Online test paper generation for a web-based mathematics testing environment

Nguyen, Minh Luan

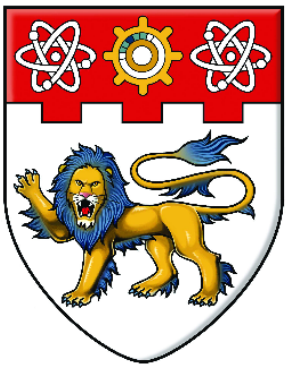
2014

Nguyen, M. L. (2014). Online test paper generation for a web-based mathematics testing environment. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/59538

https://doi.org/10.32657/10356/59538

ONLINE TEST PAPER GENERATION FOR A WEB-BASED MATHEMATICS ENVIRONMENT NGUYEN MINH LUAN 2014



NANYANG TECHNOLOGICAL UNIVERSITY

ONLINE TEST PAPER GENERATION FOR A WEB-BASED MATHEMATICS TESTING ENVIRONMENT

NGUYEN MINH LUAN

SCHOOL OF COMPUTER ENGINEERING

2014

ONLINE TEST PAPER GENERATION FOR A WEB-BASED MATHEMATICS TESTING ENVIRONMENT

NGUYEN MINH LUAN

NGUYEN MINH LUAN

School of Computer Engineering

A thesis submitted to the Nanyang Technological University

in fulfilment of the requirement for the degree of

Doctor of Philosophy

2014

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Dr. Hui Siu Cheung, for his encouragement, support and guidance on this research. As a supervisor, he gives valuable advice and inspiration on my research topic. Without his guidance, I could not have completed the work reported in this thesis.

My gratitude also goes to Dr. Alvis Cheuk M. Fong and Dr. Quan Thanh Tho for their support for my PhD application. Their help is invaluable for the early stage of my academic career.

I would like to thank the School of Computer Engineering, Nanyang Technological University for providing me the scholarship to pursue this research. I would also like to thank the technical staffs of the CAIS Lab, Mrs. Linda Ang and Mr. Thomas Loo, for their supports over the years. I would also like to thank my friends at the Nanyang Technological University for their help.

Last but not least, I am very grateful to my parents and sister for their encouragement and support. Their love accompanies me wherever I go.

Abstract

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. To make learning effective, it is important to assess and evaluate the proficiency and ability of the learners while they are learning the concepts. Currently, Web-based testing has been popularly used for automatic self-assessment especially in Web-based learning environments. Different from passive course archives like MIT OpenCourseWare, the online courses are interactive and can assess learners automatically on what they have learnt via Web-based testing. The main benefit is that learners can take classes at their own pace and get immediate feedback on their proficiency, unlike traditional classes.

In this research, we propose a Web-based Mathematics Testing Environment, which consists of three major components, namely question item calibration, online test paper generation, and automatic solution assessment. Question item calibration calibrates the attributes of each question in the question database. Online test paper generation generates test papers automatically from the question database with online runtime requirement. After the test paper has been answered online, automatic solution assessment can then evaluate the correctness of user answers. This research aims to investigate different multiobjective optimization, data mining and probabilistic techniques for supporting the three components of the Web-based Mathematics testing environment. More specifically, it focuses mainly on online test paper generation, parallel test paper generation, automatic question difficulty calibration and automatic mathematical solution assessment. As a result, we have made the following contributions in this research:

- An efficient constraint-based Divide-and-Conquer (DAC) approach has been proposed for Online-TPG. Three algorithms, namely DAC, DAC Tabu Search (DAC-TS) and DAC Memetic Algorithm (DAC-MA), have been implemented based on the proposed constraint-based DAC approach. The proposed DAC-MA approach is based on the

principle of constraint decomposition for effective multiobjective optimization. The DAC-MA approach has achieved 12 times faster in runtime performance and 38% better test paper quality than the other current TPG techniques including GA, DE, TS, ACO and PSO.

- An Integer Programming approach, called BAC-TPG, has been proposed for high quality large-scale Online-TPG. The proposed BAC-TPG approach is based on the Branch-and-Bound and Lifted Cover Cutting methods to find the near-optimal solution by exploiting the sparse matrix property of 0-1 ILP. The performance results have shown that the BAC-TPG approach has achieved 29% better test paper quality than DAC-MA on the large-scale Online-TPG problem.

- A Multiobjective Evolutionary Co-Approximation (MECA) algorithm has been proposed for k-TPG. The proposed MECA approach is a greedy-based approximation algorithm, which explores the submodular property of the objective function for combinatorial multiobjective optimization. The MECA approach has achieved 8 times faster in runtime performance and 85% better test paper quality for parallel test paper generation than the other current techniques including k-TS, k-PSO and k-ACO.

- An effective Content-based Collaborative Filtering (CCF) approach has been proposed for automatic calibration of question difficulty degree. In CCF, collaborative filtering is used to predict unknown user responses from known responses of questions. And the difficulty degree of each question in the dataset is then estimated using Item Response Theory. The proposed CCF approach has achieved promising results with 40% prediction errors less than the traditional Proportion Correct Method technique.

- An effective Probabilistic Equivalence Verification (PEV) algorithm has been proposed for automatic mathematical solution assessment. The PEV approach is a probabilistic algorithm, which verifies the equivalence of two functions by evaluating the value of the functions at random points sampled uniformly over a sufficiently large range. PEV can avoid false negative errors of the current popular Computer Algebra Systems such as Maple and Mathematica. The PEV approach can achieve an average accuracy of 93.6% while Maple and Mathematica can only achieve about 72.4% and 64% respectively.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the rapid growth of the Internet, Web-based learning is very popular nowadays with much instructional and educational course materials available on the Web. Well-known educational institutions such as MIT [5], Stanford [7, 8] and University of California, Berkeley [9] have provided course materials online freely to any user. Popular websites such as Youtube [10] and Khan Academy [4] contain millions of educational videos for users to view and learn. In addition, most academic institutions also maintain e-learning systems [2] to support the learning of their courses for their students. Such e-learning systems generally contain course materials in the form of lecture notes, recorded video lectures and exercises. However, to make learning effective, it is important to assess and evaluate the proficiency and ability of the learners while they are learning the topics and the course. To achieve this, Computer-Based Testing (CBT) [75, 149, 153, 225] or Web-based testing [52, 53, 58, 102] is a promising approach. In addition, Web-based testing is also very useful for supporting personalized learning [1, 195], in which individual learners will be able to learn adaptively according to their abilities and learning goals.

1.2 Web-based Testing

Web-based testing is an effective way for evaluating learning abilities such that proper learning strategies can be adopted to improve learning performance [103, 102]. The approach has the advantages that the generated tests could be composed dynamically based on practical requirements and there are plenty of test items available in the question database for selection

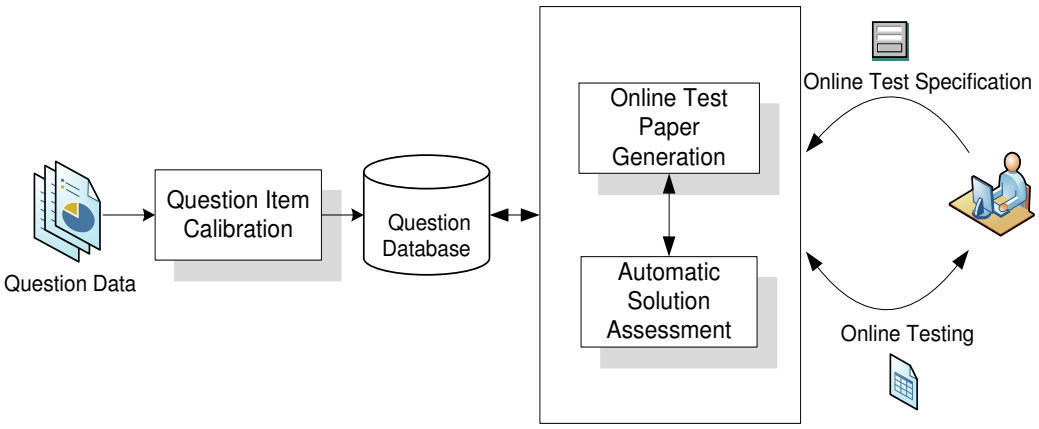


Figure 1.1: Web-based Testing

which could be presented in multimedia styles. In addition, with Web-based testing, learners can receive the evaluation results immediately, and teachers or instructors can save a lot of time for setting test papers. Furthermore, the learners' test results can be recorded automatically after the tests, and these records can be analyzed further to identify the weaknesses and strengths of the learners. Based on the analysis, relevant suggestions can be given to learners in order to improve their learning performance.

Web-based testing will not only help the teacher or instructor evaluate the learning status of the learners, but also facilitate the diagnosis of the learning problems. One of the most important and challenging issues in Web-based testing is the construction of good test papers that can meet multiple assessment requirements. Most of the current research on Web-based testing focus on Computerized Adaptive Testing [52, 53, 58, 102, 149, 218, 225]. Apart from CAT, another promising direction for Web-based testing can be based on automatic test paper generation which generates full test papers automatically based on multiple assessment criteria.

In this research, we aim to investigate a Web-based Mathematic testing environment. Figure 1.1 shows the proposed research, which consists of three major components: Online Test Paper Generation, Question Item Calibration and Automatic Solution Assessment. In addition, Question Database is used to store all question items. In the proposed research, a learner/teacher can specify a test paper based on his requirements. The test paper specification will be used as constraints for the generation of the test paper for testing. The learner will then be able to attempt the generated test paper online. The answers will then be checked automatically. And finally, the ability of the learner will be evaluated.

1.2.1 Online Test Paper Generation

In automatic test paper generation [112, 116, 117, 118, 119, 120, 147, 197, 217], test papers are generated automatically using optimization techniques. In this approach, it generates test papers automatically based on user specification on completion time, topic distribution, difficulty degree and discrimination degree. The test papers are generally generated offline. The generated test papers are then attempted by learners either online or offline in a similar manner to that of the traditional pen-and-pencil tests. As a result, the learners' abilities can be determined from the results of the attempted test papers. As can be seen, the automatic test paper generation approach creates full test papers which cover the topics and difficulty degree that are specified by the user. Automatic test paper generation has the advantage that it generates the full test paper which will be quite comprehensive according to user requirement. Also, as it is similar to traditional paper-and-pencil testing, the approach is well recognized and adopted by most people in the educational environment.

Although different techniques such as Dynamic Programming (DP) [117], Tabu Search (TS) [118, 120], Genetic Algorithms (GA) [119, 197], Particle Swarm Optimization (PSO) [112, 217], Ant Colony Optimization (ACO) [116] and Artificial Immune System (AIS) [147] have been proposed for automatic test paper generation, these techniques have not taken the generation time into consideration. The test paper generation process is generally considered as an offline process. As a result, the paper generation process requires long runtime due to the nature of these proposed techniques. In addition, the quality of the generated test paper depend very much on the satisfaction of the specified set of constraints. Therefore, in automatic test paper generation, satisfying the constraints generally is more important than obtaining a high objective function value. However, many of the current techniques have focused mainly on optimizing the objective function. As such, the generated test papers have violated heavily on the specified constraints [116, 117, 118, 120, 147, 197, 217], thereby reducing the quality of the generated test papers. In addition, to support test paper generation for Web-based testing, it is also necessary to consider the online requirement in which a user expects to generate a test paper within an acceptable response time. Moreover, in some testing situations, it is necessary to generate multiple test papers of similar quality based on a user specification. This problem is referred to as parallel test paper generation.

In this research, we will focus on investigating different optimization techniques for online test paper generation and parallel test paper generation.

1.2.2 Question Item Calibration

A calibrated question item database is necessary for automatic test paper generation. Thus, accurate question attribute values are very important in order to generate good quality test papers. There are 5 *semantic* attributes for each question: discrimination degree, difficulty degree, completion time, related topics and question type. The question attributes are defined based on the classical definitions in Item Response Theory (IRT) [27] of Computerized Adaptive Testing [218, 225]. Conventionally, these question attributes are labeled by teachers or human experts [232]. However, it is a very tedious task to label all question attributes manually. As such, it is necessary for automatic question attribute calibration for a large pool of questions.

Among these 5 attributes, the *question type* attribute is quite straightforward to label based on the format of a question. For the *related topic* attribute, it is highly domain-dependent. For example, Math question data may contain formulas which are highly symbolic and structural, whereas English question data are mainly presented in textual format. Moreover, the length of the question and its solution may be very short. Thus, it requires effective and efficient feature extraction as well as data classification techniques for calibrating the related topic attribute. The *difficulty degree* and *discrimination degree* attributes of each question are conventionally determined using statistical analysis based on a large number of correct/incorrect answers from users' responses [218, 225]. However, this process is very tedious and time consuming [150]. Therefore, it is challenging to predict accurately the values of these two attributes [102]. In addition, the *question time* attribute can be computed based on question difficulty or gathered through the students' question answering activities [142].

In this research, we will focus on investigating techniques for automatic question difficulty calibration. The discrimination degree of a question can be calibrated in a similar manner.

1.2.3 Automatic Solution Assessment

Assessing the knowledge acquired by learners is one of the most important aspects in the learning process. Automatic assessment has been shown to be effective in improving learners' performance [103] as it helps evaluate learner's proficiency and provides relevant feedback as well as suggestions to enhance the learning process. Most learning systems such as Coursera[1] and Edventure[2] have supported automatic assessment for multiple choice questions.

[1] https://www.coursera.org/
[2] https://edventure.ntu.edu.sg

Currently, automatic assessment techniques have been applied for automatic essay grading [109], essay summarization [108], and short text grading [165]. However, it has not been widely applied for mathematical questions. As equivalent mathematical expressions can be expressed in different forms, e.g., $x+\frac{1}{x}$ and $\frac{x^2+1}{x}$, automatic mathematical solution assessment which checks the equivalence of the user answer and standard solution is another challenging problem. Computer Algebra Systems (CAS) such as Maple [156] and Mathematica [158] provide the expression equivalence checking function. However, their techniques have generated many false negative errors if the provided correct answers are written in other forms different from the given standard solutions. Therefore, it is challenging to provide automatic solution assessment for different forms of math answers.

In this research, we will focus on investigating techniques for automatic solution assessment for mathematical expressions.

1.3 Objectives

The main objective of this research is to investigate optimization, data mining and automatic assessment techniques to overcome the issues of current Web-based Testing as discussed in Section 1.2. In particular, this research will investigate techniques for online test paper generation, parallel test paper generation, question difficulty calibration, and automatic mathematical solution assessment as follows:

- *Investigate new techniques for Online Test Paper Generation (Online-TPG).* In a Web-based testing environment, a user may specify the test paper criteria according to his preference, and then attempt the generated test paper online. To support Web-based testing, we need to consider the runtime performance of the generated test papers in addition to constraint satisfacation. This research will investigate a constraint-based Divide-and-Conquer approach for Online-TPG.

- *Investigate a new technique for Large-scale Online Test Paper Generation.* In large-Scale Online-TPG, apart from considering the runtime for generating the test papers, users may also want to have high quality test papers. To achieve this, this research will investigate an efficient 0-1 Integer Linear Programming approach for large-scale Online-TPG.

- *Investigate a new technique for Parallel Test Paper Generation.* In some Web-based

testing situations, it is necessary to generate multiple different test papers of similar quality. This research will investigate a greedy-based approximation algorithm for parallel test paper generation (k-TPG).

- *Investigate a new technique for Question Difficulty Calibration.* Question attributes are very important as they determine the quality of the generated test papers. They could be automatically labeled based on the question content and its solution, and users' responses for supporting Web-based testing. In this research, we will investigate a content-based collaborative filtering approach for automatic question difficulty calibration.

- *Investigate a new technique for Automatic Mathematical Solution Assessment.* In this research, we focus on investigating mathematical solution assessment, which is commonly required in most mathematical assessments. The correctness of the user answer for a question will be automatically checked with the standard solution for automatic solution assessment. This research will investigate a probabilistic equivalence verification approach for automatic mathematical solution assessment.

1.4 Contributions

As a result of this research, we have developed different novel techniques for online test paper generation, parallel test paper generation, question difficulty calibration and automatic mathematical answer verification. The contributions of this research are listed as follows:

- We proposed an efficient constraint-based Divide-and-Conquer (DAC) approach which is based on constraint decomposition and multiobjective optimization for Online-TPG. The proposed DAC approach is based on the principle of constraint decomposition for effective multiobjective optimization. Based on the proposed DAC approach, three algorithms, namely DAC [176], DAC Tabu Search (DAC-TS) [175] and DAC Memetic Algorithm (DAC-MA) [178], have been implemented. The DAC-MA approach has achieved 12 times faster in runtime performance and 38% better test paper quality than the other current TPG techniques including GA, DE, TS, ACO and PSO.

- We proposed an efficient 0-1 Integer Linear Programming approach for large-scale Online-TPG, called BAC-TPG (Branch-and-Cut for Test Paper Generation) [179]. BAC-TPG is based on the Branch-and-Bound and Lifted Cover Cutting methods to

find the near-optimal solution by exploiting the sparse matrix property of 0-1 ILP. The proposed approach has achieved 29% better test paper quality than DAC-MA on the large-scale Online-TPG problem.

- We proposed an effective Multiobjective Evolutionary Co-Approximation (MECA) approach for parallel test paper generation (k-TPG). MECA is a greedy-based approximation algorithm, which explores the submodular property of the objective function for combinatorial multiobjective optimization. The proposed approach has achieved 8 times faster in runtime performance and 85% better test paper quality for parallel test paper generation than the other current techniques including k-TS, k-PSO and k-ACO.

- We proposed an effective Content-based Collaborative Filtering (CCF) approach [177] for automatic calibration of question difficulty degree. In CCF, collaborative filtering is used to predict unknown user responses from known responses of questions. And the difficulty degree of each question in the dataset is then estimated using Item Response Theory. The proposed CCF approach has achieved promising results with 40% prediction errors less than the traditional Proportion Correct Method technique.

- We proposed an effective Probabilistic Equivalence Verification (PEV) algorithm [180] for automatic mathematical solution assessment. The PEV approach is a probabilistic algorithm, which verifies the equivalence of two functions by evaluating the value of the functions at random points sampled uniformly over a sufficiently large range. PEV can avoid false negative errors of the current popular Computer Algebra Systems such as Maple and Mathematica. The PEV approach can achieve an average accuracy of 93.6% while Maple and Mathematica can only achieve about 72.4% and 64% respectively.

1.5 Organization of the Thesis

This chapter has discussed briefly the background and motivation of this research. The objectives and contributions of the research have also been given. The rest of the thesis is organized as follows.

Chapter 2 reviews the related work on multiobjective optimization techniques and automatic test paper generation techniques.

Chapter 3 discusses the question database, test paper specification, and problem statement for online test paper generation (Online-TPG).

Chapter 4 presents the proposed constraint-based Divide-and-Conquer Memetic Algorithm (DAC-MA) for Online-TPG. The performance evaluation of the proposed DAC-MA approach is also given.

Chapter 5 discusses the proposed Integer Programming approach, called BAC-TPG, for large-scale Online-TPG. The performance of the proposed BAC-TPG approach and its comparison with other TPG techniques are given. In addition, an expert evaluation of the generated test paper quality is also discussed in this chapter.

Chapter 6 discusses the proposed Multiobjective Evolutionary Co-Approximation (MECA) algorithm for parallel test paper generation (k-TPG). The performance evaluation of the proposed MECA approach and its comparison with other k-TPG techniques are also given.

Chapter 7 presents the proposed Content-based Collaborative Filtering (CCF) approach for automatic question difficulty calibration. The performance evaluation of the proposed CCF approach is also given.

Chapter 8 discusses the proposed Probabilistic Equivalence Verification (PEV) algorithm for automatic mathematical answer verification. The performance evaluation of the proposed PEV approach is also given.

Finally, Chapter 9 gives the conclusion and discusses the directions for further research work.

Chapter 2

Related Work

In this chapter, we review the related work on multiobjective optimization (MOO), 0-1 Integer Linear Programming (0-1 ILP), distributed resource allocation and automatic test paper generation.

2.1 Multiobjective Optimization

Multiobjective Optimization (MOO) [95, 199, 210] or multi-criteria optimization aims to concurrently optimize some objective functions subject to a set of constraints. Formally, a multiobjective optimization problem can be written as

$$\begin{aligned} & \min_{x} \left[f_1(x), f_2(x), \ldots, f_n(x)\right]^T \\ s.t. \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x_l \leq x \leq x_u \end{aligned}$$

where f are objective functions, g is the inequality constraint and h is the equality constraint, and x is a vector variable representing the optimization solution. Generally, Pareto solutions are promising solutions for a multiobjective problem. A Pareto solution is a solution such that there does not exist any solution that is better than it in all of the objectives.

There are various real-word applications of multiobjective optimization problems such

as process scheduling, product design, network routing, supply chain optimization, etc. Depending on the domain of the corresponding decision variables x, i.e., discrete or continuous, there are two types of MOO problems: continuous MOO and discrete MOO. Generally, there exists more effective and efficient methods for continuous MOO problems than the discrete MOO problems. It is because in the continuous domain there are special properties of the objective functions such as linear, gradient descent or convexity that can be utilized effectively for optimization. However, most of the methods for continuous MOO might not be applicable for solving discrete MOO problems.

There are several general approaches for solving continuous MOO problems including the construction of a single aggregate objective function [65, 66], Normal Boundary Intersection (NBI) method [61, 62], Normal Constraint (NC) method [160, 161], Successive Pareto Optimization (SPO) method [169], Directed Search Domain (DSD) method [77], Multiobjective Optimization using Evolutionary Algorithms (MOEA) [56, 65].

However, there seems to be no general one-size-fits-all approach, which can effectively and efficiently solve discrete MOO problems. To effectively solve discrete MOO problems, it is often necessary to understand and make use of special formulation as well as properties of a specific problem. Many discrete MOO problems such as scheduling, timetabling, and capacitated arc routing are typically formulated into either Integer Linear Programming (ILP) or 0-1 Integer Linear Programming (0-1 ILP). The formulated ILP problem can then be solved effectively and efficiently by exploiting its properties.

2.2 0-1 Integer Linear Programming

Integer Linear Programming (ILP) [36, 172, 202] is a mathematical model that optimizes a linear objective function while satisfying a system of integral linear equality and inequality constraints:

$$\begin{aligned} &\max_{x \in \mathbb{Z}^n} && c^T x \\ &\text{subject to} && \mathbf{A}\mathrm{x} \leq \mathbf{b} \end{aligned}$$

where $\mathbf{A}= [a_{i,j}]_{m \times n}$ is a $m \times n$ matrix with $a_{ij} \in \mathbb{Z}$, $c \in \mathbb{Z}^n$, $b \in \mathbb{Z}^m$, m is the number of constraints, and n is the number of variables or dimensions.

Although 0-1ILP is a NP-hard problem [204], there are several solvers available today for solving practical 0-1ILP problems. The performance depends on the dimensions n and degree of sparsity of the constraint matrix $\mathbf{A}$.

If we add the constraint $x \in \{0,1\}^n$ to ILP, then such special form is called either a 0-1 Integer Linear Programming (0-1 ILP) or Multidimensional 0-1 Knapsack Problem (MKP) [36, 87, 172, 202]:

$$\begin{aligned} &\max_{x\in\{0,1\}^n} && c^T x \\ &\text{subject to} && \mathbf{A}\mathrm{x} \leq \mathbf{b} \end{aligned}$$

Like the general ILP, 0-1 ILP is also NP-hard. In addition, it is difficult to deal with the binary constraint $x \in \{0,1\}^n$. Thus, finding a solution for a 0-1 ILP problem is more difficult than that for a general ILP problem. For example, there is a greedy based polynomial algorithm for the fractional Knapsack problem, but there may not exist a polynomial algorithm for the 0-1 Knapsack problem due to its NP-hardness. Some further discussion of the 0-1 ILP problem can be found in [36, 87, 172, 202]. Generally, there are four major optimization techniques for solving 0-1 ILP: Dynamic Programming, Heuristic Algorithms, Exact 0-1 ILP Algorithms and Approximation Algorithms.

2.2.1 Dynamic Programming

In operational research, system control and computer science, Dynamic Programming (DP) [33] is a well-known optimization algorithm which is based on the Principle of Optimality and Divide-and-Conquer algorithm design paradigm for planning and decision-making problems. DP was originally used in the 1940s by Richard Bellman to describe the process of solving problems where one needs to nest smaller problems inside larger problems in recursive form in order to find the best decisions one after another. DP solves a complex problem by breaking it down into simpler steps. It is applicable to problems exhibiting the properties of overlapping subproblems which are only slightly smaller with optimal substructure. When applied, DP can improve the runtime considerably.

DP is efficient only when the number of constraints m is either 0 or 1. For instance, DP leads to the famous Dijkstra's algorithm [60] for the shortest path problem in network routing in which there is no constraint ($m = 0$). In addition, DP also leads to an efficient algorithm for the 0-1 Knapsack problem [60] in which there is one constraint ($m = 1$). In general, DP has the computational complexity $O(n(\delta)^m)$ [36], $\delta = \max\{b_1, b_2, ..., b_m\}$ where $b_1, b_2, ..., b_m$ are entries of the vector b in the 0-1 ILP formulation. Furthermore, DP also requires about $O(n\delta)$ memory space to store intermediate results. Thus, the computational requirement has made it impractical even for a moderate value of m. Therefore, DP is not a practical optimization method particulary for large-scale 0-1 ILP where $m \geq 2$.

2.2.2 Heuristic Algorithms

In computer science, heuristic algorithms [95, 214] are widely used to optimize a multiobjective decision problem by iteratively improving a candidate solution with respect to a given aggregation function of the quality measurement. Heuristic algorithms can search a very large space of candidate solutions. Heuristic algorithms, however, do not guarantee an optimal solution to be found. Many heuristic algorithms implement some forms of stochastic optimization. A general heuristic algorithm often consists of the following two steps:

- Initial Step: It generates an initial solution which satisfies certain constraints.
- Improvement Step: It improves the quality of the initial solution. The heuristic algorithm stops when it achieves a satisfied solution or exceeds a predefined maximum number of allowable iterations.

There are several heuristic algorithms in literature such as Local Search, Genetic Algorithm, Particle Swarm Optimization, etc. They are generally classified into the following three main categories [214]:

- *Single-Solution based Algorithms*: It is a group of techniques including Local Search, Tabu Search (TS) and Simulated Annealing (SA) [11, 90]. They use local enumeration to move sequentially from a current solution to other neighborhood solutions in order to improve the solution's quality.
- *Swarm Intelligence* (SI) [44]: It is a set of population based optimization techniques based on the collective behavior of decentralized, self-organized systems such as Particle Swarm Optimization (PSO) [55, 131, 191] and Ant Colony Optimization (ACO) [71, 72]. SI is based on the interaction optimization model of a population of agents. These agents can either interact with other agents or with the environment.
- *Biologically Inspired Algorithms* [84, 95]: They are optimization models based on the evolutionary process of biological populations. Biologically inspired algorithms such as Genetic Algorithm (GA), Artificial Immune System algorithm (AIS) [115] and Differential Evolution (DE) [211] simulate its optimization process as a natural evolution phenomenon. These techniques often involve the establishing a set of biological rules as well as a set of organisms which follow these rules, and an evolutionary process which iteratively applies these rules. After many generations of new populations, some forms of the solution may arise.

- Memetic algorithms (MA) [123, 186, 187] represent one of the recent growing research areas in evolutionary computation. Inspired by the principles of natural selection, the term "memetic algorithm" was first introduced by Moscato [166] 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 [51]. Recently, there have been increasing interest in investigating simple hybrid and adaptive hybrid for tackling multiobjective optimization problems [139]. In simple hybrid, special population-based methods [66, 123, 138] or individual improvement methods [138] are designed to deal with multiobjective optimization. In adaptive hybrid, some adaptive coordinations of individual improvement methods [46, 48] are proposed for handling multiobjective optimization. Traditionally, all of these approaches are proposed for *offline* multiobjective optimization by using either weighting parameters [159] or Pareto front [138] methods. However, these approaches are computationally expensive especially when there is a high number of multiobjective constraints [65, 122, 138, 205].

In practice, heuristic algorithms have been intensively investigated to solve many 0-1 Integer Linear Programming problems such as the 0-1 Knapsack problem, Set Covering, etc. There are several major drawbacks of heuristic algorithms for solving 0-1 ILP. Firstly, it is difficult to find a good initial solution especially when there are several constraints. Secondly, there is no guarantee to always find an optimal solution because it is easily trapped in a local optimal region. However, empirical evidence has suggested that it could find an approximately good solution for small-scale problems. The quality measurement function used in heuristic algorithms is based on weighting parameters which are not easy to determine. In practice, these weighting parameters are determined empirically. Heuristic algorithms often need a large number of iterations. Hence, heuristic algorithms require intensive computation.

2.2.3 Exact 0-1 ILP Algorithms

Three global methods can be used to find the exact optimal solution based on linear programming for 0-1 ILP problems. They are the Cutting Planes method, Branch-and-Bound, and Branch-and-Cut.

The performance of these global methods depends on the algorithms used for linear programming, preprocessing techniques, and computational processing power of the computer hardware. In the early nineties, there is not much improvement on the Simplex algorithm for LP. Since the early twenties, the development of the Dual Simplex algorithm and other techniques such as Lifted Cover Cutting Planes [98] have remarkably improved integer programming techniques [128]. The runtime performance has been improved significantly. Currently, a large ILP of about 18000 variables can be solved in less than 3 minutes. However, the LP-based ILP is still not efficient in runtime performance especially for large-scale 0-1 ILP problems. In particular, the methods implemented in popular commercial optimization software such as CPLEX [121] and GUROBI [101] are ineffective to handle 0-1 ILP with more than twenty thousands of variables [164].

Among the three methods, Branch-and-Cut (BAC) [164] is most efficient as it is able to solve and prove optimality for larger set of instances than the others. BAC is a global optimization method, which is based on the Branch-and-Bound method and Cutting Planes method such as the Gomory or Fenchel Cutting Planes [36]. The main idea of the Cutting Planes method is to add extra constraints to reduce the feasible region and find the integral optimal solution. For 0-1 ILP problems with the sparse matrix property, Lifted Cover Cutting is an effective method for enhancing runtime performance. However, the BAC method suffers from several drawbacks when solving large-sized 0-1 ILP problems. It is difficult to approximate the integral optimal solution from the fractional optimal solution of the 0-1 ILP problem. In addition, the Simplex algorithm used to solve LP relaxation is also not very efficient on large-sized ILP problems. As BAC is an exact algorithm, the size of the Branch-and-Bound search tree may combinatorially explode with the number of variables. Hence, BAC generally suffers from poor runtime performance on large-sized ILP problems. Moreover, finding lifted cover cutting planes efficiently is challenging as it is NP-hard [98].

2.2.4 Approximation Algorithms

Many problems of interest in combinatorial optimization are NP-hard problems. Thus, it is unlikely to have efficient polynomial time algorithms to find an optimal solution. Researchers in computer science and operational research have developed several approaches to deal with NP-hard problems with performance guarantees. These approaches mainly fall into one of the following 2 groups. The first group consists of algorithms such as Integer Programming that can find the optimal solution but are unable to run efficiently in polynomial time for all problem instances. The second group consists of algorithms that can run in polynomial time but are unable to find the optimal solution for all instances. Local search, heuristics, meta-heuristics and evolutionary algorithms belong to this group. Interestingly, there is a special class of methods in the second group that can produce near-optimal solutions for NP-hard problems. They are called *approximation algorithms*. Generally, approximation algorithms for NP-hard problems are polynomial time heuristics that have provable guarantees on the solution quality for all instances. Typical examples of such approximation algorithms include approximate local search [20, 189], obvious local search [105], (1+1)-EA [183], etc. Such algorithms are robust to solve intractable problems that arise in many areas of computer science and beyond. The key idea of a successful approximation algorithm is to explore the combinatorial structure of NP-hard problems.

Formally, let's consider a maximization problem $\prod$ with a objective function. Let OPT(I) be the optimal value of the objective function for each instance $I \in \prod$. Let $A(I)$ be the objective value produced by $\mathcal{A}$ on the input I, where $\mathcal{A}$ is a polynomial time algorithm, which produces a feasible solution for each $I \in \prod$. We define

$$R_{\mathcal{A}}(I) = \frac{A(I)}{OPT(I)}$$

as the performance ratio of $\mathcal{A}$ on input I. We say that algorithm $\mathcal{A}$ for the problem $\prod$ has an *approximation ratio* of $\rho(n)$ if, for any input I of size n, the objective function value of the solution returned by the algorithm is lower bounded by a factor of $\rho(n)$ of the optimal value OPT(I), i.e.,

$$R_{\mathcal{A}}(I) \geq \rho(n)$$

The parameter $\rho(n)$ is also called the performance guarantee of the approximation algorithms. By definition, it is straightforward to see that $\rho(n) \leq 1$. In most typical cases, researchers have developed constant performance guarantee algorithms, i.e., $\rho(n) = c \leq 1$, for NP-hard problems.

There are two general ways [73, 220] to design approximation algorithms for 0-1 ILP. The first is called Combinatorial Approximation Algorithm, which is purely combinatorial and does not need the fractional optimal solution of the original 0-1 ILP. The second method is Linear Programming Relaxation and Rounding Algorithm, which does require the fractional optimal solution to be solved by Linear Programming techniques. Although approximation algorithms have been successfully investigated to find robust solutions for many single objective NP-hard problems, there has not been much research work done on investigating approximation algorithms for multiobjective NP-hard problems [97, 183].

Combinatorial Approximation Algorithms

Combinatorial approximation algorithms can be categorized into the following three main groups of techniques: Approximate Local Search, Greedy-based Approximation Algorithm and Primal-Dual Method.

Approximate Local Search [11, 20, 127, 135, 136, 163, 189, 201] is a pure local search algorithm, which is able to produce solutions with some performance guarantee based on well-designed local moves. Although traditional local search may be very efficient in practice for obtaining near-optimal solutions for a large class of combinatorial optimization problems, it lacks theoretical results concerning the quality of solution obtained in the worst case. Recently, there are more theoretical results of approximate local search algorithms [20, 113, 220] for the optimization problems such as Maximum Satisfiability, Uncapacitated Facility Location, k-Mean Clustering and Scheduling Jobs on Identical Parallel Machine, etc. In other words, it is possible to prove that such locally optimal solutions are close to the optimal solution. In addition, unlike approximate local search algorithms that typically run in polynomial time, traditional local search algorithms are unable to do so. Traditional local search algorithms usually require certain restrictions to the allowable local changes in order to ensure that enough progress is made during each improving step so that a locally optimal solution can be found in polynomial time.

Greedy-based Approximation Algorithms [85, 162] construct a solution in an incremental manner by greedy selection based on a well-designed criterion. In each step of the algorithm, a

better solution is constructed locally by making some decisions. Greedy-based approximation algorithms are often very fast. Take the Set Cover problem as an example, a greedy-based approximation algorithm [60] for this problem constructs a set cover by repeatedly choosing the set that minimizes the ratio of its weight to the number of currently uncovered elements it contains. Greedy-based approximation algorithms have been used successfully to solve many NP-hard problems such as Knapsack, Scheduling Jobs on Single Machine, Traveling Salesman problem with performance guarantee.

A special and important class of greedy-based approximation algorithms for combinatorial optimization is based on the concept of submodularity [88, 151]. Specifically, submodularity of the objective function is a very important property [151], which is similar to the convexity concept in the continuous domain. Submodular functions appear in various combinatorial settings such as Set Cover, Graph Cuts, and Welfare Maximization. The submodular property has been efficiently exploited to solve many problems including Set Cover [78], Graph Cuts [91, 125], Rank Function of Matroids [76, 86], Social Welfare [80], and Influence Maximization of Social Networks [130]. Recently, there are theoretical research results, which have shown that there exists effective and efficient algorithms for solving submodular objective function optimization.

Primal-Dual (PD) [92, 228] is a method for solving Integer Linear Programming problems. This method starts from a feasible solution y of the dual program and tries to find a feasible solution x of the primal program that satisfies the complementary slackness conditions. If there is such a solution x, we can find a better solution y based on its objective value. Then, this process is repeated until y equals to x, i.e., the optimal solution is achieved. This idea can also be used to design approximation algorithms for solving 0-1 ILP problems. The Primal-Dual method finds an approximate solution to the primal 0-1 ILP simultaneously with finding a feasible solution to the dual LP. To estimate the approximation ratio, it compares the approximate solution with the dual feasible solution. The Primal-Dual method is often very efficient for combinatorial problems. In practice, PD has successfully been used to solve many problems such as Network Flow Cut [92], Set Cover [228], Vertex Cover [228] and Minimum Spanning Tree [92]. It is easy to solve the dual LP with the existing LP solver. However, it is not easy to find the corresponding approximate solution of a general primal 0-1 ILP problem due to the nature of the constraints in 0-1 ILP.

Linear Programming Relaxation and Rounding Algorithms

Linear Programming (LP) [35, 54] is a mathematical model that optimizes a linear objective function while satisfying a system of linear equality and inequality constraints

$$\begin{aligned} &\max_{x \in \mathbb{R}^n} && c^T x \\ &\text{subject to} && \mathbf{A}\mathrm{x} \leq \mathbf{b} \end{aligned}$$

In general, there are two equivalent ways to express linear programming: standard form $max\ \{c^T\ x\ |Ax \leq b, x \in \mathbb{R}^n, x \geq 0\}$, or canonical form $max\ \{c^T\ x\ |Ax = b, x \in \mathbb{R}^n, x \geq 0\}$ (by introducing slack variables), where $\mathbf{A}$ is a $m \times n$ matrix, and $\mathbf{b} \in \mathbb{R}^m$.

There are several advanced numerical algorithms such as Simplex algorithm [54], Ellipsoid algorithm [35], Primal-Dual algorithm [35], or Interior Point algorithm [132] that can be used solve the LP problem. Interior Point is known to be the best method for solving LP, which can solve very large-scale LP problems with hundred thousands of variables within a few minutes.

Given an 0-1 ILP problem, we can transform it to the corresponding LP problem by relaxing the constraint to $0 \leq x_i \leq 1$. Then, a LP solver can be used to find the relaxed optimal solution called x^*. From x^*, an appropriate rounding scheme is used to construct a binary solution x^A of the 0-1 ILP problem. This technique is known as Linear Programming Relaxation and Rounding algorithm (LPRR) [113, 220]. Let x^O be the optimal solution of the original 0-1 ILP problem and $f(x)$ be the objective function of a solution x, the difference ratio $\frac{f(x^A)}{f(x^O)}$ between x^A and x^O is called the integral gap of the solution. Therefore, the smaller the integral gap is, the better the rounding scheme is. In practice, LPRR has successfully been used to solve many problems such as Network Flow Cut [35], Set Covering [220] and Vertex Cover [220] where there exists good rounding schemes to construct a binary solution x^A from x^*. However, it is not easy to find a good rounding scheme for a general 0-1 ILP problem.

2.3 Distributed Resource Allocation

This section discusses the related approaches for multiobjective optimization of a class of problems in combinatorial optimization and game theory of economics. These approaches are related to the problem on distributed resource allocation such as Social Welfare [223], Combinatorial Auctions [40], Cost Sharing [126], Machine Load Balance [221], Multiprocessor Scheduling, Cake Cutting [194], Stanta Claus problem [23], etc. In the problem setting,

the goal is to cooperatively allocate resources to different partners of the game in order to maximize a certain objective function under some constraints. The two objective functions are to maximize either total benefit of all partners or fairness of benefit allocation among all partners.

To maximize either total benefit of all partners, the objective functions often have the submodular property [151], which is exploited to derive high-quality solutions. Most of the proposed approaches are approximation algorithms, which are capable of finding good solutions with performance guarantee. However, these approaches are unable to deal with the multiple knapsack constraints. In addition, these approaches often require the fractional optimal solution of the linear programming for the rounding steps. This is often computational expensive due to the worst case runtime performance of linear programming algorithms of about $O(n^6)$.

To maximize the fairness of benefit allocation for all partners, most of the proposed approaches are approximation algorithms, which aim at maximizing the minimal benefit received by an unlucky parter in the competitive game. However, most of these approaches are proposed for unconstrained allocation and they are unable to solve the multiple knapsack constraints. Moreover, these approaches also require computationally expensive fractional optimal solutions to be solved by linear programming.

2.4 Automatic Test Paper Generation

In this section, we review existing optimization techniques used for automatic test paper generation, which can be categorized into two main groups: linear programming-based integer programming (LP-based IP), Dynamic Programming (DP) and heuristic-based methods. These heuristic-based methods include Tabu Search (TS), Genetic Algorithm (GA), Artificial Immune System (AIS), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO).

LP-based IP, which was proposed in 1986 by Adema et al. [12, 13], used the LANDO program to solve the 0-1 ILP of TPG. It is similar to our proposed approach because of the use of linear programming and branch-and-bound. In [42, 43], Boekkooi-Timminga attempted to combine ILP with heuristics to improve runtime performance for multiple test paper generation. Although these approaches have rigorous mathematical foundations on optimization, they can only solve TPG for very small datasets of about 300-600 questions

due to the limitations of the state-of-the-art optimization methods at that time. An in-depth review of the LP-based IP for TPG can be found in [83].

Dynamic Programming was proposed in [117] to deal with the single TPG problem. The DP technique optimizes constraint satisfaction specified by an objective function with two variables. The test paper is then constructed and optimized incrementally based on the recursive optimal relation of the objective function. As the proposed technique is based on dynamic programming, it requires lots of memory space and computation time. In addition, the quality of the generated test paper is also not satisfactory.

For heuristic-based methods, Theunissen [215] used a heuristic based on the characteristics of question item information function to optimize the objective function. Later, Luecht [152] proposed an efficient heuristic to solve TPG on a dataset with 3000 questions. However, these heuristic-based methods were proposed to solve TPG for small datasets and are ineffective for larger datasets.

Since 2003, there has been a revived interest for TPG on larger datasets of about 3000-20000 questions by using modern heuristic methods.

