoff between the runtime time and quality of constraint satisfaction. An optimal value of $\delta = 0.05$ was found experimentally.

H. Complexity Analysis

We summarize the approximation results of the proposed SMA approach for multiobjective optimization of k-TPG.

Theorem 3. The proposed SMA approach achieves the following theoretical multiobjective approximation results of the total quality maximization and fairness quality maximization:

$$\sum_{i=1}^{k} f(P_i) \geq \frac{1}{2} \max_{P'_1,..,P'_k} \sum_{i=1}^{k} f(P'_i) = \frac{1}{2} OPT$$

$$\min_{l=1..k} f(P_l) \geq \frac{1}{6} \max_{P'_1,..,P'_k} \min_{l=1..k} f(P'_l)$$

In addition, the multiple knapsack equality constraints are also guaranteed to achieve a constant approximation ratio of $\frac{1}{3}$.

Moreover, polynomial time complexity is also achieved. Let $\vartheta = \max_{q \in \mathcal{Q}} f(q))$, ϵ is the parameter in the local constraint improvement step. The computational time of the objective function evaluation is denoted as τ_f in our analysis. The total time complexity of the SMA approach is in the order of:

$O(\lceil \log_2(N \times \vartheta) \rceil \times (k \times n \times N \times \tau_f + k \times n \times N \times \tau_f + k^3 \times N\tau_f + k \times n^2 \times \log \frac{1}{\epsilon} \times \log n \times \tau_f)$
$\approx O(\lceil \log_2(N \times \vartheta) \rceil \times k \times \tau_f (2nN + k^2 N + n^2 \log n \log \frac{1}{\epsilon}))$
$\approx O(\log \frac{1}{\epsilon} \times k \times \log_2(N \times \vartheta) \times n^2 \log n \times \tau_f)$

V. Performance Evaluation

In this section, we evaluate the performance of the proposed SMA approach for k-TPG. The experiments are conducted in a Windows XP environment, using an Intel Core 2 Quad 2.66 GHz CPU with 3.37 GB of memory. The performance of SMA is measured and compared with other techniques including the following 3 re-implemented algorithms for k-TPG: k-TS [4], k-PSO [5] and k-ACO [6] based on related published articles. In addition, we also compare SMA with DAC-MA [36]. As DAC-MA is designed for 1-TPG, we generate each of k test papers of DAC-MA sequentially after eliminating questions of the previously generated test papers. We run each experiment of the stochastic approaches k-TS, k-PSO and k-ACO five times and select the best result for comparison.

A. Datasets

We generate 4 large-sized synthetic datasets, namely D_1, D_2, D_3 and D_4, for performance evaluation. In the 2 datasets, D_1 and D_3, the value of each attribute is generated according to a uniform distribution. However, in the other 2 datasets, D_2 and D_4, the value of each attribute is generated according to a normal distribution. Our purpose is to measure the effectiveness and efficacy of the test paper generation process of each algorithm for both balanced datasets D_1 and D_3, and imbalanced datasets D_2 and D_4. Intuitively, it is more difficult to generate good quality test papers for the datasets D_2 and D_4 than the datasets D_1 and D_3. Table II summarizes the 4 datasets.

TABLE II
Test Datasets

	#Questions	#Topics	#Question Types	Distribution
D_1	20000	40	3	uniform
D_2	30000	50	3	normal
D_3	40000	55	3	uniform
D_4	50000	60	3	normal

B. Experiments

To evaluate the performance of the SMA approach, we aim to analyze the quality and runtime efficiency of the SMA approach for k-TPG based on 4 large-scale datasets by using different specifications. Here, 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. To evaluate the effectiveness of the proposed approach, we have conducted the experiments according to 5 different values of k, i.e., $k = 1$, $k = 5$, $k = 10$, $k = 15$, $k = 20$. In the experiments, the performance of the proposed SMA approach is evaluated and compared based on runtime and paper quality. The performance of SMA is measured and compared with other k-TPG techniques.

