Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Algorithmic Exam Generation

Omer Geiger and **Shaul Markovitch**
Department of Computer Science
Technion - Israel Institue of Technology, 32000 Haifa, Israel
{omergcs,shaulm}@cs.technion.ac.il

Abstract

Given a class of students, and a pool of questions in the domain of study, what subset will constitute a "good" exam? Millions of educators are dealing with this difficult problem worldwide, yet exams are still composed manually in non-systematic ways. In this work we present a novel algorithmic framework for exam composition. Our framework requires two input components: a student population represented by a distribution over overlay models, each consisting of a set of mastered abilities, or actions; and a *target model ordering* that, given any two student models, defines which should be given the higher grade. To determine the performance of a student model on a potential question, we test whether it satisfies a disjunctive action landmark, i.e., whether its abilities are sufficient to follow at least one solution path. We present a novel utility function for evaluating exams, using the described components. An exam is highly evaluated if it is expected to order the student population with high correlation to the target order. The merit of our algorithmic framework is exemplified with real auto-generated questions in the domain of middle-school algebra.

1 Introduction

Assessing the knowledge state of students is an important task addressed by educators worldwide [Gronlund, 1998]. Knowledge assessment is required not only for the purpose of determining the students' deserved grades, but also for diagnostic evaluation used to focus the pedagogical resources on the students' observed shortcomings [Lin *et al.*, 2012]. The most common method for such an assessment is having the students answer an exam. Composing an exam from which the students' knowledge state can be precisely evaluated is a difficult task that millions of educators encounter regularly.

The importance of exam composition has increased with two major developments in computer-aided education. The first is the growing popularity of *massive open on-line courses* (*MOOC*s) such as Coursera, Kahn Academy, edX, and Academic-Earth, which offer new educational opportunities worldwide [Yuan *et al.*, 2013]. The second is the improvement of *intelligent tutoring systems* (*ITS*). These are software products that intelligently guide students through educational activities [Polson and Richardson, 2013].

Exams are still predominantly written manually by educators. Several attempts have been made at automating this task, often referred to as *testsheet composition* [Hwang, 2003; Guzmán and Conejo, 2005; Lin *et al.*, 2012; El-Alfy and Abdel-Aal, 2008]. In many of these works, exam questions are considered to be atomic abstract objects represented only by a vector of numeric features. Common features include difficulty level, solving time, and discrimination degree. Using such a factorial representation, the problem is then defined as a *mixed integer programming* (*MIP*) problem. Usually the objective (maximization) function is the discrimination level of the entire exam while the remaining features compose the problem constraints. Different optimization algorithms have been applied to solve such *MIP* problem formulations [Hwang *et al.*, 2006; 2008; Duan *et al.*, 2012; Wang *et al.*, 2009].

In these works, assuming a feature vector per question is given, the process of exam composition is effectively automated. However, in order to apply these methods in real educational settings, the feature vectors of all candidate questions must be determined. Alas, it remains unclear how this was done, and this major framework component remains an atomic blackbox. The reader is left to speculate that perhaps the feature vectors are manually specified by a field expert. If so, these methods may be regarded as only semi-automatic.

In this paper, we present a novel algorithmic framework for exam generation, which requires minimal manual specification. We generate real candidate questions, and algorithmically determine which domain abilities they test. Student models are used to represent possible knowledge states, allowing us to determine their performance on candidate questions. Our algorithm accepts as input a target order between knowledge states, indicating the relation "more proficient than". It then searches the space of possible exams for one that best reflects this ordering over the student population.

Building the framework requires us to overcome several difficulties. First we need to define the representation of a student model or knowledge state. As the number of possible student models is typically large, we also need a method by which the user can specify the target student order compactly. Another challenge is determining the performance of a given

student model on a question and on an entire exam. We address this issue by constructing a technique based on graph search and planning, applicable in procedural domains. Finally, we need to define a utility function to guide the search process in the space of possible exams. To this end we use a correlation measure between the grade order imposed by an exam and the target student order. '

2 Problem definition

We define an examination domain as a triplet $\langle Q, A, \psi \rangle$. It is composed of a set of candidate questions Q, a set of abilities $A = \{a_1, a_2, ..., a_m\}$, and a sufficiency predicate $\psi : 2^A \times Q \rightarrow \{1, 0\}$, where $\psi(A', q) = 1$ *iff* the ability set $A' \subseteq A$ is sufficient to answer the question $q \in Q$.