In [120], Hwang et al. proposed a Tabu Search technique to solve the TPG problem. An objective function is defined based on multi-criteria constraints and weighting parameters for test paper quality. To optimize test paper quality, TS starts with an initial random test paper and then moves from the current test paper to another iteratively such that the evaluation of the objective function is improved. For efficiency, TS uses a fixed length tabu list which is the key component to restrict the number of potential candidates. However, the weighting parameters and the length of the tabu list need to be determined experimentally for each test paper specification. The performance of the TS technique based on the average discrimination degree is slightly better than other heuristic approaches.

In addition, biologically inspired algorithms such as Genetic Algorithm, Differential Evolution and Artificial Immune System have also been investigated for TPG. In [119], Hwang et al. proposed Genetic Algorithm for test paper generation. GA generates quality papers by optimizing a fitness ranking function which is defined based on multi-criteria constraints and weighting parameters. GA is based on the principle of population evolution. To optimize test paper quality, GA starts with an initial population of K random test papers. At each iteration, the one-cut-point crossover method and the randomized mutual method are used to create new instances in the population. All instances will then be re-selected for the population of next generation by the roulette wheel method based on the fitness ranking

value. The performance results have shown that GA is quite efficient on runtime based on 1500 generations, though it may not be sufficient for the search process converging to the near-optimal solution.

In [197], Wang et al. proposed a Differential Evolution technique for test paper generation. In general, 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 addition, Wang et al. have also discussed the online requirement for test paper generation. However, the proposed optimization technique has not been investigated for satisfying the online test paper generation requirement. The performance results have shown that DE is quite efficient on small-sized datasets and performs better than GA.

Lee et al. [147] proposed an Artificial Immune System technique for test paper generation. AIS is a population based optimization technique which uses the clonal selection principle to deal with the highly similar antibodies for elitist selection in order to maintain the best test papers for the next generation. The performance results based on a very small dataset with 1000 questions have shown that AIS outperforms GA in terms of test paper quality.

Recently, swarm intelligence algorithms such as Particle Swarm Optimization were investigated for TPG. In [217], Particle Swarm Optimization was proposed to generate test papers. The proposed PSO technique generates quality papers by optimizing a fitness function which is defined based on multi-criteria constraints. To optimize the test paper quality, PSO starts with an initial population of random test papers. Each test paper is considered as a particle in the swarm with its position and velocity. At each iteration, the best particle is identified based on the fitness function and all other particles will move towards the direction of the best particle by updating their positions and velocities. The quality of the generated test papers in PSO is better than that using GA.

Generally, these biologically inspired algorithms and swarm intelligence algorithms require the values for the weighting parameters and population size for each test paper generation which are not easy to determine.

2.5 Parallel Test Paper Generation

In [118], a TS approach was proposed to solve the k-TPG problem, which considers only the fairness maximization objective under multicriteria constraints. To solve this problem, the objective function is formulated as to minimize the maximal mutual difference of the average

discrimination degree of two arbitrary test papers

$$\text{minimize} \max_{\forall 1 \leq i \leq j \leq k} |f(P_i) - f(P_j)|$$

where $f(P_i), i = 1..k$ is the average discrimination degree of test paper P_i. Although this formulation seems to be reasonable, it does not have any effective method for optimization. Thus, the TS approach is purely heuristic search in a very large search space. As a result, the quality of the generated test papers is poor and the computational runtime is very expensive.

In [112], Ho et al. proposed a PSO approach to solve the same k-TPG problem as in [118] with the similar objective function. The performance results showed that this approach outperforms GA algorithm with small values of $k = 2, 3, 4$.

In [116], Ant Colony Optimization was proposed for multiple test paper generation. Different from the previous work on k-TPG, this work considered both total quality and fairness quality maximization. The objective function is formulated as follows:

$$\text{maxmize} \sum_{l=1}^{k} f(P_l) - \sum_{\forall 1 \leq i \leq j \leq k} |f(P_i) - f(P_j)|$$

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 [118, 112], 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 of ACO, the m ants are dispatched for stochastically constructing their own solutions to improve the objective function. The quality of the generated test papers of ACO is better than that of using GA.

2.6 Summary

In this chapter, we have reviewed the different multiobjective optimization techniques that are widely used to solve the 0-1 ILP problem. Each algorithm has it own merits and limitations. Depending on the performance objective we want to achieve, we can design an algorithm based on the heuristic techniques, 0-1 ILP, or approximation algorithm. The heuristic techniques are

able to produce relatively good solutions in short runtime. The exact 0-1 ILP techniques are able to produce optimal solutions but with long runtime. The approximation algorithms are able to produce near-optimal solutions with performance guarantee in polynomial runtime. In addition, this chapter has also reviewed the current techniques for automatic test paper generation and parallel test paper generation.

Chapter 3

Online Test Paper Generation

Online test paper generation aims to find an optimal test paper from a question database based on multiple assessment criteria. It is also required to satisfy the online runtime requirement. In this chapter, we discuss the question database, test paper specification, problem statement, and computational requirement for online test paper generation (Online-TPG).

3.1 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.

Table 3.1: Example of 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_1
q_7	,...,.	,...,.	5	2	3	c_2	y_1
q_8	Evaluate $\int \cos^3 x\, dx$	$\sin x - \frac{1}{3}\sin^3 x + C$	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

(b) Topic Table

$\mathcal{C}$_id	*name*
c_1	Differentiation
c_2	Integration
c_3	Limits and Continuity
c_4	Complex Numbers
c_5	Probability
c_6	Infinite Sequences and Series
c_7	Vectors and Analytic Geometry
c_8	Matrices and Determinants

(c) Question Type Table

$\mathcal{Y}$_id	*name*
y_1	Multiple Choice
y_2	Fill in Blank
y_3	Long 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.

Note that the discrimination degree and difficulty degree attributes here refer to the classical Item Response Theory (IRT) [27] definitions. Table 3.1 shows a sample Math question dataset, which consists of three main tables: question table, topic table and question type table. This dataset has 13 questions with their labeled attributes.

There are two possible ways to construct large-scale question datasets for Web-based testing. It can be constructed by gathering questions from past tests and examinations on subjects such as TOEFL[1] and GRE[1] accumulatively. In addition, it can also be constructed by gathering freely available questions from online educational websites such as Khan Academy

[1] http://www.ets.org

or Question Answering (Q&A) websites such as The Art of Problem Solving Portal[2].

The large pool of questions has posed a great challenge on labeling all question attributes accurately and automatically. In this research, we assume that question attributes are correctly calibrated. In fact, with the advancement in educational data mining techniques [30], it is feasible to automatically label all the attributes of each question. Automatic text categorization techniques such as Support Vector Machine (SVM) can be used for automatic topic classification of questions [50]. However, human labeling of topics for each question is still needed in the training phase. To calibrate the other attributes, we can use the historical correct/incorrect response information from students. These response information as well as other important information such as question time can be gathered automatically through the students' question answering activities [142] over a period of time. However, it is more difficult to calibrate the discrimination degree and the difficulty degree attributes due to missing user responses on certain questions. To overcome this, in this research, we propose to apply the collaborative filtering technique to predict missing user responses and use the IRT model to calibrate these 2 attributes automatically [177]. In addition, we also propose an effective method to calibrate new questions, which do not have any student response information. As such, automatic labeling of question attributes for large-scale question datasets can be achieved.

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. If the user does not specify the number of questions, this parameter will be left as null in $\mathcal{S}$ ($N = \emptyset$).

- *Total time T:* It is the total time specified for the test paper P.

$$T = \sum_{i=1}^{N} t_i$$

  where t_i is the time specified for question q_i.

[2]http://www.artofproblemsolving.com/Forum/portal.php?ml=1

- *Average difficulty degree D:* It specifies the average difficulty degree for all the questions in the test paper P.

$$D = \frac{\sum_{i=1}^{N} d_i}{N}$$

where d_i is the difficulty degree of question q_i.

- *Topic distribution C:* It is a set of pairs $C = \{(c_1, pc_1), (c_2, pc_2), ..., (c_M, pc_M)\}$, where c_i $(i = 1, 2, ..., M)$ is the unique topic's *id*, and pc_i is either a real value representing the corresponding proportion (%) of topic c_i or an integer representing the number of questions of topic c_i in P. If the number of questions N is not specified, pc_i is the topic proportion value for the corresponding topic c_i, where $\sum_{i=1}^{M} pc_i = 1$. If N is specified, pc_i is the number of questions of topic c_i, where $\sum_{i=1}^{M} pc_i = N$. A specification on topic c_i using the number of questions can be easily converted to the corresponding proportion value.

- *Question Type distribution Y:* It is a set of pairs $Y = \{(y_1, py_1), (y_2, py_2), ..., ((y_K, py_K)\}$, where y_j $(j = 1, 2, ..., K)$ is the unique question type *id*, py_j is either a real value representing the corresponding proportion (%) or an integer representing the number of questions of type y_j in P. If the number of questions N is not specified, py_j is the proportion value for a question type y_j, where $\sum_{j=1}^{K} py_j = 1$. If N is specified, py_j is the number of questions of type y_j, where $\sum_{j=1}^{K} py_j = N$. A specification on question type y_j using the number of questions can be easily converted to the corresponding proportion value.

Besides these attributes, which are specified explicitly by users, the specification $\mathcal{S}_\mathcal{P}$ has the implicit constraint on the average discrimination degree E

$$E = \frac{\sum_{i=1}^{N} e_i}{N}$$

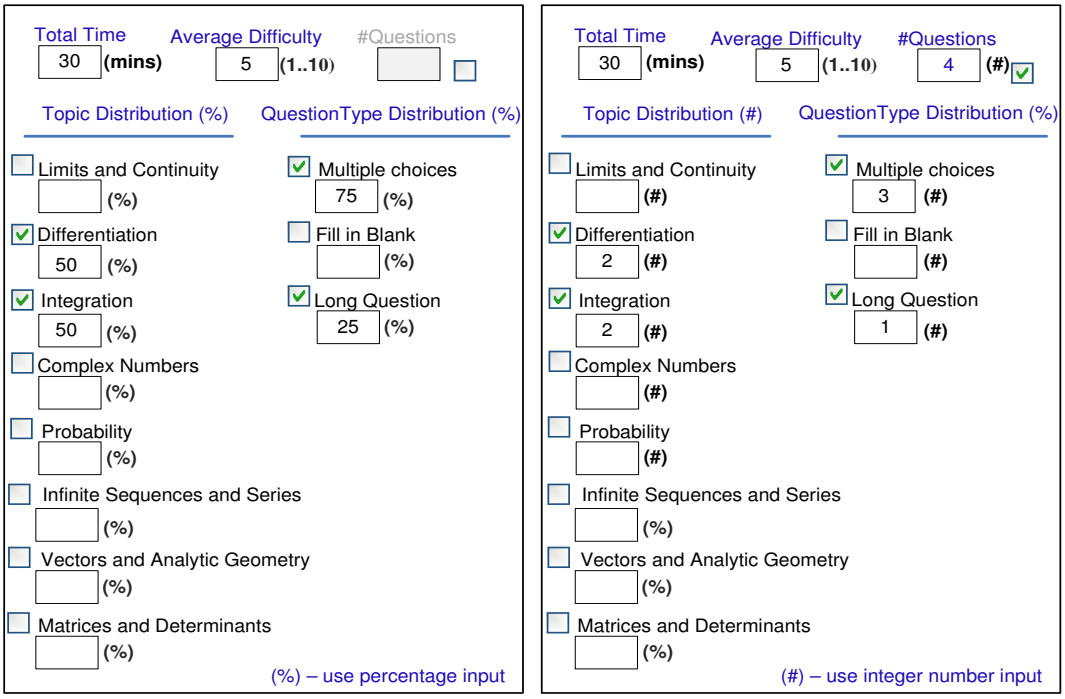
where e_i is the discrimination degree of question q_i. It implies an important expectation on maximizing E when satisfying the remaining specified constraints on attributes N, T, D, C, and Y during the test paper generation process. Table 3.2 summarizes the notations used for test paper specification.

Example 3.1. Figure 3.1 shows the two ways for test paper specification. For example, a user may specify the proportion of questions as shown in $\mathcal{S}_1 = \langle \emptyset, 30, 5, \{(c_2, 0.5), (c_3, 0.5)\},$

Table 3.2: Notations

Symbol	Explanation
$\mathcal{S}_\mathcal{P}$	specification of test paper P
N, M, K	number of questions, topics, question types in test paper P
T	total completion time of test paper P
D	average difficulty degree of test paper P
C	topic distribution in test paper P
Y	question type distribution in test paper P
E	average discrimination degree of test paper P
c_i	identifier of i^{th} topic of test paper P
pc	topic proportion of test paper P
y_i	identifier of i^{th} question type of test paper P
py	question type proportion of test paper P

$\{(y_2, 0.75), (y_3, 0.25)\}\rangle$, or he may specify the number of questions as in $\mathcal{S}_2 = \langle 4, 30, 5, \{(c_2, 0.5), (c_3, 0.5)\}, \{(y_2, 0.75), (y_3, 0.25)\}\rangle$.



(a) Specification using Proportion (S_1) (b) Specification using Number of Questions (S_2)

Figure 3.1: Test Paper Specification

3.3 Problem Definition

Online test paper generation is a combinatorial constraint optimization problem. It consists of two kinds of requirements: test paper generation and online generation.

$$
\begin{array}{lll}
\textit{maximize} & \sum_{i=1}^{n} e_i x_i / \sum_{i=1}^{n} x_i & \text{(Average Discrimination Degree)} \\
\text{subject to} & \sum_{i=1}^{n} x_i = N & \text{(Number of Questions)} \quad (3.1) \\
& \sum_{i=1}^{n} t_i x_i = T & \text{(Total Time)} \quad (3.2) \\
& \sum_{i=1}^{n} d_i x_i / \sum_{i=1}^{n} x_i = D & \text{(Average Difficulty Degree)} \quad (3.3) \\
& \sum_{i=1}^{n} r_{iu} x_i / \sum_{i=1}^{n} x_i = pc_u \quad \forall u = 1..M & \text{(Topic Distribution)} \quad (3.4) \\
& \sum_{i=1}^{n} s_{iv} x_i / \sum_{i=1}^{n} x_i = py_v \quad \forall v = 1..K & \text{(Question Type Distribution)} \quad (3.5) \\
& x \in \{0,1\}^n & \text{(Binary Value)} \quad (3.6)
\end{array}
$$

Figure 3.2: The 0-1 Fractional Programming Formulation of TPG

3.3.1 Single Test Paper Generation

Given a test paper specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$, the test paper generation process aims to find a subset of questions from a question dataset $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ 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 $\mathcal{S}_P = \mathcal{S}$.

It is often convenient to model a combinatorial constraint optimization problem in Integer Linear Programming (ILP) [36, 172, 202]. The underlying mathematical optimization formulation helps to analyze the computational hardness. To facilitate a deep understanding of the problem as well as finding efficient solutions, the TPG problem can be formalized as a 0-1 Fractional Programming (0-1 FP) problem [36, 172, 202]. Using the question attributes and the user specification $\mathcal{S}$, Figure 3.2 gives the optimization problem for test paper generation.

In Figure 3.2, constraint (3.1) is the constraint on the number of questions, where $x_i \in \{0, 1\}$ is a binary variable associated with question $q_i, i = 1, , ..., ., n$, in the dataset. Constraint (3.2) is the total time constraint. Constraint (3.3) is the average difficulty degree constraint. Constraint (3.4) is the topic distribution constraint. The relationship of a question $q_i, i = 1, , ..., ., n$, and a topic $c_u, u = 1, , ..., ., M$, is represented as r_{iu} such that $r_{iu} = 1$ if question q_i relates to topic c_u and $r_{iu} = 0$ if otherwise. Constraint (3.5) is the question type distribution constraint. The relationship of a question $q_i, i = 1, , ..., ., n$, and a question type $y_v, v = 1, , ..., .K$, is represented as s_{iv} such that $s_{iv} = 1$ if question q_i relates to question type y_v and $s_{iv} = 0$ if otherwise.

This 0-1 FP formulation, however, is not a standard optimization problem because there are variables in the denominators of constraints (3.3), (3.4) and (3.5). In order to transform it into a standard 0-1 ILP problem, these denominators can be eliminated by some standard

algebraic operations. As a result, we have the standard 0-1 ILP problem which is shown in Figure 3.3. The floor operator in the constraints (3.9)-(3.11) is due to the requirement of the standard 0-1 ILP formulation.

$$
\begin{aligned}
&\textit{maximize} && \textstyle\sum_{i=1}^{n} e_i x_i && && \\
&\text{subject to} && \textstyle\sum_{i=1}^{n} x_i = N && && (3.7)\\
& && \textstyle\sum_{i=1}^{n} t_i x_i = T && && (3.8)\\
& && \textstyle\sum_{i=1}^{n} d_i x_i = \lfloor DN \rfloor && && (3.9)\\
& && \textstyle\sum_{i=1}^{n} r_{iu} x_i = \lfloor pc_u N \rfloor && \forall u = 1..M && (3.10)\\
& && \textstyle\sum_{i=1}^{n} s_{iv} x_i = \lfloor py_v N \rfloor && \forall v = 1..K && (3.11)\\
& && x \in \{0,1\}^n && && (3.12)
\end{aligned}
$$

Figure 3.3: The Standard 0-1 ILP Problem

3.3.2 Online Generation

Although test paper generation is categorized as a *Multiobjective Optimization* (MOO) [95, 148] problem, the online requirement has not been well-studied. In contrast to the conventional multiobjective optimization problems such as timetabling [64, 200] and job-shop scheduling [28, 41, 190], where the optimization process can be done offline, it is important to note that the test paper generation process occurs online where a 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 test paper generation process has to be completed within τ minutes. The parameter τ is just an arbitrary value. It is not used as a parameter in our algorithm. Therefore, it could be set (30 seconds, 1 minute, 2 minutes, etc.) according to user expectation on runtime requirement. Although two minutes may be long, it may still be acceptable due to the NP-hardness of the TPG problem. We refer the Online-TPG problem as an *online multiobjective 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.

The primary objective of Online-TPG is to efficiently generate a test paper based on a user's specification and satisfy the online constraint. In view of this, Online-TPG is similar in spirit to online querying of relational databases. In online querying, when a user submits a query to a Relational Database Management System (RDBMS), the RDBMS processes and

$$
\begin{aligned}
\textit{maximize} \quad & 7x_1 + 5x_2 + 4x_4 + 7x_7 + 5x_8 && \text{(Average Discrimination Degree)} \\
\text{s.t.} \quad & x_1 + x_2 + x_4 + x_7 + x_8 = 4 && \text{(Number of Question)} \\
& 5x_1 + 5x_2 + 4x_4 + 5x_7 + 15x_8 = 30 && \text{(Total Time)} \\
& 2x_1 - 2x_2 - x_4 - x_7 + x_8 = 0 && \text{(Average Difficulty Degree)} \\
& x_1 + x_2 + x_4 - x_7 - x_8 = 0 && \text{(Topic Distribution)} \\
& x_1 + x_2 - 3x_4 + x_7 - 3x_8 = 0 && \text{(Question Type Distribution)} \\
& x_i \in \{0, 1\}, i = 1, 2, 4, 7, 8 && \text{(Binary Value)}
\end{aligned}
$$

Figure 3.4: An Example of Standard 0-1 ILP Formulation of TPG

returns a set of data tuples that matches with the user's query within acceptable response time. Similarly, when a user submits a test paper specification, Online-TPG should return a set of questions in a test paper that satisfies as much as possible the expected user specification within acceptable response time. While matching tuple-to-query in RDBMS is quite straightforward, finding a set of questions that satisfies the user specification of Online-TPG with the online requirement is challanging due to the combinatorial explosion of all possible solutions and complex constraints.

Example 3.2. Given $\mathcal{S} = \langle 4, 30, 5, \{(c_2, 0.5), (c_3, 0.5)\}, \{(y_2, 0.75), (y_3, 0.25)\}\rangle$ as a specification, we would like to generate a test paper P that satisfies this specification using the Math dataset given in Table 3.1. We associate each question with a binary variable x_i, $i = 1, ..., 12$. However, we eliminate inappropriate variables $x_3, x_5, x_6, x_9, x_{10}, x_{11}$, and x_{12} because they cannot satisfy the specification $\mathcal{S}$. Here, we formulate the problem as a 0-1 Fractional ILP optimization problem with 5 binary variables as shown in Figure 3.4.
A possible solution for such specification is $P = \{q_1, q_2, q_7, q_8\}$, which has an average discrimination degree $E = 6$ and specification $\mathcal{S}_P = \langle 4, 30, 5, \{(c_2, 0.5), (c_3, 0.5)\}, \{(y_2, 0.75), (y_3, 0.25)\}\rangle$. In fact, from the formulation, P is the best solution as it has the maximum average discrimination degree and exactly matches the user specification among 2^5 (i.e. 32) possible solutions.

3.4 Computational Hardness

Although some previous studies [117, 120] attempted to argue that automatic test paper generation (TPG) problem is NP-hard, we found that the constraint satisfaction of TPG is also NP-hard. In this section, we analyze the computational hardness of constraint satisfaction of the TPG problem. Specifically, in [133, 204], they show that solving a multiple 0-1 equation

problem of the form: $Ax = b$ where A is a $m \times n$ matrix, b is a vector of size m and x is a 0-1 vector of size n, is NP-hard in general. As a result, the constraint satisfaction of TPG is NP-hard because it is equivalent to the multiple 0-1 equations problem. Therefore, it is clear that if the constraint satisfaction problem of the TPG is NP-hard, then TPG is also NP-hard.

Corollary 3.1. Finding an optimal test paper of the TPG problem is NP-hard.

Since the TPG problem is NP-hard, we cannot expect to have a practical and efficient solution, which is both correct and fast.

3.5 Summary

In this chapter, we have discussed the question dataset and test paper specification. We have also defined the Online-TPG as a combinatorial constraint optimization problem with online requirement. Moreover, we have analyzed the computational hardness of the TPG problem.

Chapter 4

Divide-and-Conquer Memetic Algorithm for Online-TPG

In this chapter, we discuss the proposed constraint-based Divide-and-Conquer (DAC) approach which is based on constraint decomposition and multi-objective optimization for online test paper generation (Online-TPG). In this approach, the set of constraints are decomposed into two independent subsets of constraints which can then be solved effectively. Three algorithms, namely DAC, DAC Tabu Search (DAC-TS) and DAC Memetic Algorithm (DAC-MA), have been implemented for Online-TPG.

4.1 Divide-and-Conquer for Optimization

Divide and Conquer (DAC) [17, 60, 137, 207] refers to an important class of algorithm design paradigm in which one breaks the input into several parts, solves the problem in each part recursively and then combines the solutions of these subproblems in order to obtain an overall solution for the original problem. Due to the recurrence relation in the runtime of the original problem and its subproblems, the DAC paradigm often helps in the discovery of efficient algorithms for difficult problems. More specifically, analyzing the runtime of a DAC algorithm generally involves solving a recurrence relation that bounds the runtime recursively in terms of runtime of smaller instances. DAC has been successfully applied as the basis technique for designing efficient algorithms for many kinds of problems, such as Multiplication of Large Numbers [60, 140], Gaussian Elimination [25], Fast Fourier Transform (FFT) [60], Arithmetic Computation [111], Skyline Computation [45] and Data Stream Processing [99, 170]. In all

these examples, DAC has led to an improvement in the asymptotic computational cost of the best solution.

Besides the above mentioned examples, constraint-based DAC [22, 67, 137] has also been applied successfully to solve many optimization problems with sophisticated constraints such as Linear Programming (LP) [60] and Integer Linear Programming (ILP) [172]. Constraint-based Divide-and-Conquer is based on the principle of constraint decomposition for tackling complex optimization problems. It reduces the complexity in constraint satisfaction, thereby enhancing the quality of solutions.

4.2 Proposed DAC-MA Approach

In this section, we propose a constraint-based Divide-and-Conquer (DAC) approach for Online-TPG. As most of the constraints of the TPG problem are in the form of linear equality constraints, we can decompose the set of constraints into two independent subsets of constraints, namely *content constraints* and *assessment constraints*, which can then be solved separately and effectively. As such, we can enhance the solution quality in constraint satisfaction, and eliminate the need of using a complex objective function with its weighting parameters for multiobjective optimization that are generally not easy to determine. In the test paper specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$ shown in Figure 3.3, the content constraints include constraint (3.10) on topic distribution C and constraint (3.11) on question type distribution Y, whereas the assessment constraints include constraint (3.8) on total time T and constraint (3.9) on average difficulty degree D.

The proposed DAC approach 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.

Based on the constraint-based DAC approach, we propose the following three algorithms for Online-TPG: DAC, DAC Tabu Search (DAC-TS) and DAC Memetic Algorithm (DAC-MA). As these three algorithms are quite similar in their designs for online test paper generation, and DAC-MA has achieved better performance than the other two algorithms, we focus on discussing the constraint-based Divide-and-Conquer Memetic Algorithm (DAC-MA)

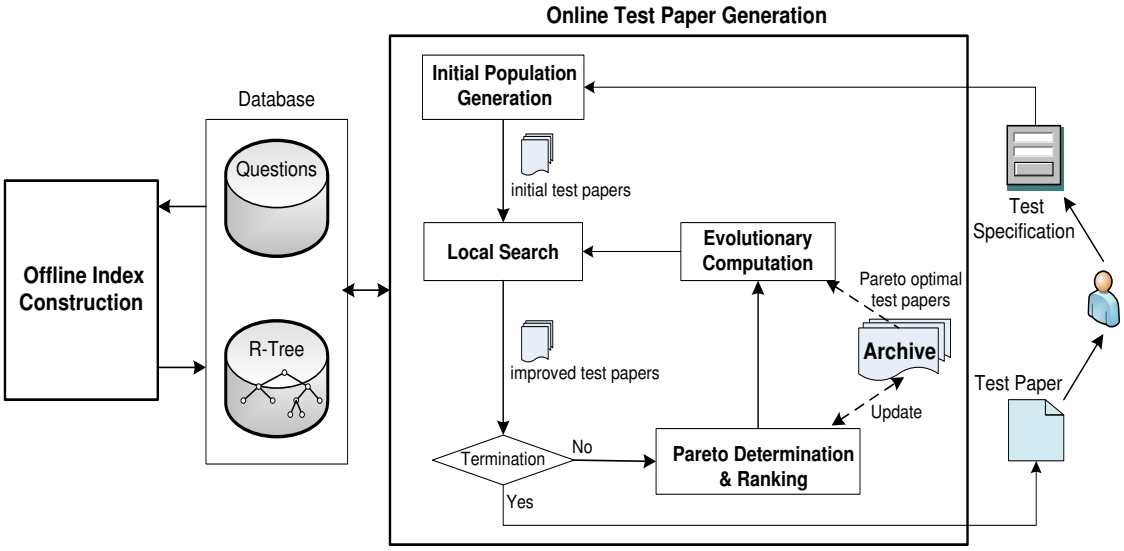


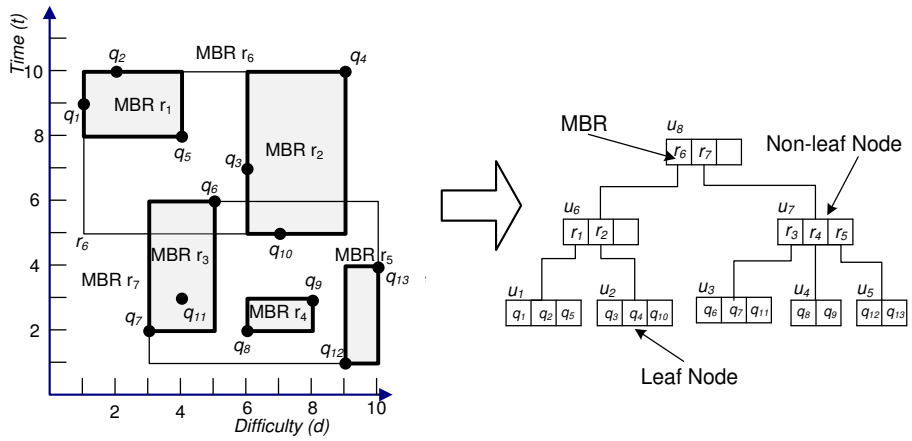
Figure 4.1: The Proposed DAC-MA Approach

algorithm in this chapter. For the other two algorithms, please refer to the published papers on DAC [176] and DAC-TS [175] respectively.

The proposed DAC-MA approach is shown in Figure 4.1. As a synergy of the diversification process in evolutionary computation with the intensification process in local search improvement, Memetic Algorithms (MA) [93, 123, 166, 185, 186, 187] have been widely used for solving NP-hard multiobjective optimization problems. It is able to converge to high quality solutions more efficiently than their conventional counterparts. In the proposed DAC-MA approach, the set of constraints are decomposed into two independent subsets of constraints which can then be solved effectively by evolutionary computation and local search of MA.

4.3 Offline Index Construction

In the Offline Index Construction process, we use an effective 2-dimensional data structure, called R-Tree [32, 155], to store questions based on the time and difficulty degree attributes (see Figure 4.2(a)). R-Tree has been widely used for processing queries on 2-dimensional or 3-dimensional spatial databases. As there is no specific 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 [32], with some modifications on index construction in order to enhance efficiency. Each leaf node in a R-Tree is a Minimum Bounding Rectangle (MBR) which has between $0.4F$ and F data points, where F is a parameter which has a value of at least 3. MBR is the smallest rectangle in the spatial representation that covers



(a) Spatial Representation of Questions

(b) R-Tree

Figure 4.2: R-Tree Index Construction for the Math Dataset

all data points located in the leaf node. Each non-leaf node has between $0.4F$ and F child nodes which contain MBRs at the lower level. As compared with the traditional R-Tree, we construct the R-Tree based on the following four modified operations:

- *Insertion:* To insert a question q into a R-Tree, we add q to a leaf node u by following a single path from the root to the leaf . There are two important issues to consider when updating the R-Tree structure by the insertion operation: subtree selection and overflow handling.

- *Subtree selection:* It minimizes the perimeter of a MBR when inserting a question point q.

- *Overflow handling:* If a node u overflows, it is split. Then, a new child of $parent(u)$ is created. If the $parent(u)$ overflows, it is also split. This process will propagate upwards recursively.

- *Node splitting:* We split the node which contains a set S of $F + 1$ points into disjoint subsets S_1 and S_2 such that: (1) $S_1 \bigcup S_2 = S$, $|S_1| \geq \lambda F$ and $|S_2| \geq \lambda F$, where $\lambda = 0.4$ is the minimal number of question points in each node; and (2) the perimeter sum of the MBRs S_1 and S_2 is the smallest.

Figure 4.2(b) illustrates the R-Tree constructed from the Math dataset given in Table 3.1.

Algorithm 4.1: Divide_And_Conquer_Memetic_Algorithm

Input: $\mathcal{S} = (N, T, D, C, Y)$ - 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 **Initial_Population_Generation** $\mathcal{I} \leftarrow \{P_1, .., P_{K_{pop}}\}$; /* content constraint */
3 **while** *Termination condition is satisfied* **do**
4 **foreach** P_i ***in*** $\mathcal{I}$ **do** /* assessment constraint */
5 $P_i' \leftarrow$ **Local_Search(**$\mathcal{S}$**,** P_i**,** $\mathcal{R}$**)**;
$\mathcal{A}_r \leftarrow \mathcal{A}_r \bigcup P_i'$; /* add new test papers into Archive */
6 **Pareto_Determination_And_Ranking**($\mathcal{A}_r$); /* Update the Archive */
7 $\mathcal{I}_{new} \leftarrow$ **Evolutionary_Computation**($\mathcal{I}$); /* new population generation */
8 **return** $P \leftarrow \underset{P_i \in \mathcal{A}_r}{\operatorname{argmin}} f(P_i)$

4.4 Online Test Paper Generation

In the Online Test Paper Generation process, an initial population of test papers is first generated by satisfying the content constraints. Next, it performs local search to improve the quality of test papers by minimizing the assessment constraint violations and selecting test papers that have the maximum average difficulty degree. 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 Figure 4.1, the Online Test Paper Generation process consists of 4 major steps: Initial Population Generation, Local Search, Pareto Determination & Ranking and Evolutionary Computation. Algorithm 4.1 presents the overall DAC-MA approach for Online-TPG.

4.4.1 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 Figure 3.3, the lowest common denominator A for all A_l, $l = 1..M$, and lowest common denominator B for all B_j, $j = 1..K$,

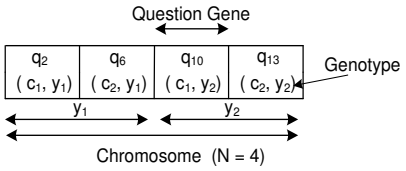


Figure 4.3: Test Paper Chromosome and Question Gene

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 [202] 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 $k = 1, 2, ...$ and check 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, 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 topic. Figure 4.3 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 diversity 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 [63], which provides a solution to count the number of all possible genotypes with duplications. By

Table 4.1: Population Size Determination

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

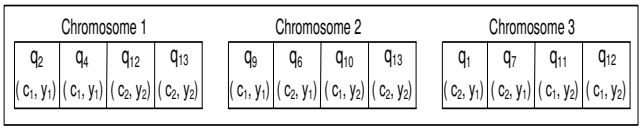


Figure 4.4: Initial Population Generation

eliminating the duplications, the number of all distinguishing genotypes 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 4.1 gives the population size K_{pop} according to the number of questions N based on [63].

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 4.4 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.

4.4.2 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 4.1 (Assessment Constraint Violation). Given a test paper specification $\mathcal{S} = \langle N, T, D, C, Y\rangle$. Let P be a generated test paper with specification $\mathcal{S}_P = \langle N, T_P, D_P, C_P, Y_P\rangle$. 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:

$$\text{Total Time Constraint Violation: } \triangle T(\mathcal{S}_P, \mathcal{S}) = \frac{T_P - T}{T}$$

$$\text{Average Difficulty Degree Constraint Violation: } \triangle D(\mathcal{S}_P, \mathcal{S}) = \frac{D_P - D}{D}$$

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 $|\triangle T(\mathcal{S}_P, \mathcal{S})| \leq \alpha$ and $|\triangle D(\mathcal{S}_P, \mathcal{S})| \leq \beta$, 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 4.2 (Fitness). Given a test paper specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$. 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 4.2 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 a better question q_m of 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 $f(P_1)$. 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 4.2: 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 $\mathcal{P} \leftarrow \{P_0\} \leftarrow \underset{P_1 \in \mathcal{P}}{\operatorname{argmin}} f(P_1)$; `/* best move */`

10 **return** $P_1 \leftarrow P_0$

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. Exhaustive search on a large question list is computationally 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 $\mathcal{S} = \langle N, T, D, C, Y \rangle$. 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:

$$T_1 = T_0 + t_m - t_k \tag{4.1}$$

$$D_1 = D_0 + \frac{d_m}{N} - \frac{d_k}{N} \tag{4.2}$$

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.

Let's consider the total time violation of P_0. If $|\triangle T(\mathcal{S}_{P_0}, \mathcal{S})| = \frac{|T_0 - T|}{T} \geq \alpha$ and $T_0 \leq T$, where α is the predefined threshold for constraint satisfaction on total time, we can select q_m

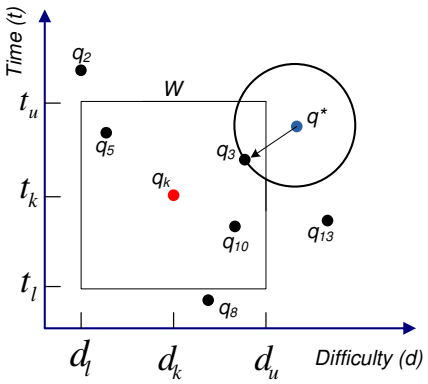


Figure 4.5: The 2-dimensional Region W and Best Question Selection

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 (4.1) 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 $|\triangle T(\mathcal{S}_{P_0}, \mathcal{S})| = \frac{|T_0 - T|}{T} \geq \alpha$ 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 $d_l = d_k + N(D - D_0) - \beta ND$, $d_u = d_k + N(D - D_0) + \beta ND$, 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 4.5 shows the region W of questions for substitution.

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 $f(t, d)$ as follows:

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

From Equations (4.1) and (4.2), we have

$$\begin{aligned} T_1 - T_0 &= t - (T - T_0 + t_k) = t - t^* \\ D_1 - D_0 &= d - (ND - ND_0 + d_k) = d - d^* \end{aligned}$$

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 4.5 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) [196] with the R-Tree. BFS recursively visits the nearest question whose MBR (Minimum Bounding Rectangle) is close to q^*. For efficiency, BFS uses a memory-resident min-heap $\mathcal{H}$ [60] 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 $O(\log N)$, we can improve the time complexity of the scanning step to $O(\log N)$.

4.4.3 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 [66, 233] 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 4.3 (**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 4.4 (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 multiobjective optimization for a long time. Although several methods have been proposed, they are not efficiently scalable due to high runtime complexity such as $O(dn^2)$ [66] and $O(dn^3)$ [233], 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 [146], 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 [146] 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.

4.4.4 Evolutionary Computation

Evolutionary Computation 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.

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 [26], 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

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

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 [26], 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 Figure 4.6, 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

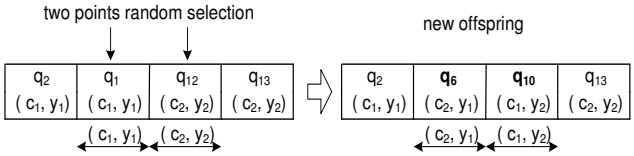


Figure 4.6: Mutation Operation

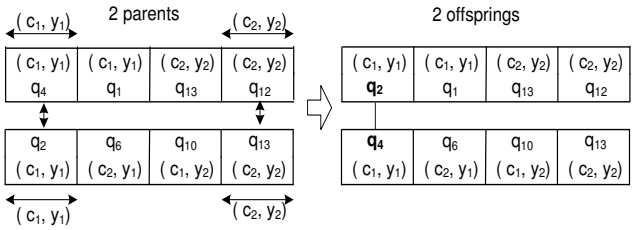


Figure 4.7: Crossover Operation

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 [26] based on two chromosomes. The crossover operator exchanges a pair of question genes that have a common topic-question type of two parent chromosomes to render two child chromosomes. Figure 4.7 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.

4.4.5 Termination

After evolutionary computation, a new generation of test paper population is generated. The test paper generation 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.

4.4.6 Computational Complexity

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

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 [60]. 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 $O(n^N)$.

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. The time complexity of a fitness evaluation is denoted as τ_{eval} in our analysis.

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

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

Comparison

To show the computational advantages of the proposed DAC-MA approach, we compare DAC-MA with a conventional memetic algorithm [138] with Pareto optimal determination [146] for multiobjective 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 4.2 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.

Table 4.2: 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	$O(n^N)$	$O(g \times \log n \times N(K_{pop} \times \tau_{eval} + n^N))$

This shows that the proposed DAC-MA approach is an effective multiobjective optimization approach for Online-TPG.

4.5 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) [117], particle swarm optimization (PSO) [112], differential evolution (DE) [197], ant colony optimization (ACO) [116], tabu search (TS) [120], and classical MA (MA) [138]. Here, we adapt the classical memetic algorithm [138] with Pareto optimal determination [146] for multiobjective optimization of Online-TPG. These techniques are re-implemented based on the published articles. All algorithms including GA, PSO, DE, ACO, TS, MA and DAC-MA were implemented in Java.

4.5.1 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 20000, 30000, 40000 and 50000 respectively for performance evaluation. There are mainly 3 question types in each dataset, namely fill-in-the-blank, multiple choice and long question. In these 4 datasets, the values of each attribute are generated according to a uniform distribution. Table 4.3 shows the summary of the 4 datasets.

4.5.2 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

Table 4.3: Test Datasets

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

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 have summarized the 12 specifications with their attributes and presented them in Table B.1 of Appendix B.

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.

4.5.3 Quality Measures for Online-TPG

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 4.5 (**Mean Discrimination Degree).** Let $P_1, P_2, ..., P_k$ be the generated test papers on a question dataset $\mathcal{D}$ w.r.t different test paper specifications $\mathcal{S}_i$, $i = 1..k$. 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 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 test 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 4.1 that consists of the total time constraint violation and average difficulty degree constraint violation.
- Content Constraint Violation: Kullback-Leibler (KL) Divergence [145] 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 generated test paper P and the test paper specification $\mathcal{S}$.

Definition 4.6 (**Content Constraint Violation).** Given a test paper specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$. 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 0-100. λ is a constant which is set equal to 100.