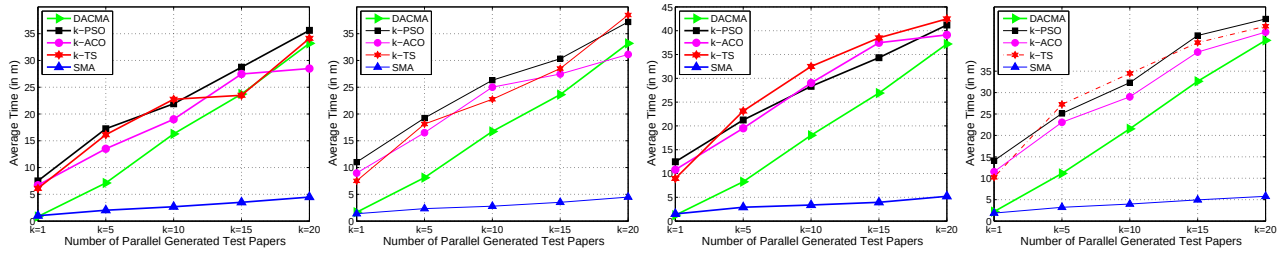
C. Quality Measures for k-TPG

To evaluate the quality of the generated test papers for parallel test paper generation, we use the Average Total Quality and Average Constraint Violation. In addition to the average quality, the corresponding Deviate Total Quality and Deviate Constraint Violation are also used to measure the similarity in quality of the k generated test papers based on the same user specification $\mathcal{S}$. To begin with, we define Average Total Quality and Average Constraint Violation.

Definition 5 (Average Total Quality). Given a specification $\langle k, \mathcal{S} \rangle$. Let $\mathbf{P} = P_1, ..., P_k$ be the set of generated test papers from the specification $\mathcal{S}$. The Average Total Quality $\mathcal{M}_d^{k,\mathcal{S}}(\mathbf{P})$ is defined as average of the quality of $f(P_i)$, $i = 1..k$.

Constraint violation indicates the differences between the test paper specification and generated test paper. Let $\mathcal{S} = \langle N,\ T, D, C, Y \rangle$ be a test paper specification and $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P \rangle$ be a generated test paper specification. Constraint violations can be measured according to total time, average difficulty degree, topic distribution and question type distribution between the test paper specification ($\mathcal{S}$) and the generated test paper specification ($\mathcal{S}_P$) as follows:

- *Total time constraint violation:* $\triangle T(\mathcal{S}_P, \mathcal{S}) = \frac{|T_P - T|}{T}$
- *Average difficulty degree constraint violation:* $\triangle D(\mathcal{S}_P, \mathcal{S}) = \frac{|D_P - D|}{D}$
- *Topic distribution constraint violation:* $\triangle C(\mathcal{S}_P, \mathcal{S}) = D_{KL}(pc_p || pc) = \sum_{i=1}^{M} pc_p(i) \log \frac{pc_p(i)}{pc(i)}$
- *Question type distribution constraint violation:* $\triangle Y(\mathcal{S}_P, \mathcal{S}) = D_{KL}(py_p || py) = \sum_{j=1}^{K} py_p(j) \log \frac{py_p(j)}{py(j)}$



(a) Dataset D_1 (b) Dataset D_2 (c) Dataset D_3 (d) Dataset D_4

Fig. 2. Performance Results Based on Average Runtime

where D_{KL} is the Kullback-Leibler Divergence [40] which is used to measure the statistical differences of the topic and question type distributions between $\mathcal{S}_P$ and $\mathcal{S}$.

The constraint violations (CV) of a generated test paper P w.r.t. the test paper specification $\mathcal{S}$ can then be calculated as the average of the 4 violations:

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

A high quality test paper P should maximize the average discrimination degree and minimize constraint violations. In other words, it should have a high value on E_P and a low value on the constraint violation $CV(P,\mathcal{S})$. The constant $\lambda = 100$ is used to rebalance the contribution of $\triangle C, \triangle Y$ and $\triangle T, \triangle D$.

Definition 6 (Average Constraint Violation and Variant Constraint Violation). Given a specification $\langle k, \mathcal{S}\rangle$. Let $\mathbf{P} = P_1, P_2, ..., P_k$ be the generated test papers on a question dataset $\mathcal{D}$ from the specification $\mathcal{S}$. The Average Constraint Violation $\mathcal{M}_c^{k,\mathcal{S}}(\mathbf{P})$ of k generated test papers $P_1, P_2, ..., P_k$ from the same specification $\mathcal{S}$ is defined as the average of the Constraint Violation $CV(P_i,\mathcal{S}), i = 1..k$. The Variant Constraint Violation $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ is defined as the variance $Var(CV(P_1,\mathcal{S}),..,CV(P_k,\mathcal{S}))$, $i = 1..k$

As such, we can determine the quality of the generated test papers for k-TPG. For high quality parallel test papers, the Mean Discrimination Degree should be high and the Mean Constraint Violation should be small. We set a threshold $\mathcal{M}_c^{k,\mathcal{D}} \leq 10$, which is obtained experimentally, for high quality parallel test papers. In addition, both of the Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ and Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$ should also be low. We note that $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ and $\mathcal{V}_c^{k,\mathcal{S}}(\mathbf{P})$ are 0 when $k = 1$. Hence, $\mathcal{V}_d^{k,\mathcal{D}}$ and $\mathcal{V}_c^{k,\mathcal{D}}$ should also be 0 in that case. Note that the variant discrimination degree $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ and variant constraint violation $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ are 0 when $k = 1$.

