ExamGAN and Twin-ExamGAN for Exam Script Generation

Zhengyang Wu, Ke Deng, Judy Qiu, and Yong Tang

Abstract—Nowadays, the learning management system (LMS) has been widely used in different educational stages from primary to tertiary education for student administration, documentation, tracking, reporting, and delivery of educational courses, training programs, or learning and development programs. Towards effective learning outcome assessment, the exam script generation problem has attracted many attentions recently. But the research in this field is still in its early stage. Two essential issues have been ignored largely by existing solutions. First, given a course, it is unknown yet how to generate an exam script which concurrently has (i) the proper difficulty level, (ii) the coverage of essential knowledge points, (iii) the capability to distinguish academic performances between students, and (iv) the student scores in normal distribution. Second, while frequently encountered in practice, it is unknown so far how to generate a pair of high quality exam scripts which are equivalent in assessment (i.e., the student scores are comparable by taking either of them) but have significantly different sets of questions. To fill the gap, this paper proposes ExamGAN (Exam Script Generative Adversarial Network) to generate high quality exam scripts, and then extends ExamGAN to T-ExamGAN (Twin-ExamGAN) to generate a pair of high quality exam scripts. Based on extensive experiments on three benchmark datasets, it has verified the superiority of proposed solutions in various aspects against the state-of-the-art. Moreover, we have conducted a case study which demonstrated the effectiveness of proposed solution in the real teaching scenarios.

Index Terms—Deep knowledge tracing, educational data mining, exam script generation, generative adversarial network

1 INTRODUCTION

The learning management system (LMS) has been widely applied in different educational stages from primary to tertiary education. Exam is the essential part of learning outcome assessment. Manually generating high quality exam scripts by teachers is a confronting task physically and intelligently. It requires a thorough understanding of all knowledge points in the course and their relative importance; requires the understanding of the questions in the question bank including the knowledge points covered and the difficulty level; requires the understanding of the knowledge level of students. The task becomes particularly challenging in MOOCs (Massive Open Online Courses) where the number of courses, classes, and students are typically much larger than those in traditional schools and the size of question banks are considerably large.

This motivates the automatic generation of exam script in the past decade. In [1], questions are randomly selected from different question databases without bias where each database contains questions of a particular type. In [2], the questions have been extracted from a question bank against a predetermined difficulty level (e.g., the average of student scores is 70 if 100 is full). In [3], the ability of exam script to distinguish academic performances of students has been studied. In [4], author has developed a model based on a genetic algorithm so as to optimize objectives including proper difficulty level and clear distinction between students in academic performance. In [5], randomization algorithm has been adopted in exam script generation with various goals including proper distribution of question types and knowledge points.

The research in this topic is still in its early stage. Two essential problems have been largely neglected. First, given a course, it is unknown yet how to generate an exam script which concurrently has (i) the proper difficulty level, (ii) the coverage of essential knowledge points, (iii) the capability to distinguish academic performances of students, and (iv) the student scores in normal distribution. The last three aspects have been investigated in exam script generation. The last aspect is lack of study even though the ideal of a normal distribution of student scores has been naturalized in education [6]. Second, in practice, teachers often need to generate two exam scripts for a class of students, for example, one for formal exam and the other for deferred exam. However, it is unknown so far how to generate a pair of high quality exam scripts which are equivalent in assessment (i.e., the student scores are comparable by taking either of them) but have significantly different sets of questions.

11354 IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 35, NO. 11, NOVEMBER 2023
To fill the gap, we develop ExamGAN (Exam Script Generative Adversarial Network) to generate high quality exam scripts featured by validity (i.e., the proper knowledge coverage), difficulty (i.e., the expected average score of students), distinguishability (i.e., the ability of distinguishing academic performances between students), and rationality (i.e., the student scores are normally distributed). The task is technically challenging because (1) the four aspects must be harmonized to ensure the consistency of their optimization objectives, and (2) it is hard, if not impossible, to have sufficient data for ExamGAN training. To attack the first challenge, we parameterize the normal distribution of student scores with the mean consistent with desirable difficulty level and with the standard deviation consistent with the desirable distinguishability. To attack the second challenge, Deep Knowledge Tracing (DKT) [7] has been explored to estimate the probability of each individual student to correctly answer questions in an exam script and then infer the score distribution of all students in the class.

More interestingly, ExamGAN has been extended to Twin-ExamGAN (or simply T-ExamGAN) which has a novel structure collaborating two generators and one discriminator, each generator for one exam script generation. With T-ExamGAN, the two exam scripts generated have significantly different sets of questions but they are equivalent in assessment. Two learning strategies have been proposed for T-ExamGAN to prioritize different optimization objectives, i.e., the question set difference or the assessment equivalence.

Table 1 introduces the notations used in this study. In summary, the contributions of this study are threefold:

- This study has demonstrated the superiority of our solutions by conducting extensive experiments on benchmark datasets and a case study in real teaching scenarios with evaluation of human experts.

2 RELATED WORK

Learning Material Recommendation & Knowledge Tracing. A personalized learning resource recommendation architecture has been proposed where, beyond personal information, the learners’ preferences, behaviors [8], and emotions [9] have been explored. In [10], it measures the learner’s satisfaction during the learning process according to the learner’s knowledge mastery level and the interaction behavior, and adjusts the organizational learning materials accordingly. In [11], the question difficulty is predicted. In [12], [13], [14], authors have proposed to model the knowledge mastery level based on the questions answered so far which can be used in personalized test question recommendation. In [15], authors have organized the interactive information generated by learners in a timely manner to predict the learner’s response to the next task.

Knowledge tracing [16] is a model that monitor students’ dynamic knowledge states. It predicts the probability that a learner can answer a given exercise correctly at next time step. In [7], authors have proposed an improved knowledge tracing model based on the deep neural networks, called Deep Knowledge Tracing (DKT), using LSTM network to predict the probability to correctly answer next exercises.

Automatic Exam Script Generation. Given a question bank, the automatic exam script generation aims to select a subset of questions from a question bank to form an exam script. In [17], it first specifies a predetermined difficulty level and then extract different questions from a question bank based on an algorithm. According to [18], the desirable difficulty level should be near 0.7, that is, the average score of all students is 70 if 100 is full. In [11], it tries to improve the objectivity of the difficulty level of the questions in exam scripts such that the test results are close to the specified difficulty level. In [2], authors have introduced a model using Apriori algorithm to match the difficulty level of the selected questions according to the difficulty level set by the examiners.