Definition 4.7 (**Mean Constraint Violation).** The Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$ of k generated test papers $P_1, P_2, ..., P_k$ on a question dataset $\mathcal{D}$ w.r.t different test paper specifications $\mathcal{S}_i$, $i = 1..k$, is defined as

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

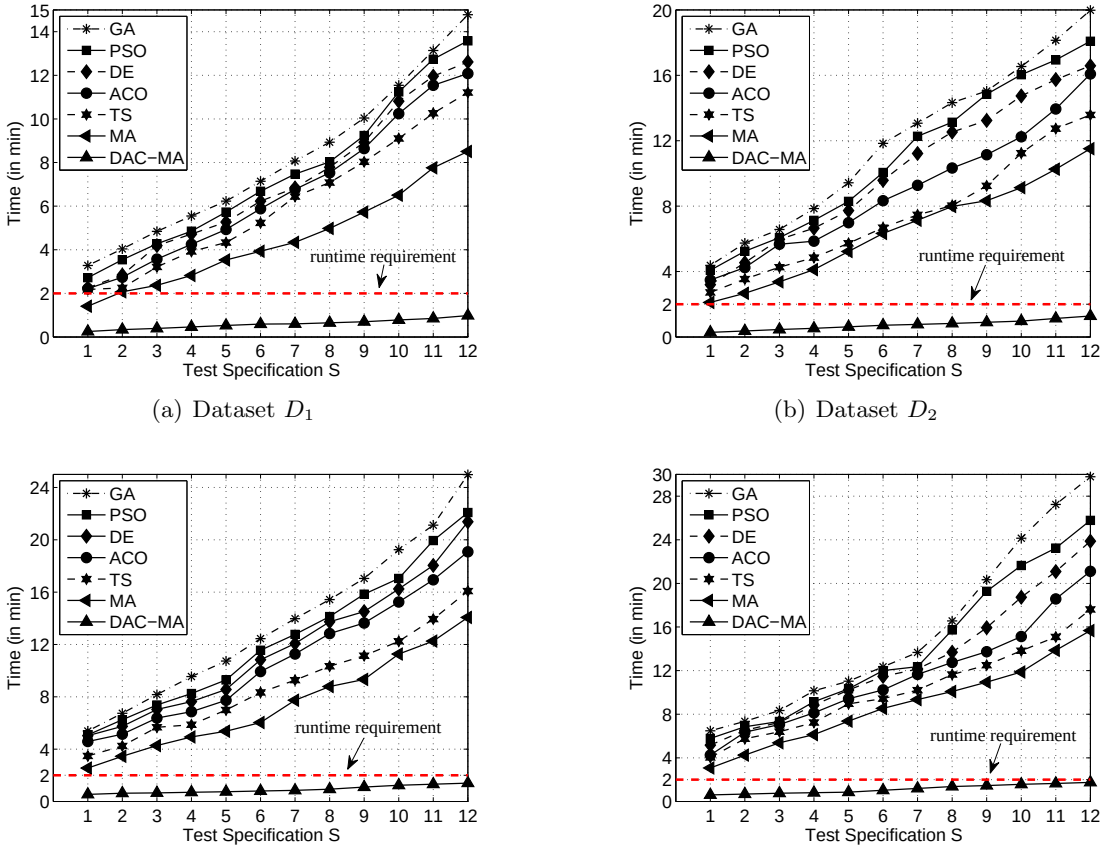
where $CV(P_i, \mathcal{S}_i)$ is the Constraint Violation of P_i w.r.t. S_i.

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 $CV(P, \mathcal{S})$. Hence, the overall quality of a generated test paper depends on user's preference between these two aspects. According to a pedagogical perspective, users often pay more attention to constraint satisfaction of test papers. To determine the quality of a generated test paper, constraint violations could be defined in a certain range. Here, we set the following 4 thresholds for a high quality test paper: $\triangle T(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\triangle D(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\log \triangle C(\mathcal{S}_P, \mathcal{S}) \leq 5$ and $\log \triangle Y(\mathcal{S}_P, \mathcal{S}) \leq 5$. The threshold values are obtained experimentally (to be discussed in Section 5.3). Based on these thresholds, we set $\mathcal{M}_c^{\mathcal{D}} \leq 10$ for high quality test papers, $10 < \mathcal{M}_c^{\mathcal{D}} \leq 30$ for medium quality test papers and $\mathcal{M}_c^{\mathcal{D}} > 30$ for low quality test papers.

4.5.4 Performance Results

Performance on Runtime: Figure 4.8 compares the runtime performance of the seven algorithms based on the 4 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 minutes to complete the paper generation process. By measuring the average runtime over the 12 specifications and the four datasets, DAC-MA is about 12 times faster in runtime performance than MA, which is the most efficient one among the other algorithms. Therefore, 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.

Performance on Quality: Figure 4.9 shows the performance results according to 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 4 datasets. As can be seen from Figure 4.9(a), 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 4 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 $\mathcal{M}_c^{\mathcal{D}} \leq 6$ for all datasets. As such, DAC-MA tends to generate higher quality test papers on larger datasets while other



(a) Dataset D_1

(b) Dataset D_2

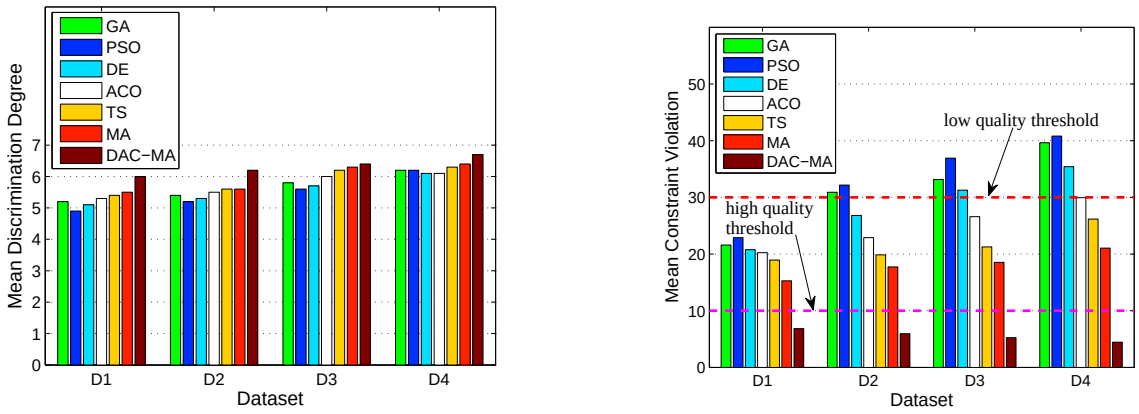
(c) Dataset D_3

(d) Dataset D_4

Figure 4.8: Performance Results based on Runtime

techniques tend to generate lower quality test papers. By measuring the average test paper quality over the 12 specifications and the four datasets, DAC-MA has achieved 69% better in Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$ and 7% better in Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$ than MA, which is the most effective one among the other algorithms. As such, DAC-MA has achieved an average of about 38% better test paper quality than MA.

Discussion: The runtime of DAC-MA is more efficient for Online-TPG due to 3 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



(a) Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$

(b) Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$

Figure 4.9: Performance Results based on Quality

Table 4.4: Performance Comparison between DAC and DAC-MA

	Algorithm	D_1	D_2	D_3	D_4
Average Runtime (seconds)	DAC	39.0	49.3	61.9	76.5
	DAC-MA	36.8	46.2	57.0	73.1
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

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.

Comparison between DAC and DAC-MA: Table 4.4 gives the performance comparison between DAC [176] 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 4 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 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.

4.6 Summary

In this chapter, we have discussed the proposed Divide-and-Conquer Memetic Algorithm (DAC-MA) approach for online test paper generation (Online-TPG). The proposed DAC-MA approach is based on constraint decomposition and memetic algorithm for multiobjective 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 chapter, we have also evaluated the performance of the proposed approach. The performance results have shown that the proposed DAC-MA approach has achieved 12 times faster in runtime performance and 38% better test paper quality than the other current TPG techniques. As such, it is particularly useful for generating test papers online for Web-based testing and intelligent tutoring.

Chapter 5

Integer Programming for Large-Scale Online-TPG

In this chapter, we discuss the proposed Integer Programming approach, called BAC-TPG, which is based on the Branch-and-Bound method and Lifted Cover Cutting method for large-scale online test paper generation (Online-TPG). Generally, there exists many topics (e.g. differentiation, integration, etc.) in a subject (e.g. mathematics). When the TPG problem is formulated in 0-1 ILP for a large question dataset, it has the sparse matrix property. The proposed BAC-TPG approach is based on the Branch-and-Bound method with the Lifted Cover Cutting method for solving the 0-1 ILP by exploiting the sparse matrix property. As Branch-and-Bound is a global and parameter-free method to deal with multiple constraints of the Online-TPG, the proposed approach avoids getting stuck in local optimal solutions to achieve high quality test papers as well as eliminates the need of using weighting parameters as in heuristic-based techniques.

5.1 Proposed BAC-TPG Approach

In this section, we discussed the proposed BAC-TPG (Branch-and-Cut for Test Paper Generation) for large scale online test paper generation. The proposed BAC-TPG approach has the following important characteristics:

- When the 0-1 ILP problem has the sparse matrix property, the proposed approach is able to approximate the binary optimal solution of the 0-1 ILP problem with the fractional optimal solution.

$$
\begin{aligned}
\textit{maximize} \quad & \textstyle\sum_{i=1}^{n} e_i x_i & & \\
\textit{subject to} \quad & \textstyle\sum_{i=1}^{n} x_i = N & & (5.1)\\
& \textstyle\sum_{i=1}^{n} t_i x_i = T & & (5.2)\\
& \textstyle\sum_{i=1}^{n} d_i x_i = \lfloor DN \rfloor & & (5.3)\\
& \textstyle\sum_{i=1}^{n} r_{iu} x_i = \lfloor pc_u N \rfloor & \forall u = 1..M & (5.5)\\
& \textstyle\sum_{i=1}^{n} s_{iv} x_i = \lfloor py_v N \rfloor & \forall v = 1..K & (5.5)\\
& x \in \{0,1\}^n & & (5.6)
\end{aligned}
$$

Figure 5.1: The 0-1 ILP Formulation of TPG

- The proposed approach uses the Primal-Dual Interior Point [143] which is the most efficient algorithm for solving the LP relaxation problem. In addition, the Simplex method [36] is also used for solving the LP relaxation problem efficiently in subsequent steps of the approach when new cutting planes are added.
- An effective branching strategy is proposed for reducing the size of the Branch-and-Bound search tree.
- An efficient approach is proposed for finding effective lifted cover cutting planes.

In the proposed BAC-TPG approach, we first re-formulate the 0-1 Fractional ILP of the TPG problem into a standard 0-1 ILP which is given in Figure 5.1. Note that as the number of questions N is a constant, the denominator of the maximizing cost function can be eliminated from the fractional ILP during re-formulation. In addition, as each question has only a few related topics and a question type, most of the coefficients in the topic constraint and question type constraint are zeros. Thus, for large-sized TPG problems, the matrix A of the 0-1 ILP is very sparse.

5.2 Branch-and-Bound

The Branch-and-Bound method is based on the divide-and-conquer strategy which iteratively partitions the original ILP problem into a series of subproblems. Each subproblem is then solved by Linear Programming (LP) relaxation to obtain an upper bound on its objective value. The key idea of the Branch-and-Bound method is that if the upper bound for the objective value of a given subproblem is less than the objective value of a known integer feasible solution, then the given subproblem does not contain the optimal solution of the

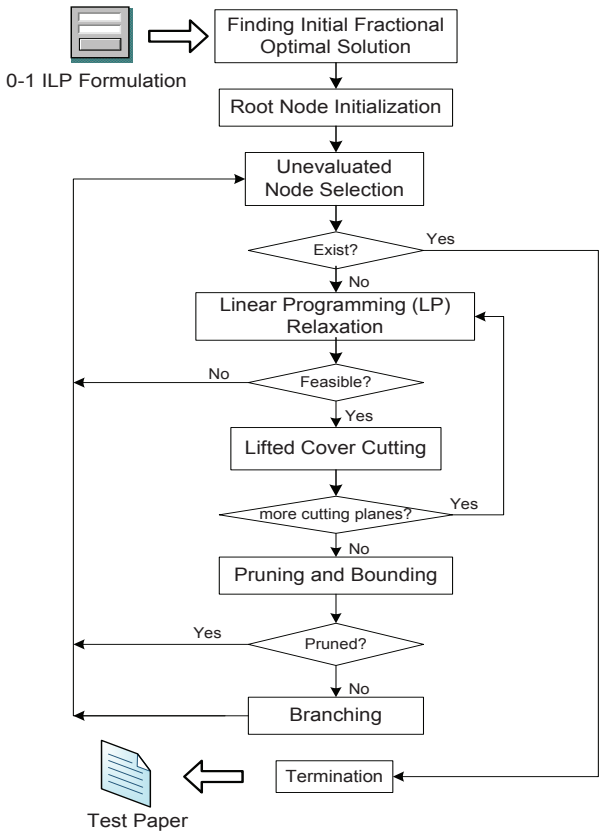


Figure 5.2: Branch-and-Bound Flowchart of the BAC-TPG

original ILP problem. Hence, the upper bounds of subproblems are used to construct a proof of optimality without exhaustive search. Figure 5.2 shows the main steps of the Branch-and-Bound in the BAC-TPG approach.

In BAC-TPG, the subproblems are organized as an *enumeration tree* which is constructed iteratively in a top-down manner with new nodes created by branching on an existing node in which the optimal solution of the LP relaxation is fractional. The problem at the root node of the tree is the original 0-1 ILP. When a new node N^i is created, it contains the corresponding 0-1 ILP subproblem and is stored in the list $\mathcal{L} = \{N^1, , ..., ., N^n\}$, $i \in \{1, ..., n\}$, of all unevaluated or leaf nodes. Let F^i be the formulation of the feasible region of the problem at node N^i. Let z^i_{ub} be the local upper bound at each node N^i and z_{lb} be the current global lower bound of the 0-1 ILP solution.

5.2.1 Finding Initial Fractional Optimal Solution

In this step, we find the fractional optimal solution x^* of the original 0-1 ILP problem. This is done by relaxing the constraints on binary value of variables. The 0-1 ILP formulation of

the TPG problem shown in Figure 5.1 is transformed into a standard Linear Program (LP) as follows:

$$\begin{aligned} &\text{maximize} \quad cx \\ &\text{subject to} \quad Ax = b, 0 \leq x \leq 1 \end{aligned}$$

or equivalently: maximize $\{cx | x \in F\}$, where $F = \{x \in \{0,1\} : Ax \leq b\}$ denotes the constraint set of feasible regions of the original ILP problem.

The LP problem can then be solved by using the most efficient Primal-Dual Interior Point (PDIP) algorithm [143]. PDIP solves the LP problem by resolving the following logarithmic barrier optimization problem:

$$\begin{aligned} &\text{maximize} \quad cx - \frac{1}{t}\sum_{i=1}^{n}(\log x_i + \log(1 - x_i)) \\ &\text{subject to} \quad Ax = b \end{aligned}$$

where t is a barrier parameter.

The optimal solution x^* of PDIP will be used to construct the corresponding tableau of the Simplex method [36]. It consists of two steps: initial tableau construction and Simplex tableau construction.

In the *initial tableau construction*, the Simplex algorithm works on inequalities of the form $\sum a_{ij}x_j \leq b_i$ and the 0-1 ILP of the TPG problem needs to satisfy the equality constraints given in Equations (7)-(12) of the form $\sum a_{ij}x_j = b_i$. Thus, we replace each constraint of the form $\sum a_{i,j}x_j = b_i$ by the following two constraints: $\sum a_{ij}x_j \leq b_i$ and $-\sum a_{ij}x_j \leq -b_i$. So far, all the replaced constraints given in Equations (7)-(12) are now in the form $\sum a_{ij}x_j \leq b_i$. By introducing new slack variables, we have the following *initial tableau*: $\{\text{maximize } cx \mid Ax + s = b\}$, where s is the vector of slack variables.

In the *Simplex tableau construction*, we perform pivoting operations on the initial tableau such that all variables x_j with $x_j^* > 0$ are basic variables while others are non-basic variables. As a result, the optimal solution x^* and its corresponding Simplex tableau $\mathcal{T}$ of the form $x_i + \sum a_{i,j}x_j = b_i$ are obtained.

5.2.2 Root Node Initialization

It first creates the root node N^1 of the enumeration tree that contains the original 0-1 ILP problem with its fractional optimal solution x^* and Simplex tableau $\mathcal{T}$. Next, it initializes the local upper bound at N^1 as $z^1_{ub} = \text{maximize}\,\{cx^* : x^* \in F^1\}$, the global lower bound $z_{lb} = -\infty$ and the current best 0-1 solution $x_{best} = \emptyset$. Then, the root node is stored in the list $\mathcal{L}$ for further processing.

5.2.3 Unevaluated Node Selection

This step selects an unevaluated node in the list $\mathcal{L}$ for processing and solving. If there is no unevaluated node, the algorithm will terminate. Otherwise, a node in the list $\mathcal{L}$ will be selected. Here, we use a greedy strategy, namely *best-bound*, to choose the most promising node N^i in $\mathcal{L}$ with largest local upper bound value z^i_{ub}.

$$N^i = \underset{N^i \in \mathcal{L}}{\text{argmax}}\, z^i_{ub}$$

5.2.4 LP Relaxation

It iteratively solves the subproblem of the selected unevaluated node N^i based on the optimal solution and Simplex tableau of its parent node's problem (except the root node). At the k^{th} iteration of processing a node N^i, it solves the following LP problem: $z^{i,k}_{ub} = \text{maximize}\,\{cx : x \in F^{i,k}\}$. If the returned result is infeasible (i.e. when $F^{i,k} = \emptyset$), it ignores this node and continues processing another node in the list $\mathcal{L}$. Otherwise, it goes to the next step on Lifted Cover Cutting for adding cutting planes. Note that for efficiency, it adds the new constraints into the Simplex tableau of its parent node and continue re-optimizing this tableau. After solving the LP Relaxation at node N^i, the fractional optimal solution x^* and its corresponding Simplex tableau are obtained.

5.2.5 Lifted Cover Cutting

The main purpose of the Lifted Cover Cutting is to add extra constraints, called *cutting planes*, to reduce the feasible region and approximate the binary optimal solution, which is nearest to x^*. Based on the current fractional optimal solution x^* and its corresponding Simplex tableau, this step helps LP Relaxation to gradually approximate more closely to the binary optimal solution x_{best} of the subproblem. It adds extra constraints or cutting

planes into the current subproblem. To achieve this, it adds some *lifted cover inequalities* to the current formulation $F^{i,k}$ such that a new formulation $F^{i,k+1}$ is formed. Then, this new formulation will go back to the LP Relaxation step for optimization. For efficiency, at most three cuts are added at each iteration according to an empirical study in [98]. It repeats until no more cutting plane is found. The Lifted Cover Cutting will be discussed later in Section 5.3.

5.2.6 Pruning and Bounding

After processing a node N^i, it will consider whether this node should be pruned. To determine this, it checks the obtained local upper bound of the LP Relaxation at node N^i (after the k^{th} iteration) and the global lower bound of the 0-1 ILP solution of the original 0-1 ILP:

- If $z_{ub}^{i,k} \leq z_{lb}$, it prunes the node N^i.
- If $z_{ub}^{i,k} \geq z_{lb}$ and the current fractional optimal solution $x^{i,k}$ is a binary solution, it updates the new global lower bound $z_{lb} = z_{ub}^{i,k}$ and the current best 0-1 ILP solution $x_{best} = x^{i,k}$. Then, it prunes this node and all unevaluated nodes in the list $\mathcal{L}$ whose upper bound z_{ub}^{i} is less than the new global lower bound z_{lb}, and processes another node in $\mathcal{L}$. If $z_{ub}^{i,k} \geq z_{lb}$ and $x^{i,k}$ is fractional, it goes to the Branching step.

5.2.7 Branching

If the solution of the LP Relaxation in node N^i is fractional, the Branching step creates two child nodes of N^i. First, it chooses the fractional variable x_j^* in the current fractional optimal solution $x^{i,k}$ of the LP Relaxation and performs the branching. We use a common choice, namely *most fractional variable* [36], to select the variable x_j^*: $j = \text{argmax}_{j \in C} \ \min[f_j, 1 - f_j]$, where $f_j = x_j^* - \lfloor x_j^* \rfloor$. Then, the two child nodes are placed into the list $\mathcal{L}$ for further processing:

- $N^{i+1} = \max \{cx : x \in F^{k+1} = \{F^k \bigcap (x_j = 0)\}\}$;
- $N^{i+2} = \max \{cx : x \in F^{k+2} = \{F^k \bigcap (x_j = 1)\}\}$.

The size of the search tree may grow exponentially if branching is not controlled properly. To effectively reduce the size of the tree, we use a heuristic based on the number of specified questions N in the generated test paper. Consider a path from the root to a given unevaluated

Table 5.1: An Example of Math Dataset

(a) Question Table

$\mathcal{Q}$_id	o	a	e	t	d	c	y
q_1	,...,.	,...,.	5	5	5	c_1	y_1
q_2	,...,.	,...,.	6	10	7	c_2	y_2
q_3	,...,.	,...,.	7	6	7	c_1	y_1
q_4	,...,.	,...,.	7	12	6	c_2	y_2
q_5	,...,.	,...,.	7	14	9	c_1	y_2
q_6	,...,.	,...,.	7	10	9	c_3	y_2
q_7	,...,.	,...,.	6	8	4	c_3	y_3
q_8	,...,.	,...,.	6	8	4	c_3	y_3

(b) Topic Table

$\mathcal{C}$_id	*name*
c_1	Integration
c_2	Differentiation
c_3	Limits

(c) Question Type Table

$\mathcal{Y}$_id	*name*
y_1	Fill-in-the-blank
y_2	Long question
y_3	Multiple choice

node of the tree, if the number of branching variables x_j with value 1 along the path is larger than or equal to N, we stop branching at that node. The reason is that we only need N questions in the generated test paper.

5.2.8 Termination

It returns the current best solution x_{best} of the 0-1 ILP.

5.2.9 An Example

Suppose that we need to generate a test paper P from the Math dataset given in Table 5.1 based on the specification $\mathcal{S} = \langle 2, 15, 6, \{(c_1, 0.5), (c_2, 0.5)\}, \{(y_1, 0.5), (y_2, 0.5)\}\rangle$. We associate each question with a binary variable x_i, $i = 1, ..., 8$. However, we eliminate inappropriate variables x_6, x_7 and x_8 because they cannot satisfy the specification $\mathcal{S}$. Here, we formulate the problem as a 0-1 fractional ILP with 5 binary variables, which is shown in Figure 5.3(a). The 0-1 fractional ILP problem is then transformed into a standard 0-1 ILP which is shown in Figure 5.3(b).

Figure 5.4 shows an example on the construction of the enumeration tree of subproblems during the BAC-TPG process. Initially, the LP relaxation of this problem is solved by the Primal-Dual Interior Point algorithm to obtain the fractional optimal solution $x^* =$

$$
\begin{aligned}
\text{maximize} \quad & \frac{5x_1 + 6x_2 + 7x_3 + 7x_4 + 7x_5}{x_1 + x_2 + x_3 + x_4 + x_5} && \text{(Average Discrimination Degree)} \\
\text{subject to} \quad & x_1 + x_2 + x_3 + x_4 + x_5 = 2 && \text{(Number of Questions)} \\
& 5x_1 + 10x_2 + 6x_3 + 12x_4 + 14x_5 = 15 && \text{(Total Time)} \\
& \frac{5x_1 + 7x_2 + 7x_3 + 6x_4 + 9x_5}{x_1 + x_2 + x_3 + x_4 + x_5} = 6 && \text{(Average Difficulty Degree)} \\
& \frac{x_1 + x_3 + x_5}{x_1 + x_2 + x_3 + x_4 + x_5} = 0.5 && \text{(Topic Distribution)} \\
& \frac{x_1 + x_3}{x_1 + x_2 + x_3 + x_4 + x_5} = 0.5 && \text{(Question Type Distribution)} \\
& x_i \in \{0, 1\}, i = 1, 2, 3, 4, 5 && \text{(Binary Value)}
\end{aligned}
$$

(a) The 0-1 Fractional ILP Formulation Example

$$
\begin{aligned}
\text{maximize} \quad & 5x_1 + 6x_2 + 7x_3 + 7x_4 + 7x_5 \\
\text{subject to} \quad & x_1 + x_2 + x_3 + x_4 + x_5 = 2 \\
& 5x_1 + 10x_2 + 6x_3 + 12x_4 + 14x_5 = 15 \\
& 5x_1 + 7x_2 + 7x_3 + 6x_4 + 9x_5 = 12 \\
& x_1 + x_3 + x_5 = 1 \\
& x_1 + x_3 = 1 \\
& x_i \in \{0, 1\}, i = 1, 2, 3, 4, 5
\end{aligned}
$$

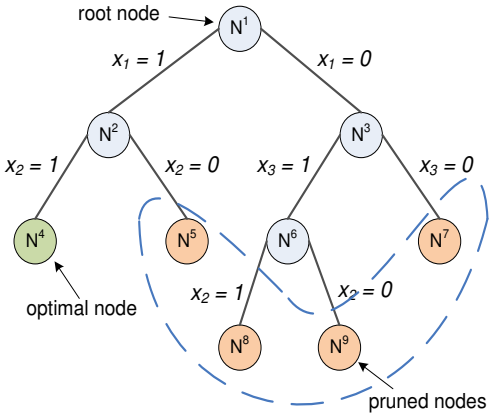
(b) The 0-1 ILP Formulation Example

Figure 5.3: The ILP Formulation Example

$(x_1, x_2, x_3, x_4, x_5) = (0.50, 0.70, 0.79, 0.00, 0.00)$. Next, the fractional optimal solution is updated when new lifted cover cutting planes are added to obtain the new fractional optimal solution x^*. Then, the local upper bound value at N^1 is set as $z^1_{ub} = 12.23$; the global lower bound is set as $z_{lb} = -\infty$; and the current best 0-1 solution is set as $x_{best} = \emptyset$.

After that, the root node N^1 is branched on the variable x_1 to create two child nodes N^2 and N^3 during branching. Then, the unevaluated node N^2 is selected for processing as it has the largest local upper bound. After processing and branching at N^2, two child nodes N^4 and N^5 are created in which $N^4 = (1.00, 1.00, 0.00, 0.00, 0.00)$ is a feasible binary solution with its objective value $z^4_{ub} = 11$. Then, the global local bound and best solution are updated as $z_{lb} = 11$ and $x_{best} = (1.00, 1.00, 0.00, 0.00, 0.00)$ respectively. The node N^5 is then pruned because its local upper bound is less than the current global lower bound. Subsequently, branching at node N^3 will create N^6 and N^7. Similar to N^5, N^7 will then be pruned. Branching at node N^6 will create N^8 and N^9, which are then pruned similarly to N^5.

Finally, the best solution $x_{best} = (1.0, 1.0, 0.0, 0.0, 0.0)$ is obtained. It corresponds to the test paper $P = \{q_1, q_2\}$ for the specification $\mathcal{S}$. The generated test paper has the average discrimination degree $E = 5.5$ and specification $\mathcal{S}_P = \langle 2, 15, 6, \{(c_1, 0.5), (c_2, 0.5)\}, \{(y_1, 0.5), (y_2, 0.5)\}\rangle$.



(a) Enumeration Tree

	$x^* = (x_1, x_2, x_3, x_4, x_5)$	z^i_{ub}	z_{lb}
N^1	0.50, 0.70, 0.79, 0.00, 0.00	12.23	$-\infty$
N^2	1.00, 0.85, 0.00, 0.15, 0.00	11.55	$-\infty$
N^3	0.00, 0.90, 0.50, 0.60, 0.00	12.10	$-\infty$
N^4	1.00, 1.00, 0.00, 0.00, 0.00	11.00	11.00
N^5	1.00, 0.00, 0.00, 0.50, 0.00	8.50	*pruned*
N^6	0.00, 0.50, 1.00, 0.20, 0.00	11.40	11.00
N^7	*infeasible*	$-\infty$	*pruned*
N^8	*infeasible*	$-\infty$	*pruned*
N^9	0.00, 0.00, 1.00, 0.30, 0.00	9.1	*pruned*

(b) Node List $\mathcal{L}$

Figure 5.4: The Enumeration Tree Example

5.3 Lifted Cover Cutting

Lifted Cover Cutting aims to generate the lifted cover cutting planes from the fractional optimal solution x^* and Simplex tableau $\mathcal{T}$ for the LP Relaxation step. Before discussing it in details, we need to define some basic terminologies. Consider the set $X = \{x \in \{0,1\}^n : \sum_{j=1}^n a_j x_j \leq b\}$, which represents a row of the Simplex tableau $\mathcal{T}$.

Definition 5.1 (Dominance). If $a_1 x \leq b_1$ and $a_2 x \leq b_2$ are two valid inequalities for $X = \{x \in R^n_+ : Ax \leq b\}$, $a_1 x \leq b_1$ *dominates* $a_2 x \leq b_2$ if $\{x \in R^n_+ : a_1 x \leq b_1\} \subseteq \{x \in R^n_+ : a_2 x \leq b_2\}$.

If there exists any non-negative coefficient a_j in X, the variable x_j can be replaced by its complementary variable $x'_j = 1 - x_j$. So that X contains only coefficients $a_j > 0$. As all coefficients in the LHS of X are now non-negative, we may assume that the RHS $b > 0$. Let $N = \{1, 2, ..., , n\}$ in the following definitions.

Definition 5.2 (Cover). Let $C \subseteq N$ be a set such that $\sum_{j \in C} a_j > b$, then C is a *cover*. A *cover* is *minimal* if $C \setminus \{j\}$ is not a *cover* for any $j \in C$.

The following 2 propositions are derived directly from the *cover* definition.

Proposition 5.1 (Cover Inequality). Let $C \subseteq N$ be a *cover* for X, the *cover inequality* $\sum_{j \in C} x_j \leq |C| - 1$ is valid for X, where $|C|$ is the cardinality of C.

Proposition 5.2 (Extended Cover Inequality). Let $C \subseteq N$ be a *cover* for X, the *extended cover inequality*: $\sum_{j \in \mathcal{E}(C)} x_j \leq |C| - 1$ is valid for X, where $\mathcal{E}(C) = C \cup \{j : a_j \geq a_i, \forall\, i \in C\}$.

Definition 5.3 (**Lifted Cover Inequality**). *Lifted Cover Inequality* (LCI) is an *extended cover inequality* which is not dominated by any other *extended cover inequalities*.

In general, the problem of finding LCI is equivalent to the finding of the best possible values for α_j for $j \in N \setminus \{C\}$ such that the inequality

$$\sum_{j \in N \setminus \{C\}} \alpha_j x_j + \sum_{j \in C} x_j \leq |C| - 1$$

is valid for X, where C is a *cover inequality*.

When the matrix A in the 0-1 ILP is sparse, the lifted cover cutting plane defined by LCI is an effective cutting plane for the pruning step in the Branch-and-Cut method. However, the problem on finding LCI has been shown to be NP-hard [98]. In this research, we propose to generate LCIs efficiently as follows.

First, we find a *minimal cover inequality* based on the N most significant basic variables in the fractional optimal solution x^* of the LP Relaxation, where N is the number of questions given in the test paper specification. Specifically, consider a row of the form $\sum a_{ij} x_j = b_i$ in the tableau, the basic variables $x_j^* > 0$ of the fractional optimal solution are then sorted in non-increasing order. Let U be a list of the first N largest coefficients, and V be the list of the remainings of the sorted list. If k is the minimal number such that the sum of the coefficients a_{ij} w.r.t. the first basic variables of the list V exceeds b_i, i.e. $\sum_{j=1}^{k} a_{ij} > b_i$, then the set $C = \{j_1, ..., j_k\}$ is a *minimal cover*.

Next, we generate *extended cover inequalities* from the *minimal cover* C as follows:

$$\sum_{t=1}^{N} \alpha_{j_t} x_{j_t} + \sum_{j \in C} x_j \leq |C| - 1$$

To generate LCIs from the *extended cover inequalities*, we need to calculate the largest lifting value $\alpha_{j_t} \in U, t = 1, ..., N$, for each variable x_{j_t}. This can be done by using an incremental algorithm to calculate α_{j_1}, then α_{j_2} until α_{j_N} in step by step manner. Specifically, the algorithm starts from calculating α_{j_1}, the result of α_{j_1} will then be used to calculate α_{j_2} and so on. To obtain α_{j_t}, $t = 1, ..., N$, we need to solve the following 0-1 Knapsack problem (0-1 KP):

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{t-1} \alpha_{j_m} x_{j_m} + \sum_{j \in C} x_j \\ \text{subject to} \quad & \sum_{m=1}^{t-1} a_{j_m} x_{j_m} + \sum_{j \in C} a_j x_j \leq b_i - a_{j_t} \\ & x \in \{0,1\}^{|C|+t-1} \end{aligned}$$

The largest lifting value α_{j_t} is then calculated as $\alpha_{j_t} = |C|-1-\gamma_t$, where γ_t, $t = 1, ..., N$, is the objective function value according to the optimal solution obtained from the 0-1 KP. It can be seen that γ_t computes the maximum weight corresponding to the set $\{j_1, j_2, ..., j_{t-1}\} \bigcup C$ in the *lifted cover inequality* when $x_{j_t} = 1$.

Gu et al. [98] solved the 0-1 KP by using a dynamic algorithm that requires high computational complexity of $O(n^4)$. The experimental results have shown that the runtime performance is poor when handling test cases with a few thousand variables. This is not acceptable in TPG in which the 0-1 ILP may have tens of thousands of variables that need to be solved efficiently. In this research, we apply an approximation algorithm from Martello and Toth [157] to efficiently solve the 0-1 KP that only requires $O(n \log n)$.

5.4 Performance Evaluation

In this section, we evaluate the performance of the proposed BAC-TPG approach for 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 BAC-TPG approach is implemented in Java with the CPLEX API package, version 11.0 [121]. From the CPLEX package, we use the Primal-Dual Interior-Point and Simplex methods. The performance of BAC-TPG is measured and compared with other techniques including genetic algorithm (GA) [117], particle swarm optimization (PSO) [112], differential evolution (DE) [197], ant colony optimization (ACO) [116], tabu search (TS) [120], Divide-and-Conquer Memetic Algorithm (DAC-MA) [178] and the conventional Branch-and-Cut (BAC) method [98]. These techniques were reimplemented in Java according to the published articles as same as in the experiment section of Chapter 4. We compare the BAC-TPG approach with the conventional BAC technique, which has been shown to be more effective for large-scale 0-1 ILP with the sparse matrix property than the commercial software [98].

5.4.1 Objectives

We have conducted 2 sets of experiments. In the first set of experiments, we aim to analyze the quality and runtime efficiency of our BAC-TPG approach based on 4 large-scale datasets by using different specifications. In the second set of experiments, we aim to evaluate the effectiveness of our proposed approach by conducting a user evaluation for the quality of test papers generated from different specifications based on the G.C.E A-Level math and the undergraduate engineering math datasets.

In the experiments, the test paper generation process is repeated until one of the following two termination conditions is reached:

- Quality satisfaction: The algorithm will terminate if a high quality test paper is generated.
- Maximum number of evaluated nodes in which no better solution is found: This parameter is experimentally set to 300 nodes for BAC and BAC-TPG. Similarly, for other heuristic techniques, this parameter is set to the maximum number of iterations in which no better solution is found. It is generally set to 200 iterations.

Here, high quality test paper is based on the following 4 thresholds as discussed in Section 4.5.3 of Chapter 4: $\triangle T(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\triangle D(\mathcal{S}_P, \mathcal{S}) \leq 0.15$, $\log \triangle C(\mathcal{S}_P, \mathcal{S}) \leq 5$ and $\log \triangle Y(\mathcal{S}_P, \mathcal{S}) \leq 5$. To obtain the best generated test paper of each algorithm for a fair comparison, we note that we do not set a hard termination condition based on runtime in the experiments.

5.4.2 Performance on Quality and Runtime

Datasets: Similar to Section 4.5, 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 5.2 summaries the 4 datasets.

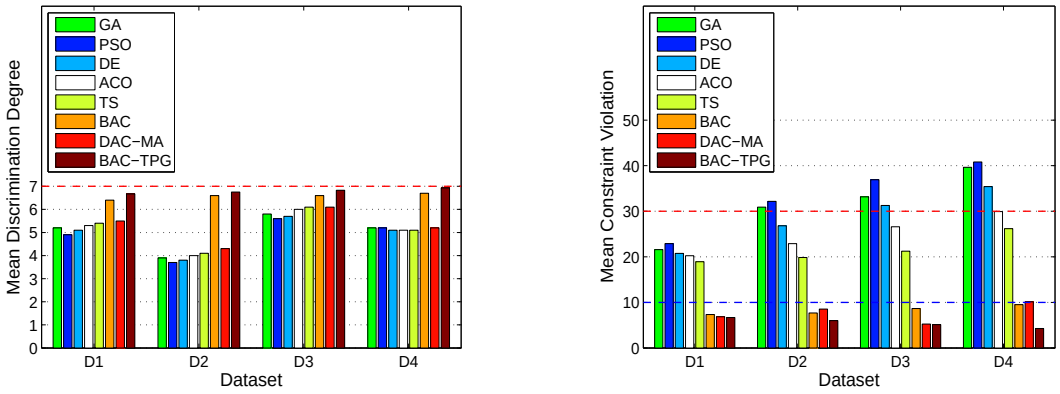
Table 5.2: Test Datasets

	#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

Experimental Procedures: To evaluate the performance of the BAC-TPG 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 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 8 algorithms: GA, PSO, DE, ACO, TS, BAC, DAC-MA and BAC-TPG. We measure the runtime and quality of the generated test papers for each experiment.

Performance on Quality: Figure 5.5 shows the quality performance results of the 8 techniques based on the Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$ and Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$. As can be seen from Figure 5.5(a), BAC-TPG has consistently achieved higher Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$ than the conventional BAC and other heuristic techniques for the generated test papers. Particularly, BAC-TPG can generate test papers with quality close to the maximal achievable value of $\mathcal{M}_d^{\mathcal{D}} \approx 7$. In addition, we also observe that BAC-TPG has consistently outperformed the other techniques on Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$ based on the 4 datasets. The average constraint violations of BAC-TPG tend to decrease whereas the average constraint violations of the other techniques increase quite fast when the dataset size or the number of specified constraints gets larger. In particular, BAC-TPG can generate high quality test papers with $\mathcal{M}_c^{\mathcal{D}} \leq 6$ for all datasets. Also, BAC-TPG is able to generate higher quality test papers on larger datasets while the other techniques generally degrade the quality of the generated test papers when the dataset size gets larger.

Performance on Runtime: Figure 5.6 compares the runtime performance of the 8 techniques based on the 4 datasets. Here, the 12 specifications are sorted increasingly according to the number of topics in the constraints. The results have clearly shown that the proposed BAC-TPG approach outperforms the conventional BAC and other heuristic techniques, except DAC-MA, in runtime for the different datasets. BAC-TPG generally requires less than



(a) Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$

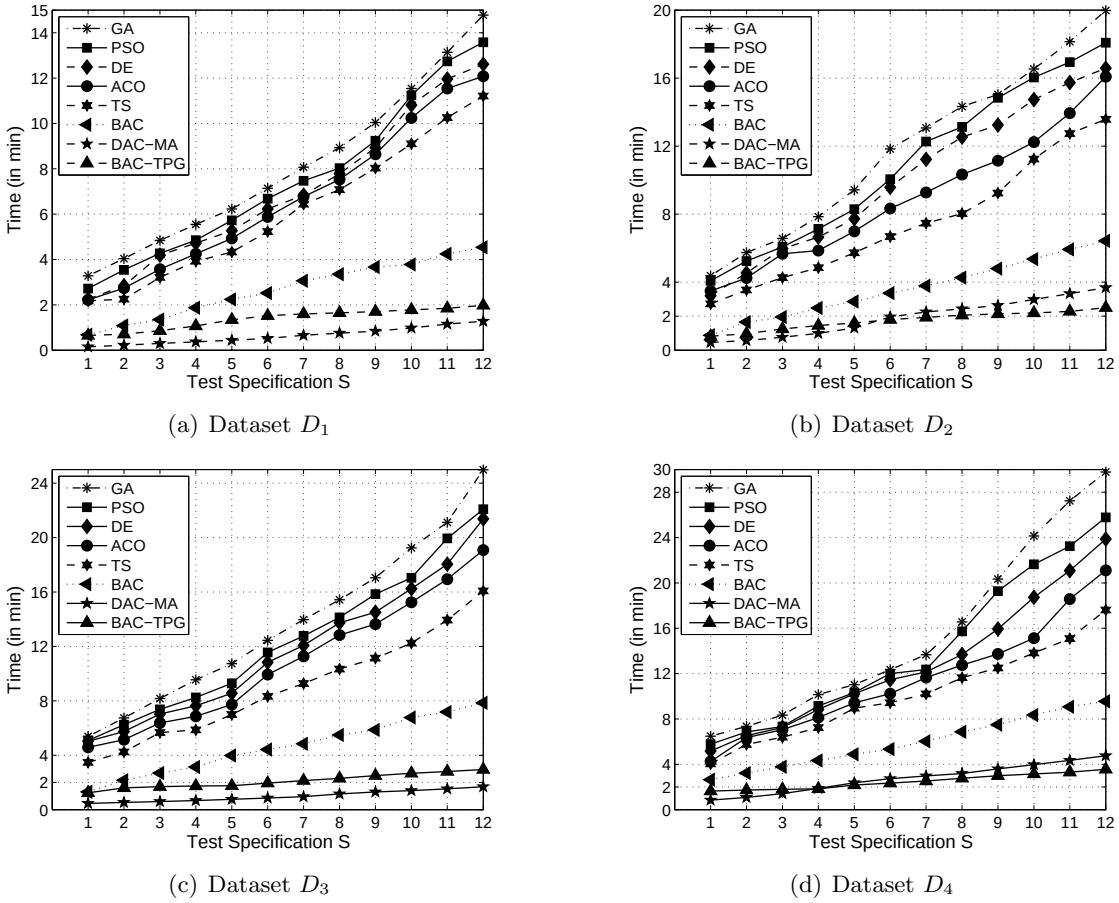
(b) Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$

Figure 5.5: Performance Results based on Average Quality of the 12 Specifications

2 minutes to complete the paper generation process. Moreover, the proposed BAC-TPG approach is quite scalable in runtime on different dataset sizes and distributions. In contrast, the other techniques (except DAC-MA) are not efficient to generate high quality test papers. Particularly, the runtime performance of these techniques degrades quite badly as the dataset size or the number of specified constraints gets larger, especially for imbalanced datasets D_2 and D_4.

Discussion: The good performance of BAC-TPG is due to 3 main reasons. Firstly, as BAC-TPG is based on LP relaxation, it can maximize the average discrimination degree effectively and efficiently while satisfying the multiple constraints without using weighting parameters. As such, BAC-TPG can achieve better paper quality and runtime efficiency as compared with other heuristic-based techniques. Secondly, BAC-TPG has a more effective branching strategy than that of conventional BAC, thereby pruning the search space more effectively. This helps BAC-TPG improve runtime and search on promising unvisited subproblem nodes of the search tree for paper quality enhancement. Thirdly, BAC-TPG uses an efficient algorithm to generate the *lifted cover inequalities*. Thus, BAC-TPG can also improve its computational efficiency on large-scale datasets as compared with the conventional BAC technique. Moreover, as there are more questions with different attribute values on larger datasets and LP relaxation is effective for global optimization, BAC-TPG is able to generate higher quality test papers. Therefore, the BAC-TPG approach is effective for TPG in terms of paper quality and runtime efficiency.