Definition 7 (Mean of Average Total Quality and Deviate of Average Total Quality). Let $\mathbf{P}_1, \mathbf{P}_2, .. ., \mathbf{P}_{12}$ be the sets of a fixed k generated test papers on a question dataset $\mathcal{D}$ w.r.t. the corresponding test paper specifications $\mathcal{S}_i$, $i = 1..12$. The Mean of Average Total Quality $\mathcal{M}_d^{k,\mathcal{D}}$ is defined as the mean of the 12 average discrimination degrees of $\mathcal{M}_d^{k,\mathcal{S}_i}(\mathbf{P}_i)$. The Deviate of Average Total Quality $\mathcal{V}_d^{k,\mathcal{D}}$ is defined as:

$$\mathcal{V}_d^{k,\mathcal{D}} = \frac{\sum_{i=1}^{12} \log(\mathcal{V}_d^{k,\mathcal{S}_i}(\mathbf{P}_i)^{\frac{1}{2}})}{12}$$

Definition 8 (Mean of Fairness Quality). Let $\mathbf{P}_1, \mathbf{P}_2, .. ., \mathbf{P}_{12}$ be the sets of a fixed k generated test papers on a question dataset $\mathcal{D}$ w.r.t. the corresponding test paper specifications $\mathcal{S}_i$, $i = 1..12$. The Mean of Fairness Quality $\mathcal{M}_f^{k,\mathcal{D}}$ is defined as the mean of the 12 fairness quality of $\mathbf{P}_i$.

Definition 9 (Mean Constraint Violation and Deviate Constraint Violation). Let $\mathbf{P}_1, \mathbf{P}_2, .. ., \mathbf{P}_{12}$ be the sets of a fixed k generated test papers on a question dataset $\mathcal{D}$ w.r.t. the corresponding test paper specifications $\mathcal{S}_i$, $i = 1..12$. The Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$ is defined as the mean of the average constraint violation $\mathcal{M}_c^{k,\mathcal{S}_i}(\mathbf{P}_i)$. The Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$ is defined as:

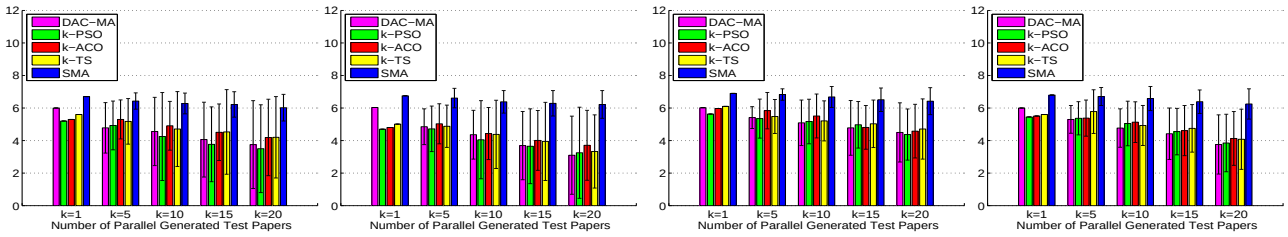
$$\mathcal{V}_c^{k,\mathcal{D}} = \frac{\sum_{i=1}^{12} \log(\mathcal{V}_c^{k,\mathcal{S}_i}(\mathbf{P}_i)^{\frac{1}{2}})}{12}$$

D. Performance Results for k-TPG

Performance on Runtime: Figure 2 compares the runtime performance of the 4 techniques based on the 4 datasets. The results have clearly shown that the proposed SMA approach significantly outperforms the other heuristic techniques in runtime for the different datasets. SMA generally requires less than 6 minutes to complete the parallel paper generation process. Moreover, the proposed SMA approach is quite scalable in runtime on different dataset sizes and distributions. In contrast, the other techniques are not efficient to generate high quality parallel test papers. Particularly, the runtime performance of these techniques degrades quite badly as the dataset size or the number of generated parallel test papers gets larger, especially for imbalanced datasets D_2 and D_4.