In addition to difficulty level, the distinguishability (aka. discrimination) has been taken into account in exam script generation. It refers to the ability to distinguish academic performances between students. According to [3], the distinguishability is measured by $\frac{P_H - P_L}{P_H}$ where $P_H$ is the average of the 27% highest scores and $P_L$ is the average score of the 27% lowest scores. In [4], author has developed a model based on a genetic algorithm such that the problem of generating exam scripts is transformed into a multi-objective optimization problem. The optimization objectives include difficulty and distinguishability.

Moreover, the distributions of question types and/or knowledge points in the exam script have been considered. In [1], authors have proposed a system which utilizes a fuzzy logic algorithm. The purpose is to randomly select questions from different question databases without bias where each database contains questions of a particular type such as objective questions or subjective questions. In [5], authors have used a shuffling algorithm as a randomization technique to ensure the consistency of their decisions according to the difficulty level set by the examiners.

| Notation | Description |
|----------|-------------|
| $\Lambda$ | A course |
| $QuB$ | The exam question bank of course $\Lambda$ |
| $ExB$ | The set of exercises in course $\Lambda$ |
| $q \in QuB$ | A question in $QuB$ |
| $e \in ExB$ | A question in $ExB$ |
| $K$ | The knowledge points of course $\Lambda$ |
| $K_q$ | The knowledge points covered by question $q \in QuB$ |
| $K_e$ | The knowledge points covered by exercise $e \in ExB$ |
| $w(k)$ | The weight of knowledge point $k \in K$ in course $\Lambda$ |
| $\lambda \in \Lambda$ | A class of course $\Lambda$ |
| $E$ | An exam script for class $\lambda$ |
| $S$ | The students in class $\lambda$ |
| $R_s(E)$ | The score of student $s \in S$ on exam script $E$ |
| $< e, a_s >$ | $e \in ExB$ is an exercise answered by student $s \in S$ and $a_s$ is whether the answer is correct or not |
technique with the attempt to avoid repetitively selecting same questions for the same examinees. In [19], authors have used the randomization algorithm to select the attributes of questions (such as difficulty, type, etc.) or parameters (such as corresponding knowledge points, scores, etc.) to generate exam scripts. In [20], authors have proposed to automatically generates aptitude-based questions with certain keywords using randomization technique.

**Generative Model.** The generative model assumes data is created by a probability distribution, which is then estimated and a distribution very similar to the original one is generated. According to [21], the generative models include Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Latent Dirichlet Allocation (LDA), Boltzmann Machine (RBM), Variational autoencoder (VAE), and Generative Adversarial Network (GAN).

The GMM assumes all data samples are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The HMM is used to generate sequences named as Markov chains. The LDA is mainly used for topic modelling, although more broadly it is considered a dimensionality reduction technique [22]. The RBM represents a class of unsupervised neural networks which generate data to form a system that closely resembles the original system [23].

The above four models cannot be used for exam script generation while GAN and VAE can. The VAE comprises three layers. The input layer $X$, the middle layer $Z$ (aka., coding layer) and the output layer $\hat{X}$. The inputs are encoded into feature-extracted representations and stored in $Z$ through weights. Similarly, an output similar to $X$ at $\hat{X}$ is generated after decoding of vector $Z$. In order to generate data, the VAE assumes that $X$ is generated by a true prior latent distribution $Z$, which is assumed to be a Gaussian. In addition to $Z$, the conditional VAE considers additional information relevant to the generation [24]. While the VAE generates data similar to the original data to an extent, the GAN [25] can achieve the higher accuracy in generating data [21]. The GAN offers a solution of training generative models without the procedure of maximizing a log likelihood (as in VAE) which are usually intractable and require numerous approximations. This motivates us to generate exam scripts based on GAN in this study.

### 3 Preliminary

Given a course denoted as $\Lambda$, the knowledge points covered by the course is denoted as $K$. For each knowledge point $k \in K$, $w(k)$ denotes the importance of $k$ in the course. The exam question bank in the course is denoted as $QuB$ and the set of exercises is denoted as $ExB$. The knowledge points covered by a question $q \in QuB$ is $K_q$. Similarly, the knowledge points covered by an exercise $e \in ExB$ is $K_e$. The students enrolled in a class of the course $\lambda \in \Lambda$ are denoted as $S$. If a student $s \in S$ has answered an exercise $e \in ExB$, it is recorded as $<e,a_s>$ where $a_s$ is whether $s$’s answer is correct or not.

#### 3.1 Student Score Estimation and Distribution

Given a question $q \in QuB$, the full score is $m(q)$. Suppose $m(q)$ is allocated to the knowledge points covered by the question (i.e., $K_q$). For one knowledge point $k \in K_q$, the score is denoted as $m(q,k)$ and $m(q) = \sum_{k \in K_q} m(q,k)$. The score of a student $s$ on question $q$ can be estimated as:

$$m_s(q) = \sum_{k \in K_q} p_{s,k} m(q,k),$$

(1)

where $p_{s,k}$ is the mastery level of student $s$ on knowledge point $k$. The estimation of $p_{s,k}$ is based on Deep Knowledge Tracing (DKT) [7].

Knowledge tracing captures the knowledge mastery level of students according to the records of exercises answered so far by these students. In [7], Deep Knowledge Tracing (DKT) model applies LSTM network to achieve this goal. The process is illustrated in Fig. 1. The input $x_i$ is the encoded form of the knowledge point(s) covered by the exercise which have been answered by a student at time $t$, and the information whether the answer is correct or not; the output $y_i$ is a vector which comprises $|K|$ elements, each element corresponding to a unique knowledge point covered by the course. After training the model, the output of LSTM network can be used to predict the probability that a student $s$ would correctly answer a particular knowledge point based on the exercises done so far. Specifically, the value of $k$-th element in $y_t$ is $p_{s,k}$, i.e., the knowledge mastery level of student $s$ on knowledge point $k$.