Comparison between BAC-TPG and DAC-MA: Divide-and-Conquer Memetic Algorithm (DAC-MA) [178] is an efficient TPG approach for online test paper generation. Table 5.3 gives



(a) Dataset D_1

(b) Dataset D_2

(c) Dataset D_3

(d) Dataset D_4

Figure 5.6: Performance Results based on Runtime of the 12 Specifications

the performance comparison between BAC-TPG and DAC-MA. The performance results are obtained based on the average results of the 12 test specifications for each dataset. In Figure 5.5, the quality performance of BAC-TPG is consistently better than DAC-MA for the 4 datasets. However, the runtime performance of BAC-TPG and DAC-MA depends on the user specifications and dataset distributions as shown in Table 5.3 and Figure 5.6.

For the balanced uniform distribution datasets D_1 and D_3, DAC-MA outperforms BAC-TPG in runtime. DAC-MA is in fact very fast as it is designed for achieving online runtime requirement. As there may have enough relevant questions on D_1 and D_3 for DAC-MA to optimize its solution without getting stuck on local optimal, the runtime performance of DAC-MA outperforms BAC-TPG in this situation because it is a heuristic-based technique whereas BAC-TPG is a global optimization method. For imbalanced normal distribution datasets D_2 and D_4, BAC-TPG achieves better runtime performance if the specified test papers contain many topics or have high total time. Otherwise, DAC-MA achieves better runtime performance. The main reason is that the sparse matrix property of 0-1 ILP formulation

Table 5.3: Performance Comparison between BAC-TPG and DAC-MA

	Algorithm	D_1	D_2	D_3	D_4
Average Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$	DAC-MA	5.50	4.30	6.25	5.20
	BAC-TPG	6.67	6.75	6.83	6.92
Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$	DAC-MA	6.85	8.53	5.25	10.15
	BAC-TPG	5.35	4.68	4.24	3.05
Average Runtime (seconds)	DAC-MA	35.4	116.8	54.0	165.8
	BAC-TPG	83.6	104.9	126.7	149.2

of BAC-TPG is satisfied in this situation, which makes lifted cover cuttings become more effective for 0-1 ILP optimization. Furthermore, as DAC-MA focuses only on optimizing the constraint satisfaction of a unique initial solution, and it could easily lead to a local optimal. It is especially the case for imbalanced datasets, where there may not have enough relevant questions for DAC-MA to optimize the unique initial solution. As shown in Figure 5.5, the quality performance of DAC-MA degrades quite badly on both D_2 and D_4.

As shown from the experimental results, the main advantage of BAC-TPG as compared with DAC-MA is the test paper quality on large-sized datasets, whereas the main advantage of DAC is its runtime efficiency in small and average-sized datasets. While the advantage of BAC-TPG on runtime is only marginal, BAC-TPG outperforms DAC-MA on test paper quality significantly in all datasets, especially on the objective function of Mean Discrimination Degree. Specifically, by measuring the average test paper quality over the 12 specifications and the four datasets, BAC-TPG has achieved 13% better in Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$ and 45% better in Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$ than DAC-MA. As such, BAC-TPG has achieved an average of about 29% better test paper quality than DAC-MA. Therefore, DAC-MA is suitable for online testing (e.g. when students need to take an online paper) in which run-time efficiency is important, whereas BAC-TPG is suitable for paper generation (e.g. when teachers or lecturers need to generate a test paper for testing or examination purpose) in which the quality of the test paper is more important than runtime.

5.4.3 User Evaluation on Generated Paper Quality

Datasets: To gain further insight into the paper quality generated by the proposed BAC-TPG approach, we have conducted a user evaluation. Here, we use 2 math datasets, namely *A-Math* and *U-Math*, which are constructed from G.C.E. A-Level Math and Undergraduate Math respectively. For experimental purposes, the question attributes are calibrated

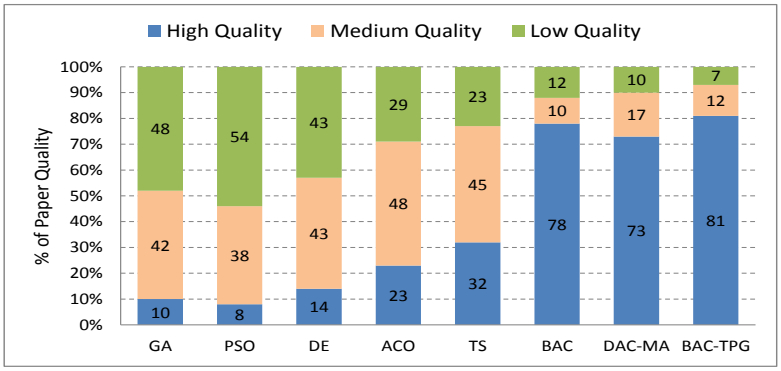
Table 5.4: Math Datasets

	#Questions	#Topics	#Question Types
$A-Math$	1500	12	3
$U-Math$	3000	36	3

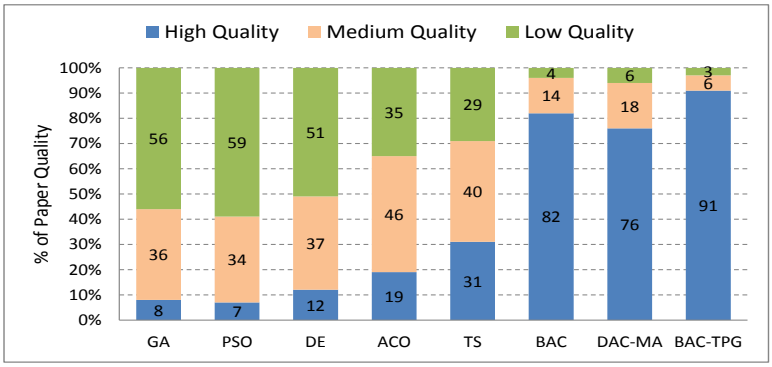
semi-automatically by ten tutors who are tutors of the first year undergraduate mathematics subject. These tutors have good knowledge of the math contents contained in the 2 math datasets. In the calibration, we adopt a rating method [226] for the tutors to rate the difficulty degree of each question from 1 to 7 corresponding to the following 7 discretized difficulty levels of IRT: extremely easy, very easy, easy, medium, hard, very hard and extremely hard. In addition, according to the IRT Normal Ogive model [27], the discrimination degree of a question can be computed initially based on its relation formulae with a fixed user's proficiency value and the difficulty degree. Table 5.4 summaries the 2 datasets.

Experimental Procedures: In the experiments, we have designed 12 new specifications for the A-Math dataset and 12 new specifications for the U-Math dataset with different parameter values. Then, we generate the test papers based on the test specifications using the 8 techniques. As a result, a total of 96 papers are generated for each dataset. To conduct the user evaluation, the same ten tutors are participated. In the experiments, the tutors are asked to evaluate the test paper quality by comparing its attributes with its original specification. The experts compare intuitively each corresponding attribute of the generated test with its specification by their own intuition without knowing our defined measures. Based on the similarities, they are asked to evaluate the overall quality of the generated test and classify it into one of the following three categories: high, medium and low. As a result, a total of 960 test paper evaluations are conducted for each dataset.

Expert Calibration on Quality Results: Figure 5.7 shows the user evaluation results on the generated test paper qualities from the 8 techniques. As can be seen, BAC-TPG has achieved better performance on quality than the other techniques. For the A-Math dataset, BAC-TPG has achieved promising results with 81% (97 papers), 12% (14 papers) and 7% (9 papers) for high, medium and low quality generated papers respectively from a total of 120 papers evaluated. Similarly, for the U-Math dataset, BAC-TPG has achieved promising results with 91% (109 papers), 6% (7 papers) and 3% (4 papers) for high, medium and low quality generated papers respectively. On average, BAC-TPG has achieved 86%, 9% and 5% for high, medium and low quality generated papers respectively. In addition, it can also



(a) *A-Math*



(b) *U-Math*

Figure 5.7: User Evaluation Results on Generated Test Paper Quality

be observed that the proposed BAC-TPG approach is able to improve the quality of the generated papers with a larger dataset. It has performed better with the U-Math dataset than the A-Math dataset.

Moreover, we have further analyzed the quality of the generated papers based on the user evaluation results. After all the generated papers are classified into high, medium and low quality, we have analyzed the constraint violations on topic, question type, difficulty degree and total time based on each of the two datasets. Table 5.5 gives the analysis results in terms of mean and standard deviation of the corresponding constraint violations. For the A-Math dataset, we have obtained the mean constraint violations of 10 ± 2.2, 21 ± 2.8 and 30 ± 1.9 for high, medium and low quality generated papers respectively. Similarly, for the U-Math dataset, we have obtained the mean constraint violations of 9 ± 1.4, 19 ± 1.4 and 29 ± 1.2 for high, medium and low quality generated papers respectively. On average, we have obtained 9.5 ± 1.3, 20 ± 1.6 and 29.5 ± 1.1 constraint violations for high, medium and low quality generated papers. In fact, we have used these values as the thresholds when we determine high, medium or low quality for the generated papers during performance evaluation on test

Table 5.5: Quality Analysis of the Generated Test Papers on the User Evaluation Results

User Assessment Criteria	*A-Math*			*U-Math*		
	High	Medium	Low	High	Medium	Low
Topic Violation ($\log \triangle C$)	5 ± 3	14 ± 4	14 ± 4	4 ± 2	13 ± 2	13 ± 2
Question Type Violation ($\log \triangle Y$)	6 ± 3	15 ± 5	17 ± 3	10 ± 3	13 ± 4	15 ± 2
Difficulty Degree Violation ($\triangle D$)	13 ± 5	25 ± 6	43 ± 4	11 ± 3	23 ± 2	42 ± 3
Time Violation ($\triangle T$)	16 ± 6	29 ± 7	48 ± 4	12 ± 3	26 ± 3	46 ± 2
Constraint Violation *CV*	10 ± 2.2	21 ± 2.8	30 ± 1.9	9 ± 1.4	19 ± 1.4	29 ± 1.2

paper quality.

5.5 Summary

In this chapter, we have proposed an efficient Integer Programming approach called BAC-TPG for high quality test paper generation from large scale question datasets. The proposed BAC-TPG approach is based on the Branch-and-Bound and Lifted Cover Cutting methods to find the near-optimal solution by exploiting the sparse matrix property of 0-1 ILP. The performance results on various datasets and a user evaluation on generated test paper quality have shown that the BAC-TPG approach has achieved 29% better test paper quality than DAC-MA on the large-scale Online-TPG problem. As such, the proposed BAC-TPG approach is particularly useful for Web-based testing and assessment for online learning environment.

Chapter 6

Multiobjective Evolutionary Co-Approximation for Parallel-TPG

In this chapter, we propose a Multiobjective Evolutionary Co-Approximation (MECA) algorithm for parallel test paper generation (k-TPG). MECA is a greedy-based approximation algorithm, which explores the submodular property of the objective function for combinatorial multiobjective optimization. Rather than probabilistically searching on a huge space of potential solutions as in evolutionary computation, the key idea is that MECA deterministically drives the search process through a finite number of evolutionary generations that can enhance the quality of the k test papers. In each generation, MECA keeps only a fixed number of k test papers and jointly optimizes them to find co-approximate solutions that are better than previously obtained solutions by using the submodular greedy-based approximation algorithm.

6.1 Problem Specification for k-TPG

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 $\mathcal{S}_{P_i} = \mathcal{S}, \forall i \in [1, k]$. 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 groups of users. As such, there are two important objective functions that need to be maximized in k-TPG: *total quality maximization* and *fairness quality maximization*. Especially, these objective functions are defined over a set of k generated test papers instead of an individual paper as in single test paper generation.

6.1.1 Total Quality Maximization

Total Quality Maximization aims to maximize the sum of discrimination degrees of k test papers $\sum_{i=1}^{n} f(P_i)$, where $f(P_i)$ is the discrimination degree of test paper $P_i, i \in [1, k]$. Formally, the k-TPG problem is stated as follows:

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

6.1.2 Fairness Quality Maximization

Fairness Quality Maximization 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 [112, 116, 118]. Instead, we can optimize for a fair allocation of the discrimination degrees of k generated test papers as follows:

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

In fact, this way of formulation is more effective for fairness optimization [23, 31, 34, 194].

6.1.3 Joint Total and Fairness Quality Maximization

Both Total Quality Maximization and Fairness Quality Maximization have been studied separately 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:

$$\underset{P_1,...,P_k}{\text{maximize}} \sum_{i=1}^{k} f(P_i) \tag{6.1}$$

and the fairness quality objective function:

$$\underset{P_1,...,P_k}{\text{maximize}} \min_{1 \leq i \leq k} f(P_i) \tag{6.2}$$

subject to the following global and local multicriteria constraints:

$$P_i \bigcap_{\forall i \neq j} P_j = \emptyset \tag{6.3}$$

$$\mathcal{S}_{P_i} = \mathcal{S} \tag{6.4}$$

We note that in the special case $k = 1$, the two objective functions become the same. Thus, the formulation also becomes exactly the same as the formulation of the single test paper generation. For the remaining discussion, we refer the objective functions (6.1) and (6.2) to as global objective functions whereas the constraints in (6.4) are referred to as local constraints.

6.1.4 Computational Complexity

For the special case $k = 1$, it was shown to be NP-hard [120]. Except this special case, we are not aware of any study on the computational complexity of the general k-TPG problem so far. Generally, the k-TPG problem aims to maximize global objective functions over a set of generated test papers while satisfying the local multiobjective constraints in (6.4). In [120], constraints in (6.4) can be formulated as multiple knapsack equality constraint satisfaction problem, which is proved to be NP-hard in Section 3.4, Chapter 3. To simplify the study, we first ignore the fairness quality objective in (6.2). Thus, the modified k-TPG problem can be formulated as the 0-1 Integer Linear Programming (0-1 ILP) as follows. Let $\mathcal{S} = \langle k, N, T, D, C, Y \rangle$ be a user specification, where k is the number of generated parallel test papers. Different from the 1-TPG problem, we associate k binary variables $x_{ij} \in \{0, 1\}$ with each question $q_i, i = 1, ..., n$, in the dataset and each of the k test papers $P_1, P_2, ..., P_k$. In other words, the k variables $x_{ij}, j = 1, ..., k$, are duplicated and indicate whether the question q_i is included in the test paper $P_j, j = 1, ..., k$, or not. This duplication is typical and necessary to formulate the local constraints that each test paper must satisfy. Figure 6.1 shows the 0-1 ILP formulation of the k-TPG. Note that the constraint in (6.11) restricts that at most one variable among the k variables $x_{ij}, j = 1, ..., k$, is included in the k test papers.

$$\textit{maximize} \quad \sum_{i=1,j=1}^{n,k} e_i x_{ij} \tag{6.5}$$

subject to

for each test paper $P_j, j = 1..k$*:*

$$\sum_{i=1}^{n} x_{ij} = N \tag{6.6}$$

$$\sum_{i=1}^{n} t_i x_{ij} = T \tag{6.7}$$

$$\sum_{i=1}^{n} d_i x_{ij} = \lfloor DN \rfloor \tag{6.8}$$

$$\sum_{i=1}^{n} r_{iu} x_{ij} = \lfloor pc_u N \rfloor \quad \forall u = 1..M \tag{6.9}$$

$$\sum_{i=1}^{n} s_{iv} x_{ij} = \lfloor py_v N \rfloor \quad \forall v = 1..K \tag{6.10}$$

for each question $q_i, i = 1..n$ *:*

$$\sum_{j=1}^{k} x_{ij} \leq 1 \tag{6.11}$$

$$x_{ij} \in \{0, 1\}^n \tag{6.12}$$

Figure 6.1: The 0-1 ILP Formulation of k-TPG with the Total Quality Objective

It is obvious that if the modified k-TPG problem is NP-hard, then the original k-TPG problem is also NP-hard. Consider the modified k-TPG problem as to maximize only the total quality objective function given in (6.1), it can be polynomially reduced from the social welfare problem [223] with equal players and a cardinality constraint. The social welfare problem is an auction problem in economics theory, which is known to be NP-hard [223]. As a result, k-TPG is also NP-hard. It is also important to note that the parallel test paper generation process occurs over the Web where users would expect to generate multiple test papers within an acceptable response time. Therefore, k-TPG is a challenging problem due to its computational NP-hardness, and it is also required to be solved efficiently in runtime.

If we consider the k-TPG problem as to maximize only the fairness quality objective function given in (6.2), k-TPG can be considered as a distributed resource allocation problem under a budget constraint [24, 80, 94] or an auction game problem in economics theory [70]. Both of these problems are known to be NP-hard. Thus, optimizing both objective functions of k-TPG is also NP-hard.

6.2 Submodular Function Optimization

In combinatorial optimization, submodularity of an objective function is a very important property [151]. In this section, we review some fundamental concepts of submodular functions.

Definition 6.1 (Discrete Derivative)**.** Given a set X, a non-negative function $f : 2^X \rightarrow \mathbb{R}^+$,

$T \subseteq X$, 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 $f(S) : \{0,1\}^n \rightarrow \mathbb{R}^+$ at element x.

Definition 6.2 (Submodularity)**.** Given a set X, a non-negative function $f : 2^X \rightarrow \mathbb{R}^+$ 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 $f : 2^X \rightarrow \mathbb{R}^+$ is *submodular* if for every set $A \subseteq B \subseteq X$ and $x \in X \setminus B$,

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

The first definition is called the decreasing marginal property of submodular functions. In addition, a submodular function is monotone if $f(S) \leq f(T), S \subseteq T$. The latter definition can be deduced from the first definition by substituting $S = A \cup \{x\}$ and $T = B$, where $A \subseteq B \subseteq X$ and $x \in X \setminus B$.

Definition 6.3 (Monotonicity and Non-monotonicity)**.** A submodular function $f : 2^X \rightarrow \mathbb{R}^+$ is *monotone* if for every $S \subseteq T \subseteq X$, $f(S) \leq f(T)$. If the condition does not hold, the submodular function f is *non-monotone*.

Note that a function f is monotone iff all its discrete derivatives are non-negative, i.e., iff for every $S \subset X$ and $x \in X$, it holds that $\triangle_f(x|S) \geq 0$. Typically, it is assumed that f is given in terms of a *value oracle*, i.e., a black box that computes $f(S)$ on any input set S.

Submodular function optimization has been extensively studied over the last 30 years since the preliminary work by L. Lovasz [151]. Subsequently, a lot of research work have been conducted on maximizing or minimizing a submodular function, often subjected to combinatorial constraints [88, 204]. Generally, there exists efficient polynomial algorithms for minimizing a submodular function [82, 124, 125, 188, 203]. However, maximizing a submodular function is known to be NP-hard [173]. Although NP-hard, the decreasing marginal property has led to the existence of a general Greedy-based Approximation Algorithm for

Algorithm 6.1: Greedy_based_Approximation_Algorithm

Input: $f : 2^X \rightarrow \mathbb{R}^+$ - monotone submodular functions; k - cardinality constraint
Output: $\mathcal{A} \subset X$ - approximate solution
begin
1 $\mathcal{A} \leftarrow \emptyset$;
2 **for** *i=1 to* k **do**
foreach $x \in X \setminus \mathcal{A}$ **do**
3 Compute the marginal improvement value: $\triangle(x|\mathcal{A}) \leftarrow f(\mathcal{A} + \{x\}) - f(\mathcal{A})$;
4 $x^* \leftarrow \text{argmax}_{x \in X \setminus \mathcal{A}} \triangle(x|\mathcal{A})$;
5 $\mathcal{A} \leftarrow \mathcal{A} \bigcup \{x^*\}$;
6 **return** $\mathcal{A}$

the maximization problem. Algorithm 6.1 gives the basic Greedy-based Approximation Algorithm [204, 88] for monotone submodular function optimization under the cardinality constraint. Lemma 6.1 gives the performance guarantee of the Greedy-based Approximation Algorithm. It can achieve a good approximation ratio of $(1 - \frac{1}{e}) \approx 0.63$. This is known to be the best achievable ratio until now. Moreover, the simplicity of the Greedy-based Approximation Algorithm makes it useful in various applications where other algorithms may not be appropriate. Although the Greedy-based Approximation Algorithm can also be used for non-monotone submodular functions, it performs quite poorly.

Lemma 6.1 (Nemhauser et al. [173])**.** The Greedy-based Approximation Algorithm gives an approximation ratio of $1 - \frac{1}{e} \approx 0.63$ for the problem of $\max_{|\mathcal{A}| \leq k} f(\mathcal{A})$ where $f : 2^X \rightarrow \mathbb{R}^+$ is a monotone submodular function. Specifically, it means that

$$f(\mathcal{A}) \geq (1 - (1 - \frac{1}{k})^k)\, OPT \underset{k \rightarrow \infty}{=} (1 - \frac{1}{e}) OPT$$

where $OPT = \max\limits_{|\mathcal{A}| \leq k} f(\mathcal{A})$ is the value of the optimal solution.

Lemma 6.2 (Some Submodular Properties [88])**.** Submodular functions have many useful basic properties, which are given as follows:

- *Linear function*: A linear function $f(S) = \sum_{x_i \in S} w_i$ is submodular, where w_i is a weight associated with x_i.
- *Linear combinations*: If f_1 and f_2 are two submodular functions, then $f = \alpha_1 f_1 + \alpha_2 f_2$ is also a submodular function, where α_1 and α_1 are non-negative constants.
- *Truncation*: If $f : 2^X \rightarrow \mathbb{R}^+$ is a monotone submodular function, then the function

$f_c = \min\{f(S), c\}$ is also submodular, where $S \subset X$ and $c \geq 0$ is a constant.

6.3 Solving k-TPG with Approximation Algorithm

In this section, we further analyze the special properties of k-TPG on the two objective functions as well as the multicriteria constraints. 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 optimization method.

It is obvious that the objective function for total quality maximization is submodular. In fact, the discrimination degree function $f(P)$ is submodular due to the linearity property. Hence, the total quality objective, which is a linear combination of k discrimination degree functions, is also submodular.

Lemma 6.3. Given a test paper P generated from a question set $\mathcal{Q}$ ($P \subseteq \mathcal{Q}$), the discrimination degree function $f(P) : 2^{\mathcal{Q}} \rightarrow \mathbb{R}^+$ is submodular and monotone.

The function $f(P)$ is linear, therefore it is submodular.

Corollary 6.1. The total quality objective function $\sum_{i=1}^{k} f(P_i), P_i \subseteq \mathcal{Q}$ defined on k generated test papers is submodular and monotone.

However, we note that optimizing the total quality objective alone is not sufficiently good to ensure for fairness quality maximization.

Let $f_\phi(P) = \min\{f(P), \phi\}$ be a truncated function, where ϕ is a non-negative constant. From Lemma 6.2, $f_\phi(P)$ is also submodular and monotone. For any constant ϕ, we have the following important observation:

$$\min_{l \in \{1,\ldots,k\}} f(P_l) \geq \phi \iff \sum_{l=1}^{k} f_\phi(P_l) = k\phi \tag{6.13}$$

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

Theorem 6.1. Given the optimal fairness value ϕ^*, i.e., $\max_{P_1,\ldots,P_k} \min_l f(P_l) = \phi^*$, then it is sufficient to jointly optimize the two objective functions of the original k-TPG problem by solving the following equivalent problem:

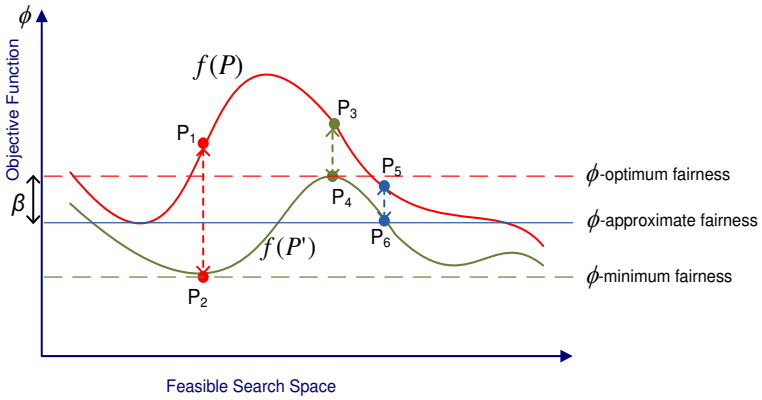


Figure 6.2: Motivating Idea of Using Greedy-based Approximation Algorithm for k-TPG

$$\max_{P_1,...,P_k} \sum_{l=1}^{k} f_{\phi^*}(P_l) \text{ s.t. } P_i \bigcap_{\forall i \neq j} P_j = \emptyset \ , \ \mathcal{S}_{P_i} = \mathcal{S} \tag{6.14}$$

Proof: The original k-TPG problem is equivalent to the problem given in (6.14) since, by (6.13), 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 test papers $P_1, ..., P_k$ of the problem in (6.14), 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 (6.13).

In other words, this result means that we can solve the problem in (6.14) to jointly optimize the two objective functions. Although we do not know the optimal fairness value ϕ^* in general, we could use the binary search strategy to find this value. As such, our approach will progressively solve several reformulated instances of the problem in (6.14) together with different estimated values of ϕ^*.

Now, the k-TPG problem becomes the problem on solving (6.14) with multiple knapsack constraints. Here, we devise an effective and efficient approach to solve (6.14) based on a greedy-based approximation algorithm for the special 1-TPG problem.

For illustration, let's consider a 2-TPG problem. Figure 6.2 shows the motivating idea of the greedy-based approximation algorithm for solving this problem. Assume that we need to generate 2 test papers according to a user specification and we have a greedy-based approximation algorithm for the 1-TPG problem, called $\mathcal{A}$. In Figure 6.2, the upper curve represents the objective landscape of the first generated test paper $f(P), P \subset \mathcal{Q}$, whereas the lower curve represents the objective landscape of the second test paper $f(P'), P' \subset \mathcal{Q} \setminus P$

after all questions in the first one are excluded. Note that the pair of corresponding solutions is represented by a vertical line. Obviously, function $f(P')$ is upper bounded by function $f(P)$ because of the deterministic behavior of the greedy-based approximation algorithm $\mathcal{A}$ for solving the same 1-TPG problem, i.e., the same user specification, on a subset of the question dataset. Figure 6.2 shows 3 possible pairs of solution of 2-TPG. Among the 3 pairs, the solution pair of P_1 and P_2 is the worst because $f(P_1) + f(P_2)$ is not near-optimal and $f(P_2)$ is minimal. The solution pair of P_3 and P_4 is the best choice because $f(P_3) + f(P_4)$ is near-optimal and $f(P_2)$ is maximal. However, due to NP-hardness, it is not easy to find the solution pair of P_3 and P_4. In fact, in this research, we aim to find the solution pair of P_5 and P_6 as the best choice because $f(P_5) + f(P_6)$ is near-optimal and $f(P_6)$ is near-maximal.

To find approximate solution, assume that by some ways we can estimate a feasible fairness value ϕ. Let P_1 and P_2 be the first and second test papers found by $\mathcal{A}$ with the adjusted objective function $f_\phi(P)$. If the value of ϕ is appropriately estimated, we can find P_1 and P_2 such that $f_\phi(P_1) \geq \beta\phi$ and $f_\phi(P_2) \geq \beta\phi$, where $\beta \leq 1$ is the approximation ratio of 1-TPG. It is possible if we are able to devise a good approximation algorithm for 1-TPG. This can be done by adjusting gradually the value of ϕ by the binary search strategy until the optimal value is obtained. As such, the problem on finding an approximate solution for the problem in (6.14) is completely solved.

Therefore, the k-TPG problem can be tackled by solving the 1-TPG problem, which aims to maximize $f_\phi(P), P \subset \mathcal{Q}$ with multiple knapsack constraints. As far as we know, we can only find approximate solution for 1-TPG problem in polynomial time due to its NP-hardness [120]. However, it requires the expensive fractional optimal solution to deal with the multiple knapsack constraints. There is a more effective way to deal with this issue for 1-TPG. We observe that the content and question type constraints can be easily satisfied given the number of questions in a test paper. After satisfying these constraints, it becomes a simpler problem with two remaining constraints on total time and average difficulty degree. Therefore, we can divide the set of constraints in (6.4) of the problem formulation into two subsets of relevant constraints, namely *content constraints* and *assessment constraints*, which can then be solved separately and effectively. In the test paper specification $\mathcal{S} = \langle N, T, D, C, Y \rangle$ shown in Figure 6.1, the content constraints include the constraint (6.9) on topic distribution C and constraint (6.10) on question type distribution Y, whereas the assessment constraints include the constraint (6.7) on total time T and constraint (6.8) on average difficulty degree D.

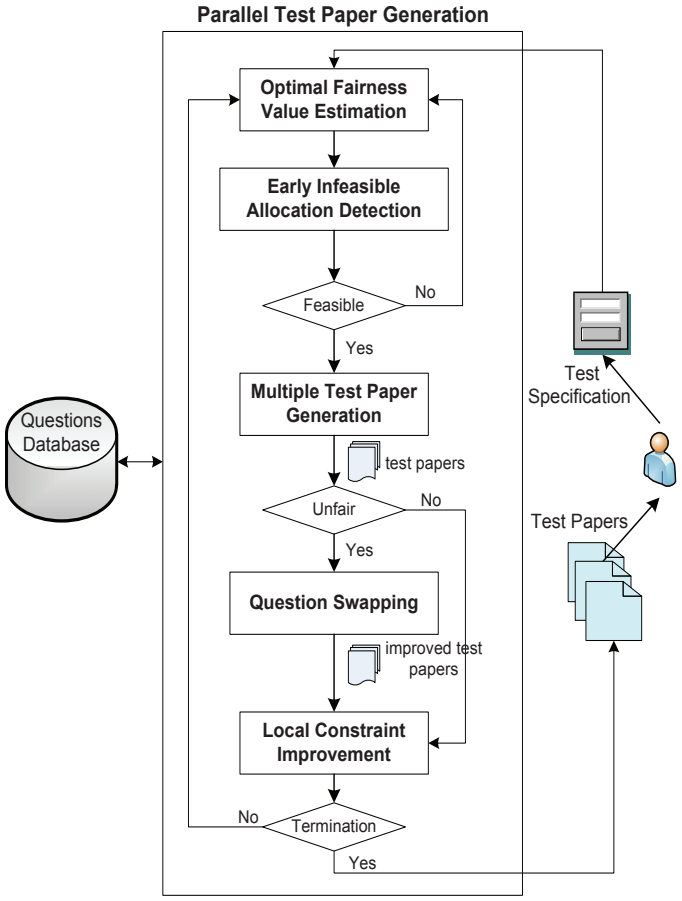


Figure 6.3: Proposed MECA Approach

6.4 Proposed MECA Approach

In this research, we propose a novel Multiobjective Evolutionary Co-Approximation (MECA) 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 achieve a fraction $\beta \leq 1$ of the optimal objective value of ϕ, where β is a constant that will be determined in Section 6.4.4. As such, our goal is to allocate questions into k test papers such that for all test papers, we have $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., k$.

Figure 6.3 shows the proposed MECA approach, which consists of 5 main steps as follows:

- *Optimal Fairness Value Estimation*: It first estimates the optimal fairness value ϕ_{min}, ϕ_{max} and $\phi = (\phi_{min} + \phi_{max})/2$ for the co-optimization process.
- *Early Infeasible Allocation Detection*: It detects whether it is possible to generate k test papers such that $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., k$ for early infeasible allocation detection.

Algorithm 6.2: Multiobjective_Evolutionary_Co-Approximation (MECA)

Input: $\mathcal{S} = (k, N, T, D, C, Y)$ - test paper specification; $\mathcal{Q}$ - question dataset;
Output: $\mathbf{P} = P_1, P_2, ..., P_k$ - test papers
begin

```
1  Initialize φ_min, φ_max  P_best ← ∅;
2  while φ_max − φ_min ≥ ϵ do   /* termination condition is not satisfied */
3      Optimal_Fairness_Value_Estimation: φ = (φ_min + φ_max)/2;   /* new φ */
4      infeasible ← Early_Infeasible_Allocation_Detection(φ, S, Q);
5      if infeasible then
6          Update φ_max = φ;
           break;                          /* early infeasible detection */
       else
7          for l ← 1 to k do                          /* generate k papers */
8              P = {P_1, P_2, ..., P_k} ← Multiple_Test_Paper_Generation(φ, S, Q);
9          if P is unfair allocation then
10             P ← Question_Swapping(P, φ, Q);        /* question swapping */
11         P' ← Local_Constraint_Improvement(P); /* assessment constraint */
12         P_best = P';
13         Update φ_min = φ;
14 return P_best
```

- *Multiple Test Paper Generation*: It progressively generates k test papers by using an approximation algorithm. The generated test papers are then tested on whether $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., k$.

- *Question Swapping*: It swaps questions of the generated test papers for improving fairness allocation if there exists a test paper P_l such that $f_\phi(P_l) \leq \beta\phi$.

- *Local Constraint Improvement*: It optimizes the equality constraint satisfaction of the generated test papers. Finally, the termination condition is checked to determine whether to continue the next evolutionary generation process by adjusting ϕ or not.

Algorithm 6.2 presents the proposed MECA approach for k-TPG.

6.4.1 Optimal Fairness Value Estimation

This step aims to find a new optimal value of ϕ to continue the co-optimization process. Although the optimal value ϕ^* is unknown, we can search for it by using the binary search on the initial 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,

Algorithm 6.3: Early_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* ;

it goes to the Early Infeasible Allocation Detection step to check whether it is possible to generate k satisfied test 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 Multiple Test Paper Generation 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.

6.4.2 Early Infeasible Allocation Detection

This step aims to detect whether it is possible to generate k test papers such that $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., k$ for early infeasible detection. In other words, it checks whether the estimated value of ϕ is appropriate or not. Our proposed algorithm for this step is a greedy-based approximation algorithm that generalizes the basic approximation algorithm given in Algorithm 6.1 for submodular function optimization. Note that in this step, we consider only the content constraints 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 6.3.

Recall that each of the k test 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 insertable into a test paper P_t. If yes, we then compute the marginal improvement value $\Psi_{t,q}$ of inserting question q into P_t. Next, we greedily choose the best pair of test paper and question (P_{t^*}, q^*) with the maximum $\Psi_{t,q}$ value. Finally, we insert q^*

into P_{t^*}. To verify whether q is insertable into test paper P_t, we first check whether P_t is full, i.e., $|P_t| = N$. Then, we check whether inserting q into P_t will violate the content constraints. Specifically, the maximum number of questions of each topic c_l is $pc_l * N$, $l = 1, ..., M$. Similarly, the maximum number of questions of each question type y_j is $py_j * N$, $j = 1, ..., K$. Hence, we check if these maximum numbers are exceeded when adding q into P_t.

We provide a theoretical result of this step. For the case $k = 1$, it is obvious that this algorithm is the same as the basic Greedy-based Approximation Algorithm 6.1. Therefore, the performance is guaranteed to obtain a solution with at least a constant approximation ratio of $1 - \frac{1}{e} \approx 0.63$ of the optimal value. For general cases, i.e., $k \geq 2$, it achieves a weaker approximation ratio of $\frac{1}{2}$.

Theorem 6.2. Consider only the content constraints for the general cases, i.e., $k \geq 2$, the Early Infeasible Allocation Detection 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 test 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 test paper. Let H' be the problem on the $n-1$ remaining questions after the first question q_t is selected for test 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 6.3 can be considered as first allocating question q_t to test paper P_j and then allocating 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 6.3, $OPT(H)$ 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 test 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 test 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' also consists of submodular quality evaluation function:

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

∎

Theorem 6.2 means that Algorithm 6.3 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 larger 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 6.3 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.

6.4.3 Multiple Test Paper Generation

This step aims to solve progressively k problem instances of 1-TPG to find a set of k test papers $P_1, P_2, ..., P_k$. Based on the monotone submodular property of the objective function $f_\phi(P)$, we greedily select questions that maximize the objective functions while paying attention to satisfy the multiple knapsack constraints.

In some applications, we would like to maximize a submodular function under a knapsack constraint as follows:

$$\max_S f(S) \text{ s.t. } \sum_{x \in S} c(x) \leq b$$

where $c(x) \geq 0$ is a non-negative cost function and $b \in \mathbb{R}$ is a budget constraint. Sviridenko [213] proposed a greedy-based approximation 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 $c(x)$. Then, the algorithm selects the next element of maximum marginal gain by considering the remaining budget available. This algorithm starts with $S = \emptyset$, and then iteratively adds the element x that maximizes the marginal gain ratio among all elements that still satisfy the remaining budget constraint as follows:

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

We extend the idea of submodular function maximization algorithm under a knapsack constraint for solving the case of multiple knapsack constraints. Algorithm 6.4 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 the current solution. The algorithm is based on a main loop in Line 4. In each iteration of that loop, the algorithm extends the current solution with an available question that maximizes the sum of marginal gain ratio normalized by the weights as follows:

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

When the loop terminates, the algorithm can return a set of generated test papers, which are feasible according to the multiple knapsack constraints. Recall from Definition 6.1 that $\Delta_{f_\phi}(q_j|P_l)$ is the incremental marginal value of question q_j to the test paper P_l.

Lemma 6.4. Algorithm 6.4 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 test paper in the user specification.

Proof: Let $P_l, l = 1, ..., k$, be a generated test 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 question that induces a violation in some constraints. Specifically, 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,

Algorithm 6.4: Multiple_Test_Paper_Generation

Input: $\mathbf{Aq} \leq \mathbf{b}, q \in \{0,1\}^n$ - 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 $P_l \leftarrow P_l \cup \{q_j\}$;
7 **for** *i=1 to m* **do** $h_i \leftarrow h_i \mu^{\frac{A_{ij}}{b_i}}$;
;
8 $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{q_j\}$;
9 **return** $P_1, P_2, ..., P_k$

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

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 6.4 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 6.4 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.

6.4.4 Question Swapping

Recall that our goal is to allocate questions into k test papers such that for all test 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, the Greedy-based Approximation Algorithm 6.4 aggressively selects all questions with high value $f_\phi(q)$ for some of the generated test papers $P_1, P_2, ..., P_s$ to satisfy $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., s$. As a result, subsequent test papers $P_{s+1}, ..., P_k$, might not have enough good questions to satisfy this requirement. In this case, we need to swap some of the questions from the satisfied test papers with questions from unsatisfied test papers to achieve a fair

Algorithm 6.5: Question_Swapping

Input: $P_1, P_2, ..., P_k$ - test papers with unfair allocation
Output: $P'_1, P'_2, ..., P'_k$ - test papers with fair allocation
begin
  while $\exists i, j \leq k : f_\phi(P_i) \geq 3\beta\phi; f_\phi(P_j) < \beta\phi$ **do**
1   **foreach** $q^i \in P_i$ **do**
2     $P_j \leftarrow \{P_j \setminus \{q_j\}\} \cup \{q_i\}$;
3     $P_i \leftarrow \{P_i \setminus \{q_i\}\} \cup \{q_j\}$;
4   **if** $f_\phi(P_j) \geq \beta\phi$ **then** break;
      ;
5 **return** $P_1, P_2, ..., P_k$

allocation.

We will move questions from satisfied test papers to unsatisfied test papers and vice versa, until all test papers satisfy the requirement $f_\phi(P_l) \geq \beta\phi, \forall l = 1, ..., k$. Let's define the *swapping operation* as follows. Select a satisfied test paper $P_i = \{q_1^i, q_2^i, , ..., .q_N^i\}$ for which $f_\phi(P_i) \geq 3\beta\phi$. Such a test paper is always ensured. Then, we select an unsatisfied test paper P_j, i.e., $f_\phi(P_j) \leq \beta\phi$. In the test 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. We reallocate the questions of test 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 test papers remain feasible. More importantly, this operation improves the fairness allocation between test papers. The swapping operation is given in Algorithm 6.5.

Lemma 6.5. The swapping operation improves the fairness allocation of test papers P_i and P_j as follows:

$$f_\phi((P_j \setminus \Lambda_j) \cup \Lambda_i) \geq \beta\phi, \text{ and } f_\phi((P_i \setminus \Lambda_i) \cup \Lambda_j) \geq f_\phi(P_i) - 2\beta\phi$$

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

$$f_\phi(\Lambda_i) \leq f_\phi(\{q_1^i, q_2^i, , ..., .q_{t-1}^i\}) + f_\phi(\{q_t^i\}) < 2\beta\phi$$

because of the submodularity of f_ϕ and the assumption $f_\phi(\{q_t^i\}) < \beta\phi$. Hence, inserting Λ_i to P_i' increases more than inserting Λ_i to the empty set. It is contrary to the submodular property of f_ϕ. ∎

Therefore, swapping questions will decrease the objective value of $f_\phi(P_i)$ an amount smaller than $2\beta\phi$. Thus, P_i is still satisfied. Moreover, the unsatisfied test paper becomes satisfied by this operation. To complete this step, it is necessary to guarantee that the question swapping operation can be performed 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 6.3 in Section 6.4.2 for the total quality maximization.