Next, we define a *student model*, using the relatively simple approach known as the *binary overlay model* [Brusilovskiy, 1994]. By this approach, a *student model* is defined as a subset of domain abilities, $s \subseteq A$, mastered by the student. Therefore, a student s answers a question $q \in Q$ correctly *iff* $\psi(s, q) = 1$. The student model, also sometimes referred to as a *knowledge state*, may be alternatively represented by a binary vector with each coordinate indicating mastery of a matching ability or lack thereof.

We denote the set of all possible models as $\mathcal{M} = 2^A$ but assume that not all student models are equally likely. Therefore we denote by $\mathcal{P}_\mathcal{M} = \{\langle \bar{s_i}, p_i \rangle\}$ the distribution over the possible student models, where $\bar{s_i} \in \mathcal{M}$ and p_i is its proportion in the population.

An exam e of order k_e is defined as a vector of k_e questions, and a matching vector of associated non-negative grading weights: $e = \big\langle \langle q_1, ..., q_{k_e} \rangle, \langle w_1, ..., w_{k_e} \rangle \big\rangle$. The grade of a student model $\bar{s} \in \mathcal{M}$ on exam e is simply the sum of grading weights for questions answered correctly by the student : $g(\bar{s}, e) = \sum_{1 \leq i \leq k_e} w_i \cdot \psi(\bar{s}, q_i)$.

We turn to define the notion of a good exam. Suppose an educator teaching a certain domain is asked: What is the perfect exam for this domain? Ideally, the educator would specify, for each pair of students, which is more proficient and thus deserves a higher grade. Obviously a student knowing nothing should be ranked inferior to all others, while a student knowing everything should be ranked superior to all others. However, to determine the complete order, we rely upon the educator's expert knowledge to define the ordering. We call the desired order, given by the educator, the *target student order*, and denote it $\preceq_* \subseteq \mathcal{M}^2$. This is a partial order defining the binary relation "is more proficient than" between pairs of students.

Observe that any exam (e) also defines such a partial order between students ($\preceq_e$) according to their grade. For $\bar{s_1}, \bar{s_2} \in \mathcal{M}$, we have that $\bar{s_1} \preceq_e \bar{s_2} \Leftrightarrow g(\bar{s_1}, e) \leq g(\bar{s_2}, e)$. A good exam is one for which the resulting student grades accurately reflect the target order, while taking into account the model distribution $\mathcal{P}_\mathcal{M}$. That is to say, it is more important to correctly order more likely models than less likely ones. For this purpose we must make use of some correlation function C between orders.

We are now ready to define a utility function for evaluating exams. Given an exam e, an order correlation function C, and a target student ordering $\preceq_*$, we define the utility of e as $U(e) = C(\preceq_e, \preceq_*)$. Note that the absolute grades resulting from our generated exams can always be curved to match any desired absolute grade distribution.

3 Model-based exam generation

In this section we present our algorithmic framework for *MOdel-based Exam Generation* (MOEG).

3.1 Examination domains

A reasonable source for candidate questions is a curriculum textbook or a collection of previous exams. This means that Q is some finite set of questions selected by the educator or curriculum supervisor and coded once for the purpose of all future exam generations.

A more generic approach is to devise a question generating procedure. In section 4.1 we present an algorithm for automatically generating questions in algebra. Naturally, this approach becomes increasingly more difficult with the complexity of the domain. A hybrid approach is to algorithmically produce variations of existing questions based on user refinement of constraints [Singh *et al.*, 2012].

The set of abilities A and a sufficiency predicate ψ are assumed to be given by the educator. However, for procedural domains, we introduce an algorithm that automatically induces the sufficiency predicate. Procedural domains are those where questions are solved by applying a sequence of operators or actions. Such domains can therefore be represented as a search graph, where the vertices are possible solution states S, and the actions are steps executed for solving the exercise. We assume that the set of search graph actions are in fact the set of domain abilities A. Examples of applicable procedural domains include algebra, geometry, trigonometry, classical mechanics, and motion problems.