Given an exam script $E$, the score estimation of student $s$ is the summation of scores that this student gets on all questions in $E$:

$$R_s(E) = \sum_{q \in E} m_s(q).$$

(2)

Given a class of students $S$ and an exam script $E$, the expected score for each student can be obtained using Equ. (2). The distribution of students’ expected scores is denoted as $P(S,E)$ which indicates the percentage of students at each score (e.g., 25% at 70, 5% at 90 if 100 is the full).
4.1 Knowledge Mastery Level Representation (KMLR)

As shown in Fig. 2, the student exercise records are the input of KMLR. Using Deep Knowledge Tracing [7], the knowledge mastery level of student \( s \) on all knowledge points \( K \) in this course is inferred, denoted as \( p_{s,k} \). For all students, a knowledge matrix (KMM) is obtained where each column corresponds to a student and each row corresponds to one of the \( K \) knowledge points. The output of KMLR is denoted as \( c \) which is derived from KMM. In \( c \), each knowledge point keeps only two values, i.e., average and standard deviation of knowledge mastery levels of all students on the knowledge point.

4.2 Generator and Discriminator

The generator does not generate exam script \( E \) directly. Instead, given the input condition \( c \) from KMLR, it generates the probability distribution \( V_E \) of all questions in the exam question bank to be selected in the exam script. The discriminator checks whether the generated \( V_E \) is desirable. The objective function of the generator and discriminator is:

\[
\max_G \min_D V(D, G) = E_{x\sim p_{data}(x)}[\log D(x|c)] + E_{z\sim p_{z}(z)}[\log (1 - D(G(z|c)))].
\]

where \( \log \) denotes the log probability.

An ExamGAN is trained for course \( \Lambda \). Given a collection of classes \( \lambda_1, \ldots, \lambda_i \) in \( \Lambda \) where each class has a set of student exercise records, the training data for class \( \lambda_i \) are the desirable exam scripts. Unfortunately, it is hard, if not impossible, to collect sufficient training data directly from teaching practices. In this situation, we need an alternative approach to effectively obtain training data.

4.3 Desirable Properties of Exam Scripts

Given a class of students, the training data are exam scripts which have the desirable validity, difficulty, distinguishability, and rationality.

4.3.1 Desirable Difficulty and Distinguishability

The desirable difficulty is that the average score of students is 70 if 100 is full [18]. According to [3], the distinguishability is measured by \( \frac{P_{HI} - P_{IL}}{P_{HI}} \) where \( P_{HI} \) is the average of the 27% highest scores and \( P_{IL} \) is the average of the 27% lowest scores. The desirable value of distinguishability is shown in Table 2, that is, \( \frac{P_{HI} - P_{IL}}{P_{HI}} > 0.39 \).

4.3.2 Desirable Validity

An exam script needs not cover all knowledge points. But the more important knowledge points should be more likely to be included in the exam scripts. Given a knowledge point \( k \in K \), if \( k \) is more important, it appears in more questions in the exam question bank of the course. That is, \( w(k) \) is the frequency of \( k \) in the exam question bank. The desirable knowledge point distribution in an exam script should be similar to the knowledge point distribution of the course.
The knowledge point distribution of the course is denoted as $C_j$, whose value for each knowledge point is its frequency in the exam question bank; the knowledge point distribution of an exam script is denoted as $E_j$, whose value for each knowledge point is its frequency in the exam script. The cosine similarity is used to measure the similarity between $E_j$ and $C_j$. The higher similarity is preferable.

4.3.3 Desirable Rationality

The ideal of a normal distribution of student scores has been naturalized in education [6]. In practice, students’ scores are commonly adjusted over a normal distribution known as grading on a curve or bell curving [32], [33]. We aim to generate the exam script working on which the distribution of student scores in a class is (or close to) normal without score adjustment.

Given an exam script of high quality, let $X$ be the variable of student score. Suppose the score range is $0-100$. The desirable distribution of $X$ is $P^*$, i.e., normally distributed with the mean $\mu$ corresponding to the desirable difficulty level and with the standard deviation $\sigma$ corresponding to the excellent distinguishability. It is straightforward $\mu = 70$. We next derive the proper value of $\sigma$ leading to the excellent distinguishability as in Table 2.

Let $\alpha$ be the 27%-quantile of $P^*$ and $\beta$ the 73%-quantile of $P^*$. According to the definition of Moment of normal distribution, the expectation of $X$ conditioned on the event that $X$ lies in an interval $[a, b]$ is given by

$$E[X|a \leq X \leq b] = \mu - \sigma^2 \frac{f(b) - f(a)}{F(b) - F(a)},$$

where $f$ and $F$ respectively are the density and the cumulative distribution function of $X$. We set $\bar{\sigma}$ represents the mean of the 27% lowest values, and $\bar{\beta}$ represents the mean of the 27% highest values. Then, $\bar{\sigma}$ can be expressed as:

$$\bar{\sigma} = \mu - \sigma^2 \frac{f(0) - f(\alpha)}{F(0) - F(\alpha)},$$

and $\bar{\beta}$ can be expressed as:

$$\bar{\beta} = \mu - \sigma^2 \frac{f(100) - f(\beta)}{F(100) - F(\beta)},$$

where $f(0) = 0$, $F(0) = 0$ and $f(100) = 0$, $F(100) = 1$. Furthermore, we set the distance between $\bar{\sigma}$ and $\bar{\beta}$ is $Dis$. As shown in Fig. 3, we assume there is a point $\bar{\beta} - \epsilon$ on the x-axis which makes the areas of two cumulative distribution $F(\alpha)$ and $1 - F(\beta - \epsilon)$ equal. $Dis_\beta$ is the distance between $\bar{\sigma}$ and $\bar{\beta} - \epsilon$. Obviously, we have

$$Dis \geq Dis_\beta,$$

$$\geq \sigma^2 \left( \frac{f(\alpha) + f(\beta - \epsilon)}{F(\alpha)} \right),$$

where $\alpha$ is the 27%-quantile of $P^*$, then we can get $\alpha$ by checking the standard normal distribution table and do the normal distribution transformation, i.e.,

$$\alpha = (0.1103063 \sigma) + \mu,$$

where we set $\mu = 70$, and let

$$\sigma^2 \left( \frac{f(\alpha)}{F(\alpha)} \right) = 39.$$

By substituting $\alpha$ into Equ. (10), we obtain the value of $\sigma$ which makes the distance between the average of the 27% highest score and the average of the 27% lowest scores greater than 39. That is, $\sigma = 15$ leads to excellent distinguishability.