Lemma 6.6. 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. Let α be the ratio of the objective value of an allocation $\mathbf{P}$ of k test papers obtained by Algorithm 6.4 and the optimal objective value. 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 test papers that can have. Let θ be the fraction of unsatisfied test papers. We have

$$k\theta\beta + k(1-\theta) \geq \alpha k$$

Since the optimal value θ is obtained if all the satisfied test papers are achieved, i.e., having the total quality of about $k(1-\theta)$, and the unsatisfied test 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 test papers. We know that

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

The first question swapping is possible if

$$\frac{\Phi}{k(1-\alpha)} \geq 3\beta$$

Since if the total quality remained of all $(1-\theta)k$ satisfied test papers is 3β, then there do exists a test paper such that question swapping can be carried out. Since each swapping operation decreases the total quality Φ no larger than 2β, as proved in Lemma 6.5, and since we need θk swapping operations to satisfy all unsatisfied test papers, the following condition is necessary:

$$\begin{aligned} & \frac{\Phi - 2\theta k\beta}{k(1-\theta)} \geq 3\beta \\ \Leftrightarrow \quad & \Phi - 2\theta k\beta \geq 3\beta k - 3\beta\theta k \\ \Leftrightarrow \quad & \Phi \geq 3\beta k - \beta\theta k \end{aligned}$$

Hence, a necessary condition for β such that sufficient swapping operations can be carried out to satisfy all unsatisfied test papers is

$$\begin{aligned} & \frac{\alpha(1-\beta) - \beta(1-\alpha)}{1-\beta} \geq 3\beta - \beta\frac{1-\alpha}{1-\beta} \\ \Rightarrow \quad & 3\beta^2 - (3+\alpha)\beta + \alpha \geq 0 \\ \Rightarrow \quad & \beta \leq \alpha/3 \end{aligned}$$

by solving this inequality according to β and removing the infeasible solution $\beta \geq 1$. Because β is the approximation ratio, we wish to maximize β with respect to the constraint. Thus, we select $\beta = \alpha/3 = 1/6$. ∎

6.4.5 Local Constraint Improvement

So far, we have achieved a set of k test papers, having guarantee on the global objective value and satisfying the multiple knapsack constraints. 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 value and other constraints. Here, we propose an

effective method for assessment constraint optimization based on the submodular property.

Starting from each generated test paper $P^0 = P_l = \{q_1, q_2, ..., q_N\}, l = 1, ..., k$, 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 [106], which is known to be NP-hard in general. However, as shown in Section 6.4.3, the existing test paper P_l has completely satisfied the content constraints and cardinality constraint. Hence, we only need to optimize the two assessment constraints on total time and difficulty degree. 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 discrimination degree objective function f discussed in Lemma 6.4 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 optimization by introducing Lagrange multipliers $\alpha_1 < 1$ and $\alpha_2 < 1$ 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 known that maximizing this problem *maximize* $f_\xi(P)$ is equivalent to achieving simultaneously the following three objectives: *maximize* $f(P)$, *minimize* $|\sum_{l=1}^{n} a_{il}q - b_i|$ and *minimize* $|\sum_{l=1}^{n} a_{jl}q - b_j|$. We will show that the problem $\max f_\xi(P_l)$ is an unconstrained non-monotone submodular maximization, which is NP-hard in general. The two Lagrange multipliers λ_1 and λ_2 are set smaller than 1 such that the local search process will not degrade the current objective value $f(P)$ while improving the local constraint satisfaction. Thus, we are only able to achieve an approximate solution q^1 of the assessment constraint objective $\mathbf{A}q^1 \approx \mathbf{b}$. Here, we adapt an efficient deterministic local search proposed by Feige et al. [79] for unconstrained non-monotone submodular maximization.

Lemma 6.7. The modified objective function $f_\xi(P)$ is non-monotone and submodular.

Algorithm 6.6: Local_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** $P_1, P_2, ..., P_k$

Proof: By rewriting $f_\xi(P)$ as follows:

$$f_\xi(P) = \begin{cases} f(P) + \lambda_1(b_i - \sum_{l=1}^{n} a_{il}q) + \lambda_2(b_j - \sum_{l=1}^{n} a_{jl}q) & if \quad \sum_{l=1}^{n} a_{il}q > b_i, \sum_{l=1}^{n} a_{jl}q > b_j \\ f(P) + \lambda_1(b_i - \sum_{l=1}^{n} a_{il}q) + \lambda_2(\sum_{l=1}^{n} a_{jl}q - b_j) & if \quad \sum_{l=1}^{n} a_{il}q > b_i, \sum_{l=1}^{n} a_{jl}q < b_j \\ f(P) + \lambda_1(\sum_{l=1}^{n} a_{il}q - b_i) + \lambda_2(b_j - \sum_{l=1}^{n} a_{jl}q) & if \quad \sum_{l=1}^{n} a_{il}q < b_i, \sum_{l=1}^{n} a_{jl}q > b_j \\ f(P) + \lambda_1(\sum_{l=1}^{n} a_{il}q - b_i) + \lambda_2(\sum_{l=1}^{n} a_{jl}q - b_j) & if \quad \sum_{l=1}^{n} a_{il}q < b_i, \sum_{l=1}^{n} a_{jl}q < b_j \end{cases}$$

We observe that $f_\xi(P)$ is always rewritten as a linear combination of 3 submodular functions with non-negative constants. Hence, $f_\xi(P)$ is a submodular function too. In addition, $f_\xi(P)$ is also a combination of increasing monotone submodular function and decreasing monotone submodular function. As such, $f_\xi(P)$ is non-monotone and submodular. ∎

Different from the deterministic local search [79], our adapted method starts with an existing solution instead of an empty set. Our local search algorithm aims to find better questions to substitute the existing questions in the test paper in order to minimize assessment constraint violations. Algorithm 6.6 presents the local search algorithm. It starts from an original test paper P^0 and then iteratively moves to its neighboring solution $P^1 \in \mathcal{N}(P^0)$, where $\mathcal{N}(P^0)$ is a neighborhood region. Here, the neighborhood region of P^0 is defined so that the content constraints are preserved. More specifically, the neighborhood region of P^0 is any test paper that has the same content constraint satisfaction as P^0. As such, the neighborhood region of P^0 could be very large. To form a new test paper, each question q_i in the original test paper P^0 is substituted by another better question q_j 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 $f_\xi(P^1)$, $P^1 \in \mathcal{N}(P^0)$. As shown in [79], 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 an arbitrary small value.

6.4.6 Termination

In each evolutionary generation, a new set of k test papers are generated according to a new fairness value ϕ. The test paper generation process is repeated until the following termination condition is reached: $\phi_{max} - \phi_{min} \leq \epsilon$, where ϵ is a small predefined threshold. This is because the existing fairness value ϕ is near-optimal.

6.4.7 Complexity Analysis

In this section, we summarize the approximation results of the proposed MECA approach for multiobjective optimization of k-TPG.

Corollary 6.2. The proposed MECA approach achieves the following theoretical multiobjective approximation results of the total quality maximization and fairness quality maximization:

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

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

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

Moreover, polynomial time complexity is also achieved. Let $\vartheta = \max_{q \in \mathcal{Q}} f(q))$. The time complexity involved in the various steps of our proposed MECA approach is summarized as follows:

1. Optimal Fairness Value Estimation: $O(\lceil \log_2(N \times \vartheta) \rceil)$
2. Early Infeasible Allocation Detection: $O(k \times n \times N \times \tau_f)$
3. Multiple Test Paper Generation: $O(k \times n \times N \times \tau_f)$
4. Question Swapping: $O(k^3 \times N\tau_f)$
5. Local Constraint Optimization: $O(k \times n^2 \times \log \frac{1}{\epsilon} \times \log n \times \tau_f)$

where n is the number of questions in the dataset, N is the number of questions in a test paper from a given user specification, 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.

Hence, the total time complexity of the MECA approach is in the order of

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

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

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

6.5 Performance Evaluation

In this section, we evaluate the performance of the proposed MECA approach for k-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 MECA is measured and compared with other techniques including the following 3 re-implemented algorithms for k-TPG: k-TS [118], k-PSO [112] and k-ACO [116] based on their related published articles. All algorithms were implemented in Java.

6.5.1 Experiments

To evaluate the performance of the MECA approach, we aim to analyze the quality and runtime efficiency of the MECA approach for k-TPG based on 4 large-scale datasets by using different specifications.

Here, we have used the 4 large-scale synthetic datasets presented in Table 5.2, Section 5.4.2, Chapter 5. In addition, we have also used the 12 test specifications designed in Section 5.4.2, Chapter 5 for the experiments. To evaluate the effectiveness of the proposed approach, we have conducted the experiments according to 5 different values of k, i.e., $k = 1$ (Online-TPG), $k = 5$, $k = 10$, $k = 15$, $k = 20$.

For the case $k = 1$, we perform the experiments according to the 12 test specifications for each of the following 11 algorithms: GA, PSO, DE, ACO, TS, DAC-MA, BAC-TPG, k-PSO, k-ACO, k-TS and MECA. We measure the runtime and quality of the generated test papers for each experiment. For parallel test paper generation, i.e., $k \geq 2$, the performance of the proposed MECA approach is evaluated based on runtime and paper quality. The performance of MECA is measured and compared with other k-TPG techniques including k-PSO, k-ACO

and k-TS.

6.5.2 Quality Measures for k-TPG

To evaluate the quality of the generated test papers for parallel test paper generation, we use the Average Discrimination Degree and Average Constraint Violation. In addition to the average quality, the corresponding Deviate Discrimination Degree 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 $\mathcal{S}$. Note that the following definitions are defined for k-TPG, which are different from the definitions on quality measures for Online-TPG given in Chapter 4.

Definition 6.4 (**Average Discrimination Degree and Variant Discrimination Degree).** Given a specification $\langle k, \mathcal{S} \rangle$. Let $\mathbf{P} = P_1, P_2, ..., P_k$ be the set of generated test papers from the user specification $\mathcal{S}$.

The Average Discrimination Degree $\mathcal{M}_d^{k,\mathcal{S}}(\mathbf{P})$ is defined as

$$\mathcal{M}_d^{k,\mathcal{S}}(\mathbf{P}) = \frac{\sum_{i=1}^{k} E_{P_i}}{k}$$

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

The Variant Discrimination Degree $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ is defined as

$$\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P}) = Var(E_{P_1}, E_{P_2}, ..., E_{P_k}) = \frac{\sum_{i=1}^{k} (E_{P_i} - \mathcal{M}_d^{k,\mathcal{S}}(\mathbf{P}))^2}{k}$$

where $Var(E_{P_1}, E_{P_2}, ..., E_{P_k})$ is the variance of $E_{P_i}, i = 1, ..., k$.

Definition 6.5 (**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 user specifications $\mathcal{S}$.

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

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

where $CV(P_i, \mathcal{S})$ is the Constraint Violation of P_i w.r.t. $\mathcal{S}$, which was discussed in Section 4.5.3, Chapter 4 for Online-TPG.

The Variant Constraint Violation $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$ is defined as

$$\mathcal{V}_c^{k,\mathcal{S}}(\mathbf{P}) = Var(CV(P_1,\mathcal{S}), CV(P_2,\mathcal{S}), ..., CV(P_k,\mathcal{S})) = \frac{\sum_{i=1}^{k}(CV(P_i,\mathcal{S}) - \mathcal{M}_c^{k,\mathcal{S}}(\mathbf{P}))^2}{k}$$

where $Var(CV(P_1,\mathcal{S}), CV(P_2,\mathcal{S}), ..., CV(P_k,\mathcal{S}))$ is the variance of $CV(P_i,\mathcal{S}), i = 1, ..., k$.

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 6.6 (**Mean Discrimination Degree and Deviate Discrimination Degree).** 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 Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ is defined as

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

where $\mathcal{M}_d^{k,\mathcal{S}_i}(\mathbf{P}_i)$ is the Average Discrimination Degree of $\mathbf{P}_i$.

The Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ is defined as

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

where $\mathcal{V}_d^{k,\mathcal{S}_i}(\mathbf{P}_i)$ is the Variant Discrimination Degree of $\mathbf{P}_i$.

Definition 6.7 (**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

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

where $\mathcal{M}_c^{k,\mathcal{S}_i}(\mathbf{P}_i)$ is the Average Constraint Violation of $\mathbf{P}_i$.

The Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$ is defined as

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

where $\mathcal{V}_d^{k,\mathcal{S}_i}(\mathbf{P}_i)$ is the Average Constraint Violation of $\mathbf{P}_i$.

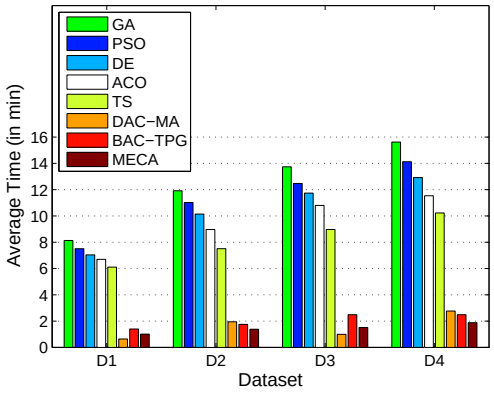
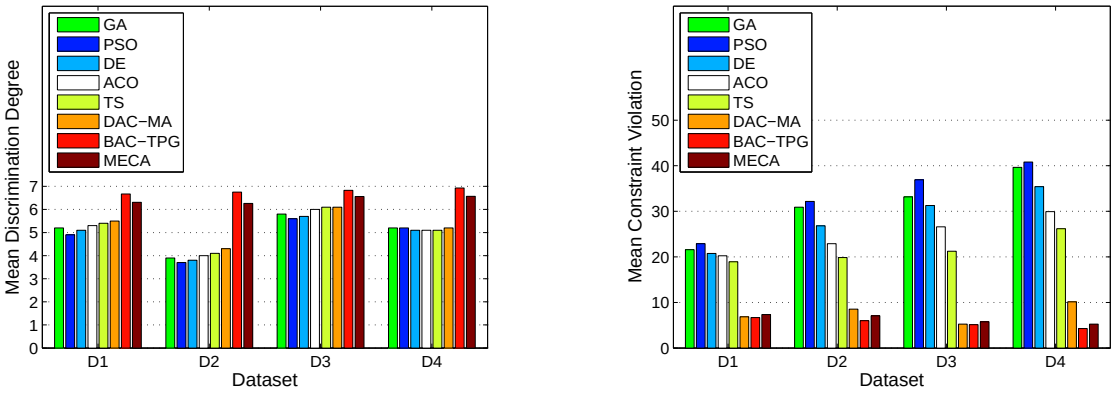


Figure 6.4: Performance Results based on Average Runtime



(a) Mean Discrimination Degree $\mathcal{M}_d^{\mathcal{D}}$

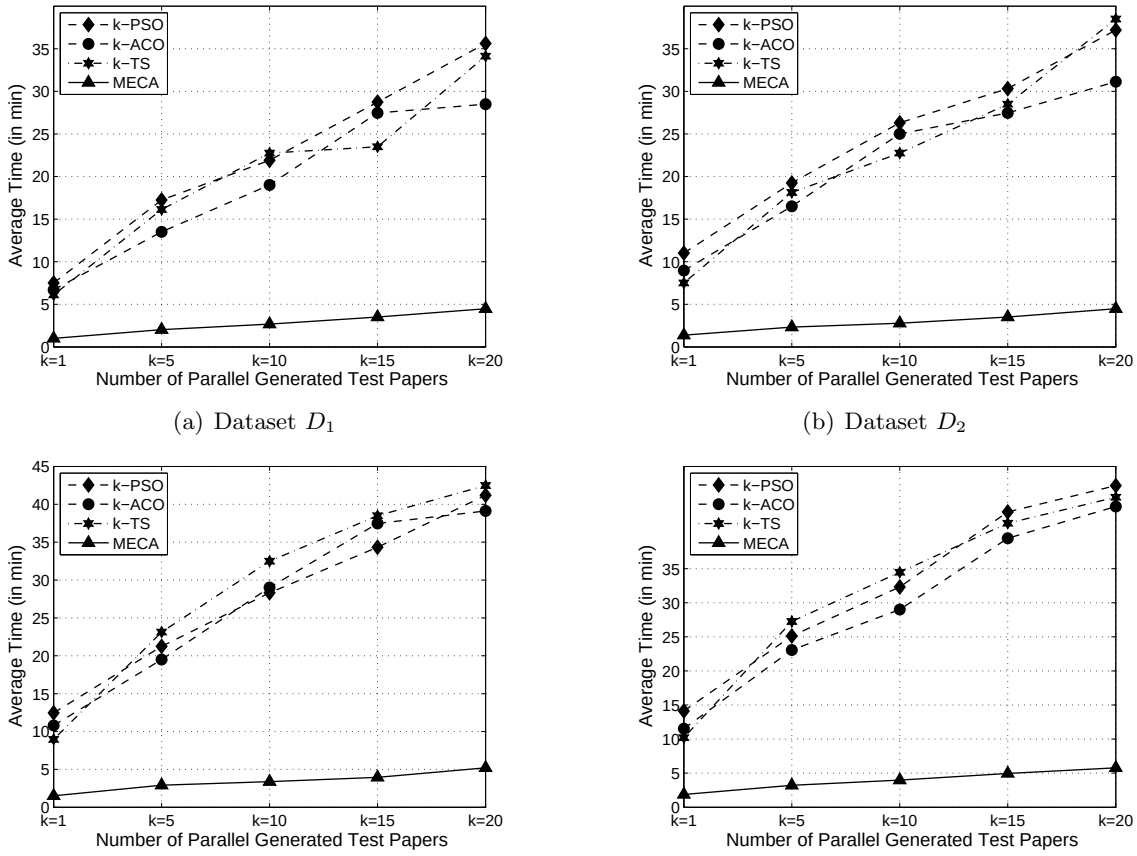
(b) Mean Constraint Violation $\mathcal{M}_c^{\mathcal{D}}$

Figure 6.5: Performance Results based on Average Quality

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.

6.5.3 Performance Results for 1-TPG

To evaluate the quality performance for 1-TPG, we use the Mean Discrimination Degree and Mean Constraint Violation measures for Online-TPG discussed in Section 4.5.3, Chapter 4. The performance results are obtained based on the average results of the 12 test specifications for each dataset.



(a) Dataset D_1

(b) Dataset D_2

(c) Dataset D_3

(d) Dataset D_4

Figure 6.6: Performance Results Based on Average Runtime

Divide-and-Conquer Memetic Algorithm (DAC-MA) [176] and Branch-and-Cut for Test Paper Generation (BAC-TPG) [179] are efficient TPG approaches for online test paper generation. Figure 6.4 gives the runtime performance of MECA in comparison with other 1-TPG techniques. When compared with DAC-MA and BAC for the runtime performance on the balanced uniform distribution datasets D_1 and D_3, DAC-MA outperforms MECA and BAC-TPG. For imbalanced normal distribution datasets D_2 and D_4, MECA achieves better runtime performance. Figure 6.5 gives the quality performance comparison between MECA and other 1-TPG techniques. As shown in Figure 6.5, the quality performance of MECA is consistently better than DAC-MA for the 4 datasets. In Figure 6.5, the quality performance of DAC-MA degrades quite badly on both D_2 and D_4. However, the quality performance of MECA is outperformed by BAC-TPG. As such, MECA can be seen as a tradeoff between the runtime performance and quality performance in comparison with DAC-MA and BAC-TPG.

6.5.4 Performance Results for k-TPG

Performance on Runtime: Figure 6.6 compares the runtime performance of the 4 techniques based on the 4 datasets. The results have clearly shown that the proposed MECA approach outperforms the other heuristic techniques in runtime for the different datasets. MECA generally requires less than 6 minutes to complete the parallel paper generation process. Moreover, the proposed MECA 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. By measuring the average runtime over the 12 specifications and the four datasets, MECA is about 8 times faster in runtime than k-ACO, which is the most efficient one among the other k-TPG techniques.

Performance on Quality: Table 6.1 and Figure 6.7 show 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 6.7, MECA 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, MECA 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.

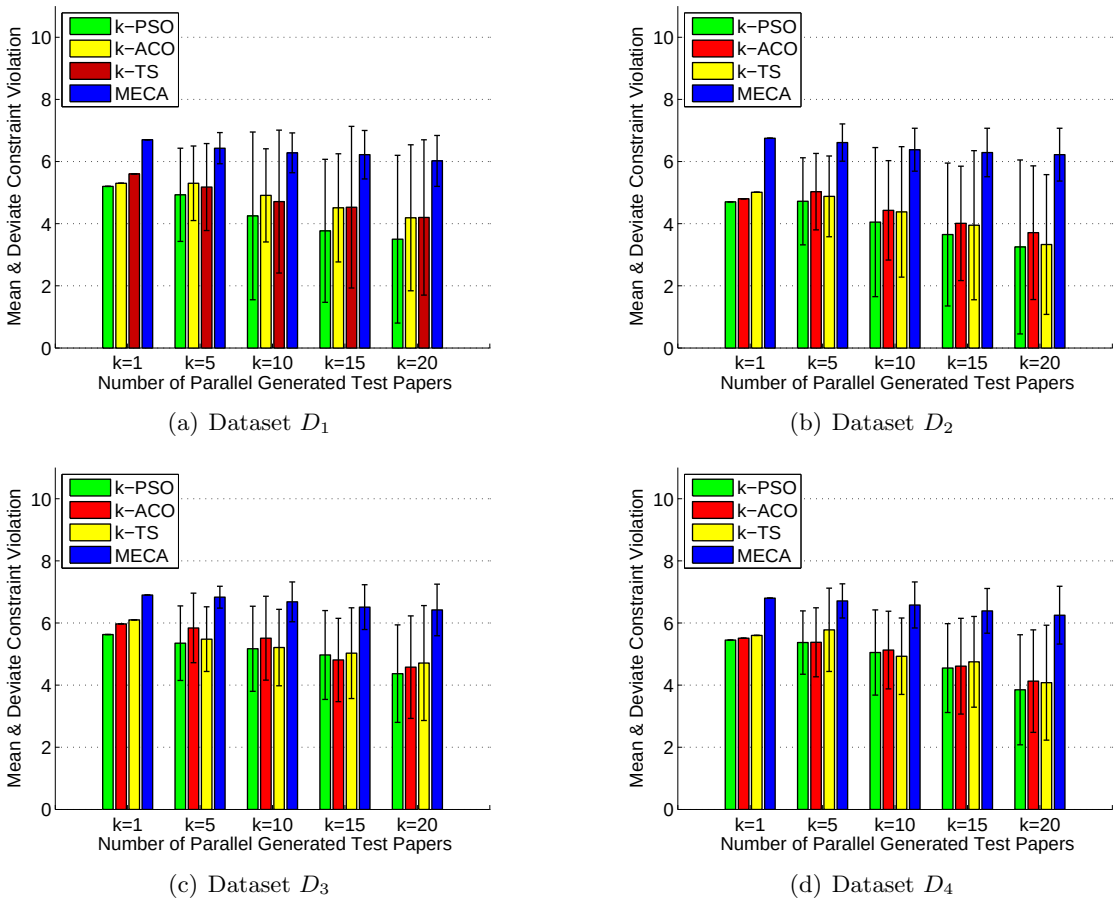
Generally, for a specific value of k, we observe that the quality of the generated parallel test papers of all 4 techniques based on the Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and the Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$ tend to be improved when the dataset size gets larger. Table 6.1 gives the quality performance based on discrimination degree of the 4 techniques. As can be observed from Table 6.1, 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 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.

Table 6.1: Quality Performance Comparison based on Discrimination Degree of MECA 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
$k = 1$	k-PSO	5.20 ± 0	4.70 ± 0	5.63 ± 0	5.45 ± 0
	k-ACO	5.30 ± 0	4.80 ± 0	5.97 ± 0	5.51 ± 0
	k-TS	5.60 ± 0	5.01 ± 0	6.10 ± 0	5.60 ± 0
	MECA	6.70 ± 0	6.75 ± 0	6.90 ± 0	6.80 ± 0
$k = 5$	k-PSO	6.70 ± 1.50	4.72 ± 1.40	5.35 ± 1.20	5.37 ± 1.02
	k-ACO	5.30 ± 1.20	5.03 ± 1.23	5.84 ± 1.12	5.38 ± 1.11
	k-TS	5.18 ± 1.40	4.88 ± 1.30	5.48 ± 1.04	5.78 ± 1.34
	MECA	6.43 ± 0.50	6.61 ± 0.60	6.83 ± 0.35	6.71 ± 0.55
$k = 10$	k-PSO	4.25 ± 2.70	4.05 ± 2.40	5.17 ± 1.37	5.05 ± 1.37
	k-ACO	4.91 ± 1.50	4.43 ± 1.60	5.51 ± 1.35	5.13 ± 1.25
	k-TS	4.71 ± 2.30	4.38 ± 2.10	5.21 ± 1.23	4.93 ± 1.23
	MECA	6.28 ± 0.64	6.38 ± 0.69	6.68 ± 0.64	6.58 ± 0.74
$k = 15$	k-PSO	3.77 ± 2.30	3.65 ± 2.30	4.97 ± 1.43	4.55 ± 1.43
	k-ACO	4.51 ± 1.74	4.01 ± 1.84	4.81 ± 1.34	4.61 ± 1.54
	k-TS	4.53 ± 2.60	3.95 ± 2.40	5.03 ± 1.46	4.75 ± 1.46
	MECA	6.22 ± 0.78	6.29 ± 0.78	6.51 ± 0.72	6.39 ± 0.72
$k = 20$	k-PSO	3.50 ± 2.70	3.25 ± 2.80	4.37 ± 1.57	3.85 ± 1.77
	k-ACO	4.19 ± 2.35	3.71 ± 2.15	4.58 ± 1.65	4.13 ± 1.65
	k-TS	4.20 ± 2.50	3.33 ± 2.25	4.71 ± 1.85	4.08 ± 1.85
	MECA	6.02 ± 0.82	6.22 ± 0.85	6.42 ± 0.83	6.25 ± 0.93
Average	k-PSO	4.33 ± 1.84	4.07 ± 1.78	5.10 ± 1.11	4.85 ± 1.12
	k-ACO	4.84 ± 1.36	4.40 ± 1.36	5.34 ± 1.10	4.95 ± 1.11
	k-TS	4.84 ± 1.76	4.31 ± 1.61	5.31 ± 1.12	5.03 ± 1.15
	MECA	6.33 ± 0.55	6.45 ± 0.53	6.67 ± 0.47	6.71 ± 0.48

In addition, we observe from Table 6.1 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 MECA 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.

Table 6.2 and Figure 6.8 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 MECA 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



(a) Dataset D_1

(b) Dataset D_2

(c) Dataset D_3

(d) Dataset D_4

Figure 6.7: Performance Results on Quality based on Mean Discrimination Degree $\mathcal{M}_d^{k,\mathcal{D}}$ and Deviate Discrimination Degree $\mathcal{V}_d^{k,\mathcal{D}}$

Constraint Violation of MECA tend to decrease whereas the Mean Constraint Violation of the other 3 techniques increase quite fast when the dataset size or the number of specified constraints gets larger. In particular, MECA can generate high quality parallel test papers with $\mathcal{M}_c^{\mathcal{D},k} \leq 10$ for all datasets. Also, MECA 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.

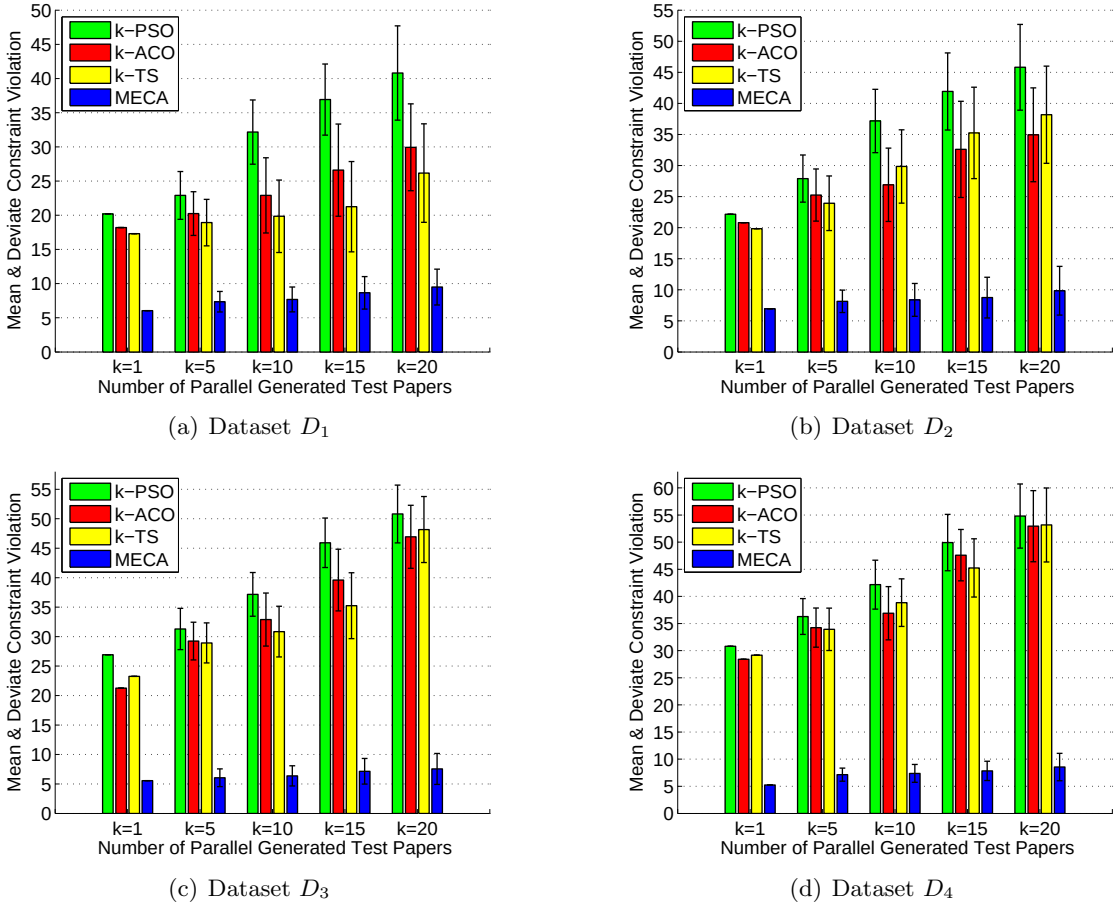
In addition, we find that when k gets larger, the Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$ on a specific dataset $\mathcal{D}$ tends to decrease. Similarly, the Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$ quality tends to increase when k gets larger. These results have shown that the quality based on constraint violation on a specific dataset $\mathcal{D}$ tends to degrade when k gets larger. Table 6.2 gives the quality performance based on constraint violation of the 4 techniques. As can be seen, the quality degradation is small for the proposed MECA approach whereas it is quite significant for the other k-TPG techniques.

Table 6.2: Quality Performance Comparison based on Constraint Violation of MECA 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
$k = 1$	k-PSO	20.19 ± 0	22.17 ± 0	26.92 ± 0	30.81 ± 0
	k-ACO	18.20 ± 0	20.81 ± 0	21.26 ± 0	28.41 ± 0
	k-TS	17.30 ± 0	19.85 ± 0	23.25 ± 0	29.17 ± 0
	MECA	6.05 ± 0	6.95 ± 0	5.56 ± 0	5.25 ± 0
$k = 5$	k-PSO	22.90 ± 3.50	27.90 ± 3.80	31.29 ± 3.50	36.29 ± 3.30
	k-ACO	20.25 ± 3.20	25.25 ± 4.20	29.25 ± 3.20	34.25 ± 3.60
	k-TS	18.93 ± 3.40	23.93 ± 4.40	28.93 ± 3.40	33.93 ± 3.90
	MECA	7.35 ± 1.50	8.15 ± 1.80	6.05 ± 1.50	7.15 ± 1.22
$k = 10$	k-PSO	32.17 ± 4.70	37.17 ± 5.10	37.17 ± 3.70	42.17 ± 4.51
	k-ACO	22.90 ± 5.50	26.90 ± 5.90	32.90 ± 4.50	36.90 ± 4.90
	k-TS	19.85 ± 5.30	29.85 ± 5.90	30.85 ± 4.30	38.85 ± 4.39
	MECA	7.68 ± 1.81	8.38 ± 2.64	6.38 ± 1.71	7.38 ± 1.64
$k = 15$	k-PSO	36.92 ± 5.20	41.92 ± 6.20	45.92 ± 4.20	49.92 ± 5.20
	k-ACO	26.60 ± 6.74	32.60 ± 7.74	39.60 ± 5.24	47.60 ± 4.74
	k-TS	21.25 ± 6.60	35.25 ± 7.36	35.25 ± 5.60	45.25 ± 5.36
	MECA	8.65 ± 2.38	8.75 ± 3.28	7.15 ± 2.18	7.85 ± 1.78
$k = 20$	k-PSO	40.81 ± 6.90	45.81 ± 6.90	50.81 ± 4.90	54.81 ± 5.90
	k-ACO	29.94 ± 6.35	34.94 ± 7.55	46.94 ± 5.35	52.94 ± 6.55
	k-TS	26.17 ± 7.20	38.17 ± 7.82	48.17 ± 5.60	53.17 ± 6.82
	MECA	9.5 ± 2.62	9.85 ± 3.92	7.55 ± 2.62	8.55 ± 2.52
Average	k-PSO	30.59 ± 4.06	34.99 ± 4.40	38.42 ± 3.26	42.80 ± 3.78
	k-ACO	23.57 ± 4.35	28.10 ± 5.07	33.99 ± 3.65	40.022 ± 3.95
	k-TS	20.7 ± 4.50	29.41 ± 5.12	33.29 ± 3.78	40.07 ± 4.09
	MECA	7.84 ± 1.66	8.42 ± 1.82	6.54 ± 1.50	7.23 ± 1.42

In summary, by measuring the average test paper quality over the 12 specifications and the four datasets, MECA has achieved an average of about 85% better test paper quality than k-PSO, which is the most effective one among the other k-TPG techniques. This improvement quantity is computed as an average of improvement on $\mathcal{M}_d^{k,\mathcal{S}}(\mathbf{P})$, $\mathcal{V}_d^{k,\mathcal{S}}(\mathbf{P})$, $\mathcal{M}_c^{k,\mathcal{D}}$, $\mathcal{V}_c^{k,\mathcal{D}}$, $\mathcal{M}_d^{k,\mathcal{D}}$ and $\mathcal{V}_d^{k,\mathcal{D}}$.

Discussion: The good performance of MECA is due to 2 main reasons. Firstly, as MECA 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, MECA can achieve better paper quality and runtime efficiency as compared with other heuristic-based k-TPG



(a) Dataset D_1

(b) Dataset D_2

(c) Dataset D_3

(d) Dataset D_4

Figure 6.8: Performance Results on Quality based on Mean Constraint Violation $\mathcal{M}_c^{k,\mathcal{D}}$ and Deviate Constraint Violation $\mathcal{V}_c^{k,\mathcal{D}}$

techniques. Secondly, MECA is a submodular greedy-based algorithm, which is able to produce good solution in efficient polynomial runtime. Thus, MECA can also improve its computational efficiency on large-scale datasets as compared with the other k-TPG techniques. Moreover, as there are more questions with different attribute values on larger datasets and submodular greedy-based approximation algorithm is effective for optimization, MECA is able to generate high quality test papers with high discrimination degree and small constraint violation.

Furthermore, the deviation of discrimination degree and deviation of constraint violation of the MECA approach tend to increase slightly while these deviations grow quite fast for the other 3 techniques when k gets larger. It is because the MECA 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. For the proposed MECA approach, the deviations are slightly increased when k gets larger because of the statistical nature of the variance. The variance tends to increase slightly when k gets larger even though the values on discrimination degree and constraint violation of the k generated parallel test papers are not much different.

6.6 Summary

In this chapter, we have proposed an effective and efficient Multiobjective Evolutionary Co-Approximation (MECA) approach for parallel test paper generation (k-TPG). The proposed MECA approach is a submodular greedy-based approximation algorithm to find the near-optimal solution by exploiting the submodular property of objective functions. The performance results on various datasets have shown that the MECA approach has achieved 8 times faster in runtime performance and 85% better test paper quality for parallel test paper generation than the other current techniques including k-TS, k-PSO and k-ACO. As such, the proposed MECA approach is a promising approach for k-TPG.

Chapter 7

Automatic Question Difficulty Calibration

For Web-based testing, the difficulty degree of each question in the question database is the most important attribute that needs to be determined. In this chapter, we propose a Content-based Collaborative Filtering (CCF) approach for automatic calibration of difficulty degree of questions. Based on an input question query, whether it is an existing question in the dataset or a new question, the proposed CCF approach will return the difficulty degree for the input question.

7.1 Related Work

In practice, Item Response Theory (IRT) [27] is traditionally applied to calibrate the difficulty degree of each question in the question database based on a large number of historical correct/incorrect information gathered from surveys and tests. However, it requires a large sample of users for calibrating the questions. The recommended sample size varies between 50 and 1000 users [134]. Therefore, it is very expensive to gather the sample information. In [150], Lopez et al. conducted a sampling survey to collect a large number of responses to calibrate 252 multiple-choice Basque language questions. Although many strategies have been applied to reduce human efforts, this process remains very time-consuming.

To overcome this problem, several automatic and semi-automatic methods [227] have been proposed for practical question calibration including proportion correct, learner feedback, expert rating, paired comparison by learners, paired comparison by experts and Elo

rating. Performance results on real data have shown that Proportion Correct Method (PCM) [224] gives the most accurate estimation as compared to IRT-based calibration. PCM is a unique fully automatic calibration method, which was proposed to approximate question difficulty degree. PCM computes question difficulty degree as $\beta_i = \log[\frac{1-n_i/N}{n_i/N}]$, where β_i is the question's difficulty degree, n_i is the number of users who have answered correctly out of the total N users who have answered that question. Although PCM is straightforward to compute and able to estimate the initial difficulty degree even with missing user responses, PCM still requires a large number of historical response information for accurate estimation.

Another way to gather question and response data is to obtain them from Web-based learning environments. With the recent emergence of educational data mining [3, 30], question data and student response information are getting more readily available. These data may come from different sources including standardized tests combined with student demographic data and classroom interactions [6, 29, 142, 208]. With these data, it will certainly help reduce the efforts required for calibrating the difficulty degree of questions. This approach is a much more feasible and inexpensive approach. However, there is one major problem in gathering data from the learning and testing environment. The data often contain missing response information on some questions. This is because learners are free to select questions they want to practice, thereby resulting in the possibly large number of items that have not been answered by learners. Even though IRT can deal with structural incomplete datasets [27], the huge number of missing values can easily lead to non-converging estimation in the IRT model. In addition, the calibration process of IRT is computational expensive. To overcome the impediments of the IRT-based question calibration, other estimation methods [150, 227] based on small datasets have also been investigated. However, the accuracy of these methods is not comparable to IRT-based calibration under the same settings.

In this research, we aim to propose a feasible, efficient and automatic question difficulty calibration for large question datasets gathered from the logging data of Web-based learning and testing environments.

7.2 Proposed Approach

Content-based and collaborative filtering approaches have been extensively used in recommender systems [14, 212]. The content-based approach examines the properties of the items for recommendation, while collaborative filtering (CF) approach recommends items based on

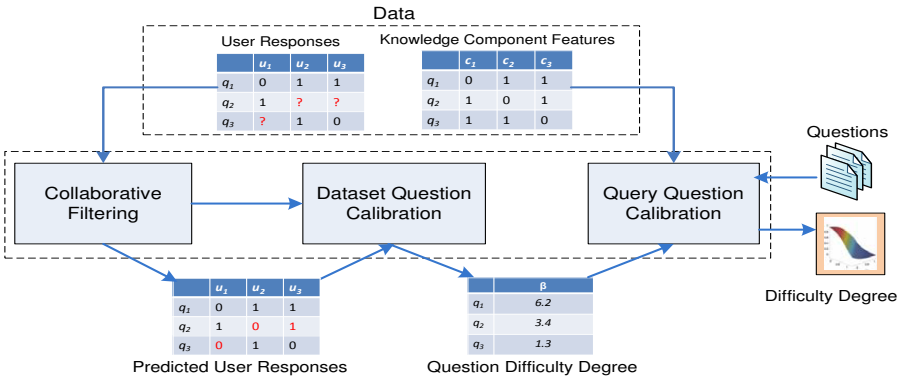


Figure 7.1: The Content-based Collaborative Filtering Approach

similarity measures between users or items from user-item interactions. The items recommended to a user are those preferred by similar users. The CF approach can also be categorized into memory-based and model-based according to user-to-user similarity and item-to-item similarity respectively. Empirically, the item-to-item CF approach has outperformed the user-to-user approach in many real-world applications. Recently, matrix factorization [144] has also been introduced for collaborative filtering. It captures the information by associating both users and items with latent profiles represented by vectors in a low dimensional space. Generally, the CF and matrix factorization approaches are used for recommending warm-start items which have already existed in the dataset. The content-based approach is used for recommending cold-start items which are new items as the CF and matrix factorization approaches are unable to function when the vector representing a new item is empty.

In this chapter, we propose a hybrid Content-based Collaborative Filtering (CCF) approach for question difficulty calibration. The proposed CCF approach combines the advantages of both content-based approach and item-to-item collaborative filtering approach. Collaborative filtering is used to predict unknown responses of questions of a user (or learner) from the known responses of other users. All the gathered user responses are then used for estimating the difficulty degree of questions according to Item Response Theory (IRT). When a user submits a new question as input query, the content-based similarities of the new question with the existing questions from the dataset are calculated and the difficulty degree of the new question is then estimated accordingly. Figure 7.1 shows the proposed CCF approach which consists of 3 main processes: Collaborative Filtering, Dataset Question Calibration and Query Question Calibration.

7.3 Question Difficulty Dataset

There are mainly two types of entities in the question difficulty dataset: users/learners and question items. Let $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ be a dataset consisting of n questions, $\mathcal{K} = \{k_1, k_2, ..., k_m\}$ be a set of m knowledge components. Each question $q_i \in \mathcal{Q}$, where $i \in \{1, 2, ..., n\}$, has 6 attributes $\mathcal{A} = \{q, o, a, t, \beta, k\}$, where q is the question identity, o is the question content, a is the question answer, t is the question time, β is the difficulty degree, and k is the list of knowledge components needed to solve the question. Let $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ be a set of users. Each user has answered some questions with each question $q_i \in \mathcal{Q}$. And the responses to user answers are stored in a binary two-dimensional matrix $\mathcal{R}^{n \times N}$. For each response $r_{ij} \in \mathcal{R}$, its value is 1 if the user answer is correct and 0 if the user answer is incorrect. Otherwise, r_{ij} is marked as unknown.