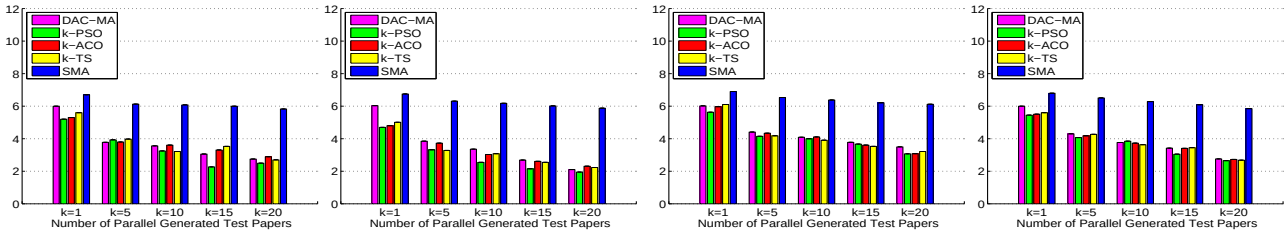
Performance on Quality: Table III and Figure 3 shows the quality performance results of the 4 techniques based on the Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$. As can be seen from Figure 3, SMA has consistently achieved higher Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and lower Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ than the other heuristic k-TPG techniques for the generated parallel test papers. Particularly, SMA can generate high quality test papers with $\mathcal{M}_d^{k,\mathcal{D}} \approx 7$. Note that the lower Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ value indicates that the generated parallel test papers have similar quality in terms of discrimination degree.

Generally, for a specific value of k, we observe that the quality of the generated parallel test papers of all 4 techniques based on the Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and the Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ tend to be improved when the dataset size gets larger. Table III gives the quality performance based on discrimination degree of the 4 techniques. As can be observed from Table III, the quality of the generated parallel test papers for the other 3 techniques (i.e., k-PSO, k-ACO and k-TS) seem not to be improved



(a) Dataset D_1 (b) Dataset D_2 (c) Dataset D_3 (d) Dataset D_4

Fig. 3. Performance Results based on Mean of Average Total Quality $\mathcal{M}_d^{k,\mathcal{D}}$ and Deviate of Average Total Quality $\mathcal{V}_d^{k,\mathcal{D}}$



(a) Dataset D_1 (b) Dataset D_2 (c) Dataset D_3 (d) Dataset D_4

Fig. 4. Performance Results based on Mean Fairness Quality $\mathcal{M}_f^{k,\mathcal{D}}$

on D_2 as compared with D_1 even though the dataset size of D_2 is 1.5 times larger than D_1. On the other hand, the quality improvement of the other 3 techniques on datasets D_3 and D_4 as compared with D_1 is quite significant. Note that the dataset sizes of D_3 and D_4 are 2 and 2.5 times larger than D_1 respectively. This shows that a slight increase of the imbalanced dataset size may not help improve the quality while a significant increase may help improve the quality of the generated parallel test papers.

In addition, we observe from Table III that when the number of generated parallel test papers k gets larger, the Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ on a specific dataset $\mathcal{D}$ tends to decrease. Similarly, the Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ tends to increase when k gets larger. These results have shown that the quality of the generated parallel test papers on a specific dataset $\mathcal{D}$ tends to degrade when k gets larger. This degradation is small for the proposed SMA approach. However, it is quite significant for the other k-TPG techniques. We also find that this degradation is less significant for all 4 techniques when the dataset size gets larger.

Figure 4 shows the performance results of the 4 techniques based on the Mean Fairness Quality $\mathcal{M}_f^{k,\mathcal{D}}$. As can be seen from Figure 4, SMA has consistently achieved higher Mean Fairness Quality $\mathcal{M}_f^{k,\mathcal{D}}$ than the other k-TPG techniques. Particularly, SMA can generate high quality test papers with $\mathcal{M}_f^{k,\mathcal{D}} \approx 6$. Note that the higher $\mathcal{M}_f^{k,\mathcal{D}}$ value indicates that the generated parallel test papers have similar optimal quality.

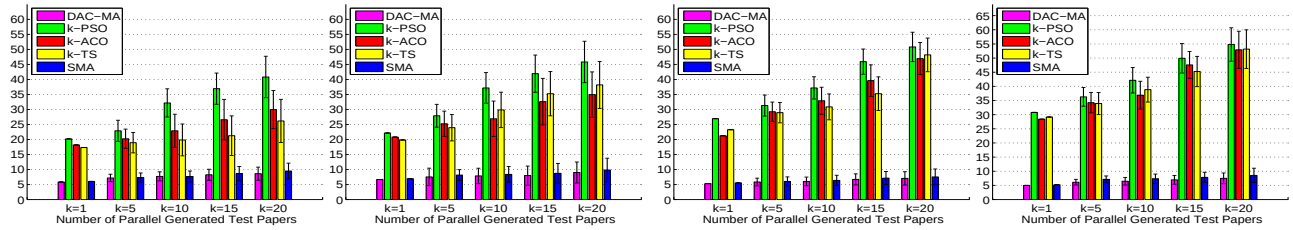
Table IV and Figure 5 give the quality performance results of the 4 techniques based on the Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D},k}$ and Deviate Constraint Violation $\mathcal{V}_c^{\mathcal{D},k}$. We also observe that SMA has consistently outperformed the other techniques on Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$ and Deviate Constraint Violation $\mathcal{V}_c^{\mathcal{D},k}$ based on the 4 datasets. The Mean Constraint Violation of SMA tends to decrease whereas the Mean Constraint Violations of the other 3 techniques increase quite fast when the dataset size or the number of specified constraints gets larger. In particular, SMA can generate high quality parallel test papers with $\mathcal{M}_c^{\mathcal{D},k} \leq 10$ for all datasets.