Now, we have the mean and standard deviation of the desirable normal distribution $P^*$. Given an exam script $E$ for a class of students $S$, the student score distribution, denoted as $P(S, E)$, should be similar to $P^*$. We use the Kullback-Leibler (KL) divergence [34] as the similarity measure, that is

$$Dissim(P(S, E), P^*) = 1 - KL(P(S, E), P^*).$$

The higher similarity is preferable.

4.4 Creating Training Data

Given a class of a course and the exercise records of students in the class, we randomly generate a large number of exam scripts from the exam question bank. Among them, those with desirable properties (discussed in Section 4.3) are selected as the training data. This process is time consuming but acceptable to create training data offline. Once the generative model is trained with the training data, exam scripts can be online generated for any classes of the course.

Given a class of students, the training data is created by taking five steps: (1) Based on the exercise records, the knowledge
mastery level (i.e., c) of students in the class is captured as the condition of exam script generation (detailed in Section 4.1). (ii) Among all knowledge points, we randomly pick up n knowledge points based on their frequencies in the exam question bank, i.e., the more frequent one is more likely to be selected. (iii) For each of the n knowledge points, it may be covered by a number of questions from which one and only one is selected in random and inserted into the exam script. This is performed repeatedly until the exam script contains n questions. By this way, m exam scripts are created (m = 1000 in this study). (iv) For each of the newly created m exam scripts, denoted as $E_i$, the student score is estimated using knowledge tracing where it requires the records of exercises done by students as explained in Section 3.1. The distribution of estimated student scores $R^E_{ij}$ is obtained; and the similarity between $R^E_{ij}$ and $Z$ is calculated following Equ. (11). (v) Among all m exam scripts, the 1% with the highest similarity to $Z$ are adopted. For each adopted exam script, an instance of training data $< c, V_E >$ is created where $V_E$ is the vector, each element corresponding to a question in the exam question bank. The elements in $V_E$ corresponding to the n questions in the exam script are 1; and the value of other elements is 0.

To have more training data, we use all students who have exercise record in a course and form them into classes randomly. For each class, the above (i)-(v) steps are repeated.

Algorithm 1. Training ExamGAN

Input: Training data (a set of $< c, V_E >$)  
Output: $\theta_G$ (the parameters of $G$)  
Initialize $\theta_G$ and $\theta_D$ (the parameters of $D$)  
while $\theta_D$ not converged do  
while $\theta_D$ not converged do  
Extract a batch of training data, i.e., a set of $< c_i, V_{E_i} >$;  
Using $G$ to generate a batch of data, i.e., a set of $< c_j, V_{E_j} >$;  
$\theta_D \leftarrow \theta_D + \eta \nabla_D (\theta_D); \quad$  
$\theta_G \leftarrow \theta_G + \eta \nabla_G (\theta_G); \quad$  
4.5 Training and Using ExamGAN Model

The training algorithm of ExamGAN is presented in Algorithm 1. $\theta_G$ and $\theta_D$ are the parameters of the generator and discriminator in ExamGAN respectively. $< c_i, V_{E_i} >$ is an instance in the training data, i.e., a sample from $p_{data}(x)$ as shown in Equ. (3). For $< c_j, V_{E_j} >$, $V_{E_j}$ is the output of generator on condition $c_j$. The discriminator model is trained to recognize the difference between $< c_i, V_{E_i} >$; and $< c_j, V_{E_j} >$; and the generator model is trained to generate $< c_j, V_{E_j} >$ such that the discriminator model cannot recognize their difference.

Once ExamGAN has been trained, given any new class $\lambda_{new}$ in the course, the knowledge mastery level of students is identified following Section 4.1 and fed to the generator model. The output is the probability distribution of questions in the exam question bank to appear in the exam script. In the exam question bank, $n$ questions with the highest probabilities will be selected to form the new exam script for $\lambda_{new}$.

5 T-ExamGAN Model

We extend ExamGAN to T-ExamGAN which generates a pair of equivalent exam scripts for the same class of students.

The two exam scripts should have significantly different sets of questions while both of them retain the high quality.

For a pair of exam scripts, they may have same questions but the percentage must be restrained. The Jaccard coefficient is applied to measure to which level two exam scripts have the same questions:

$$f(E_A, E_B) = \frac{|E_A \cap E_B|}{|E_A \cup E_B|},$$

where $E_A$ and $E_B$ represent a pair of exam scripts generated.

5.1 Framework of T-ExamGAN

Difference from ExamGAN, T-ExamGAN comprises four components, i.e., knowledge mastery level representation (KMLR), generator-$A$, generator-$B$, and discriminator, as shown in Fig. 4.

KMLR is exactly same as that in ExamGAN. As introduced in Section 4.1, the output of KMLR is the condition $c$ of generators. The generator-$A$ (denoted as $G_A$) outputs $V_{E_A}$ and the generator-$B$ (denoted as $G_B$) outputs $V_{E_B}$. $V_{E_A}$ and $V_{E_B}$ are the probability distribution of all questions in the exam question bank to appear in the exam script $E_A$ ($E_B$). To ensure $V_{E_A}$ and $V_{E_B}$ have significantly different sets of questions, $G_A$ and $G_B$ are trained alternatively. The loss function is cross entropy between $V_{E_A}$ and $V_{E_B}$

$$H(V_{E_A}, V_{E_B}) = - \sum_{x \in Q} V_{E_A}(x) \log V_{E_B}(x),$$

where $V_{E_A}(x)$ is the probability of the question $x$ in $V_{E_A}$. If the loss function $H(V_{E_A}, V_{E_B})$ is greater, $V_{E_A}$ and $V_{E_B}$ are more similar and thus one of the generators needs to be adjusted so as to reduce the loss function value (i.e., making $V_{E_A}$ and $V_{E_B}$ less similar).

At the same time, to ensure the high quality of the generated exam scripts by $G_A$ and $G_B$, the discriminator model checks $V_{E_A}$ and $V_{E_B}$ and the generator $G_A$ and $G_B$ are optimized accordingly as in ExamGAN. Note the training data are same as those in ExamGAN.