Table 7.1 shows an example Math dataset with some sample values for the question attributes, knowledge components and responses. Apart from the general question information, question attributes also store the salient question content features, called Knowledge Component Features (KCF), that are highly related to the difficulty degree of the question. Specifically, a knowledge component represents the knowledge that can be used to accomplish certain tasks, that may also require the use of other knowledge components. The knowledge component feature is a generalization of terms representing the concept, principle, fact or skill, and cognitive science terms representing the schema, production rule, misconception or facet. The KCF can be labeled manually by human experts based on the contents of questions and their solutions. The Response Log stores responses based on user-question item pairs which are collaborative information such as the correct/incorrect response to a question item by a user. Such data are generally obtained from the interactions between the learners and any computer-aided tutoring systems.

7.4 Collaborative Filtering

Collaborative Filtering (CF) aims to predict unknown user responses of questions from known user responses. The probability that whether a question item can be answered correctly/incorrectly by a certain user can be predicted collaboratively based on the past user responses of all other questions. In CF, it works by gathering known responses for question items and exploiting similarities in response behavior amongst several question items to predict unknown responses. CF is a neighborhood-based approach, in which a subset of question

Table 7.1: An Example Math Dataset

(a) Question

$\mathcal{Q}$_id	o	a	t	β	k
q_1	,...,.	,...,.	8	?	k_1, k_3
q_2	,...,.	,...,.	9	?	k_1, k_4
q_3	,...,.	,...,.	6	?	k_1
q_4	,...,.	,...,.	9	?	k_1, k_3, k_4
q_5	,...,.	,...,.	7	?	k_1, k_5
q_6	,...,.	,...,.	4	?	k_3, k_4

(b) Knowledge Component

$\mathcal{K}$_id	*knowledge component*
k_1	angles
k_2	triangles
k_3	polygons
k_4	congruece
k_5	similarity

(c) Response Log

$\mathcal{U}$_id	u_1	u_2	u_3	u_4	u_5	u_6
q_1	0	1	1	0	?	1
q_2	1	?	?	1	?	0
q_3	0	0	1	0	?	1
q_4	1	?	0	1	?	?

items are chosen based on their similarities to an active question item, and a weighted combination of the responses is used to predict the response for the active question item. In this research, we use the question item-based similarity approach, which is more scalable than the user-based similarity approach. It is because the number of users could be extremely large, often up to millions, while the number of question items is much smaller. In practice, the item-based similarity approach can generally achieve faster performance and result in better predictions [198].

To predict an unknown response r_{ij} of user u_j for an active question q_i, the CF process performs the following 3 steps: assigning weights to question items, finding neighboring items and computing predicted responses.

Assigning Weights to Question Items: This step assigns a weight $w_{i,l}$ to each question item $q_l, l = 1, ..., n$ with respect to its similarity to the active question item q_i. To achieve this, three most commonly used similarity measures, namely the Cosine Similarity, Adjusted Cosine Similarity and Pearson Correlation Similarity [69, 198], could be used. Let U be the set of all users who have answered both question items q_i and q_l, let $\bar{r}_i$ be the average responses of the question item q_i among all the users in U and let $\bar{r}_v$ be the average responses of any user u_v. Table 7.2 gives the formulas for the three similarity measures.

The Cosine Similarity measures the responses of two question items as vectors in the n-dimensional space, and computes the similarity based on the cosine of the angle between them. When computing the cosine similarity, unknown responses are treated as having the response

Table 7.2: Similarity Measures

Name	Formula
Cosine Similarity	$w_{i,l} = \cos(\vec{r_i}, \vec{r_l}) = \frac{\vec{r_i}.\vec{r_l}}{\|\vec{r_i}\|\|\vec{r_l}\|} = \frac{\sum_{v=1}^{N} r_{i,v} r_{l,v}}{\sqrt{\sum_{v=1}^{N} r_{i,v}^2}\sqrt{\sum_{v=1}^{N} r_{l,v}^2}}$
Adjusted Cosine Similarity	$w_{i,l} = \frac{\sum_{v \in U}(r_{i,v} - \bar{r}_v)(r_{l,v} - \bar{r}_v)}{\sqrt{\sum_{v \in U}(r_{i,v} - \bar{r}_v)^2}\sqrt{\sum_{v \in U}(r_{l,v} - \bar{r}_v)^2}}$
Pearson Correlation Similarity	$w_{i,l} = \frac{\sum_{v \in U}(r_{i,v} - \bar{r}_i)(r_{l,v} - \bar{r}_l)}{\sqrt{\sum_{v \in U}(r_{i,v} - \bar{r}_i)^2}\sqrt{\sum_{v \in U}(r_{l,v} - \bar{r}_l)^2}}$

of zero. However, there is one major drawback for the basic Cosine Similarity measure in the item-based CF approach, i.e., the difference in response scale between different users is not taken into account. The Adjusted Cosine Similarity measure overcomes this drawback by subtracting the corresponding user average from each co-response pair. The Pearson Correlation Similarity measures the extent of linear dependence between two variables. To compute the correlation, it is necessary to isolate the co-response cases, where the users have answered both questions q_i and q_l.

Finding Neighborhood Items: Based on the question item weight $w_{i,l}, l = 1, ..., n$ obtained from the previous step, this step aims to find the top-K (a predefined value) question items, that have responses by the user u_j, having the highest similarities with the question item q_i. These items are called neighborhood items, and they are denoted as $\mathcal{N}(q_i, u_j)$. Finding top-K question items is in fact a K-select problem [140] which can simply be done by scanning the list of items and maintaining a priority queue.

Computing Predicted Responses: This step aims to compute a predicted response r_{ij} of the active question item q_i from the weighted combination of its K selected neighborhood items' responses. Specifically, it computes the weighted average of u_j's responses on the top-K selected items weighted by similarities as the predicted response for the question item q_i by user u_j. It is computed as follows:

$$\widehat{r_{i,j}} = \frac{\sum_{l \in \mathcal{N}(q_i, u_j)} r_{l,j} w_{i,l}}{\sum_{l \in \mathcal{N}(q_i, u_j)} |w_{i,l}|}$$

where $w_{i,l}$ is the similarity between question items q_i and q_l.

7.5 Dataset Question Calibration

In Dataset Question Calibration, it uses the response information to calibrate the difficulty degree of each question item based on the Item Response Theory (IRT) [27]. In psychometrics, IRT models the correct/incorrect response of an examinee of given ability to each question item in a test. It is based on the idea that the probability of a correct response to an item is a statistical function of the examinee (or person) and item difficulty parameters. The person parameter is called latent trait or ability that represents the person's intelligence. Here, we use the popular one parameter logistic model (or 1-PL) [27] for discussing the calibration of difficulty degree. Calibration based on other models such as 2-PL or 3-PL could be done in a similar way.

In question calibration, user responses (i.e. r_{ij}) are used as the observed data. We need to estimate the values of the model parameters $\beta_i, i = 1, ..., n$ and $\theta_j, j = 1, ..., N$ such that these parameters maximize the probability of the observed data. Here, we follow the traditional approach for question calibration by using Maximum Likelihood Estimation (MLE), in which the model parameters $\beta_i, i = 1, ..., n$ and $\theta_j, j = 1, ..., N$ are considered as constants. The 1-PL model has an advantage that the estimation of question difficulty parameters does not require the estimation of user ability parameters.

The mathematical representation of the 1-PL model is based on a 2-dimensional data matrix obtained by the response information on the set of N users with the set of n question items. Figure 7.2 shows the response data matrix. The responses to the question are dichotomously scored, i.e., $r_{ij} = 0$ or 1, where $i = 1, ..., n$ is the question index and $j = 1, ..., N$ is the user index. There is a column vector (r_{ij}) of item response of length n for each user. Recall that $\mathcal{R}_{ij} = [r_{ij}]^{n \times N}$ denotes the response matrix. The vector of column marginal totals $(r_{\cdot j})$ contains the total responses t_j of the user. The vector of row marginal totals $(r_{i\cdot})$ contains the total responses s_i of the question item.

The probability of a correct/incorrect response in the 1-PL model is given as

$$P(r_{ij}|\theta_j, \beta_i) = \frac{e^{r_{ij}(\theta_j - \beta_i)}}{1 + e^{(\theta_j - \beta_i)}}$$

Under the assumption that item responses are stochastically independent, the probability of user u_j yielding item response vector (r_{ij}) for the n questions is

Users

Questions	u_1	u_2	...	u_j	...	u_N	$\sum$
q_1	r_{11}	r_{12}	...	r_{1j}	...	r_{1N}	$r_{1.} = s_1$
q_2	r_{21}	r_{22}	...	r_{2j}	...		$r_{2.} = s_2$
...	...	...	...	...	...	...	...
q_i	r_{i1}	r_{i2}	...	r_{ij}	...	r_{iN}	$r_{i.} = s_i$
...	...	...	...	...	...	...	...
q_n	r_{n1}	r_{n2}	...	r_{nj}	...	r_{nN}	$r_{n.} = s_n$
$\sum$	$r_{.1} = t_1$	$r_{.2} = t_2$	...	$r_{.j} = t_j$	...	$r_{.N} = t_N$	

Figure 7.2: Response Data Matrix

$$P((r_{ij})|\theta_j,(\beta_i)) = \prod_{i=1}^{n} P(r_{ij}|\theta_j,\beta_i) = \prod_{i=1}^{n} \frac{exp[r_{ij}(\theta_j-\beta_i)]}{1+e^{(\theta_j-\beta_i)}} = \frac{exp(t_j\theta_j - \sum_{i=1}^{n} r_{ij}\beta_i))}{\prod_{i=1}^{n} 1+e^{(\theta_j-\beta_i)}}$$

It is necessary to find the probability of a given user's total responses t_j. A user of ability θ_j can obtain a given user's total responses t_j in $\binom{n}{t_j}$ different ways and the sum of probabilities of the individual ways is the probability that $r_{.j} = t_j$. Thus,

$$P(r_{.j} = t_j|\theta_j,(\beta_i)) = \sum_{(r_{ij})}^{t_j} P((r_{ij})|\theta_j,(\beta_i)) = \frac{e^{t_j\theta_j}\gamma_{t_j}}{\prod_{i=1}^{n} 1+e^{(\theta_j-\beta_i)}}$$

where $\sum_{(r_{ij})}^{t_j}$ represents the sum of all group combinations of $\{r_{ij}\}, i = 1, ..., n$ such that $\sum r_{ij} = t_j$, and γ_{t_j} is the elementary symmetric function under the logarithmic transformation of the parameters:

$$\gamma_{t_j} = \sum_{(r_{ij})}^{t_j} exp(-\sum_{i=1}^{n} r_{ij}\beta_i)) = \gamma(t_j;\beta_1,\beta_2,...,\beta_n)$$

Let $d(\theta_j) = \prod_{i=1}^{n} 1+e^{(\theta_j-\beta_i)}$. The probability of user u_j having the total responses t_j is

$$P(r_{.j} = t_j|\theta_j,(\beta_i)) = \frac{e^{t_j\theta_j}\gamma_{t_j}}{d(\theta_j)}$$

Then, if user u_j has the total responses t_j, the item response vector (r_{ij}) has the conditional probability of

$$P((r_{ij})|r_{.j} = t_j) = \frac{P((r_{ij})|\theta_j,(\beta_i))}{P(r_{.j} = t_j|\theta_j,(\beta_i))} = \frac{\frac{exp(t_j\theta_j - \sum_{i=1}^{n} r_{ij}\beta_i))}{d(\theta_j)}}{\frac{e^{t_j\theta_j}\gamma_{t_j}}{d(\theta_j)}} = \frac{exp(-\sum_{i=1}^{n} r_{ij}\beta_i))}{\gamma(t_j;\beta_1,\beta_2,...,\beta_n)}$$

Note that t_j is used in place of the user's ability parameter θ_j. Hence, this probability is not a function of the user's ability level. Thus, the conditional probability of the item response vector is based on the total responses and the set of item parameters (β_i). Assuming that the responses from different users are independent and N users have responded to all question items. If given the values of raw responses t_1, t_2,...,t_N of the N users, then the conditional probability of the item response matrix is

$$P([r_{ij}]|t_1, t_2, ..., t_N, \beta_1, \beta_2, ..., \beta_n) = \frac{exp(-\sum_{j=1}^{N}\sum_{i=1}^{n} r_{ij}\beta_i))}{\prod_{j=1}^{N}\gamma(t_j; \beta_1, \beta_2, ..., \beta_n)}$$

The derived equation can be used as the likelihood function for the estimation of the item parameters $\beta_2, ..., \beta_n$. Here, we note that the possible values of the total responses $t_j, j = 1, ..., N$ are usually less than the number of users possessing these responses. As a result, the denominator of the equation can be rewritten as

$$\prod_{j=1}^{N}\gamma(t_j; \beta_1, \beta_2, ..., \beta_n) = \prod_{r=0}^{n}[\gamma(t_j; \beta_1, \beta_2, ..., \beta_n)]^{f_r}$$

where f_r is the number of users having the total responses t. Let $s_i = \sum_{j=1}^{N} r_{ij}$ be the item response and $\beta \equiv (\beta_i) = (\beta_1, \beta_2, ..., \beta_n)$, the likelihood function is

$$l = \frac{exp(-\sum_{i=1}^{n} s_i\beta_i))}{\prod_{r=0}^{n}[\gamma(t; \beta)]^{f_r}}$$

The log likelihood is used to obtain the maximum of the likelihood function:

$$\mathcal{L} = \log l = -\sum_{i=1}^{n} s_i\beta_i - \sum_{t=1}^{n-1} f_t \log\gamma(t, \beta)$$

To obtain the maximum of the log likelihood estimates, it is necessary to take first order derivatives $\frac{\partial L}{\partial \beta_h}$ of the log likelihood function with respect to different parameters $\beta_h, h = 1, ..., n$. Note that these first derivative values are zeros at maximum point. Hence, there are n of these equations, which need to be solved simultaneously for the estimation of β_h. Here, as n is large, these n equations can be solved approximately and iteratively by the Newton-Raphson [182] method. To use the Newton-Raphson method, the n^2 second-order derivatives $\mathcal{L}_{hk} = \frac{\partial^2 \mathcal{L}}{\partial h \partial k}$ of the log-likelihood are also needed in order to establish the iterative approximation equation for the estimation of question difficulty degrees:

Algorithm 7.1: Newton_Raphson_for_Question_Calibration

Input: $\mathcal{R}_{ij} = [r_{ij}]$ - response information matrix; $t = \{t_1, t_2, ..., t_N\}$ - column total responses; $s = \{s_1, s_2, ..., s_n\}$ - row total responses

Output: $\widehat{\beta_1}, \widehat{\beta_2}, ..., \widehat{\beta_n}$ - question difficulty parameters

begin

1 Estimate initial item difficulty parameters: $\widehat{\beta_i}^{(0)} = \log(\frac{N-s_i}{s_i}) - \frac{\sum_{i=1}^{n} \log(\frac{N-s_i}{s_i})}{n}$;

2 **repeat**

3 Use the current set of parameters $\widehat{\beta_i}$, evaluate the symmetric functions, $\gamma(t, \beta)$, $\gamma(r-1, \beta(h))$, and $\gamma(r-2, \beta(h,k))$;

4 Use these values to evaluate the first and second derivatives of the log likelihood function: $\widehat{\mathcal{L}_h}$, $\widehat{\mathcal{L}_{hh}}$, and $\widehat{\mathcal{L}_{hk}}$;

5 Solve the Newton-Raphson Equations iteratively to obtain improved estimates of the n item difficulty parameters;

until *the convergence criterion is satisfied:* $\sum_{i=1}^{n-1} \frac{(\widehat{\beta_i}^{(p+1)} - \widehat{\beta_i}^{(p)})^2}{n} < 0.001$;

6 **return** $\widehat{\beta_1}, \widehat{\beta_2}, ..., \widehat{\beta_n}$

$$\begin{pmatrix} \widehat{\beta_1} \\ \widehat{\beta_2} \\ \vdots \\ \widehat{\beta_n} \end{pmatrix}_{p+1} = \begin{pmatrix} \widehat{\beta_1} \\ \widehat{\beta_2} \\ \vdots \\ \widehat{\beta_n} \end{pmatrix}_{p} - \begin{pmatrix} \mathcal{L}_{11} & \mathcal{L}_{12} & \cdots & \mathcal{L}_{1n} \\ \mathcal{L}_{21} & \mathcal{L}_{22} & \cdots & \mathcal{L}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{L}_{n1} & \mathcal{L}_{n2} & \cdots & \mathcal{L}_{nn} \end{pmatrix}_{p}^{-1} \times \begin{pmatrix} \widehat{\mathcal{L}_1} \\ \widehat{\mathcal{L}_2} \\ \vdots \\ \widehat{\mathcal{L}_n} \end{pmatrix}_{p}$$

Algorithm 7.1 gives the Newton-Raphson method, which solves the conditional estimates of the item difficulty parameters.

The limiting aspect of the conditional maximum likelihood estimation procedure is the expensive computation of the elementary symmetric functions $\gamma(t; \beta_1, \beta_2, ..., \beta_n)$. Traditionally, this function is computed based on combinatorial enumeration and summing up. As such, a symmetric function $\gamma(t; \beta_1, \beta_2, ..., \beta_n)$ of order t consists of the sum of $\binom{n}{t} \approx O(n^t)$ products, each of which consists of t terms. For instance, when $n = 50$ and $t = 25$, the symmetric function is defined as a sum of about 1.264×10^{14} terms. Thus, the computational demands are very high when computing the symmetric functions directly. As a result, considerable effort has been devoted to find efficient computing procedure that can handle calibrations with a practical number of items. However, the current state-of-the-arts symmetric function computation can handle only at most 100 items [27]. Here, we propose a fast and exact computation of the symmetric function based on the existence of the recursive formulas related to the symmetric functions. Different from previous approaches, our algorithm is more efficient and can handle up to few thousands of items parameters.

Lemma 7.1 (Newton's Formula [209])**.** Given a ground set of n elements $\mathcal{E} = \{e_1, e_2, ..., e_n\}$. Let $\sigma_t(e_1, e_2, ..., e_n) = \sum_{1 \leq i_1 < i_2 \leq n} e_{i_1} e_{i_2}, ..., e_{i_t}$ be the elementary symmetric function of degree t in $\{e_i\}, i = 1, ..., n$. Let $\phi_k(e_1, e_2, ..., e_n) = \sum_{i=1}^{n} e_i^t$ be the power sum function of degree t in $\{e_i\}, i = 1, ..., n$. We can compute efficiently the symmetric functions σ_t based on the power sum functions ϕ_k and vice versa using the following two recursive formulas:

$$\phi_t + t(-1)^t \sigma_t = -\sum_{i=1}^{t-1} (-1)^i \sigma_i \phi_{t-i}$$

$$\sigma_t = \frac{1}{t} \sum_{i=1}^{t} (-1)^{i+1} \phi_i \sigma_{t-i}$$

Therefore, we can compute the symmetric function $\gamma(t; \beta_1, \beta_2, ..., \beta_n)$ with time complexity $O(2n + t^2 + t)$, which is much faster than $O(n^t)$.

7.6 Query Question Calibration

In the proposed approach, users can submit an input query on a question item and its difficulty degree will be returned. In the warm-start scenario, i.e., if the question has already existed in the dataset, its estimated difficulty degree will then be returned immediately. Otherwise, in the cold-start scenario, in which there is no collaborative information available for the given users or items, the proposed approach will predict the new question item's response based on the question items, which are similar in content with the new question item, the user has answered before. Intuitively, two questions are considered to be similar if they have similar content-based knowledge component features.

Let $A \in R^{|\mathcal{Q}| \times m}$ be the matrix of knowledge component attributes, where a_{ik} is 1 iff question item q_i has the knowledge component feature k. There are totally m different knowledge component features or attributes. To measure the content-based similarity for new questions, we use the Jaccard Set Similarity [89] based on the set of knowledge component features as follows:

$$w_{i,l} = Jaccard(A_i, A_l) = \frac{|A_i \cap A_l|}{|A_i \cup A_l|}$$

where A_i and A_l are rows of matrix A w.r.t. the knowledge component attributes of question items q_i and q_l.

After calculating the similarity, the remaining steps are similar to predicting the unknown

Table 7.3: Datasets

Datasets	# Attributes	# KC Features	# Users	# Questions	# Test Sets Questions	
Algebra	23	8197	3310	5013	Test set 1	4400
					Test set 2	100
					Test set 3	513
Bridge-to-Algebra	21	7550	6043	8459	Test set 1	7400
					Test set 2	100
					Test set 3	959

responses for existing questions in the dataset. Specifically, the top-K neighborhood questions, denoted as $\mathcal{N}(q_i)$, which are most similar to the cold-start question q_i are selected. Then, the difficulty degree β_i is computed by aggregating the weighted average of the difficulty degrees on the top-K nearest neighbors weighted by their similarities as follows:

$$\widehat{\beta_i} = \frac{\sum_{l \in \mathcal{N}(q_i)} \beta_l w_{i,l}}{\sum_{l \in \mathcal{N}(q_i)} |w_{i,l}|}$$

7.7 Performance Evaluation

Two datasets, namely *Algebra 2008-2009* and *Bridge-to-Algebra 2008-2009*, obtained from the Knowledge Discovery and Data Mining (KDD) Challenge Cup 2011 [3] are used for performance evaluation. In the experiments, we split each dataset further to form 3 test sets. Test set 1 contains questions whose number of user responses is less than 200. For the remaining questions in each dataset, 100 questions are randomly selected to form test set 2. This test set will be used for evaluating the Query Question Calibration process for cold-start question calibration. The other questions will then be used to form test set 3. Table 7.3 gives a summary on the statistics of the two datasets. In the experiments, test set 1 will be used for populating the data source. Test set 3 will be used for testing both the Collaborative Filtering and Dataset Question Calibration processes. To do this, we randomly hide 75% of the user response information in this test set during the experiment for evaluation. Test set 2 will be used for Query Question Calibration. The user response information are hidden in this test set during the experiment for evaluation.

We evaluate the performance based on each process of the proposed CCF approach: Collaborative Filtering, Dataset Question Calibration and Query Question Calibration. The performance of the Collaborative Filtering process can be evaluated by comparing the predictions with the actual user response information. The most commonly used metric is *Root Mean Squared Error* (RMSE) [110] which is defined as

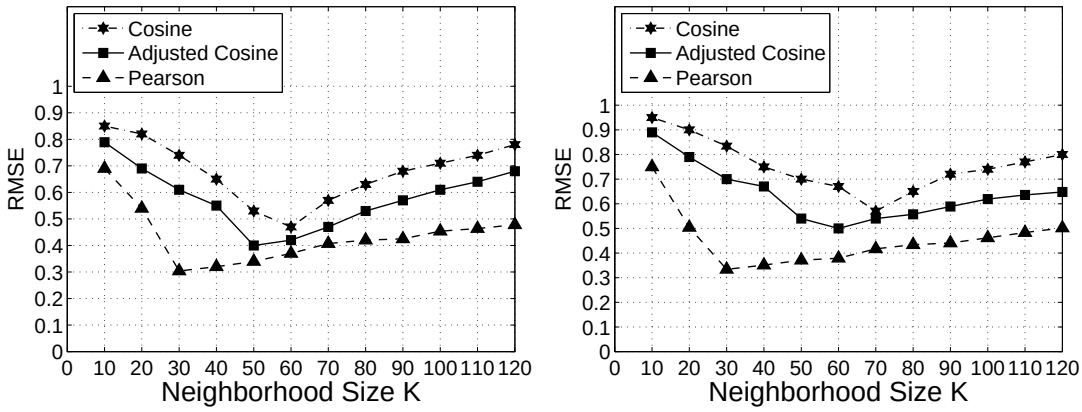
$$RMSE = \sqrt{\frac{\sum_{\{i,j\}}(\widehat{r_{i,j}} - r_{i,j})^2}{N}}$$

where $\widehat{r_{i,j}}$ is the predicted responses for user u_j on item q_i, $r_{i,j}$ is the actual responses, and N is the total number of responses in the test set. Similarly, we also use RMSE for evaluating the performance of the Dataset Question Calibration process of the proposed CCF approach and the Proportion Correct Method (PCM) [226] in comparison with the ground truth IRT calibration.

Figure 7.3 shows the performance results of the Collaborative Filtering process with different parameter settings on similarity measures and neighborhood sizes based on RMSE. In this experiment, we use the 3 different similarity measures: Cosine Similarity, Adjusted Cosine Similarity, and Pearson Correlation Similarity. We also vary the neighborhood size, ranging from 10 to 120 in an increment of 10. It can be observed that the Pearson Correlation Similarity measure has achieved significantly lower RMSE values and the performance has outperformed the two other measures. Moreover, we also observe that the Collaborative Filtering process based on the Pearson Correlation Similarity measure can achieve the best performance with the neighborhood size of about 30. Hence, we have used the Pearson Correlation Similarity measure with the neighborhood size of 30 for subsequent experiments.

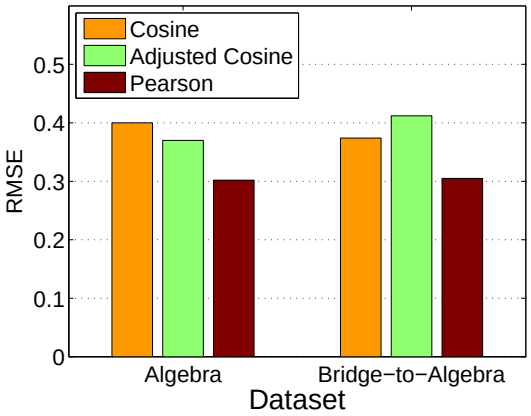
To evaluate the performance of the Dataset Question Calibration process, we compare the performance of the proposed CCF approach with PCM and IRT. Figure 7.4(a) shows the performance results of the Dataset Question Calibration process of CCF and PCM based on RMSE. Note that IRT, which has achieved RMSE of 0, is not shown in the figure. As can be seen, the CCF approach has consistently outperformed PCM on the 2 datasets with less than 40% prediction errors. It has shown that the Collaborative Filtering process is effective to provide sufficient user response information for enhancing the accuracy of the Dataset Question Calibration process. Meanwhile, PCM has achieved considerably lower performance because it calibrates question difficulty with missing user response information.

To evaluate the performance of Query Question Calibration, we only evaluate the performance of the CCF approach because both IRT and PCM cannot work with new cold-start questions. Figure 7.4(b) gives the performance results of the Query Question Calibration process. By definition, the RMSE value ranges from 0 to ∞. It will give a relatively high value to large errors. The lower the value the better the RMSE is. This means the RMSE is most useful when large errors are particularly undesirable. In our experiment, we observe



(a) Algebra

(b) Bridge-to-Algebra



(c) RMSE

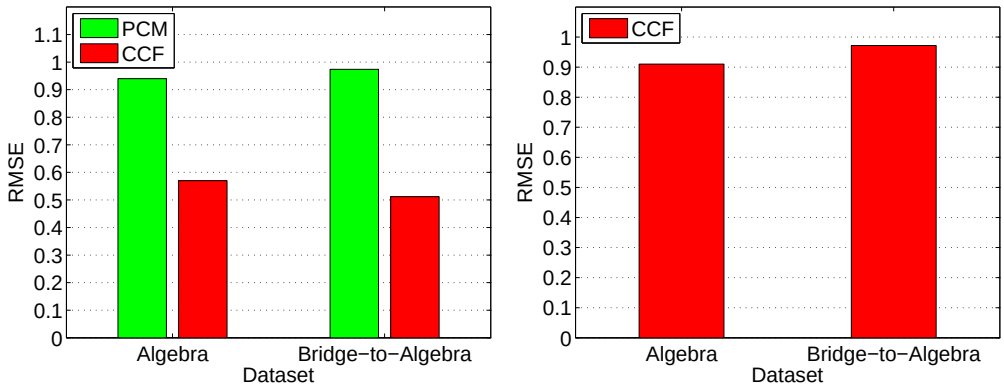
Figure 7.3: Performance Results of the Collaborative Filtering Process based on RMSE

that although the RMSE values are higher than that of the dataset questions, they are still smaller than 1. According to the pre-studies [100, 110] on RMSE metric for recommendation systems, this implies that large errors are unlikely to have occurred. As other approaches are unable to calibrate new questions, the calibrated question difficulty degrees are still very useful. In addition, the performance results have also shown that the knowledge component features are effective to predict the difficulty similarity among questions.

Table 7.4 summarizes the performance results of the 3 processes of the proposed CCF approach. Based on the evalutation criteria of RMSE error [110], the Collaborative Filtering process has achieved reliable predictions of unknown user responses. Moreover, under the situation of missing user responses, the performance of the Dataset Question Calibration process can be enhanced with predicted user responses by the Collaborative Filtering process. In addition, although cold-start question calibration has achieved about 40% higher RMSE error than that of the Dataset Question Calibration, the calibrated question difficulty degrees

Table 7.4: Performance Summary of the 3 Processes in the Proposed CCF Approach

	RMSE	
	Algebra	Bridge-to-Algebra
Collaborative Filtering	0.302	0.305
Dataset Question Calibration	0.57	0.512
Query Question Calibration	0.91	0.93



(a) Dataset Question Calibration (b) Query Question Calibration

Figure 7.4: Performance Results based on the RMSE of the 2 Question Calibration Processes

are still very useful to provide the initial estimation of question difficulty, which can be refined further later when user response information can be gathered more sufficiently.

7.8 Summary

In this chapter, we have proposed an effective Content-based Collaborative Filtering (CCF) approach for question difficulty calibration. The proposed CCF approach consists of 3 main processes, namely Collaborative Filtering, Dataset Question Calibration and Query Question Calibration. The proposed CCF approach has achieved promising performance results with 40% prediction errors less than the traditional Proportion Correct Method technique.

Chapter 8

Automatic Mathematical Solution Assessment

Web-based testing is an effective approach for self-assessment and intelligent tutoring in an e-learning environment. In Web-based testing, automatic solution assessment is a major component, which automatically verifies the correctness of users' answers. In this research, we focus on mathematical solution assessment, which checks the correctness of users' answers by verifying the equivalence of mathematical algebraic expressions in the users' answers and the standard solutions stored in the database. In this chapter, we propose an effective Probabilistic Equivalence Verification (PEV) approach for automatic mathematical solution assessment. Specifically, PEV is a randomized approach based on the numerical equivalence testing of two mathematical expressions.

8.1 Related Work

Automatic mathematical solution assessment is an exact verification problem, which is different from other similarity-based solution assessment problems in educational language subjects such as English and Chinese. Mathematical solution assessment aims to verify whether the mathematical expressions in the answer and the standard solution are equivalent or not, whereas similarity-based solution assessment aims to measure the similarity between the answer and the standard solution according to a threshold. However, the similarity-based techniques for solution assessment are inappropriate for mathematical solution assessment. In

this section, we focus on reviewing 3 existing approaches based on symbolic algebraic computation for automatic mathematical solution assessment. These approaches include computer algebraic approach, rule-based approach and structural approach.

Computer algebraic approaches, which are supported by commercial Computer Algebra Systems (CAS) such as Mathematica [158] and Maple [156], are based on symbolic computation [57, 222] for automatic mathematical solution assessmen. However, the implementation details on the assessment techniques of CASs are unavailable.

Rule-based approaches [47, 167] are based on mathematical rules for equivalence assessment. Starting with a mathematical expression, these approaches transform it into another equivalent expression and compare it with the intended target expression. This process terminates when the target expression is obtained or after a predetermined number of transformation steps. Due to the large number of possible mathematical transformation rules, it is not easy to obtain a correct sequence of transformation steps, which can transform an expression into the equivalent target expression. Moreover, the transformation steps are carried out in a recursive manner. As such, the memory space required [74, 81] to store all the intermediate steps is very huge and grows exponentially with the number of possible mathematical rules applied as well as the complexity of the mathematical expression.

Structural approaches [206, 230] are based on the tree representation of a mathematical expression. In these approaches, two mathematical expressions are first represented as two mathematical trees [74, 81]. Then, these approaches use exact tree matching algorithms [21, 114] to compare the two trees by using dynamic programming [231] for equivalence verification. The structural approach is more efficient than the rule-based approach in terms of memory usage. By aligning the structural information of the two trees, it could find a promising sequence of transformation steps without the need of storing the intermediate steps. However, the structural approaches have a similar problem to the rule-based approaches. It can only work well if the two mathematical expressions are simple and the equivalence can be obtained after a few transformation steps. For two large mathematical expressions or two expressions written in very different forms, it is not easy to obtain a correct sequence of tree transformation steps, which can match the two trees exactly. In addition, the exact tree matching problem is known to be NP-hard. Therefore, the structural approaches are not efficient for verifying the equivalence of two large mathematical expressions.

Due to the nature of symbolic computation [229, 57], all these approaches can avoid false positive errors completely. However, these approaches have generated many false negative

errors if the provided correct answers are written in different forms from the given standard solutions.

8.2 Proposed Approach

The probabilistic or randomized algorithms [18, 168] have been studied quite extensively for the last 30 years. As compared with purely deterministic algorithms, probabilistic algorithms are generally faster, simpler, and easier to implement. Probabilistic algorithms have been applied successfully for solving various fundamental problems [19, 68, 129, 141] ranging from searching and hashing, sorting, network/graph problems, distributed/parallel computing to scheduling. In particular, probabilistic algorithms are the most effective and efficient algorithms for a class of semantic or equivalence verification such as fingerprinting [193], DNA screening [154, 174], primality testing [16, 192] and boolean expression equivalence testing [39].

In most mathematics questions, mathematical expressions are the most common form of the required answers. In this chapter, we propose a novel Probabilistic Equivalence Verification (PEV) approach for equivalence verification of two mathematical expressions. PEV is a general and efficient randomized approach based on probabilistic numerical testing method [18]. The proposed PEV approach consists of the following two main steps:

- *Mathematical Expression Category Identification*: This step aims to identify a given pair of mathematical expressions according to the following 5 mathematical expression categories: Constant, Univariate Polynomial, Multivariate Polynomial, Rational Function and General Function. It can be done based on the Latex format of mathematical expressions (or functions), which are stored in the question database.

- *Probabilistic Equivalence Verification*: This step aims to verify whether the two mathematical expressions are equivalent or not. As each mathematical expression category has a different complexity level for equivalence verification, the proposed PEV approach selects the appropriate verification algorithm according to the different category of the mathematical expression to enhance effectiveness and efficiency.

Table 8.1: Examples of Mathematical Expression Categories

Category	Example	
	Answer	Solution
Constant (CO)	$\frac{2}{\sqrt{5}}$	$\frac{2\sqrt{5}}{5}$
Univariate Polynomial (UP)	$(x+1)^2(1-x)$	$(1+x)(1-x^2)$
Multivariate Polynomial (MP)	$x^4y^2-3x^2y-10$	$(x^2y-2)(x^2y-5)$
Rational Function (RF)	$\frac{x^2+3x+2}{x+1}$	$x+2$
Multivariate General Function (MF)	$\sqrt{x^2-6x+9y^2}$	$\lvert x-3y\rvert$

8.3 Mathematical Expression Category Identification

It identifies the corresponding category of a given pair of mathematical expressions based on their Latex format. The category will then be used for selecting an appropriate algorithm for equivalence verification. Each mathematical expression is classified according to the following 5 mathematical expression categories based on the increasing order of complexity for equivalence verification: Constant (CO), Univariate Polynomial (UP), Multivariate Polynomial (MP), Rational Function (RF) and General Function (GF). The category of a pair of expressions is classified according to the highest order of the two. Table 8.1 gives some examples of answers and solutions for the different mathematical expression categories.

To identify a mathematical expression category, content features of the Latex format of the mathematical expression is used. For example, the expression $(a+b)^2$ has its Latex format of (a+b)^2. Similarly, the expression $e^{2+\ln x}$ has its Latex format of e^{2+\ln x}. The content features of the mathematical expression are first extracted based on a basic lexical analysis technique [222] in symbolic computation. There are four main types of content features: constant, univariate variable, multivariate variable and elementary function. Table 8.2 shows some examples of the content features of answers and solutions. Based on the extracted content features, we can classify the category of the expression according to the decision table given in Table 8.3.

8.4 Probabilistic Equivalence Verification

The mathematical solution assessment problem is specified as follows. Given 2 mathematical functions $f(x_1, x_2, .., x_n)$ and $g(x_1, x_2, .., x_n)$, mathematical answer verification aims to verify the equivalence of two mathematical functions f and g. Mathematically, two functions f

Table 8.2: Examples of Category Features

Content Feature	Example
Constant	2, $\sqrt{5}$, $\frac{3}{10}$, $\sin \pi/6$,...
Univariate Variable	{a}, {b}, {c}, {x}, {y}, {z},...
Multivariate Variable	{a, b, c}, {x, y}, {p, q, r, s},...
Elementary Function	\ln, \frac, \sqrt, $a\hat{}$, \sin, \cot, ...

Table 8.3: Content Features for Mathematical Expression Categories

Category	Content Feature			
	Cons	Uni Variable	Mul Variable	Elem Function
Constant	✔	✘	✘	✘
Univariate Polynomial	✔	✔	✘	✘
Multivariate Polynomial	✔	✘	✔	✘
Rational Function	✔	✔	✔	✘
General Function	✔	✔	✔	✔

and g are said to be *equivalent* ($f \equiv g$), if and only if they have the same value at every point in the domain $\mathcal{D}$ (e.g., $\mathbb{Z}^n, \mathbb{R}^n$, etc.). Note that $f \equiv g$ iff $h = f - g \equiv 0$. Therefore, mathematical equivalence verification can be considered as to check whether function h is equivalent to zero, i.e., $h(x_1, .., x_n) \equiv 0, \forall x_1, .., x_n \in \mathcal{D}$.

In fact, the equivalence definition used by most modern Computer Algebra Systems (CAS) such as Maple and Mathematica are not as strict as the formal equivalence definition in mathematics. CASs such as Maple and Mathematica widely accept that $f(x) = \frac{x2+3x+2}{x+1}$ is equivalent to $g(x) = x + 2$. Mathematically, this equivalence definition is not strictly correct because it ignores the valid domain of variables. However, this equivalence has a practical significance for symbolic computation software. Solving the indefinite computational problem is generally hard for most CASs because it requires finding roots of a mathematical function or factorizing a function into irreducible factor functions. To the best of our knowledge, both of these problems have not been solved completely by the research community [96]. In our research, we follow the less-strict definition of equivalence used by modern CASs.

Algorithm 8.1 presents the high level description of the PEV approach. It repeatedly evaluates the value of f at random points $r_1, r_2, .., r_n$ of the corresponding multivariate variable $x = x_1, x_2, .., x_n$, which are sampled uniformly from a sufficiently large range. If the evaluation result is equal to 0, it returns true. Otherwise, it returns false. Different from

CASs such as Maple and Mathematica, PEV can avoid false negative errors while guaranteeing small possibility for false positive errors to occur. As such, the proposed PEV algorithm is effective for verifying the equivalence of 2 mathematical expressions which are equivalent but may be expressed in different forms.

As PEV verifies the equivalence of two functions by evaluating the values of two functions at random points sampled uniformly over a large range, it inherits a limitation on the computational precision. This is the case when two mathematical functions are indeed different. Unfortunately, PEV may verify incorrectly that they are equivalent. For example, PEV might verify that $f(x) = e^{ax}$ is equivalent to $g(x) = 1 + ax$, where a is a very small constant, e.g. $a = 10^{-20}$. In this case, the traditional symbolic algorithm may outperform our algorithm. However, the computational precision issue may not be very common in mathematical expression verification. This issue is caused by the approximation properties of elementary transcendental functions such as $\sin, \cos, e^x, \ln x$, etc. at a special point such as $x \to 0$. As computer has a limit on the computational precision, our PEV algorithm may not be able to distinguish the difference between $\sin(10^{-50}x)$ and $10^{-50}x$, thereby causing false positive errors.

With the advancement in arithmetic computation, modern computer algebra software can evaluate the value of a mathematical function up to few hundred decimal places of precision. To reduce the errors due to the computational precision, we propose to use either a higher precision or very large values of variable x for equivalent verification. In the first way, we implement our algorithm with a higher precision, e.g. 100 decimal places, rather than using the default 15 decimal places. In the second way, we implement our algorithm with very large values of variables. However, as computer algebra software cannot evaluate a function if either its variable value or the function value is too large, our algorithm is still unable to handle special cases such as verifying the equivalence of: $\sin(10^{-500}x)$ and $10^{-500}x$.

Before discussing this step in details, we need to define the following definitions. Let D be a set in a finite field $\mathbb{F}$ and R be the real number field. In the mathematical verification problem, we aim to determine probabilistically how *close* a test function $f : \mathbb{F} \to \mathbb{R}$ is to a target function g according to a well-defined measure. We introduce the concept of distance to formalize the notion of "closeness".