TABLE III
QUALITY PERFORMANCE COMPARISON BASED ON AVERAGE TOTAL QUALITY OF SMA AND 3 k-TPG TECHNIQUES ($\mathcal{M}_d^{k,\mathcal{D}} \pm \mathcal{V}_d^{k,\mathcal{D}}$)

Algorithm	D_1	D_2	D_3	D_4
1-DACMA	6.0 ± 0	6.03 ± 0	6.02 ± 0	6.0 ± 0
1-PSO	5.20 ± 0	4.70 ± 0	5.63 ± 0	5.45 ± 0
1-ACO	5.30 ± 0	4.80 ± 0	5.97 ± 0	5.51 ± 0
1-TS	5.60 ± 0	5.01 ± 0	6.10 ± 0	5.60 ± 0
1-SMA	6.70 ± 0	6.75 ± 0	6.90 ± 0	6.80 ± 0
5-DACMA	4.78 ± 1.55	4.85 ± 1.1	5.41 ± 0.67	5.3 ± 0.85
5-PSO	6.70 ± 1.50	4.72 ± 1.40	5.35 ± 1.20	5.37 ± 1.02
5-ACO	5.30 ± 1.20	5.03 ± 1.23	5.84 ± 1.12	5.38 ± 1.11
5-TS	5.18 ± 1.40	4.88 ± 1.30	5.48 ± 1.04	5.78 ± 1.34
5-SMA	6.43 ± 0.50	6.61 ± 0.60	6.83 ± 0.35	6.71 ± 0.55
10-DACMA	4.56 ± 2.1	4.36 ± 1.5	5.09 ± 1.4	4.77 ± 1.18
10-PSO	4.25 ± 2.70	4.05 ± 2.40	5.17 ± 1.37	5.05 ± 1.37
10-ACO	4.91 ± 1.50	4.43 ± 1.60	5.51 ± 1.35	5.13 ± 1.25
10-TS	4.71 ± 2.30	4.38 ± 2.10	5.21 ± 1.23	4.93 ± 1.23
10-SMA	6.28 ± 0.64	6.38 ± 0.69	6.68 ± 0.64	6.58 ± 0.74
15-DACMA	4.06 ± 2.3	3.69 ± 2.1	4.78 ± 1.68	4.42 ± 1.58
15-PSO	3.77 ± 2.30	3.65 ± 2.30	4.97 ± 1.43	4.55 ± 1.43
15-ACO	4.51 ± 1.74	4.01 ± 1.84	4.81 ± 1.34	4.61 ± 1.54
15-TS	4.53 ± 2.60	3.95 ± 2.40	5.03 ± 1.46	4.75 ± 1.46
15-SMA	6.22 ± 0.78	6.29 ± 0.78	6.51 ± 0.72	6.39 ± 0.72
20-DACMA	3.75 ± 2.7	3.1 ± 2.4	4.5 ± 1.82	3.76 ± 1.82
20-PSO	3.50 ± 2.70	3.25 ± 2.80	4.37 ± 1.57	3.85 ± 1.77
20-ACO	4.19 ± 2.35	3.71 ± 2.15	4.58 ± 1.65	4.13 ± 1.65
20-TS	4.20 ± 2.50	3.33 ± 2.25	4.71 ± 1.85	4.08 ± 1.85
20-SMA	6.02 ± 0.82	6.22 ± 0.85	6.42 ± 0.83	6.25 ± 0.93
Avg-DACMA	4.63 ± 1.73	4.40 ± 1.42	5.16 ± 1.11	4.85 ± 1.08
Avg-PSO	4.33 ± 1.84	4.07 ± 1.78	5.10 ± 1.11	4.85 ± 1.12
Avg-ACO	4.84 ± 1.36	4.40 ± 1.36	5.34 ± 1.10	4.95 ± 1.11
Avg-TS	4.84 ± 1.76	4.31 ± 1.61	5.31 ± 1.12	5.03 ± 1.15
Avg-SMA	6.33 ± 0.55	6.45 ± 0.53	6.67 ± 0.47	6.71 ± 0.48

Also, SMA is able to generate higher quality parallel test papers on larger datasets while the other techniques generally degrade on the quality of the generated test papers when the dataset size gets larger.