5.2 Training T-ExamGAN

T-ExamGAN is defined as the following minmax game

$$\min_{G_A, G_B} \max_D V(G_A, G_B, D) = E_{c \sim p_{data}(c)}[\log(D(E(c))]$$

$$+ E_{c \sim p_{data}(c)}[1 - \log(D(G_A(z(c)))$$

$$+ E_{c \sim p_{data}(c)}[1 - \log(D(G_B(z(c)))$$

$$+ |\psi - H(V_{GA}, V_{GB})|. \quad (14)$$
The objective function of discriminator can be expressed as:
\[
V_D = \frac{1}{r} \sum_{i=1}^{r} \log(D(E_i|c)) + \frac{1}{r} \sum_{i=1}^{r} (1 - \log(D(G_A(z_i|c)))) + \frac{1}{r} \sum_{i=1}^{r} (1 - \log(D(G_B(z_i|c)))).
\] (15)

We maximize the objective function by updating discriminator parameters \(\theta_D\):
\[
\theta_D \leftarrow \theta_D + \eta \nabla V_D(\theta_D),
\] (16)
where \(\eta\) is the step size and so in the following equations.

For training generator \(G_A\) and \(G_B\), the objective functions are below respectively:
\[
V_{G_A} = \frac{1}{r} \sum_{i=1}^{r} \log(D(G_A(z_i|c))),
\] (17)
\[
V_{G_B} = \frac{1}{r} \sum_{i=1}^{r} \log(D(G_B(z_i|c))),
\] (18)
where \(r\) is the number of the training sample. We maximize the objective function by updating parameters \(\theta_{G_A}\) and \(\theta_{G_B}\):
\[
\theta_{G_A} \leftarrow \theta_{G_A} + \eta \nabla V_{G_A}(\theta_{G_A}).
\] (19)
\[
\theta_{G_B} \leftarrow \theta_{G_B} + \eta \nabla V_{G_B}(\theta_{G_B}).
\] (20)

To make the exam scripts generated by generator \(G_A\) and generator \(G_B\) with different sets of questions, we aim to minimize
\[
L(c) = |\psi - H(E_A, E_B)|,
\] (21)
where \(\psi\) is a hyper-parameter indicating the required difference between the \(E_A\) and \(E_B\). The objective function \(L(c)\) is optimized by updating generator parameters \(\theta_{G_B}\) and \(\theta_{G_A}\):
\[
\theta_{G_A} \leftarrow \theta_{G_A} + \eta \nabla L(c)(\theta_{G_A}).
\] (22)
\[
\theta_{G_B} \leftarrow \theta_{G_B} + \eta \nabla L(c)(\theta_{G_B}).
\] (23)

Note it is less intuitive to specify the hyper-parameter \(\psi\). So, we take an alternative approach. \(E_A\) (\(E_B\)) is formed by the \(n\) questions with the highest probabilities in \(V_{E_A}\) (\(V_{E_B}\)). According to Equ. (12), if \(f(E_A, E_B)\) is no more than a specified ratio, say 30%, the stop criteria of optimization is met.

5.3 Two Training Strategies

Two training strategies can be applied, i.e., individual quality priority and twin difference priority.

5.3.1 Individual Quality Priority

The optimization priority is on the quality of individual exam script. Algorithm 2 illustrates the training process. (1) The generators and discriminator are trained simultaneously. The goal is that the discriminator should be able to identify true and false samples, and each generator generates fake samples to deceive the discriminator. Since the generators are trained separately, for each of them it is same as training ExamGAN. This step is repeated until two generators have been well trained. (2) The difference between the generated exam scripts from two generators are checked. The parameters in one of the two generators will be updated once. The above two steps are repeated by \(\gamma\) times.

Algorithm 2. T-ExamGAN Training - Individual Quality Priority

**Input:** Training data \(< c, V_E > ; \gamma\)
**Output:** \(G_A, G_B\)

Initialize \(\theta_{G_A}, \theta_{G_B}, \theta_D, r = 0;\)
while \(r < \gamma\) do
while \(\theta_{G_A}, \theta_{G_B}\) not converged do
while \(\theta_D\) not converged do
// Given training data \(< c, V_E > s\)
// using \(G_A\) generate \(< c_i, V_{E_i} > s\)
// using \(G_B\) generate \(< c_i, V_{E_i} > s\)
\(\theta_D \leftarrow \theta_D + \eta \nabla V_D(\theta_D);\)
\(\theta_{G_A} \leftarrow \theta_{G_A} + \eta \nabla V_{G_A}(\theta_{G_A});\)
\(\theta_{G_B} \leftarrow \theta_{G_B} + \eta \nabla V_{G_B}(\theta_{G_B});\)
if \(r\) is even then
\(\theta_{G_A} \leftarrow \theta_{G_A} + \eta \nabla L(c)(\theta_{G_A});\)
else
\(\theta_{G_B} \leftarrow \theta_{G_B} + \eta \nabla L(c)(\theta_{G_B});\)
end if
end while
end while
end while
end while

5.3.2 Twin Difference Priority

It takes the difference between the generated exam scripts as the primary goal. Algorithm 3 illustrates the training process. (1) The two generators are trained if the difference between the exam scripts does not meet the requirement, i.e., the parameters in two generators are alternatively updated to increase the difference. The training stops until the difference requirement is met. (2) The generators and discriminator are trained simultaneously. The goal is that the discriminator should be able to identify true and false samples, and each generator generates fake samples to deceive the discriminator. Each generator is trained independently in the same way as training the generator in ExamGAN. Note that the parameters of discriminator and generators are updated once only. The above two steps are repeated by \(\gamma\) times.

6 EXPERIMENTS

The experiments evaluate the performance of proposed ExamGAN and T-ExamGAN. All tests were run on a desktop computer and a laptop. The desktop computer with Intel(R) Core i9 Duo 2.4GHz CPU, and 16GB of main memory. The laptop with Intel(R) Core i7 Duo 2.4GHz CPU, and 16GB of main memory. The code used for programming is python 3.7, and sklearn 0.20.3. The development environment for deep learning is Tensorflow 1.10.0 and Keras 2.2.4.

All tests explore three datasets. Two datasets are from ASSISTments [35] which is an electronic tutor teaching and evaluating students in grade-school math. One of the two datasets was gathered in the school year 2009-2010 (denoted as assistments9010) and the other was gathered in 2012-2013 (denoted as assistments1213). The third dataset named OLI Engineering Statics is from a college-level engineering statics...
course (denoted as *olies2011*); the dataset is available online (pslcdatashop.web.cmu.edu). The detailed information of the three datasets are shown in Table 3. Note exercise questions in Table 3 are unnecessary to be the exam questions. To test different situations, the exam question bank has 10,000 questions for *assistsments0910* and *assistsments1213* which are randomly selected from exercise questions; for *olies2011*, the exam question bank has 10,000 questions which include the exercise questions and synthetic questions generated following the distribution of real questions on knowledge points.