Definition 8.1 (Distance)**.** Let f be the test function and g be the target function. The closeness of two functions f and g is measured in terms of the *Hamming* distance over D as

Algorithm 8.1: Probabilistic_Equivalence_Verification

Input: $f(x), g(x)$ - test and target functions
Output: YES if $f \equiv g$, NO if $f \not\equiv g$
begin
1 Choose a set of trial points $V \subset \mathcal{D}$ randomly such that $|V|$ is sufficiently large;
2 **repeat**
3 Select a random point x from V uniformly and independently;
4 Test $f(x) \neq g(x)$ and Update $\Delta(f, g)$;
until $|V|$;
5 **if** $\Delta(f, g) \geq 1 - \epsilon$ **then return** *NO* ;
6 **else return** *YES* ;

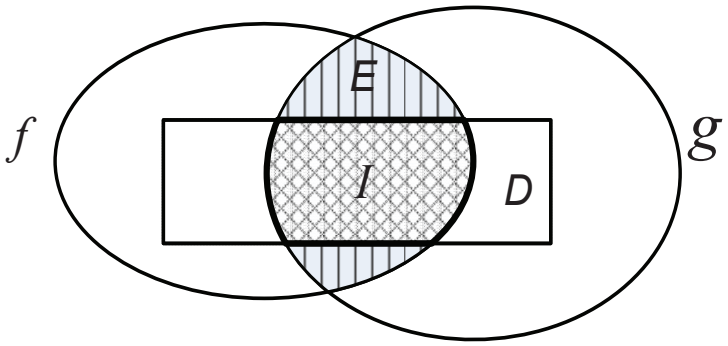


Figure 8.1: Distance Concept Illustration by Using Venn Diagram

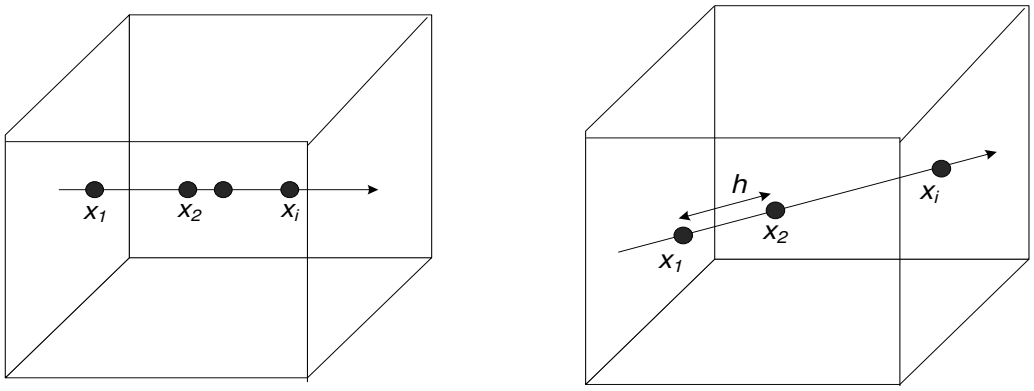
follows:

$$\Delta(f, g) = \Pr_{x \in D}[f(x) \neq g(x)] = \frac{|\{x \in D | f(x) \neq g(x)\}|}{|D|}$$

Figure 8.1 shows a Venn Diagram to illustrate the distance concept. In the diagram, let E be the intersection region of the domains of f and g where $f(x)$ coincides $g(x)$, i.e., $f(x) = g(x)$. Let $I = E \cap D$ be the intersection of E and D. Therefore, the Hamming distance can be seen as the complement of the ratio of the areas of I and D, i.e., $\Delta(f, g) = 1 - \frac{area(I)}{area(D)}$. As such, we can estimate the Hamming distance $\Delta(f, g)$ of f and g by sampling random points of $x \in D$ and testing whether $f(x) \neq g(x)$.

We wish to design an equivalence verification algorithm which has the following properties:

- **Correctness:** If the test function f is equivalent to the target function g, then the probabilistic equivalence verification algorithm is likely to return *YES*, i.e., $\Delta(f, g)$ is small with high probability. If the test function f is not equivalent to the target function g, then the probabilistic equivalence verification algorithm is likely to return *NO*, i.e., $\Delta(f, g)$ is high with high probability.



(a) Axis-based Selection (b) Line-based Selection

Figure 8.2: Query Point Selection Strategy

- **Query Size:** We would like to evaluate the functions $f(x)$ and $g(x)$ with as few points of x as possible. The query size determines the efficiency of the verification algorithm.

Firstly, the correctness property ensures that the algorithm will return *YES* with a high probability if the two expressions have the same value on sufficiently large number of random points of x. In this case, the correctness property determines the false negative error. To guarantee the correctness property, we need to prove that the Hamming distance $\Delta(f, g)$ is small. Secondly, the correctness property also guarantees that the algorithm will return *NO* with a high probability if the two expressions do not agree on sufficiently large number of random points of x. The correctness property determines the false positive error. To guarantee the correctness property, we need to prove that the Hamming distance $\Delta(f, g)$ is high. We wish to design an effective and efficient equivalence verification algorithm, which can minimize the false positive error, false negative error and the query size. Depending on the mathematical category of the two expressions $f(x)$ and $g(x)$, we have different appropriate verification algorithms accordingly. These algorithms differ on the number of query points, the strategy to select query points and the domain values of query points.

We use the following two strategies to select query points: axis-based selection and line-based selection. The axis-based selection strategy is used for univariate function verification whereas the line-based selection strategy is used for multivariate function verification. The axis-based selection strategy chooses random points along the axis of a variable. The line-based selection strategy chooses random points along a line $l = \{x_1 + th\}$, where $t = 1, 2, .., k$, which passes through x_1 with a constant slope h: $x_1 \in \mathbb{F}^m, h \in \mathbb{F}^m$. Figure 8.2 gives an illustration of the two strategies.

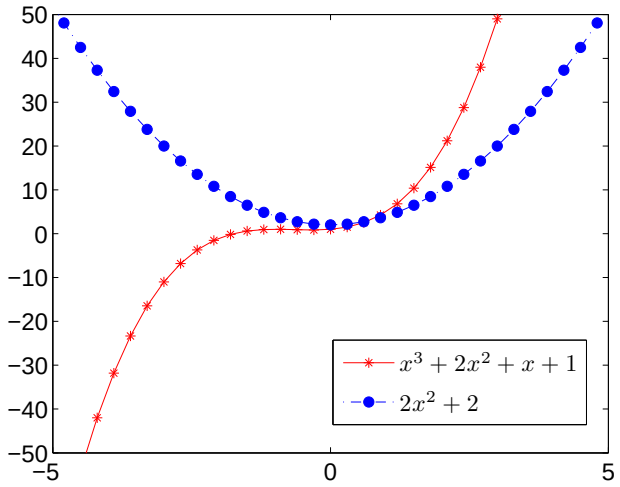


Figure 8.3: Univariate Polynomial Equivalence Verification

In this research, we assume that the functions $f(x)$ and $g(x)$ can be evaluated accurately by using a computer algebra system. Thus, this assumption simplifies the computational implementation issues to obtain fixed point accuracy.

8.4.1 Univariate Polynomial Verification

This section aims to propose an algorithm for equivalence verification of a univariate polynomial of degree at most d. The basic idea of our algorithm is based on the implication of the Lagrange Interpolation Polynomial Theorem [107], which states that two univariate polynomials $g(x)$ and $f(x)$ with degree at most d can coincide at most $d+1$ points of x. Otherwise, they must be equal to zero, i.e., $f(x) \equiv g(x) \equiv 0$. Therefore, it is sufficient to test on $d+2$ random points of $x \in \mathbb{F}$ to verify the equivalence of two univariate polynomial expressions. For efficiency, these random points are chosen based on the axis-based selection strategy with integer points for evaluating the values of $f(x)$ and $g(x)$. Figure 8.3 shows an example of equivalence verification for univariate polynomials $x^3 + 2x^2 + x + 1$ and $2x^2 + 2$. Algorithm 8.2 presents the equivalence verification algorithm for univariate polynomials.

Definition 8.2 (Univariate Polynomial)**.** Let $\mathbb{F}$ be a finite field and $|\mathbb{F}| = q$ and $d < q$. A function $f : \mathbb{F}^m \leftarrow \mathbb{R}$ is said to be a univariate polynomial with degree d if it can be expressed as follows:

$$f(x) = \sum_{i \leq d} a_i x^i \; x \in \mathbb{F}$$

Theorem 8.1 (Lagrange Interpolation Polynomial [107])**.** Given a univariate polynomial

Algorithm 8.2: Univariate_Polynomial_Verification

Input: $f(x), g(x)$ - test and target functions
Output: YES if $f \equiv g$, NO if $f \not\equiv g$
begin
1 Determine the maximum degree d of $f(x)$ and $g(x)$;
2 Choose a set V of $2d+2$ random points $x \in \mathbb{Z}$;
3 **repeat**
4 Select a random point x from V uniformly and independently;
5 Test $f(x) \neq g(x)$ and Update $\Delta(f, g)$;
until $|V|$;
6 **if** $\Delta(f, g) \geq 1 - \epsilon$ **then return** *NO* ;
7 **else return** *YES* ;

$y = f(x)$ and its values at a set of k + 1 data points:

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

where $x_i \neq x_j, \forall i \neq j$. Then, there is a unique univariate polynomial $L(x)$ with degree k, which is equivalent to $f(x)$. The Lagrange interpolation polynomial $L(x)$ of y can be determined as follows:

$$L(x) = \sum_{j=0}^{k} y_j \ell_j(x)$$

where $\ell_j(x)$ are Lagrange basis polynomials, which are determined as follows:

$$\ell_j(x) = \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

Based on the Lagrange Interpolation Polynomial Theorem, we can straightforwardly derive the following theoretical results of Algorithm 8.2.

Corollary 8.1 (Correctness)**.** Let d be the maximum degree of the univariate test polynomial $g(x) : \mathbb{V} \to \mathbb{R}$ and the univariate target polynomial $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least $d + 2$ integer points of x. If there exists a function $f(x)$ satisfying that $f(x_i) = g(x_i), x_i \in V, \forall i = 1, .., d+2$, then $f(x) \equiv g(x)$. In other words, the two polynomials $f(x)$ and $g(x)$ are equivalent, i.e., $\Delta(f, g) = 0$. In addition, if $f(x) \not\equiv g(x)$, then $f(x)$ agrees with $g(x)$ at at most $d + 1$ points. Hence, we have the probability of false positive error:

$$\Pr[f(x) = g(x)] = \frac{|\{x \in V | f(x) = g(x)\}|}{|V|} \leq \frac{d+1}{|V|} = \epsilon$$

By definition, we have $\Delta(f, g) \geq 1 - \epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability.

Corollary 8.2 (Query Size)**.** Let d be the maximum degree of the univariate test polynomial $g(x) : \mathbb{V} \rightarrow \mathbb{R}$ and the univariate target polynomial $g(x) : \mathbb{V} \rightarrow \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of integer of points x. The minimal query size of V such that the correctness property is guaranteed is $d + 2$.

In fact, the larger the query size V is, the smaller the false positive error is. In this work, we choose $|V| = 4d + 2$ for equivalence verification of univariate polynomials. As such, the probability of false positive error is at most 0.25.

8.4.2 Multivariate Polynomial Verification

This section aims to propose an algorithm for equivalence verification of a multivariate polynomial of degree at most d. Note that a multivariate polynomial $f(x_1, .., x_n)$ of degree d can be transformed into an equivalent univariate polynomial $g(y)$ of degree d^{n+1} by using Kronecker substitution [222]. However, it is not efficient to apply the univariate verification algorithm for the transformed univariate polynomial because of the large degree d^{n+1}. Here, we propose a novel algorithm based on the properties of multivariate polynomials [107]. Similar to the univariate polynomial, in this section, we will prove that two multivariate polynomials $g(x)$ and $f(x)$ with degree at most d can coincide at most d points of x. Otherwise, they must be equal to zero, i.e., $f(x) \equiv g(x) \equiv 0$. Therefore, it is sufficient to test on $d + 1$ random points of $x \in \mathbb{F}$ to verify that the two expressions are different. To verify the equivalence of two multivariate polynomial expressions, we choose random points according to the line-based selection strategy. For efficiency, these random points are selected based on integer points for evaluating the values of $f(x)$ and $g(x)$. Algorithm 8.3 presents the equivalence verification algorithm for multivariate polynomials.

Definition 8.3 (Multivariate Polynomial)**.** Let $\mathbb{F}$ be a finite field and $|\mathbb{F}| = q$ and $d < q$. A function $f : \mathbb{F}^m \leftarrow \mathbb{R}$ is said to be a multivariate polynomial with degree d if it can be

Algorithm 8.3: Multivariate_Polynomial_Verification

Input: $f(x), g(x)$ - test and target functions
Output: YES if $f \equiv g$, NO if $f \not\equiv g$
begin
1 Determine the maximum degree d of $f(x)$ and $g(x)$;
2 Choose a set V of $d+1$ random points $x \in \mathbb{Z}$;
3 **repeat**
4 Select a random point x and h from V uniformly and independently;
5 Test $f(x+(i-1)h) \neq g(x+(i-1)h)$ and Update $\Delta(f,g)$;
until $|V|$;
6 **if** $\Delta(f,g) \geq 1-\epsilon$ **then return** *NO* ;
7 **else return** *YES* ;

expressed as follows:

$$f(x) = f(x_1, x_2, ..., x_m) = \sum_{i_1+...+i_m \leq d} a_{a_1...i_m} x^{i_1} ... x^{i_m},\ \forall (x_1, ..., x_m) \in \mathbb{F}^m$$

The degree of a particular term in a polynomial f is defined as the sum of its variable exponents in that term, whereas the degree of f is defined as the maximum degree of all the terms in f. For example, the degree of the term $x_1^6 x_2 x_3^5$ is 12, and the degree of $f = x_1^3 x_2^2 + x_1^6 x_2 x_3^5$ is 12.

Definition 8.4 (Line-based Point Selection)**.** A set of points $x_1, x_2, ..., x_n, x_i \in \mathbb{F}^m$ satisfies the line-based selection strategy if there exists h such that $x_i = x_1 + (i-1)h$.

Here, we first prove that the PEV algorithm has a guaranteed error bound for polynomial functions.

Lemma 8.1 (Hardy et al. [107])**.** Let $f(x)$ be a multivariate polynomial with degree d. The line-based points $\{(x_i, y_i) | i \in \{1, ..., d+2\}\}$, where $x_i = x_1 + (i-1)h; x_i \in \mathbb{Z}^m, h \in \mathbb{Z}^m$, are located on $f(x)$ if and only if $\sum_{i=1}^{d+1} \alpha_i y_i = 0$, where $\alpha_i = (-1)^{i+1} \binom{d+2}{i}$.

Based on Lemma 8.1, it is effective and efficient to select line-based query points for multivariate polynomial verification.

Theorem 8.2 (Correctness)**.** Let d be the maximum degree of the multivariate test polynomial $f(x) : \mathbb{V} \to \mathbb{R}$ and the multivariate target polynomial $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least d multivariate integer points of $x \in \mathbb{Z}^m$. If $f(x) \not\equiv g(x)$, then the probability that the PEV algorithm has false positive error to be bounded by

$$\Pr[f(x) = g(x)] \leq \frac{d}{|V|} = \epsilon$$

By definition, we have $\Delta(f, g) \geq 1 - \epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability.

Proof: It is straightforward that the PEV algorithm has no false negative error as it is a trial-and-error method. The false positive error bound can be proved using mathematical induction on the number of variables n of the polynomial. Let $P(x) = f(x) - g(x)$ be a multivariate polynomial. It is equivalent to prove that $\Pr[P(r_1, r_2, .., r_n) = 0] \leq \frac{d}{|V|}$.

For the case $n = 1$, if $P \not\equiv 0$, then P is a univariate variable polynomial with degree d. Thus, it has at most d roots. Hence, $\Pr[P(r_1, r_2, .., r_n) = 0] \leq \frac{d}{|V|}$.

For the inductive step, we can solve the multivariate case by fixing $n-1$ variables $x_2, .., x_n$ to $r_2, .., r_n$ and apply the result from the univariate case. Assume that $P \not\equiv 0$, we compute $\Pr[P(r_1, .., r_n) = 0]$. First, let k be the highest power of x_1 in P. We can rewrite P as

$$P(x_1, x_2, .., x_n) = x_1^k A(x_2, .., x_n) + B(x_1, x_2, .., x_n)$$

for some polynomials A and B. Let $\mathcal{X}_1$ be the case that $P(r_1, .., r_n)$ is evaluated to 0, and $\mathcal{X}_2$ be the case that $A(r_2, .., r_n)$ is evaluated to 0. Using Bayes rule, we can rewrite the probability that $P(r_1, .., r_n) = 0$ as

$$\begin{aligned}
\Pr[P(r) = 0] &= \Pr[\mathcal{X}_1] \\
&= \Pr[\mathcal{X}_1|\mathcal{X}_2]\Pr[\mathcal{X}_2] + \Pr[\mathcal{X}_1|\neg\mathcal{X}_2]\Pr[\neg\mathcal{X}_2] \\
&\leq \Pr[\mathcal{X}_2] + \Pr[\mathcal{X}_1|\neg\mathcal{X}_2] \qquad (8.1)
\end{aligned}$$

Consider the probability of $\mathcal{X}_2$ (or $A(r_2, .., r_n) = 0$), the polynomial A has degree $d - k$ with $n - 1$ variables. So, by inductive hypothesis on the number of variables, we obtain

$$\Pr[\mathcal{X}_2] = \Pr[A(r_2, .., r_n) = 0] \leq \frac{d - k}{|V|} \qquad (8.2)$$

Similarly, if $\neg\mathcal{X}_2$ (or $A(r_2, .., r_n) \neq 0$), P is a univariate polynomial degree k. By inductive hypothesis, we have

$$\Pr[\mathcal{X}_1|\neg\mathcal{X}_2] = \Pr[P(r_1,.,r_n) = 0|A(r_2,.,r_n) \neq 0] \leq \frac{k}{|V|} \tag{8.3}$$

Finally, we can substitute the results from Equations (8.1) and (8.2) into Equation (8.3) to complete the inductive step. ∎

Theorem 8.3 (Query Size)**.** Let d be the maximum degree of the multivariate test polynomial $f(x) : \mathbb{V} \to \mathbb{R}$ and the multivariate target polynomial $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ be a set of integer points of x. The minimal query size of V such that the correctness property is guaranteed is $d + 1$.

Based on Theorem 8.4 for the case of polynomials, the probability of false positive errors can be guaranteed to any desired small number ϵ (e.g., $\epsilon = 10^{-3}$) by conducting sufficiently large number of trials with $|V| \geq \frac{d}{\epsilon}$.

8.4.3 General Function Verification

This section aims to propose an algorithm for equivalence verification of a general function. The proposed algorithm is based on the idea that every mathematical expression can be represented in terms of the quotient of two polynomials [222] with respect to the same number of variables. Hence, we can adapt the verification algorithms of univariate polynomial and multivariate polynomial for general function. In the proposed approach, the probability of correct equivalence verification is at least $1 - \epsilon$, where ϵ is the probability of the false positive error. Figure 8.4 shows an example of equivalence verification for univariate general functions $e^{-2\cos x}$ and $e^{-2\sin x}$. Algorithm 8.4 and Algorithm 8.5 present the equivalence verification algorithms for univariate general functions and multivariate general functions respectively.

Theorem 8.4 (Gathen et al. [222])**.** Every mathematical expression $f(x)$ can be represented as a quotient of two polynomials i.e, $f(x) = \frac{f_1(x)}{f_2(x)}$ for some polynomials $f_1(x)$ and $f_2(x)$.

Example 8.1. The following elementary functions can be represented as polynomials as follows:

- $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^5}{5!}$
- $\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5}$

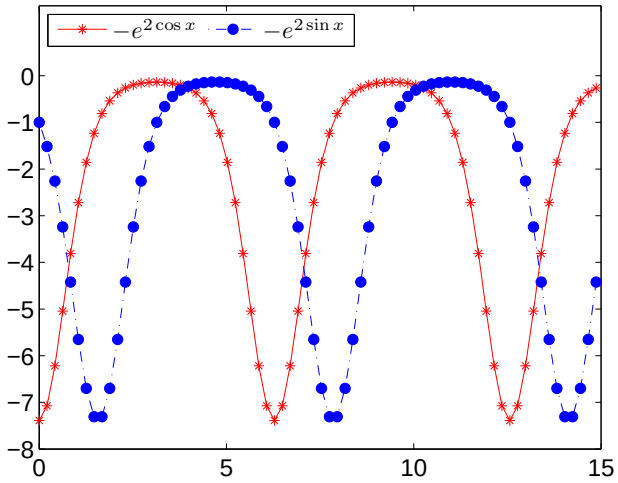


Figure 8.4: General Function Verification

Algorithm 8.4: Univariate_General_Function_Verification

Input: $f(x), g(x)$ - test and target functions
Output: YES if $f \equiv g$, NO if $f \not\equiv g$
begin
1 Determine the maximum degree d of $f(x)$ and $g(x)$;
2 Choose a set V of 2^d random points $x \in \mathbb{Z}$;
3 **repeat**
4 Select a random point x from V uniformly and independently;
5 Test $f(x) \neq g(x)$ and Update $\Delta(f, g)$;
until $|V|$;
6 **if** $\Delta(f, g) \geq 1 - \epsilon$ **then return** *NO* ;
7 **else return** *YES* ;

- $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}$
- $\frac{1}{x-1} + \frac{2x-1}{x+1} = \frac{x+1+(2x-1)(x-1)}{x^2-1}$
- $\sqrt{x} = 1 + \frac{1}{2}(x-1) - \frac{1}{8}(x-1)^2 + \frac{1}{16}(x-1)^3 - \frac{5}{128}(x-1)^4$

One issue to utilize equivalence verification algorithms for polynomials is that a general function does not have the definition of degree similar to polynomials. Although Gathen et al. [222] proved the existence of representing $f(x)$ by the quotient of two polynomials $f_1(x)$ and $f_2(x)$, it did not specify the procedure on how to find $f_1(x)$ and $f_2(x)$. In fact, it is not necessary to find $f_1(x)$ and $f_2(x)$ concretely. It is sufficient to estimate the upper bounded degree of $f_1(x)$. In [38], it showed that the degree of the numerator polynomial $f_1(x)$ is bounded by $2^{M(f)}$, where $M(f)$ is the number of multiplications and divisions used to compute f.

Algorithm 8.5: Multivariate_General_Function_Verification

Input: $f(x), g(x)$ - test and target functions
Output: YES if $f \equiv g$, NO if $f \not\equiv g$
begin
1 Determine the maximum degree d of $f(x)$ and $g(x)$;
2 Choose a set V of 2^d random points $x \in \mathbb{Z}$;
3 **repeat**
4 Select a random point x and h from V uniformly and independently;
5 Test $f(x + (i-1)h) \neq g(x + (i-1)h)$ and Update $\Delta(f, g)$;
until $|V|$;
6 **if** $\Delta(f, g) \geq 1 - \epsilon$ **then return** *YES*;
;
7 **else return** *NO*;
;

To measure the runtime, let $T(f)$ be the complexity of evaluating a general expression f. $T(f)$ counts the number of arithmetic operations needed to evaluate the expression. Here, each basic operation is counted as one. For example, $T((x^8z^2 + w)^2) = 7$ because $T(x^8) = 3$, $T(z^2) = 1$ (square function), $T(x^8z^2 + w) = 6$ and $T((x^8z^2 + w)^2) = 7$.

Here, we prove that the PEV algorithm has a guaranteed error bound for univariate and multivariate general functions.

Theorem 8.5 (Correctness)**.** Let $d = \max\{2^{M(f)}, 2^{M(g)}\}$ be the maximum degree of the test function $f(x) : \mathbb{V} \to \mathbb{R}$ and the target function $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least d multivariate integer points $x \in \mathbb{Z}$ or $x \in \mathbb{Z}^m$. If $f(x) \not\equiv g(x)$, then the probability that the PEV algorithm has false positive error to be bounded by

$$\Pr[f(x) = g(x)] \leq \frac{d}{|V|} = \epsilon$$

By definition, we have $\Delta(f, g) \geq 1 - \epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability.

Proof: Let $f(x)$ and $g(x)$ be represented in terms of quotients of polynomials, i.e., $f(x) = \frac{f_1(x)}{f_2(x)}$ and $g(x) = \frac{g_1(x)}{g_2(x)}$. Rewriting $f(x)$ and $g(x)$ as

$$f(x) = \frac{f_1(x) * g_2(x)}{f_2(x) * g_2(x)} \text{ and } g(x) = \frac{g_1(x) * f_2(x)}{f_2(x) * g_2(x)}$$

Then, the problem becomes verifying the equivalence of two polynomials $f'(x) = f_1(x) * g_2(x)$ and $g'(x) = g_1(x) * f_2(x)$. Similar to the proofs in Theorem 8.2 and Theorem 8.4, we can

show that

$$\Pr[f'(x) = g'(x)] \leq \frac{d}{|V|} = \epsilon$$

i.e., $\Delta(f, g) \geq 1 - \epsilon$. ∎

Based on Theorem 8.5, we can straightforwardly derive the following theoretical result of Algorithm 8.5.

Corollary 8.3 (Query Size)**.** Let d be the maximum degree of the test function $f(x) : \mathbb{V} \to \mathbb{R}$ and the target function $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of integer points of x. The minimal query size of V such that the correctness property is guaranteed is $\max\{2^{M(f)}, 2^{M(g)}\} + 1$.

Based on Theorem 8.7, the probability of false positive errors can be guaranteed to any desired small number ϵ (e.g., $\epsilon = 10^{-3}$) by conducting sufficiently large number of trials with $|V| \geq \frac{d}{\epsilon}$.

Table 8.4 shows the comparison of the 4 equivalence verification algorithms for different mathematical expression categories.

Table 8.4: Comparison of Equivalence Verification Algorithms based on the Mathematical Expression Category (where d is the maximum degree of f and g, m is the number of variables and ϵ is the probability of false positive error).

	Query Size	Runtime	Error Bound
Univariate Polynomial	$O(\frac{d+2}{\epsilon})$	$O(d^2)$	ϵ
Multivariate Polynomial	$O(\frac{d+1}{\epsilon})$	$O(m^2d^2)$	ϵ
Univariate General Function	$O(\frac{2^d}{\epsilon})$	$O(T_f^2)$	ϵ
Multivariate General Function	$O(\frac{2^d}{\epsilon})$	$O(m^2T_f^2)$	ϵ

8.5 Performance Evaluation

To evaluate the proposed PEV approach, we have devised 100 answer-solution pairs with 50 correct (positive) answers, and 50 incorrect (negative) answers for 100 test questions based on the Undergraduate Math dataset. The test questions are general questions selected according to some standard mathematical functions. Table C.1 and Table C.2 in Appendix C list the pairs of answers and solutions for the test cases. Table 8.5 shows a summary of the test data.

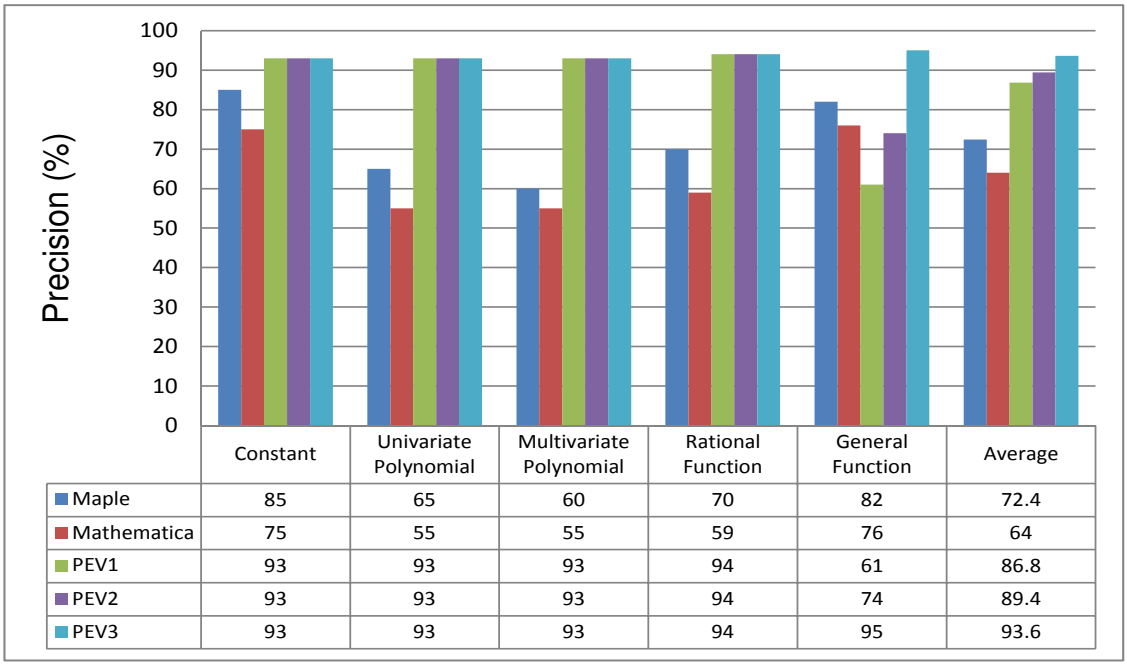
We use the precision measure to evaluate the performance of the proposed PEV algorithm. Precision is defined as the percentage of total correct verifications from all verifications. In addition, we also measure the false negative errors and false positive errors. To evaluate the

Table 8.5: Test Data

	Unequal Answer	Equal Answer	Total
Constant (CO)	5	10	15
Univariate Polynomial (UP)	5	10	15
Multivariate Polynomial (MP)	5	10	15
Rational Function (RF)	7	10	17
General Function (GF)	28	10	38
Total	50	50	100

performance, we conducted 3 sets of experiments corresponding to three different parameter settings. In the first set of experiments, we aim to evaluate the performance of our proposed approach using the default 15 decimal places for precision and small values of variables (called PEV1). In the second set of experiments, we aim to evaluate the effectiveness of the proposed approach using 100 decimal places for precision and small values of variables (called PEV2). In the third set of experiments, we aim to evaluate the effectiveness of the proposed approach using the default 15 decimal places and very large values of variables, e.g. $x = 10^{120}$, for random testing point selection (called PEV3). We also compare the performance of PEV3 with Maple and Mathemetica. In our experiments, we used the supporting functions provided by Maple and Mathematica to verify the equivalence of two expressions, $expr1$ and $expr2$. More specifically, the corresponding function call in Mathematica is: $expr1 === expr2$ and the corresponding function call in Maple is: $is(expr1 = expr2)$. For both Maple and Mathematica functions, if the two expressions are equivalent, the returned result is true, and false if otherwise.

Figure 8.5 shows the performance results of PEV1, PEV2 and PEV3 on mathematical answer verification based on precision. We observe that the proposed PEV1, PEV2 and PEV3 algorithms have the same performance results on four mathematical categories Constant, Univariate Polynomial, Multivariate Polynomial and Rational Function. However, on the General Function category, PEV3 and PEV2 outperform PEV1. This shows that increasing the decimal places for precision and using large values of random testing points help improve the performance of equivalence verification. In particular, we also observe that PEV3 outperforms PEV1 and PEV2 significantly. Specifically, PEV3 can achieve an average accuracy of 93.6% while PEV1 and PEV2 can only achieve about 86.8% and 89.4% respectively. Therefore, PEV3 is more effective than PEV1 and PEV2 in avoiding false positive errors.



	Constant	Univariate Polynomial	Multivariate Polynomial	Rational Function	General Function	Average
Maple	85	65	60	70	82	72.4
Mathematica	75	55	55	59	76	64
PEV1	93	93	93	94	61	86.8
PEV2	93	93	93	94	74	89.4
PEV3	93	93	93	94	95	93.6

Figure 8.5: Performance Results based on Precision

Figure 8.5 also shows the performance comparison of PEV3 with Maple and Mathematica on mathematical answer verification based on precision. It shows that the proposed PEV3 algorithm has consistently outperformed Maple and Mathematica in all mathematical categories. PEV3 can achieve an average accuracy of 93.6% while Maple and Mathematica can only achieve about 72.4% and 64% respectively. This is because PEV3 can avoid false negative errors and it can also reduce false positive errors quite effectively.

Table 8.6 gives the performance results of for Maple, Mathematica and PEV3 based on the false negative errors and false positive errors. For a given mathematical category, the percentage of false negative error can be calculated as the number of false negative cases divided by the total number of test cases with equal answer (shown in Table 8.5) for that category. The percentage of false positive errors is calculated similarly. As shown in Table 8.6, although both commercial CAS systems can avoid false positive errors, they incur false negative errors of 50% in Maple and 68% in Mathematica. In contrast, PEV3 can avoid false negative errors while only incurring false positive errors of 12%. As equivalent expressions are commonly occurred in mathematical answer assessment, this has shown the effectiveness of the proposed PEV algorithm in verifying the equivalence of two mathematical expressions numerically rather than symbolically. As Maple and Mathematica are commercial systems,

Table 8.6: Performance Results based on Error Types

(a) Errors in Maple

	False Negative Error		False Positive Error	
	#	%	#	%
Constant (CO)	2	20.0	0	0.0
Univariate Polynomial (UP)	5	50.0	0	0.0
Multivariate Polynomial (MP)	6	60.0	0	0.0
Rational Function (RF)	5	50.0	0	0.0
General Function (GF)	7	70.0	0	0.0
Total	25	50.0	0	0.0

(b) Errors in Mathematica

	False Negative Error		False Positive Error	
	#	%	#	%
Constant (CO)	4	40.0	0	0.0
Univariate Polynomial (UP)	7	70.0	0	0.0
Multivariate Polynomial (MP)	7	70.0	0	0.0
Rational Function (RF)	7	70.0	0	0.0
General Function (GF)	9	90.0	0	0.0
Total	34	68.0	0	0.0

(c) Errors in PEV3

	False Negative Error		False Positive Error	
	#	%	#	%
Constant (CO)	0	0.0	1	20.0
Univariate Polynomial (UP)	0	0.0	1	20.0
Multivariate Polynomial (MP)	0	0.0	1	20.0
Rational Function (RF)	0	0.0	1	14.0
General Function (GF)	0	0.0	2	7.1
Total	0	0.0	6	12.0

their technical details are not available publicly. To the best of our knowledge, Maple and Mathematica mainly use symbolic approaches for mathematical verification. This is also shown from their high false negative errors on verifying complex mathematical expressions. As such, it is difficult for Maple and Mathematica for verifying equivalent solutions, which can be expressed in different forms. This limitation is caused by the essence of the symbolic approaches for mathematical verification. Table 8.7 shows some error cases occurred in Maple, Mathematica and PEV3. As can be seen from Table 8.7(c), most error cases of PEV3 are caused by the computational limit of evaluating a function if its variables have very large values.

Table 8.7: Some Error Cases

(a) Error Cases in Maple

Test Case (#)	Answer	Solution	Category
55	$16\sin 10^\circ \sin 30^\circ \sin 50^\circ \sin 70^\circ$	1	CO
69	$x^8 + 3x^4 + 4$	$(x^4 + x^2 + 2)(x^4 - x^2 + 2)$	UP
71	$3x + 10xy - 5y - 6x^2$	$(3x - 5y)(1 - 2x)$	MP
85	$x \sin x + e^{\ln x}$	$x(\sin x + 1)$	GF
94	$\frac{9\pi x^2}{y-\sqrt{3}}$	$\frac{9\pi x^2(y+\sqrt{3})}{y^2-3}$	RF

(b) Error Cases in Mathematica

Test Case (#)	Answer	Solution	Category
55	$16\sin 10^\circ \sin 30^\circ \sin 50^\circ \sin 70^\circ$	1	CO
65	$x^5 + x^4 + 1$	$(x^2 + x + 1)(x^3 - x + 1)$	UP
74	$x^4y^2 - 3x^2y - 10$	$(x^2y + 2)(x^2y - 5)$	MP
86	$\frac{(x^2-1)^9}{x-1}$	$(x^2 - 1)^8(x + 1)$	RF
100	$\log_x y^2 + \log_{x^2} y^4$	$\log_x y$	GF

(c) Error Cases in PEV3

Test Case (#)	Answer	Solution	Category
1	$\frac{2^{1000}3^{500}}{\sqrt{5}}$	$\frac{2^{500}6^{500}\sqrt{5}}{5}$	CO
7	$x^{1000} + x^7 + 1$	$(x^{200} + x + 1)(x^6 - x^4 + x^3 + x + 1)$	UP
12	$x^{1000}y^2 - 3x^2y - 10$	$(x^{500}y - 2)(x^{500}y - 5)$	MP
21	$\frac{(x^{500}-1)^9}{x-1}$	$(x^{500} - 1)^8(x - 1)$	RF
28	$\sin(10^{-400}x)$	$10^{-400}x$	GF
29	$e^{10^{-1000}x}$	$1 + 10^{-1000}x$	GF

8.6 Summary

In this chapter, we have proposed a novel approach for automatic mathematical solution assessment in Web-based mathematics testing. The proposed PEV approach is a probabilistic algorithm, which verifies the equivalence of two functions by evaluating the value of the functions at random points sampled uniformly over a sufficiently large range. PEV can avoid false negative errors of the current popular Computer Algebra Systems such as Maple and Mathematica. The proposed PEV approach can achieve an average accuracy of 93.6% while Maple and Mathematica can only achieve about 72.4% and 64% respectively.

Chapter 9

Conclusion

In this chapter, we first summarize the work that has been done in this research. Then, we give the directions for further research work.

9.1 Summary

In this thesis, we have presented the motivation, objectives and related work of this research for Web-based testing. In this research, we mainly focus on investigating efficient multiobjective optimization techniques for online test paper generation (Online-TPG) and parallel test paper generation (k-TPG), and automatic question difficulty calibration and automatic mathematical solution assessment to support Web-based testing.

In conclusion, the contributions in this research are summarized as follows:

- An efficient constraint-based Divide-and-Conquer (DAC) approach has been proposed for Online-TPG. Three algorithms, namely DAC [176], DAC Tabu Search (DAC-TS) [175] and DAC Memetic Algorithm (DAC-MA) [178], have been implemented based on the proposed constraint-based DAC approach. Among the three proposed DAC algorithms, the DAC-MA algorithm has achieved the best overall performance. Specifically, the performance results have shown that the DAC-MA approach has achieved 12 times faster in runtime performance and 38% better test paper quality than the other current TPG techniques including GA, DE, TS, ACO and PSO.

- An Integer Programming approach, called BAC-TPG [179], has been proposed for Online-TPG. The proposed BAC-TPG approach is based on the Branch-and-Bound and Lifted Cover Cutting methods to find the near-optimal solution by exploiting the sparse

matrix property of 0-1 ILP. The performance results on various datasets and a user evaluation on generated test paper quality have shown that the BAC-TPG approach has achieved 29% better test paper quality than DAC-MA on the large-scale Online-TPG problems.

- A Multiobjective Evolutionary Co-Approximation (MECA) algorithm has been proposed for k-TPG. The proposed MECA approach is a greedy-based approximation algorithm, which explores the submodular property of the objective function for combinatorial multiobjective optimization. The MECA approach has achieved 8 times faster in runtime performance and 85% better test paper quality for parallel test paper generation than the other current techniques including k-TS, k-PSO and k-ACO.

- An effective Content-based Collaborative Filtering (CCF) approach [177] has been proposed for automatic calibration of question difficulty degree. In CCF, collaborative filtering is used to predict unknown user responses from known responses of questions. And the difficulty degree of each question in the dataset is then estimated using Item Response Theory. The proposed CCF approach has achieved promising results with 40% prediction errors less than the traditional Proportion Correct Method technique. Question calibration is an interesting and important area for web-based testing. Although only preliminary results are obtained, further extension of the current work is possible. For future work, recommendation techniques such as matrix factorization and incremental nearest neighbor collaborative filtering approach can be investigated for tackling the question difficulty calibration problem.

- An effective Probabilistic Equivalence Verification (PEV) algorithm [180] has been proposed for automatic mathematical solution assessment. The PEV approach is a probabilistic algorithm, which verifies the equivalence of two functions by evaluating the value of the functions at random points sampled uniformly over a sufficiently large range. PEV can avoid false negative errors of the current popular Computer Algebra Systems such as Maple and Mathematica. The PEV approach can achieve an average accuracy of 93.6% while Maple and Mathematica can only achieve about 72.4% and 64% respectively. Currently, there seems to be no ideal algorithm that can verify the equivalence of two general math functions perfectly. Both of the traditional symbolic algorithm and our numerical algorithm have its own merits and disadvantages. Although our proposed PEV approach may cause false positive errors, it can overcome

false negative errors by symbolic algorithms, which occur quite commonly in mathematical answer verification.

9.2 Future Work

In this research, effective techniques have been developed for supporting Web-based Testing. For future work, this research can be further extended in the following directions: Interactive Test Paper Generation, Constraint-based Computerized Adaptive Testing, Online Incremental Question Item Calibration and Automatic Solution Assessment.

9.2.1 Interactive Test Paper Generation

From a pedagogical perspective on generating high quality test papers, the current Web-based testing framework does not allow further changes on individual questions of a generated test paper after the test paper generation process. However, this could be enhanced by allowing users to edit, modify and update individual questions of a generated test paper. As such, the Web-based testing framework will be more flexible in creating high quality test papers.

For future work, we can investigate an interactive test paper generation approach for generating tailored test papers after the generation of online test papers. In this approach, a test paper is first generated automatically by one of our proposed TPG techniques. Then, the user can choose and replace some of the questions in the generated test papers following his questions of interest (QOI) without violating the content constraints, i.e., topic and question type constraints. However, this replacement could violate the assessment constraint. To overcome this potential problem, an effective algorithm could be investigated to dynamically replace the remaining questions, i.e., not QOI, to optimize assessment constraint satisfaction. For efficiency, an effective indexing structure of questions, called IR-Tree [59], could be used for enhancing runtime performance. This question replacement process can be repeated until a satisfiable test paper is obtained.

9.2.2 Constraint-based Computerized Adaptive Testing

Web-based adaptive testing is a promising approach for assessing learners' abilities. Most Web-based adaptive testing systems support Computerized Adaptive Testing (CAT) [225, 149, 53, 52, 102, 58, 218] which is based on Item Response Theory (IRT) [27, 219, 104] for evaluating learners' abilities. In CAT, test questions are generated dynamically according to

the learner's ability. It first selects a question of a suitable difficulty level according to the learner's ability. And the learner attempts the question. CAT then evaluates the response, and estimates the learner's new ability according to the response. The process repeats until one of the predefined termination conditions is met. As a result, the learner's ability or proficiency level will be determined. The CAT approach has the advantage that it can estimate a learner's ability without requiring the learner to answer all the questions in a test as typically needed in the traditional pen-and-pencil testing environment.

For future work, we can investigate a novel algorithm that is based on Item Response Theory, Online Prediction [49, 181] and Linear Programming-based Sequential Optimization (LPSO) [171, 216] for Computerized Adaptive Testing based on user specification on multiple assessment criteria. We may consider an adaptive test as a vector of questions which is the solution of an 0-1 Integer Linear Programming (0-1 ILP) problem. More specifically, the constraint-based CAT problem can be formulated as a 0-1 ILP problem and the corresponding solution of the 0-1 ILP problem can then be found. To construct an optimal adaptive test, the questions are selected adaptively and sequentially according to the learner's ability and multiple assessment criteria. The Item Response Theory framework controls the sequential question selection process. In Constraint-based CAT, it first selects a question of a suitable difficulty level according to the learner's ability. Then, the learner attempts the question. Next, the Constraint-based CAT evaluates the response. After that, the online prediction technique evaluates the learner's new ability according to the learner's response. The LPSO technique finds the updated solution for the 0-1 ILP problem. Based on the updated solution, LPSO searches for a question that best matches with the learner's ability as well as guarantees that the constraints will not be violated until the termination condition is reached.