We turn to define the sufficiency predicate $\psi : 2^A \times Q \rightarrow \{1, 0\}$ for procedural domains. An ability set is sufficient to answer a question *iff* it contains all abilities needed in at least one solution path. This type of logical condition over actions is known as an *action landmark*. Helmert and Domshlak [2011] define a *disjunctive action landmark* as a set of actions such that any solution must include at least one of them. We expand the definition to a *set of sets of actions*, such that each solution must contain at least one of the sets. Let $S(q)$ be the set of all solution paths for a question q. The disjunctive action landmark of q is therefore $l(q) \triangleq \{\{a | a \text{ appears in } t\} | t \in S(q)\}$, or equivalently, as a DNF formula: $l(q) \triangleq \bigvee_{t \in S(q)} [\wedge_{a \in t} a]$. For $A' \subseteq A, q \in Q$ we have that $\psi(A', q) = 1$ *iff* $\exists A_i \in l(q) : A_i \subseteq A'$.

In very simple domains, the set of solutions $S(q)$ can be obtained via exhaustive search. In more complex domains, however, such a procedure is computationally infeasible. We can approximate ψ using an anytime algorithm which collects solutions by sampling operator sequences of limited length, defined fully in the pseudo-code of Figure 1.

3.2 Student population

Describing the student model distribution in the general case requires explicitly defining the probability for each possible

Procedure APPROXIMATE-ACTION-LANDMARK(q)
Constants: runtime limit (T_{lim}), search depth limit (D_{lim}), a limit on the number of collected solutions (SOL_{lim})
$Sols \leftarrow \{\}$
Repeat until (TIMEUP(T_{lim}))
$sol \leftarrow$ Follow random action sequence upto length D_{lim}
If $sol \neq None$
$Sols \leftarrow Sols \cup \{sol\}$
$ShortestSols$upto $\leftarrow SOL_{lim}$ shortest solutions in $Sols$
Return $\{\{a \in A | a \in sol\} | sol \in ShortestSols\}$

Figure 1: Pseudo-code for action landmark approximation method

model in $\mathcal{M} = 2^A$. Due to the exponential size of this model set, we adopt a simplifying independence assumption between abilities. By doing so, we reduce the complexity of distribution specification from exponential to linear in $|A|$, while retaining reasonable flexibility. Formally, this simplification means we assume that a randomly selected student masters each ability $a_i \in A$ with probability $p_i \in [0, 1]$. Furthermore, we assume that the probability of mastering each ability is independent and that students are mutually independent as well. It follows that the probability of a model is:

$$Pr(\langle a_1, a_2, ..., a_{|A|}\rangle) \equiv \prod_{i:a_i=1} p_i \cdot \prod_{i:a_i=0} (1 - p_i)\,.$$

In future work we intend to relax this assumption and use Bayesian networks for allowing arbitrary dependencies [Geiger *et al.*, 1990].

3.3 Target student order

Explicitly specifying an order over the set of student models is also infeasible in the general case due to the exponential size of the model set $\mathcal{M}$. We therefore propose three methods for simple order specification. In the first method, the educator is required to specify a vector of non-negative ability weights $\bar{w} = \langle w_1, ..., w_{|A|}\rangle$, indicating the importance of each ability to domain mastery. Given these, the proficiency level of a student model $\bar{s} = \langle s_1, s_2, ..., s_{|A|}\rangle \in \mathcal{M}$ is defined as the sum of its mastered ability weights, i.e., the dot product $(\bar{s}, \bar{w}) = \Sigma_i w_i \cdot s_i$. Having defined a scalar proficiency level per student, the target order definition is straightforward. For any $\bar{s_1}, \bar{s_2} \in \mathcal{M}$:

$$\bar{s_1} \preceq_{\bar{w}} \bar{s_2} \Leftrightarrow (\bar{s_1}, \bar{w}) \leq (\bar{s_2}, \bar{w})\,.$$

The second method uses the order induced by the subset relation ($\preceq_{\subseteq}$). A student who has mastered all the abilities of another, as well as some additional ones, is naturally considered more proficient:

$$\bar{s_1} \preceq_{\subseteq} \bar{s_2} \Leftrightarrow \forall i[\bar{s_1}[i] = 1 \rightarrow \bar{s_2}[i] = 1]\,.$$

The advantage of this method over the first one is that it requires no input from the educator. However, the first method allows more refined orders to be specified and is thus preferable when the additional input is available.