Table:<br>Overview of Datasets (KPs: Knowledge Points, EQs: Exercise Questions)<br><br| Dataset | KPs | Students | EQs | Records |
|---------|-----|----------|-----|---------|
| assistments0910 | 123 | 4,163 | 26,688 | 278,607 |
| assistments1213 | 265 | 28,834 | 53,091 | 2,506,769 |
| olies2011 | 85 | 335 | 1,223 | 45,002 |

6.1 Effectiveness of ExamGAN<br>Given a class of students, the high quality exam scripts should have desirable (i) *validity*, (ii) *difficulty*, (iii) *distinguishability*, and (iv) *rationality*. As discussed in Section 4.3, the desirable *validity* is that the knowledge point distribution in exam script is similar to that in exam question bank; the desirable *difficulty* is that the students’ average score is close to 70 (the full score 100); the desirable *distinguishability* is that the difference between average of the 27% highest score and the average of the 27% lowest scores is greater than 0.39 (as shown in Table 2); the desirable *rationality* is that the student score distribution is similar to a user specified distribution (i.e., normal distribution in range 0-100 with mean 70 and standard deviation 15).

6.1.1 Baseline Methods<br>We compare ExamGAN against three baseline methods which are the state-of-the-art in automatic exam script generation.

- **Random Sampling and Filtering (RSF)** randomly samples a subset of questions from the exam question bank to form an exam script. Among 100 exam scripts, 10 with the highest coverage of knowledge points are selected; then, among the 10, RSF picks up the one with difficulty level closest to the desired level. This method is a variant of the method proposed in [19].
- **Genetic Algorithm (GA)** regards an exam script as an individual and each question is regarded as a chromosome. The expected difficulty level and the coverage of knowledge points are taken as the goal of evolution [4]. In GA, the unique parameters of genetic operation include the *crossover rate* (0.8 by default), the *mutation rate* (0.003 by default), and the size of population (1000 by default).
- **Conditional Variational Autoencoders (ExamVAE)** is based on VAEs (briefed in Section 2) conditioned on $c$ from KMLR as ExamGAN. Also, both ExamGAN and ExamVAE assume the exam scripts are generated by the same prior latent distribution $z$, i.e., normal distribution $N(0,1)$ [24].

While RSF and GA do not require training datasets, ExamVAE uses the training dataset as ExamGAN and T-ExamGAN. In ExamGAN, the generator is a two-layer network where the first layer uses the *sigmoid* activation function and the second layer using the *tanh* activation function; the discriminator is also a two-layer network where both layers use the *sigmoid* activation function. Model optimization is based on *gradient descent optimizer* [36] with a learning rate 0.001. For hyper-parameter optimization, we set drop-out rate to 0.3 for all layers. ExamGAN is trained by 200 epochs.

In ExamVAE, the encoder consists of four fully connected layers, one input layer and three hidden layers. The decoder consists of a custom layer and two fully connected layers where the custom layer receives input noise and re-parameter changes [24] and the other two layers decode the output. The activation function of encoder is *ReLU* and the activation function of decoder is *ReLU* and *sigmoid*. The model is trained by using *Adam Optimizer* [37] with a learning rate 0.01. ExamVAE is trained by 200 epochs.

6.1.2 Experiment Results<br>Figs. 5, 6, and 7 report the quality of exam scripts generated using ExamGAN and baselines (each generates 100) for the three datasets. Compared with GA and RSF in terms of the distribution mean and variance of the 100 exam scripts, the exam scripts generated using ExamGAN and ExamVAE tend to have the better *difficulty* (closer to 0.7), the similar *validity* (closer to 1), the better *rationality* (closer to 1), the
better distinguishability \( (> 0.39) \). Compared with ExamVAE, ExamGAN is slightly better in terms of distribution mean and has more advantage in terms of distribution variance. That is, the quality of exam scripts generated using ExamGAN is more reliable.

Interestingly, it is hard for dataset assistments1213 to generate exam scripts with difficulty level greater than 0.5. It implies that the questions in the exam question bank are too easy considering the knowledge mastery level of the students in the class based on records of exercises answered. This reveals the side benefit of this study is to verify whether an update of the exam question bank is necessary to maintain the quality assessment.

Figs. 8, 9, 10, and 11 further illustrate the quality of generated exam scripts for the three datasets. For each method, the box plot of the generated 100 exam scripts are illustrated in terms of difficulty in Fig. 8, validity in Fig. 9, rationality in Fig. 10, and distinguishability in Fig. 11. In Fig. 8, it clearly shows the exam scripts generated using ExamGAN are always closer to the desirable difficulty. In Figs. 10 and 11,
the exam scripts generated using ExamGAN are obviously better than those using baselines in terms of distinguishability and rationality. Fig. 9 illustrates that all methods have the considerably high validity. It is interesting to note that RSF has the even higher validity compared with other methods. The reason is that covering the required knowledge points is the main optimization objective of RSF.

6.1.3 Training Scripts versus Generated Scripts

Recall 100 classes are formed randomly for each dataset. Each class with 50 students has 10 training exam scripts generated following Section 4.4. To train ExamGAN model, the 100 classes are split into training set (80 classes), the validation set (10 classes), and the test set (10 classes). Fig. 13 illustrates the properties of 15 randomly selected training exam scripts for 5 classes (3 for each) from the test set, and the properties of 2 exam scripts generated by ExamGAN for the same 5 classes. Clearly, the exam scripts generated by ExamGAN have the highly similar properties as the training exam scripts.

While having the similar properties, it is interesting whether the exam scripts generated by ExamGAN and the training exam scripts for the same classes have the similar sets of questions or not. Fig. 12 shows the 10,000 questions in the exam question bank for each dataset with a $100 \times 100$ matrix where the highlighted points indicate the questions in an exam script. For a random class in the test set, two training exam scripts and two exam scripts generated by ExamGAN are illustrated. As show in Fig. 12, the exam scripts generated by ExamGAN have very different sets of questions from the training exam scripts.