In addition, we find that when k gets larger, the Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$ on a specific dataset $\mathcal{D}$ tends to decrease. Similarly, the Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$ quality tends to increase when k gets larger. These results have shown that the quality based on constraint violation on a specific dataset $\mathcal{D}$ tends to degrade when k gets larger. Table IV gives the quality performance based on constraint violation of the 4 techniques. As can be seen, the quality degradation



(a) Dataset D_1 (b) Dataset D_2 (c) Dataset D_3 (d) Dataset D_4

Fig. 5. Performance Results on Quality based on Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$ and Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$

TABLE IV
QUALITY PERFORMANCE COMPARISON BASED ON CONSTRAINT VIOLATION OF SMA AND 3 k-TPG TECHNIQUES ($\mathcal{M}_c^{k,\mathcal{D}} \pm \mathcal{V}_c^{k,\mathcal{D}}$)

Algorithm	D_1	D_2	D_3	D_4
1-DACMA	5.85 ± 0	6.65 ± 0	5.31 ± 0	5.01 ± 0
1-PSO	20.19 ± 0	22.17 ± 0	26.92 ± 0	30.81 ± 0
1-ACO	18.20 ± 0	20.81 ± 0	21.26 ± 0	28.41 ± 0
1-TS	17.30 ± 0	19.85 ± 0	23.25 ± 0	29.17 ± 0
1-SMA	6.05 ± 0	6.95 ± 0	5.56 ± 0	5.25 ± 0
5-DACMA	7.19 ± 1.2	7.55 ± 2.9	5.85 ± 1.28	6.13 ± 1.0
5-PSO	22.90 ± 3.5	27.90 ± 3.8	31.29 ± 3.5	36.29 ± 3.3
5-ACO	20.25 ± 3.2	25.25 ± 4.2	29.25 ± 3.2	34.25 ± 3.6
5-TS	18.93 ± 3.4	23.93 ± 4.4	28.93 ± 3.4	33.93 ± 3.9
5-SMA	7.35 ± 1.5	8.15 ± 1.8	6.05 ± 1.5	7.15 ± 1.2
10-DACMA	7.72 ± 1.5	7.87 ± 2.57	6.05 ± 1.43	6.48 ± 1.35
10-PSO	32.17 ± 4.7	37.17 ± 5.1	37.17 ± 3.7	42.17 ± 4.5
10-ACO	22.90 ± 5.5	26.90 ± 5.9	32.90 ± 4.5	36.90 ± 4.9
10-TS	19.85 ± 5.3	29.85 ± 5.9	30.85 ± 4.3	38.85 ± 4.4
10-SMA	7.68 ± 1.8	8.38 ± 2.6	6.38 ± 1.7	7.38 ± 1.6
15-DACMA	8.26 ± 1.8	7.98 ± 3.18	6.72 ± 1.83	6.98 ± 1.55
15-PSO	36.92 ± 5.2	41.92 ± 6.2	45.92 ± 4.2	49.92 ± 5.2
15-ACO	26.60 ± 6.7	32.60 ± 7.7	39.60 ± 5.2	47.60 ± 4.7
15-TS	21.25 ± 6.6	35.25 ± 7.4	35.25 ± 5.6	45.25 ± 5.4
15-SMA	8.65 ± 2.4	8.75 ± 3.3	7.15 ± 2.2	7.85 ± 1.8
20-DACMA	8.61 ± 2.2	9.01 ± 3.49	7.06 ± 2.21	7.51 ± 1.89
20-PSO	40.81 ± 6.9	45.81 ± 6.9	50.81 ± 4.9	54.81 ± 5.9
20-ACO	29.94 ± 6.4	34.94 ± 7.6	46.94 ± 5.4	52.94 ± 6.6
20-TS	26.17 ± 7.2	38.17 ± 7.8	48.17 ± 5.6	53.17 ± 6.8
20-SMA	9.5 ± 2.6	9.85 ± 3.9	7.55 ± 2.6	8.55 ± 2.5
Avg-DACMA	7.53 ± 1.35	7.81 ± 2.42	6.20 ± 1.35	6.42 ± 1.16
Avg-PSO	30.59 ± 4.1	34.99 ± 4.4	38.42 ± 3.3	42.80 ± 3.8
Avg-ACO	23.57 ± 4.4	28.10 ± 5.1	33.99 ± 3.7	40.022 ± 4.0
Avg-TS	20.7 ± 4.5	29.41 ± 5.1	33.29 ± 3.8	40.07 ± 4.1
Avg-SMA	7.84 ± 1.7	8.42 ± 1.8	6.54 ± 1.5	7.23 ± 1.4

is small for the proposed SMA approach whereas it is quite significant for the other k-TPG techniques.