9.2.3 Online Incremental Question Difficulty Calibration

In Chapter 7, we have proposed an effective approach for automatic question difficulty calibration. However, this is an offline approach because of the following two reasons. Firstly, the collaborative filtering and the dataset question calibration are offline processes. In addition, these processes are computational expensive and affect the scalability of question database calibration when dealing with a large pool of question items. Secondly, the new question's difficulty calibration requires that all new questions must be labeled with the knowledge skill components manually by human experts. To cope with the large pool of questions and users in the online Web-based testing environment, we need to investigate an online incremental

technique for automatic question difficulty calibration.

For future work, we can investigate an online incremental question difficulty calibration technique for the Mathematics domain based on the question content, question solution, and online user responses. In online calibration, we aim to calibrate incrementally the difficulty degree of each question. To achieve this, we first apply the question dataset calibration technique discussed in Chapter 7 to obtain the difficulty degree of existing questions that have user responses. However, we note that the question difficulty obtained may not be accurate due to the missing of many user response information. After that, for a new question without any user response, we can use the question contents and its solution to predict the initial value of the difficulty degree of the new question by using the content-based collaborative filtering proposed in the query question calibration of Chapter 7. Here, we can find similar questions of the new question based on extracted features of the question contents and its solution instead of using the knowledge skill components. Next, in the Web-based testing environment, there will be many user responses to questions to be gathered subsequently. Therefore, we can incrementally tune the difficulty degree of the questions when there are new user responses. To do that, we may investigate a new online incremental model, which could be based on the traditional 2-PL or 3-PL models and online machine learning techniques [49]. In the online incremental model, the difficulty degree of a new or existing question is considered as a latent variable in a latent factor model [15]. The variable value could be updated incrementally according to some adaptive formulas and new user responses.

9.2.4 Automatic Solution Assessment

In Chapter 8, we have proposed an effective approach for verifying mathematical answers in the form of mathematical expressions. However, there are other forms of mathematical solutions including answer steps, text description with math formula, mathematical proofs, and geometric and graph sketches. Verifying the answers from these forms is difficult and challenging.

For future work, we can investigate automatic solution assessment approaches for the different forms of mathematical answers. To verify the correctness of the solution steps such as equation solving, we need to verify the correctness of a sequence of inter-related steps. It is a difficult problem for such semantic verification requirement. To verify the correctness of a mathematical proof, we also need to verify the correctness of a logical sequence, which could be presented with text and math formula. In addition, the solution steps and mathematical

proofs could also be solved by applying the automated theorem proving technique [37]. As graph sketches are often drawn by hand, we need to recognize the graphical hand-drawn components such as lines, curves and the coordinates of the sketch answers, and compare these components with that of the sketch solution. This is a difficult problem for such visual semantic verification. To achieve this, we could apply the hand-drawn sketch recognition technique [184] to recognize the basic graphical components and their relationships in graph sketches. Then, a graph matching technique could be used to compare the two graphs for automatic answer verification.

Besides mathematical solutions, we can also investigate different solution forms in other subjects such as chemical formulas and expressions, chemical diagrams, physics formulas and expressions, and programming code. Basically, automatic programming code and physics expression verification is similar to mathematical answer verification. As such, we can apply the numerical testing approach, which checks the returned results of two programs or physics expressions based on different test cases for automatic verification. The other problems are much more challenging. For example, different from the numerical requirement in mathematical answer verification, chemical answer verification requires semantic verification for equivalence checking. In addition, hand-drawn chemical diagram assessment requires both the chemical hand-drawn recognition technique and graph matching technique for answer verification.

Appendix A

List of Publications

A.1 Journal Papers

1. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. Large-Scale Multiobjective Test Paper Generation for Web-based Testing with Integer Programming. *IEEE Transactions on Learning Technologies (TLT)*, Volume 6(1), pages 46-59, 2013.
2. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. Constraint-based Divide-and-Conquer Memetic Algorithm for Online Multiobjective Test Paper Generation. *Memetic Computing Journal, Springer*, Volume 4(1), pages 33-47, 2012.

A.2 Conference Papers

1. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. Probabilistic Equivalence Verification Approach for Automatic Mathematical Solution Assessment. *The 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2028-2034, 2013.
2. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. Content-based Collaborative Filtering for Question Difficulty Calibration. *The 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, LNCS Volume 7458, pages 359-371, 2012.
3. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. Web-based Mathematics Testing with Automatic Assessment. *The 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, LNCS Volume 7458, pages 347-358, 2012.
4. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. An Efficient Multiobjective Optimization Approach for Online Test Paper Generation. *IEEE Symposium*

on Computational Intelligence in Multicriteria Decision-Making (SSCI), pages 182-189, 2011.

5. Minh Luan Nguyen, Siu Cheung Hui, and Alvis C.M. Fong. A Divide-and-Conquer Tabu Search Approach for Online Test Paper Generation. *The 24th Australasian Joint Conference on Artificial Intelligence (AI)*, LNCS Volume 7106, pages 717-726, 2011.

Appendix B

Test Specifications

Table B.1 lists the 12 test specifications as discussed in Chapters 4, 5, and 6.

Table B.1: 12 Test Paper Specifications

	Total Time	Avg Difficulty Degree	Topic		Question Type	
			#Topic	Distribution	#Type	Distribution
$\mathcal{S}_1$	20	9	2	Uniform	3	Uniform
$\mathcal{S}_2$	40	8	4	Uniform	3	Normal
$\mathcal{S}_3$	60	7	6	Normal	3	Uniform
$\mathcal{S}_4$	80	3	8	Normal	3	Normal
$\mathcal{S}_5$	100	6	10	Uniform	3	Uniform
$\mathcal{S}_6$	120	8	12	Uniform	3	Normal
$\mathcal{S}_7$	140	4	16	Normal	3	Uniform
$\mathcal{S}_8$	160	7	20	Normal	3	Normal
$\mathcal{S}_9$	180	5	24	Uniform	3	Uniform
$\mathcal{S}_{10}$	200	4	28	Uniform	3	Normal
$\mathcal{S}_{11}$	220	6	32	Normal	3	Uniform
$\mathcal{S}_{12}$	240	5	40	Normal	3	Normal

Appendix C

Test Cases

Table C.1 and C.2 list the 100 pairs of user answers and standard solutions that have been used for performance evaluation on automatic mathematical solution assessment as discussed in Chapter 8.

Table C.1: Test Dataset for Unequal Answers

#	Test Cases		Category	Remark
	Answer	Solution		
1	$\frac{2^{1000}3^{500}}{\sqrt{5}}$	$\frac{2^{500}6^{500}\sqrt{5}}{5}$	CO	
2	$e^{\ln\ln 3}$	$\ln 2$	CO	
3	$\sin 6^\circ - \sin 42^\circ - \sin 66^\circ + \sin 78^\circ$	$1/2$	CO	
4	$\cot^2 36^\circ \cot^2 72^\circ$	$1/4$	CO	
5	$\frac{\log_2 24}{\log_{96} 2} - \frac{\log_2 192}{\log_{12} 2}$	4	CO	
6	$2x^{10} + 5x^5 + 3$	$(2x^5 + 3)(x^5 - 1)$	UP	
7	$x^{1000} + x^7 + 1$	$(x^{200} + x + 1)(x^6 - x^4 + x^3 + x + 1)$	UP	
8	$(x + 1)^2(1 - x)$	$(1 + x)(1 + x^2)$	UP	
9	$x^8 + 3x^4 + 4$	$(x^4 + x^2 + 2)(x^4 - x^2 + 1)$	UP	
10	$x^6 - x^4 - 2x^3 + 2x^2$	$x^2(x - 1)^2(x^2 + x + 2)$	UP	
11	$x^5 + y^5$	$(x - y)(x^4 - x^3y + x^2y^2 - xy^3 + y^4)$	MP	
12	$x^{1000}y^2 - 3x^2y - 10$	$(x^{500}y - 2)(x^{500}y - 5)$	MP	
13	$4x^4 + y^4$	$(2x^2 + y^2 - xy)(2x^2 + y^2 + 2xy)$	MP	
14	$(x + y + z)^3 - 4(x^3 + y^3 + z^3) - 12xyz$	$4(x + y - z)(y + z - x)(z + x - y)$	MP	
15	$a^3 + b^3 + c^3 - 3abc$	$(a + b - c)(a^2 + b^2 + c^2 - ab - bc - ca)$	MP	
16	$e^{2+\ln x}$	e^x	GF	
17	$\frac{\sin x + \cos x}{\sqrt{2}}$	$\sin(x + \frac{\pi}{3})$	GF	

Table C.1 Test Dataset for Unequal Answers (Cont.)

#	Test Cases		Category	Remark
	Answer	Solution		
18	$\sqrt{x^2-6x+9}$	$\lvert x-2\rvert$	GF	
19	$\frac{(x^2+2)(x+1)}{x^3}$	$\frac{x^3+x^2+2x+2}{x^2}$	RF	
20	$x\sin x+e^{\ln x}$	$x(\sin x+2)$	UF	
21	$\frac{(x^{500}-1)^9}{x-1}$	$(x^{500}-1)^8(x-1)$	RF	
22	$\frac{x^{10}+x^9+..+x}{x^9+..+x+1}$	$x+1$	RF	
23	$(\frac{\sqrt{x}}{\sqrt{x}-2}+\frac{\sqrt{x}}{\sqrt{x}+2})\frac{x-4}{\sqrt{4x}}$	$\sqrt{x}+2$	GF	
24	$\frac{x^2+2x-1}{2x^3+3x^2-2x}$	$\frac{1}{2x}+\frac{1}{5(2x+1)}+\frac{1}{10(x+2)}$	RF	
25	$\log_{0.5x}x^2-14\log_{16x}x^3+40\log_{4x}\sqrt{x}$	$\frac{2\log_2 x}{\log_2 x-1}-\frac{42\log_2 x}{\log_2 x+4}+\frac{20\log_2 x}{\log_2 x-2}$	GF	
26	$\sin(10^{-8}x)$	$10^{-8}x$	GF	
27	$\sin(10^{-100}x)$	$10^{-100}x$	GF	
28	$\sin(10^{-400}x)$	$10^{-400}x$	GF	
29	$e^{10^{-1000}x}$	$1+10^{-1000}x$	GF	
30	$\arctan(10^{-15}x)$	$10^{-15}x-\frac{(10^{-15}x)^3}{3}$	GF	
31	$\ln(1+10^{-10}x)$	$10^{-10}x-\frac{10^{-20}x^2}{3}$	GF	
32	$\arcsin(10^{-50}x)$	$10^{-50}x+\frac{10^{-150}x^3}{6}$	GF	
33	$\tan(10^{-100}x)$	$10^{-100}x+\frac{(10^{-100}x)^3}{3}$	GF	
34	$\cos(10^{-20}x)$	$1-\frac{10^{-40}x^2}{2}$	GF	
35	$\sec(10^{-30}x)$	$10^{-30}x+\frac{10^{-60}x^2}{2}$	GF	
36	$\frac{\sin(10^{-8}x)}{x}$	10^{-8}	GF	
37	$\frac{\sin(10^{-20}x)}{x^2+1}$	$\frac{10^{-20}x}{x^2+1}$	GF	
38	$\frac{\sin(10^{-100}x)}{x^3+x-1}$	$\frac{10^{-100}x}{x^3+x-1}$	GF	
39	$\frac{e^{10^{-5}x}}{2x+3}$	$\frac{1+10^{-5}x}{2x+3}$	GF	
40	$\frac{\arctan(10^{-15}x)}{10^{-15}x}$	$1-\frac{(10^{-15}x)^2}{3}$	GF	
41	$\frac{\ln(1+10^{-10}x)}{x^2}$	$\frac{10^{-10}}{x}-\frac{10^{-20}}{3}$	GF	
42	$\frac{\arcsin(10^{-50}x)}{x+1}$	$\frac{10^{-50}x}{x+1}+\frac{10^{-150}x^3}{6(x+1)}$	GF	
43	$\frac{\tan(10^{-100}x)}{e^{10^{-50}x}}$	$\frac{10^{-100}x}{1+10^{-50}x}+\frac{(10^{-100}x)^3}{3(1+10^{-50}x)}$	GF	
44	$\frac{\cos(10^{-20}x)}{\ln(1+10^{-10}x)}$	$\frac{1}{10^{-10}x}-\frac{10^{-30}x}{2}$	GF	
45	$\frac{\sec(10^{-30}x)}{\tan(10^{-10}x)}$	$10^{-20}+\frac{10^{-50}x}{2}$	GF	
46	$\frac{x^2y}{xy^2+3y}$	$\frac{x^2}{xy^2+3}$	RF	
47	$\frac{9\pi x^2}{y-\sqrt{3}}$	$\frac{9\pi x^2(y+\sqrt{3})}{y^2+3}$	RF	
48	$\frac{x^5+y^5}{x+y}$	$x^4-x^3y+x^2y^2+xy^3+y^4$	RF	
49	$(\frac{x+\sqrt{x}}{\sqrt{x}-y}+\frac{\sqrt{x}+1}{\sqrt{x}+y})\frac{\sqrt{x}-y}{\sqrt{x}+1}$	$\sqrt{x}-\frac{4}{\sqrt{x}+y}$	GF	
50	$\log_x y^2+\log_{x^2}y^4$	$\log_y x$	GF	

Table C.2: Test Dataset for Equal Answers

#	Test Cases		Category	Remark
	Answer	Solution		
51	$\frac{2}{\sqrt{5}}$	$\frac{2\sqrt{5}}{5}$	CO	
52	$2^{10}3^5$	$2^5 6^5$	CO	
53	$\frac{\log_2 25}{\log_2 5}$	2	CO	
54	$e^{\ln \ln 3}$	$\ln 3$	CO	
55	$16 \sin 10° \sin 30° \sin 50° \sin 70°$	1	CO	
56	$\frac{666}{37!}$	4.8610^{-41}	CO	
57	$\cot^2 36° \cot^2 72°$	1/5	CO	
58	$\log_3 5 \log_4 9 \log_5 2$	1	CO	
59	$\frac{\log_2 24}{\log_{96} 2} - \frac{\log_2 192}{\log_{12} 2}$	3	CO	
60	$\binom{7}{3} + \binom{7}{4}$	$\binom{8}{4}$	CO	
61	$x^3 - 15x^2 - 37x + 51$	$(x-1)(x+3)(x-17)$	UP	
62	$25x^4 - 16$	$(5x^2-4)(5x^2+4)$	UP	
63	$2x^{10} + 5x^5 + 3$	$(2x^5+3)(x^5+1)$	UP	
64	$12x^{10} + 42x^9 + 18x^8$	$6x^8(2x+1)(x+3)$	UP	
65	$x^5 + x^4 + 1$	$(x^2+x+1)(x^3-x+1)$	UP	
66	$x^8 + x^7 + 1$	$(x^2+x+1)(x^6-x^4+x^3-x+1)$	UP	
67	$x^7 + x^2 + 1$	$(x^2+1+x)(x^5+x^2-x^4-x+1)$	UP	
68	$(x+1)^2(1-x)$	$(1+x)(1-x^2)$	UP	
69	$x^8 + 3x^4 + 4$	$(x^4+x^2+2)(x^4-x^2+2)$	UP	
70	$x^6 - x^4 - 2x^3 + 2x^2$	$x^2(x-1)^2(x^2+2x+2)$	UP	
71	$3x + 10xy - 5y - 6x^2$	$(3x-5y)(1-2x)$	MP	
72	$x^5 + y^5$	$(x+y)(x^4-x^3y+x^2y^2-xy^3+y^4)$	MP	
73	$x^5 - y^5$	$(x-y)(x^4+x^3y+x^2y^2+xy^4+y^4)$	MP	
74	$x^4y^2 - 3x^2y - 10$	$(x^2y+2)(x^2y-5)$	MP	
75	$x^4 - y^4$	$(x+y)(x-y)(x^2+y^2)$	MP	
76	$\sum_{r=1}^{n} r(r+4)$	$1/6n(n+1)(2n+3)$	GF	$r \in \mathbb{N}$, $n \in \mathbb{N}$
77	$ab(a+b) - bc(b+c) + ca(c+a) + abc$	$(a+b+c)(ab-bc+ac)$	MP	
78	$4x^4 + y^4$	$(2x^2+y^2-2xy)(2x^2+y^2+2xy)$	MP	
79	$(x+y+z)^3 - 4(x^3+y^3+z^3) - 12xyz$	$3(x+y-z)(y+z-x)(z+x-y)$	MP	
80	$a^3 + b^3 + c^3 - 3abc$	$(a+b+c)(a^2+b^2+c^2-ab-bc-ca)$	MP	
81	$e^{2+\ln x}$	$e^2 x$	GF	

Table C.2 Test Dataset for Equal Answers (Cont.)

#	Test Cases		Category	Remark
	Answer	Solution		
82	$\frac{\sin x+\cos x}{\sqrt{2}}$	$\sin(x+\frac{\pi}{4})$	GF	
83	$\sqrt{x^2-6x+9}$	$\lvert x-3\rvert$	GF	
84	$\frac{(x^2+2)(x+1)}{x^3}$	$\frac{x^3+x^2+2x+2}{x^3}$	RF	
85	$x\sin x+e^{\ln x}$	$x(\sin x+1)$	GF	
86	$\frac{(x^2-1)^9}{x-1}$	$(x^2-1)^8(x+1)$	RF	
87	$\frac{x^{10}+x^9+..+x}{x^9+..+x+1}$	x	RF	
88	$(\frac{\sqrt{x}}{\sqrt{x}-2}+\frac{\sqrt{x}}{\sqrt{x}+2})\frac{x-4}{\sqrt{4x}}$	$\sqrt{x}$	GF	
89	$\frac{x^2+2x-1}{2x^3+3x^2-2x}$	$\frac{1}{2x}+\frac{1}{5(2x-1)}+\frac{1}{10(x+2)}$	RF	
90	$\log_{0.5x}x^2-14\log_{16x}x^3+40\log_{4x}\sqrt{x}$	$\frac{2\log_2 x}{\log_2 x-1}-\frac{42\log_2 x}{\log_2 x+4}+\frac{20\log_2 x}{\log_2 x+2}$	GF	
91	$x^{\sqrt{\log_x y}}$	$y^{\sqrt{\log_y x}}$	GF	
92	$\frac{x^2y}{xy^2+3y}$	$\frac{x^2}{xy+3}$	RF	
93	$\sum_{r=1}^{n}\frac{3r+5}{(r+1)(r+2)(r+3)}$	$7/6-\frac{3n+7}{(n+2)(n+3)}$	RF	$r\in\mathbb{N}$, $n\in\mathbb{N}$
94	$\frac{9\pi x^2}{y-\sqrt{3}}$	$\frac{9\pi x^2(y+\sqrt{3})}{y^2-3}$	RF	
95	$\sum_{r=1}^{n}\frac{1}{r(r+1)}$	$\frac{k}{k+1}$	RF	
96	$\frac{x^5+y^5}{x+y}$	$x^4-x^3y+x^2y^2-xy^3+y^4$	RF	
97	$\frac{x^5-y^5}{x-y}$	$x^4+x^3y+x^2y^2+xy^4+y^4$	RF	
98	$(\frac{x+\sqrt{x}}{\sqrt{x}-y}+\frac{\sqrt{x}+1}{\sqrt{x}+y})\frac{\sqrt{x}-y}{\sqrt{x}+1}$	$\sqrt{x}+1-\frac{4}{\sqrt{x}+y}$	GF	
99	$\ln(xy)-\ln y$	$\ln x$	GF	
100	$\log_x y^2+\log_{x^2}y^4$	$\log_x y$	GF	

Bibliography

[1] Advance personalized learning, http://www.engineeringchallenges.org/cms/8996/9127.aspx.

[2] Blackboard academic suite. http://elearning.mdis.edu.sg/webapps/login/.

[3] Kddcup 2010: Educational data mining challenge. https://pslcdatashop.web.cmu.edu/kddcup/.

[4] Khan academy. http://www.khanacademy.org/.

[5] Mit opencourseware. http://ocw.mit.edu/index.htm.

[6] The pslc datashop. https://pslcdatashop.web.cmu.edu/.

[7] Stanford engineering everywhere. http://see.stanford.edu/see/courses.aspx.

[8] Stanford university explore courses. http://explorecourses.stanford.edu/CourseSearch/.

[9] University of california, berkeley webcasts. http://webcast.berkeley.edu/courses.php.

[10] Youtube. http://www.youtube.com/.

[11] E. H. L. Aarts and J. K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[12] J. J. Adema, E. Boekkooi-Timminga, and W. J. Van Der Linden. Achievement test construction using 0-1 linear programming. *European Journal of Operational Research*, 55(1):103–111, 1991.

[13] J. J. Adema and W. J. Van der Linden. Algorithms for computerized test construction using classical item parameters. *Journal of Educational and Behavioral Statistics*, 14(3):279–290, 1989.

[14] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[15] D. Agarwal and B. C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28, 2009.

[16] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. *Annals of mathematics*, 160(2):781–793, 2004.

[17] A. V. Aho and J. E. Hopcroft. *Design and Analysis of Computer Algorithms*. Pearson Education.

[18] N. Alon and J. H. Spencer. *The probabilistic method*, volume 73. Wiley-Interscience, 2008.

[19] A. Amir, M. Farach, and Y. Matias. Efficient randomized dictionary matching algorithms. In *Combinatorial Pattern Matching*, pages 262–275. Springer, 1992.

[20] Eric Angel. A survey of approximation results for local search algorithms, 2006.

[21] A. Apostolico and Z. Galil. *Pattern matching algorithms*. Oxford University Press, USA, 1997.

[22] K. R. Apt. *Principles of constraint programming*. Cambridge Press, 2003.

[23] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 10–20, 2008.

[24] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989, 2010.

[25] K. E. Atkinson. *An introduction to numerical analysis*. Wiley-India, 2009.

[26] T. Back, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1: basic algorithms and operators*. IOP Publishing Ltd., UK, 1999.

[27] F. B. Baker and S. H. Kim. *Item response theory*. Marcel Dekker New York, 1992.

[28] K. R. Baker. *Introduction to sequencing and scheduling.* John Wiley & Sons, 1974.

[29] R. Baker, J. Walonoski, N. Heffernan, I. Roll, A. Corbett, and K. Koedinger. Why students engage in gaming the system behavior in interactive learning environments. *Journal of Interactive Learning Research*, 19(2):185–224, 2008.

[30] R. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 1(1):3–17, 2009.

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

[32] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331, 1990.

[33] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I, II.* Athena Scientific, 2007.

[34] D. Bertsimas, V. F. Farias, and N. Trichakis. The price of fairness. *Operations research*, 59(1):17–31, 2011.

[35] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific Belmont, MA, 1997.

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

[37] W. Bibel. *Automated theorem proving*, volume 21987. Vieweg, 1987.

[38] L. Blum. *Complexity and real computation.* Springer Verlag, 1998.

[39] M. Blum, A. K. Chandra, and M. N. Wegman. Equivalence of free boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters*, 10(2):80–82, 1980.

[40] L. Blumrosen and N. Nisan. *Algorithmic Game Theory*, chapter Combinatorial Auction, pages 267–300. Cambridge University press, Cambridge, UK, 2007.

[41] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Opertions Research*, 10(2):63–211, 1983.

[42] E. Boekkooi-Timminga. *Simultaneous test construction by zero-one programming.* Department of Education of the University of Twente, 1986.

[43] E. Boekkooi-Timminga. The construction of parallel tests from irt-based item banks. *Journal of educational and behavioral statistics*, 15(2):129–145, 1990.

[44] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence- From Natural to Artificial Systems.* Oxford University Press, 1999.

[45] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of ICDE*, 2001.

[46] P.A.N. Bosman and E.D. de Jong. Combining gradient techniques for numerical multi-objective evolutionary optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 627–634. ACM, 2006.

[47] B. Buchberger and R. Loos. Algebraic simplification. *Computer Algebra-Symbolic and Algebraic Computation*, pages 11–43, 1982.

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

[49] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006.

[50] S. Cetintas, L. Si, Y.P. Xin, D. Zhang, and J.Y. Park. Automatic text categorization of mathematical word problems. In *Proceedings of the 22nd International FLAIRS Conference*, pages 27–32, 2009.

[51] 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.

[52] C. Chou. Constructing a computer-assisted testing and evaluation system on the world wide web-the cates experience. *IEEE Transactions on Education*, 43(3):266–272, 2002.

[53] C. Chou and C. T. Sun. Constructing a cooperative distance learning system: The coral experience. *Educational Technology Research and Development*, 44(4):71–84, 1996.

[54] V. Chvatal. *Linear programming.* WH Freeman, 1983.

[55] M. Clerc. *Particle Swarm Optimization.* Hermes Science Lavoisier, 2005.

[56] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems.* Springer-Verlag New York, 2007.

[57] J.S. Cohen. *Computer algebra and symbolic computation: Mathematical methods.* Universities Press, 2003.

[58] R. Conejo, E. Guzman, E. Millan, M. Trella, J. L. Perez-De-La-Cruz, and A. Rios. Siette: a web-based tool for adaptive testing. *International Journal of Artificial Intelligence in Education*, Volume 14(1):29–61, 2004.

[59] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2(1):337–348, 2009.

[60] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition.* McGraw-Hill Science, 2001.

[61] I. Das and J. Dennis. Normal-boundary intersection: An alternate method for generating pareto optimal points in multicriteria optimization problems. *Institute for Computer Applications in Science and Engineering*, 1996.

[62] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[63] N. G. de Bruijn. Polya's theory of counting. *Applied Combinatorial Mathematics, pages 144-184*, 1964.

[64] D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162, 1985.

[65] K. Deb. *Multi-objective optimization using evolutionary algorithms.* Wiley, 2001.

[66] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[67] R. Dechter. *Constraint processing.* Morgan Kaufmann, 2003.

[68] A. L. Delcher, A. Phillippy, J. Carlton, and S. L. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research*, 30(11):2478–2483, 2002.

[69] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[70] S. Dobzinski and J. Vondrak. From query complexity to computational complexity. In *Proceedings of the 44th ACM symposium on Theory of Computing*, pages 1107–1116, 2012.

[71] M. Dorigo and C. Blum. Ant colony optimization theory: a survey. *Theory of Computer Science*, 344(3):243–278, 2005.

[72] M. Dorigo and T. Tsutzle. *Ant Colony Optimization*. Bradford Book, 2004.

[73] S. Hochbaum Dorit. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1997. 241938.

[74] P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpression problem. *Journal of the ACM (JACM)*, 27(4):758–771, 1980.

[75] F. Drasgow and J. O. Buchanan. *Innovations in computerized assessment*. Lawrence Erlbaum, 1999.

[76] J. Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial structures and their applications*, pages 69–87, 1970.

[77] T. Erfani and S. V. Utyuzhnikov. Directed search domain: A method for even generation of pareto frontier in multiobjective optimization. *Engineering Optimization*, 2010.

[78] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[79] U. Feige, V.S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2007.

[80] 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*, pages 667–676, 2006.

[81] P. Flajolet, P. Sipala, and J. M. Steyaert. Analytic variations on the common subexpression problem. *Automata, Languages and Programming*, pages 220–234, 1990.

[82] L. Fleischer. Recent progress in submodular function minimization. 2000.

[83] R.B. Fletcher. A review of linear programming and its application to the assessment tools for teaching and learning (as ttle) projects. Technical report, 2000.

[84] D. Floreano and C. Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[85] C. A. Floudas and P. M. Pardalos. *Encyclopedia of optimization*. Kluwer Academic Pub, 2001.

[86] A. Frank. Matroids and submodular functions. *Annotated Biblographies in Combinatorial Optimization*, pages 65–80, 1997.

[87] A. Freville. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research, Volume 155(1), pages 1-21*, 2004.

[88] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[89] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

[90] F. Glover and F. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[91] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[92] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems, Volume 4*, 1996.

[93] C. K. Goh, Y. S. Ong, and K. C. Tan. *Multi-Objective Memetic Algorithms*. 2009.

[94] D. Golovin. Max-min fair allocation of indivisible goods. 2005.

[95] T. F. Gonzalez. *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/Crc Computer & Information Science Series, 2007.

[96] Timothy Gowers, June Barrow-Green, and Imre Leader. *The Princeton companion to mathematics*. Princeton University Press, 2010.

[97] F. Grandoni, R. Ravi, and M. Singh. Iterative rounding for multi-objective optimization problems. *Algorithms-ESA 2009*, pages 95–106, 2009.

[98] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Complexity. *INFORMS Journal on Computing*, 11(1):117–123, 1999.

[99] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[100] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research (JMLR)*, 10:2935–2962, 2009.

[101] GUROBI. Gurobi optimizer (version 5.0). 2011.

[102] E. Guzman and R. Conejo. Self-assessment in a feasible, adaptive web-based testing system. *IEEE Transactions on Education*, 48(4):688–695, 2005.

[103] R. Guzman, E.and Conejo. Improving student performance using self-assessment tests. *IEEE Intelligent Systems*, 22(4):46–52, 2007.

[104] R. K. Hambleton, H. Swaminathan, and H. J. Rogers. *Fundamentals of item response theory*. Sage Publications, 1991.

[105] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.

[106] S. Har-Peled. Lecture notes on 473g: Introduction to algorithms, uiuc. http://valis.cs.uiuc.edu/ sariel/teach/05/b/, Fall Semester 2007.

[107] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press, USA, 1980.

[108] Y. He, S. C. Hui, and T. T. Quan. Automatic summary assessment for intelligent tutoring systems. *Computers and Education*, 53(3):890–899, 2009.

[109] M. A. Hearst. The debate on automated essay grading. *IEEE Intelligent Systems and their Applications*, 15(5):22–37, 2000.

[110] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[111] N. J. Higham. The accuracy of floating point summation. *SIAM Journal on Scientific Computing*, 14(1):783–783, 1993.

[112] 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*, 12(4):193–206, 2008.

[113] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[114] C. M. Hoffmann and M. J. O'Donnell. Pattern matching in trees. *Journal of the ACM (JACM)*, 29(1):68–95, 1982.

[115] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(443-473):443–473, 2000.

[116] 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*, 39(6):659–669, 2009.

[117] G. J. Hwang. A test-sheet-generating algorithm for multiple assessment requirements. *IEEE Transactions on Education*, 46(3):329–337, 2003.

[118] 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*, 51(3):1058–1072, 2008.

[119] G. J. Hwang, B. Lin, H. H. Tseng, and T. L. Lin. On the development of a computer-assisted testing system with genetic test sheet-generating approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(4):590–594, 2005.

[120] G. J. Hwang, P. Y. Yin, and S. H. Yeh. A tabu search approach to generating test sheets for multiple assessment criteria. *IEEE Transactions on Education*, 49(1):88–97, 2006.

[121] IBM. Ilog cplex optimizer (version 11.0). 2011.

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

[123] 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*, 7(2):204–223, 2003.

[124] S. Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45–64, 2008.

[125] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[126] K. Jain and M. Mahdian. Cost sharing. *Algorithmic game theory*, pages 385–410, 2007.

[127] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences (JCSS)*, 37(1):79–100, 1988.

[128] E. L. Johnson, G. L. Nemhauser, and M. W. P. Savelsbergh. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing*, 12(1):2–23, 2000.

[129] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[130] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[131] J. Kennedy and R. C. Eberhart. The particle swarm: social adaptation in information-processing systems. In *New ideas in optimization, pages 379-388*. McGraw-Hill Ltd., UK, 1999.

[132] L. G. Khachiian. Polynomial algorithms in linear programming. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 20(1):51–68, 1980.

[133] S. Khot and D. Moshkovitz. Np-hardness of approximately solving linear equations over reals. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC'11)*, pages 413–420, New York, NY, USA, 2011.

[134] S. Kim. A comparative study of irt fixed parameter calibration methods. *Journal of Educational Measurement*, 43(4):355–381, 2006.

[135] H. Klauck. On the complexity of approximation, local search, and local approximation. Master's thesis, Uni-GH Paderborn, 1995.

[136] H. Klauck. On the hardness of global and local approximation, 1996.

[137] J. Kleinberg and E. Tardos. *Algorithm design*. Pearson Education, 2003.

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

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

[140] D. E. Knuth. The art of computer programming, volume 3: sorting and searching, 1998.

[141] D. E. Knuth, J. H. Morris Jr, and V. R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[142] K. R. Koedinger, R. S. J. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. *A data repository for the EDM community: The PSLC DataShop*. CRC Press, 2010.

[143] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61(1):263–280, 1993.

[144] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer Magazine*, 42(8):30–37, 2009.

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

[146] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476, 1975.

[147] C. L. Lee, C. H. Huang, and C. J. Li. Test-sheet composition using immune algorithm for e-learning application. *New Trends in Applied Artificial Intelligence*, 4570(1):823–833, 2007.

[148] J. Y. T. Leung. *Handbook of scheduling: algorithms, models, and performance analysis.* Chapman & Hall, 2004.

[149] J. M. Linacre. Computer-adaptive testing: A methodology whose time has come. *Development of Computerised Middle School Achievement Tests, MESA Research Memorandum, Volume 69*, 2000.

[150] J. Lopez-Cuadrado, T. A. Perez, J. A. Vadillo, and J. Gutierrez. Calibration of an item bank for the assessment of basque language knowledge. *Computers and Education*, 55(3):1044–1055, 2010.

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

[152] R. M. Luecht. Computer-assisted test assembly using optimization heuristics. *Applied Psychological Measurement*, 22(3):224–236, 1998.

[153] R. M. Luecht. Some useful cost-benefit criteria for evaluating computer based test delivery models and systems. *University of North Carolina, Greensboro*, 2005.

[154] A. J. Macula. Probabilistic nonadaptive group testing in the presence of errors and dna library screening. *Annals of Combinatorics*, 3(1):61–69, 1999.

[155] Y. Manolopoulos, A. Nanopoulos, and Y. Theodoridis. *R-trees: Theory and Applications.* Springer Verlag, 2006.

[156] Mapple. Version 15, http://www.maplesoft.com/. 2011.

[157] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* 1990.

[158] Mathematica. http://www.wolfram.com/mathematica/. 2011.

[159] Y. Mei, K. Tang, and X. Yao. Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation, Volumr 15(2), pages 151-165*, 2011.

[160] A. Messac, A. Ismail-Yahaya, and C. A. Mattson. The normalized normal constraint method for generating the pareto frontier. *Structural and Multidisciplinary Optimization*, 25(2):86–98, 2003.

[161] A. Messac and C. A. Mattson. Normal constraint method with guarantee of even representation of complete pareto frontier. *Journal of AIAA*, 42(10):2101–2111, 2004.

[162] J. Mestre. Greedy in approximation algorithms. *AlgorithmsESA 2006*, pages 528–539, 2006.

[163] J. Mestre. Adaptive local ratio. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 152–160. Society for Industrial and Applied Mathematics, 2008.

[164] J. E. Mitchell. *Encyclopedia of Optimization, Volume II, pages 519-525*, chapter Integer programming: Branch-and-cut algorithms. Kluwer Press, 2001.

[165] M. Mohler, R. Bunescu, and R. Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 752–762, 2011.

[166] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report, Volume 826*, 1989.

[167] J. Moses. Algebraic simplification a guide for the perplexed. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 282–304, 1971.

[168] R. Motwani and P. Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28(1):37, 1996.

[169] D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded pareto front of practical multiobjective optimization problems. *SIAM Journal on Optimization*, 20(2):915–934, 2009.

[170] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers, 2005.

[171] R. B. Myerson. *Game theory: analysis of conflict*. Harvard University Press, 1997.

[172] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, 1988.

[173] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - i. *Mathematical Programming*, 14(1):265–294, 1978.

[174] H. Q. Ngo and D. Z. Du. A survey on combinatorial group testing algorithms with applications to dna library screening. *Discrete mathematical problems with medical applications*, 55:171–182, 2000.

[175] M. L. Nguyen, S. C. Hui, and A. C. M. Fong. A divide-and-conquer tabu search approach for online test paper generation. *AI 2011: Advances in Artificial Intelligence*, pages 717–726, 2011.

[176] M. L. Nguyen, S. C. Hui, and A. C. M. Fong. An efficient multi-objective optimization approach for online test paper generation. In *IEEE Symposium onComputational Intelligence in Multicriteria Decision-Making (MDCM)*, pages 182–189, 2011.

[177] M. L. Nguyen, S. C. Hui, and A. C. M. Fong. Content-based collaborative filtering for question difficulty calibration. *PRICAI 2012: Trends in Artificial Intelligence, pages 359-371*, 2012.

[178] 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*, 4(1):33–47, 2012.

[179] M. L. Nguyen, S. C. Hui, and A. C. M. Fong. Large-scale multiobjective static test generation for web-based testing with integer programming. *IEEE Transactions on Learning Technologies (TLT)*, 2012.

[180] M. L. Nguyen, S. C. Hui, and A. C. M. Fong. Web-based mathematics testing with automatic assessment. *PRICAI 2012: Trends in Artificial Intelligence, pages 347-358*, 2012.

[181] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[182] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer verlag, 1999.

[183] P. S. Oliveto, J. He, and X. Yao. Analysis of the (1+ 1)-ea for finding approximate solutions to vertex cover problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1006–1029, 2009.

[184] M. Oltmans. *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches*. PhD thesis, Massachusetts Institute of Technology, 2007.

[185] Y. S. Ong. Memetic algorithm package (map) toolkits. http://www3.ntu.edu.sg/home/asysong/MC/.

[186] Y. S. Ong and A. J. Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, 2004.

[187] Y. S. Ong, M. Lim, and X. Chen. Research frontier: Memetic computation - past, present & future. *IEEE Computational Intelligence Magazine*, 5(2):24–31, 2010.

[188] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[189] J. B. Orlin, A. P. Punnen, and A. S. Schulz. Approximate local search in combinatorial optimization. *SIAM Journal on Computing*, 33(5):1201–1214, 2004.

[190] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer Verlag, 2008.

[191] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[192] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

[193] M. O. Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, University, 1981.

[194] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can*. AK Peters, 1998.

[195] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135–146, 2007.

[196] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD, pages 71-79*, 1995.

[197] W. F. Rui, W. W. Hong, P. Q. Ke, Z. F. Chao, and J. J. Liang. A novel online test-sheet composition approach for web-based testing. In *Symposium on IT in Medicine and Education, pages 700 - 705*, 2009.

[198] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th ACM international conference on World Wide Web*, pages 285–295, 2001.

[199] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of multiobjective optimization.* Academic Press, 1985.

[200] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[201] A. Schaffer. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[202] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., 1986.

[203] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

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

[205] O. Schutze, A. Lara, and C. A. C. Coello. On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Transactions on Evolutionary Computation, Volume 15(4), pages 444-455*, 2011,.

[206] M. Shatnawi and A. Youssef. Equivalence detection using parse-tree normalization for math search. In *2nd International Conference on Digital Information Management, ICDIM*, volume 2, pages 643–648, 2007.

[207] S. S. Skiena. *The algorithm design manual.* Springer, 1998.

[208] J. Stamper, K. Koedinger, R. S. J. Baker, A. Skogsholm, B. Leber, J. Rankin, and S. Demi. Pslc datashop: a data analysis service for the learning science community. In *Intelligent Tutoring Systems*, pages 455–455. Springer, 2010.

[209] R. P. Stanley. *Enumerative combinatorics*, volume 49. Cambridge university press, 2011.

[210] R. E. Steuer. Multiple criteria optimization: theory, computation, and application. 1986.

[211] R. Storn and K. Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *International Computer Science Institution Publication*, 1995.

[212] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[213] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[214] E. Talbi. *Metaheuristics: From Design to Implementation.* Wiley Series on Parallel and Distributed Computing, 2009.

[215] T. Theunissen. Some applications of optimization algorithms in test design and adaptive testing. *Applied Psychological Measurement*, 10(4):381–389, 1986.

[216] P. R. Thie and G. E. Keough. *An introduction to linear programming and game theory.* Wiley-Interscience New York, NY, USA, 2008.

[217] 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*, 37(1):774–786, 2009.

[218] W. J. Van Der Linden and C. A. W. Glas. *Computerized adaptive testing: Theory and practice.* Springer Netherlands, 2000.

[219] W. J. Van der Linden and R. K. Hambleton. *Handbook of modern item response theory.* Springer Verlag, 1997.

[220] V. V. Vazirani. *Approximation algorithms.* springer, 2004.

[221] B. Vocking. Selfish load balancing. *Algorithmic Game Theory*, pages 517–542, 2007.

[222] J. Von Zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 2003.

[223] 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*, pages 67–74, 2008.

[224] H. Wainer. *Computerized adaptive testing.* Wiley Online Library, 1990.

[225] H. Wainer, N. J. Dorans, D. Eignor, R. Flaugher, B. F. Green, R. J. Mislevy, and L. Steinberg. Computerized adaptive testing: A primer. *Quality of Life Research*, 10(8):733–734, 2001.

[226] K. Wauters, P. Desmet, and W. Van Den Noortgate. Acquiring item difficulty estimates: a collaborative effort of data and judgment. In *Educational Data Mining*, 2011.

[227] K. Wauters, P. Desmet, and W. Van Den Noortgate. Item difficulty estimation: An auspicious collaboration between data and judgment. *Computers and Education*, 58(4):1183–1193, 2011.

[228] D. P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming*, 91(3):447–478, 2002.

[229] C. K. Yap. *Fundamental problems of algorithmic algebra*, volume 54. Oxford University Press Oxford, 2000.

[230] A. Youssef and M. Shatnawi. Math search with equivalence detection using parse-tree normalization. In *The 4th International Conference on Computer Science and Information Technology*, 2006.

[231] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[232] W. Zhou. Teachers' estimation of item difficulty: What contributes to their accuracy. In *Proceedings of the 31st annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, pages 261–264, 2009.

[233] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *International Conference on Parallel Problem Solving from Nature, pages 292-301, Springer Verlag*, 1998.