6.2 Effectiveness of T-ExamGAN

Using T-ExamGAN, a pair of exam scripts are generated. In the experiments, we compare (i) the two training strategies of T-ExamGAN, and (ii) T-ExamGAN against a baseline based on ExamGAN where two exam scripts are generated independently using the ExamGAN twice to form a pair. Here, the ExamGAN-based baseline is implemented in two settings: the first is denoted as ExamGAN@500 where ExamGAN is trained by 500 epochs; the second is denoted as ExamGAN@1500 where ExamGAN is trained by 1500 epochs.

6.2.1 Training T-ExamGAN

As introduced in Section 5, T-ExamGAN has two generators. The two generators are initialized by training each individually as training ExamGAN. Both are trained by 1500 epochs. The discriminator in T-ExamGAN is same as the discriminator in ExamGAN. The training datasets for
T-ExamGAN are same as those for training ExamGAN. After initializing the two ExamGAN models, T-ExamGAN is trained. Using individual quality priority training strategy, the T-ExamGAN model is denoted as T-ExamGAN@S1; using twin difference priority training strategy 2, the T-ExamGAN model is denoted by T-ExamGAN@S2.

### 6.2.2 Experiment Results

For T-ExamGAN@S1, T-ExamGAN@S2, ExamGAN@500 and ExamGAN@1500, each generates 100 pairs of exam scripts. It is desirable that (i) both two exam scripts have the similar quality in terms of difficulty, distinguishability, validity and rationality; (ii) the two exam scripts have significantly different sets of questions.

**Different Question Sets.** The two exam scripts of a pair should have significantly different sets of questions. In Figs. 14, 15, and 16 (the right diagram of each figure), the Jaccard coefficient measures to which extent the two exam scripts have different sets of questions by following Equ. (12).

Clearly, T-ExamGAN@S1 and T-ExamGAN@S2 ensure the significant difference between the two exam scripts of each pair while ExamGAN@500 and ExamGAN@1500 generate two exam scripts with 20-45% overlap. Also, T-ExamGAN@S2 demonstrates the better performance compared to T-ExamGAN@S1. We attribute the advantage to the twin difference priority training strategy applied in T-ExamGAN@S2.

**Assessment Equivalency.** The two exam scripts of a pair should be equivalent in assessment (i.e., the scores of students can be compared to rank their academic performances even though they take different exam scripts). Specifically, the two exam scripts should have similar difficulty, distinguishability, validity and rationality. In Figs. 14, 15, and 16 (the left three diagrams of each figure), the difference between the exam scripts of each pair is illustrated by the length of the link between them. Compared with ExamGAN@500 and ExamGAN@1500, it is believed conceptually that the pairs generated by T-ExamGAN@S1 and T-ExamGAN@S2 tend to have the longer link between them (i.e., greater difference), and such loss in assessment equivalency is the cost for the optimization goal, i.e., the two exam scripts in a pair should have different sets of questions. However, it is interesting that the loss in assessment equivalency is trivial overall as shown in Figs. 14, 15, and 16. Moreover, we observe that T-ExamGAN@S1 is slightly better than T-ExamGAN@S2. The training strategy applied in T-ExamGAN@S1 enforces high quality of the exam scripts such that the exam scripts tend to share more similar quality.

**Exam Script Quality.** As shown in Figs. 14, 15, and 16 (the left three diagrams in each figure), the quality of individual exam scripts generated by ExamGAN@500 and ExamGAN@1500 does not have noticeable advantage compared to that by T-ExamGAN@S1 and T-ExamGAN@S2. Look closely, the key reason is that the generators and discriminator in T-ExamGAN@S1 and T-ExamGAN@S2 are initialized as those in T-ExamGAN@S2.
In summary, T-ExamGAN@S1 and T-ExamGAN@S2 demonstrate their unique advantages in generating a pair of high quality exam scripts which are equivalent in assessment and have significantly different sets of questions.

7 CASE STUDY

A case study has been conducted in a real teaching scenario in the School of Computer Science, South China Normal University. The undergraduate course “Programming” is selected. The course has 68 knowledge points and a set of 240 exercises, each exercise contains 1 to 3 knowledge points. The students can do the exercises online. In the case study, we used 533 students’ 384,945 exercise answer records, from 09/2020 to 01/2021, to train a DKT model (AUC of 0.83606). In order to train ExamGAN, following method introduced in Section 4.4, we randomly generated 100 groups from these 533 students (each with 30 students) to generate the training data.

Three real classes of the same course in the new semester from 02/2021 to 04/2021 are involved in the case study. For each class, a real exam script is generated at the end of the semester using ExamGAN. For each class, the knowledge mastery level of students is identified based on exercises answered in the semester following Section 4.1 and fed to the generator. Each exam script consists of questions from the exam question bank which includes the set of exercises and 100 more questions. Each exam script contains 40 questions, and the full score is 100, 2.5 for each question. From test results of the three classes, difficulty, distinguishability, rationality, and validity (denoted as Dif, Dis, Rat and Val respectively) of the generated exam scripts are derived and shown in column 2-4 in Table 4. Clearly, they are highly desirable and consistent for all classes.

For each class, we also generate additional exam scripts using ExamGAN and baselines. In the same way as in Section 6.1.2, the trained DKT model is used to predict the results of students in the class if working on these exam scripts. From the predicted results, the four properties of the exam scripts are derived. In column 5-8 in Table 4, average of the properties over three classes are reported. The similarity between column 2-4 and column 5 verifies the experiment results reported for ExamGAN in Section 6.1.2 are reliable, i.e., similar to asking students to really work on the exam scripts. Compared with baselines, ExamGAN has demonstrated the noticeable advantage.

In the case study, we also invited three teachers of the course as the human experts to evaluate the exam scripts generated using ExamGAN and baselines. Based on their
teaching experiences, they gave score 0-5 (higher is better) to each exam script from the perspective that human experts can evaluate, i.e., difficulty, distinguishability, and validity. The average ratings are 4.2, 3.8, and 3.9 for the exam scripts generated by ExamGAN; 4.0, 3.7 and 3.3 for those by ExamVAE; 4.0, 3.2, and 3.7 for those by GA; 3.7, 3.8, and 4.3 for those by RSF. The ratings further evidence the effectiveness of ExamGAN even though the evaluation of human experts is influenced by subjective experiences.