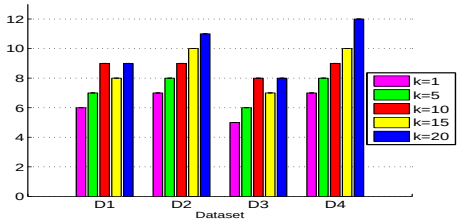


Fig. 6. Number of Better Quality Instances of SMA and other Approaches

Figure 6 shows the number of significant better quality instances of SMA as compared to the best solution among k-TS, k-PSO and k-ACO for each pair of dataset and a given k. A set of test papers is significant better in quality than another set of test papers if it is strictly better in 3 aspects including Total Quality ($\mathcal{M}_d^{k,\mathcal{D}}$ and $\mathcal{V}_d^{k,\mathcal{D}}$), Fairness Quality ($\mathcal{M}_f^{k,\mathcal{D}}$) and Constraint Violation ($\mathcal{M}_c^{k,\mathcal{D}}$ and $\mathcal{V}_c^{k,\mathcal{D}}$). As can be seen, SMA has a consistent high number of significant better quality instances in the 4 datasets. For small $k = 1, 5$, SMA has approximately 67% of better quality instances in the total of 96 instances. For large $k = 10, 15, 20$, SMA has achieved a better performance of 91% better quality instances in the total of 144 instances.

Compare SMA with DAC-MA: As we can observe in Figure 2, the runtime of DAC-MA is not scalable. It increases linearly with the number of test papers k while that of SMA increases much slower. This is because DAC-MA is designed for 1-TPG and it was run k times to generate k test papers. Figure 3 clearly shows that SMA has consistently outperformed DAC-MA in terms of Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$. This shows the effectiveness of SMA in optimizing the two objective functions simultaneously over k test papers while DAC-MA only optimizes the quality function of each paper separately. However, DAC-MA has consistently outperformed SMA in terms of the Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$. It is because DAC-MA pay more attention to constraint violation minimization when optimizing the quality objective function of each paper.

Discussion: The good performance of SMA is due to 2 main reasons. Firstly, as SMA is an approximation algorithm with constant performance guarantee, it can find the near-optimal solution for objective functions of k-TPG effectively and efficiently while satisfying the multiple constraints without using weighting parameters. As such, SMA can achieve better paper quality and runtime efficiency as compared with other heuristic-based k-TPG techniques. Secondly, SMA is a submodular greedy-based algorithm, which is able to produce good solution in efficient polynomial runtime. Thus, SMA can also improve its computational efficiency on large-scale datasets as compared with the other k-TPG techniques.

Furthermore, the deviation of discrimination degree and deviation of constraint violation of the SMA approach tend to increase slightly while these deviations grow quite fast for the other 3 techniques when k gets larger. It is because the SMA approach has effective techniques to generate parallel test papers with fairness quality while the other 3 techniques do not. Intuitively, when k gets larger, it needs more questions with appropriate attributes to generate parallel quality test papers of similar quality. However, with a fixed question dataset, the number of questions with appropriate attributes is limited. Hence, without an appropriate technique to generate parallel test papers of similar quality, the deviation of discrimination degree and deviation of constraint violation of the generated parallel test papers tend to grow fast.

VI. CONCLUSION

In this paper, we have proposed an effective and efficient Submodular Memetic Approximation (SMA) approach for multiobjective parallel test paper generation (k-TPG). The key

success of SMA lies in the synergy of the iterative diversification process in a population-based evolutionary computation with the intensification process of local improvement. This process uses a submodular local search mechanism and submodular crossover operator to jointly optimize the *total quality maximization* objective and the *fairness quality maximization* objective of the k test papers. Specifically, SMA employs submodular optimization techniques for enhancing steps of an iteratively multiobjective memetic algorithm framework. The proposed SMA approach is a deterministic greedy-based approximation algorithm to find the near-optimal solution by exploiting the special submodular structure of objective functions. To the best of our knowledge, SMA is a pioneering multiobjective memetic algorithm, which can achieve provably near-optimal solutions in polynomial runtime for a real-world problem. The performance results on various datasets have shown that the SMA approach has achieved generated parallel test papers with not only high quality, but also runtime efficiency when compared with other k-TPG techniques.

REFERENCES

[1] F. G. Martin, "Will massive open online courses change how we teach?" *Commun. ACM*, vol. 55, no. 8, pp. 26–28, Aug. 2012. [Online]. Available: http://doi.acm.org/10.1145/2240236.2240246

[2] J. Vondrak, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the 40th annual ACM symposium on Theory of computing*, 2008, pp. 67–74.