The third method for target order specification is *question-pool* based. It requires a representative set of questions P, and defines the order between student models according to the number of questions in the pool they answer correctly:

$$\bar{s_1} \preceq_P \bar{s_2} \Leftrightarrow \Sigma_{q_i \in P} \psi(\bar{s_1}, q_i) \leq \Sigma_{q_i \in P} \psi(\bar{s_2}, q_i)\,.$$

The pool is potentially much larger than the desired exam size, and thus cannot serve as an exam. In some cases it may be reasonable to use the entire question set Q as the pool.

3.4 Order correlation measure

We have reduced the problem of evaluating an exam e to comparing its induced student order ($\preceq_e$) with the target student order ($\preceq_*$). For this, we require a correlation measure that evaluates the similarity between the two orders. We considered several alternatives such as Kendall's τ [1938], Goodman and Kruskal's Γ [1954], Somers' d [1962], and Kendall's τ_b [1945]. Eventually we selected the classic Kendall's τ for this work, but the others are also adequate candidates. Kendall's τ compares the number of concordant student pairs (N_c) with the number of discordant student pairs (N_d):

$$\begin{aligned} N_c &\triangleq |\{\bar{s}_1, \bar{s}_2 \in \mathcal{M} : \bar{s}_1 \preceq_e \bar{s}_2 \wedge \bar{s}_1 \preceq_* \bar{s}_2\}| \\ N_d &\triangleq |\{\bar{s}_1, \bar{s}_2 \in \mathcal{M} : \bar{s}_1 \preceq_e \bar{s}_2 \wedge \bar{s}_1 \succeq_* \bar{s}_2\}| \\ \tau &\triangleq \frac{N_c - N_d}{\binom{|\mathcal{M}|}{2}}\,. \end{aligned}$$

A value of 1 implies complete correlation, while a value of -1 implies complete inverse correlation. Note that this measure does not account for ties in either of the partial orders, deeming the full range $[-1, 1]$ unreachable in the presence of ties.

3.5 Exam utility function

Calculating the order correlation over all possible models is computationally infeasible. Due to this practical constraint, we resort to an approximation measure based on a model sample drawn from the given distribution $\mathcal{P}_{\mathcal{M}}$. We define our utility function as Kendall's τ computed over the model sample between the target order and the exam-induced order.

Recall from section 2 that we require the order correlation measure to reflect the distribution of student models, stressing the significance of more likely models over less likely ones. Our sample-based correlation measure meets this requirement: The likely models are more likely to be sampled, perhaps more than once, and thus have a stronger influence on the measure. The resulting measure approximates the generalization of Kendall's τ to non-uniform element weights [Kumar and Vassilvitskii, 2010].

3.6 Searching the space of exams

Our algorithm for exam generation involves three main phases: adding questions, swapping questions, and adjusting grading weights. Starting with an empty set of questions, the algorithm iteratively adds the question for which the resulting set yields maximal utility value, using uniform grading weights. When the question set has reached the desired exam size, the algorithm turns to consider single swaps between an exam question and an alternative candidate. The swap maximizing the utility is performed until a local optimum is

reached, at which point the algorithm proceeds to adjusting the grading weights.

Recall that the absolute grades of the students are of no importance to us as we are only interested in the order imposed on the student sample. It follows that, theoretically, a good set of weights would meet the desired property that every two subsets have a different sum. Using such a weight set makes it possible to differentiate between any two students who answered differently on at least one exam question, i.e., they will surely receive different grades. Constructing a weight set with this property is not difficult, for example: $\{1 + \frac{1}{p_i} : p_i \text{ is the ith prime number}\}$ or $\{1 + \frac{1}{2^i} : i \in \mathbb{N}\}$ or even a weight set randomly generated from a continuous range (with a theoretical probability of 1). Of course not all such candidate weight sets are equivalent in terms of the orderings they may impose between subsets.

The weight adjustment performed by our algorithm enables the construction of such a desired weight set by applying weight perturbations of exponentially decreasing granularity. Starting from the local optimum reached at the end of the question swapping phase, the algorithm proceeds to perform a local search over the space of weight vectors. The search operators include the addition of a small constant Δ (e.g. 0.05) or its negation to any question weight. When a local optimum is reached, the increment step Δ is halved and the process continues this way until no improvement is made.