8 CONCLUSION AND FUTURE WORK

To fill the gap in the research field of automatic exam script generation, this paper has delivered ExamGAN to generate high quality exam scripts featured by validity (i.e., the proper knowledge coverage), difficulty (i.e., the expected average score of students), distinguishability (i.e., the ability of distinguishing academic performances between students), and rationality (i.e., the desirable student score distribution). Interestingly, ExamGAN can be used to verify whether an update of the exam question bank is necessary to maintain the quality assessment. In addition, ExamGAN has been extended to T-ExamGAN to generate a pair of high quality exam scripts which are equivalent in assessment and have significantly different sets of questions. Based on extensive experiments on three benchmark datasets and a case study in the real teaching scenario, the superiority of the proposed solutions has been verified in various aspects.

REFERENCES

[1] S. Kamya, M. Sachdeva, N. Dhalival, and S. Singh, “Fuzzy logic based intelligent question paper generator,” in Proc. IEEE Int. Conf. Comput. Sci. Technol., vol. 4, no. 11, pp. 707–710, 2016.

[2] W. Yuan, C. Deng, H. Zhu, and J. Li, “The statistical analysis and evaluation of examination results of materials research methods course,” Creative Edu., vol. 3, no. 7, pp. 162–164, 2012.

[3] M. Yildirim, “A genetic algorithm for generating test from a question bank,” Comput. Appl. Eng. Educ., vol. 18, no. 2, pp. 298–305, 2010.

[4] S. A. El-Rahman and A. H. Zolait, “Automated test paper generation using utility based agent and shuffling algorithm,” Int. J. Web-Based Learn. Teach. Technol., vol. 14, no. 1, pp. 69–83, 2019.

[5] L. Fendler and I. Muzaffar, “The history of the bell curve: Sorting and the idea of normal,” Educ. Theory, vol. 58, no. 1, pp. 63–82, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1471-5441.2007.00276.x

[6] C. Piech et al., “Deep knowledge tracing,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2015, pp. 505–513.

[7] R. Rajendran, A. Kumar, K. E. Carter, D. T. Levin, and G. Biswas, “Predicting learning by analyzing eye-gaze data of reading behavior,” in Proc. 11th Int. Conf. Educ. Data Mining Soc., 2018, pp. 453–461.

[8] T. J. Tiam-Lee and K. Sumi, “Analysis and prediction of student emotions while doing programming exercises,” in Proc. Int. Conf. Intell. Tutoring Syst., 2019, pp. 24–33.

[9] T. Zhu et al., “MT-MCD: A multi-task cognitive diagnosis framework for student assessment,” in Proc. Int. Conf. Database Syst. Appl. Adv., 2018, pp. 318–335.

[10] A. Chavan, M. Mohandas, R. Manjarekar, D. Karekar, and S. Man-dhare, “Automated question paper generator system using apriori algorithm and fuzzy logic,” Int. J. Innov. Res. Sci. Technol., vol. 2, no. 11, pp. 707–710, 2016.

[11] M. Yu, W. Zhang, J. Wang, and Y. Yu, “SeqGAN: Sequence generative adversarial nets with policy gradient,” in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 2852–2858.

[12] M.-C. Lee, B. Gao, and R. Zhang, “Rare query expansion through generative adversarial networks in search advertising,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 500–508.

[13] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, “Recent progress on generative adversarial networks (GANs): A survey,” IEEE Access, vol. 7, pp. 36 322–36 333, 2019.

[14] M. Zamorski, A. Zdobylak, M. Zieba, and J. Swiatek, “Generative adversarial networks: Recent developments,” in Proc. Int. Conf. Artif. Intell. Soft Comput., 2019, pp. 248–258.

[15] G. Kulick and R. Wright, “The impact of grading on the curve: A simulation analysis,” Int. J. for Scholarship Teach. Learn., vol. 2, no. 2, Art. no. 5, 2008.

[16] T. Bratz, “Explained: The bell curve at king’s,” 2018. [Online]. Available: https://westerngazette.ca/culture/explained-the-bell-curve-at-kings

[17] M. S. R. Chim and G. V. Kale, “Automatic question paper generation using parametric randomization,” J. Gujarat Res. Soc., vol. 21, no. 2, pp. 444–455, 2008.

[18] A. T. Corbett and J. R. Anderson, “Knowledge tracing: Modeling the acquisition of procedural knowledge,” User Model. User-Adapted Interact., vol. 4, no. 4, pp. 253–278, 1994.

[19] M. A. Hameed, M. Nazir, and M. J. Khan, “A comprehensive survey and analysis of generative models in machine learning,” Comput. Sci. Rev., vol. 38, 2020, Art. no. 100285.
Zhengyang Wu received the PhD degree in educational technology and science from the South China Normal University, in 2015. He is now an associate professor with the School of Computer Science, South China Normal University. He is an executive member of the Technical Committee on Collaborative Computing, China Computer Federation (CCF). He visited and studied with the Chinese University of Hong Kong in 2016, and served as a visiting scholar with RMIT University in 2019. His research interests include data mining, big data analysis, visualization technology, etc.

Ke Deng received the PhD degree in computer science from the University of Queensland (UQ), in 2007. He is currently a senior lecturer with the School of Computing Technologies, RMIT University. Before joining RMIT, he has been a research scientist with Huawei Noah Ark’s Research Lab, Hong Kong for two years. He has been a post-doctoral research fellow with CSIRO ICT center and an ARC Australian postdoctoral fellow with UQ. His research interests include data mining, big data, high performance computing, and optimisation, etc. He has more than 80 research publications.

Judy Qiu received the PhD degree in finance from the University of Queensland, Australia. She is currently an assistant teaching professor of finance with the Pennsylvania State University. She has more than 10 years teaching experience, including positions with Ithaca College, the University of Western Australia, and the University of Queensland. Her research interests focus on financial data and educational data analysis. She has published in international refereed journals such as the Entropy, PLoS One, and Australian Journal of Management.

Yong Tang received the BS and MSc degrees in computer science from Wuhan University, in 1985 and 1990, respectively, and the PhD degree in computer science from the University of Science and Technology of China, in 2001. He is now a professor and dean with the School of Computer Science, South China Normal University(SCNU). He serves as the director of Services Computing Engineering Research Center of Guangdong. He was vice dean with the School of Information of Science and Technology, Sun Yat-sen University. He is a distinguished member and the director of Technical Committee on Collaborative Computing of CCF. He served as general or program committee co-chair of more than 10 conferences, and published more than 200 publications.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.