[3] A. Asadpour and A. Saberi, "An approximation algorithm for max-min fair allocation of indivisible goods," *SIAM Journal on Computing*, vol. 39, no. 7, pp. 2970–2989, 2010.

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

[5] T. F. Ho, P. Y. Yin, G. J. Hwang, S. J. Shyu, and Y. N. Yean, "Multi-objective parallel test-sheet composition using enhanced particle swarm optimization," *Journal of Educational Technology and Society*, vol. 12, no. 4, pp. 193–206, 2008.

[6] X. M. Hu, J. Zhang, H. S. H. Chung, O. Liu, and J. Xiao, "An intelligent testing system embedded with an ant-colony-optimization-based test composition method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 6, pp. 659–669, 2009.

[7] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1808 – 1820, 2014.

[8] M. Li, S. Yang, and X. Liu, "Diversity comparison of pareto front approximations in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2568 – 2584, 2014.

[9] K. Li, S. Kwong, Q. Zhang, and K. Deb, "Interrelationship-based selection for decomposition multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 2076 – 2088, 2014.

[10] S. Jiang and S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts," *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 421 – 437, 2015.

[11] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *IEEE World Congress on Evolutionary Computation*. IEEE, 2008, pp. 2419–2426.

[12] U. Feige and J. Vondrak, "Approximation algorithms for allocation problems: Improving the factor of 1-1/e," in *47th Annual IEEE Symposium on Foundations of Computer Science*, 2006, pp. 667–676.

[13] D. Bertsimas and R. Weismantel, *Optimization Over Integers*. Belmont, MA: Dynamic Ideas, 2005.

[14] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations research*, vol. 59, no. 1, pp. 17–31, 2011.

[15] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *Automatic Control, IEEE Transactions on*, vol. 55, no. 4, pp. 922–938, 2010.

[16] K. H. Tsai, T. I. Wang, T. C. Hsieh, T. K. Chiu, and M. C. Lee, "Dynamic computerized testlet-based test generation system by discrete pso with partial course ontology," *Expert Systems with Applications*, vol. 37, no. 1, pp. 774–786, 2009.

[17] L. Lovasz, "Submodular functions and convexity," *Mathematical programming: the state of the art*, pp. 235–257, 1983.

[18] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000.

[19] L. Fleischer, "Recent progress in submodular function minimization," 2000.

[20] S. Iwata, "Submodular function minimization," *Mathematical Programming*, vol. 112, no. 1, pp. 45–64, 2008.

[21] J. B. Orlin, "A faster strongly polynomial time algorithm for submodular function minimization," *Mathematical Programming*, vol. 118, no. 2, pp. 237–251, 2009.

[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[23] A. Schrijver, *Combinatorial Optimization, Polyhedra and Efficiency: Volume A, B, C*. Springer, 2003.

[24] S. Fujishige, *Submodular functions and optimization*. Elsevier Science, 2005, vol. 58.

[25] S.-C. Horng, S.-Y. Lin, L. H. Lee, and C.-H. Chen, "Memetic algorithm for real-time combinatorial stochastic simulation optimization problems with performance analysis," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1495–1509, 2013.

[26] D. Liu, K. Tan, C. Goh, and W. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume 37(1), pages 42-50*, 2007.

[27] Y. Mei, K. Tang, and X. Yao, "A memetic algorithm for periodic capacitated arc routing problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 6, pp. 1654–1667, 2011.

[28] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, 2006.

[29] X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, 2011.

[30] J. Knowles, D. Corne, and K. Deb, *Multiobjective problem solving from nature: from concepts to applications*. Springer-Verlag New York Inc, 2008.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[32] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[33] J. Knowles and D. Corne, "Memetic algorithms for multiobjective optimization: issues, methods and prospects," *Recent advances in memetic algorithms, pages 313-352*, 2005.

[34] P. Bosman and E. de Jong, "Combining gradient techniques for numerical multi-objective evolutionary optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 627–634.

[35] A. Caponio and F. Neri, "Integrating cross-dominance adaptation in multi-objective memetic algorithms," *Multi-Objective Memetic Algorithms*, pp. 325–351, 2009.

[36] M. L. Nguyen, S. C. Hui, and A. C. M. Fong, "Divide-and-conquer memetic algorithm for online multi-objective test paper generation," *Memetic Computing*, vol. 4, no. 1, pp. 33–47, 2012.

[37] J. Barbanel, *The geometry of efficient fair division*. Cambridge University Press, 2005.

[38] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[39] U. Feige, V. Mirrokni, and J. Vondrak, "Maximizing non-monotone submodular functions," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1133–1153, 2007.

[40] S. Kullback, *Information theory and statistics*. Dover Publisher, 1997.