The algorithm produces exams expected to have good discriminating capabilities. It rejects questions for which the student answers are extremely homogeneous, i.e., very difficult or very easy ones, since such questions contribute little to the induced order correlation. Moreover, the desired discrimination is defined by the target order, given as input. Questions for which proficient students, as defined by input, are more likely to answer correctly than others, are preferred. The grading weights of exam questions are expected to behave similarly. Perhaps contrary to initial intuition, difficult exam questions are not expected to receive high grading weights. This behavior, attributed to the lower discriminative capability of such questions, is reasonable in real educational settings. An exam weighted directly by difficulty generally results in a distorted grading curve, as only the few most proficient students will answer correctly the highly weighted questions.

3.7 Wrap-up

We show in Figure 2 a high-level pseudo-code for the entire exam generation procedure described. It accepts as input the domain's ability set A and the student model distribution $\mathcal{P}_{\mathcal{M}}$. For simplicity of presentation, the pseudo-code uses the default method for defining a target student order ($\preceq_{\bar{w}}$). Therefore the third input parameter is the teacher-specified ability weight vector $\bar{w}$.

4 Evaluation

In this section we present an empirical evaluation of the complete MOEG framework over a procedural domain.

Procedure MOEG($A, \mathcal{P}_{\mathcal{M}}, \bar{w}$)
  $Q \leftarrow$ GENERATE-QUESTIONS()
  Foreach $q \in Q$
    $\hat{l}(q) \leftarrow$ APPROXIMATE-ACTION-LANDMARK(q, A)
  $\hat{\mathcal{M}} \leftarrow$ SAMPLE-STUDENT-POPULATION($\mathcal{P}_{\mathcal{M}}$)
  Foreach $\bar{s} \in \hat{\mathcal{M}}$
    $Proficiency(\bar{s}) \leftarrow \Sigma_i w_i \cdot s_i$
  $\preceq_* \leftarrow \{(\bar{s}_1, \bar{s}_2) | Proficiency(\bar{s}_1) \leq Proficiency(\bar{s}_2)\}$
  Foreach $(s, q) \in \hat{\mathcal{M}} \times Q$ *# student s in set notation*
    $\psi(s, q) \leftarrow \begin{cases} 1 & \exists t \in \hat{l}(q) \text{ s.t. } s \subseteq t \\ 0 & \text{otherwise} \end{cases}$
  Foreach exam e and students $\bar{s_1}, \bar{s_2} \in \mathcal{M}$:
    $grade(\bar{s_1}, e) \triangleq \sum_{1 \leq i \leq k_e} w_i \cdot \psi(\bar{s_1}, q_i)$
    $\preceq_e \triangleq \{(\bar{s}_1, \bar{s}_2) | grade(\bar{s_1}, e) \leq grade(\bar{s_2}, e)\}$
    $U(e) \triangleq \tau(\preceq_*, \preceq_e)$ *# or other correlation measure*

EXAM BUILD:
  Initialize $exam \leftarrow Empty\ Exam$
  $exam \leftarrow$ ADD-QUESTIONS($exam, U, k_e$)
  $exam \leftarrow$ SWAP-QUESTIONS($exam, U$)
  $\Delta \leftarrow 0.05$ *# or any other small value*
  $improved \leftarrow TRUE$
  While improved
    $(exam, improved) \leftarrow$ ADJUST-WEIGHTS($exam, U, \Delta$)
    $\Delta \leftarrow \Delta / 2$

Figure 2: MOEG pseudo-code

4.1 The Domain

The domain over which the evaluation was performed is that of single variable linear equations, typical of middle-school algebra courses. Domain questions ask students to solve for x in equations, e.g., $2x + 5 = 13$, and $2 - (-4x + 2(x + 6 - 3x)) = 4x$.

The ability set

The ability set A, consists of 18 types of algebraic manipulations. We define the following 5 main types of actions and later decompose them into subtypes:

(U) Unite: $-2x + 7x \Rightarrow_U 5x$.

(O) Open multiplication: $-3(x + 2) \Rightarrow_O -3x - 6$.

(D) Divide: $2x = -8 \Rightarrow_D x = -4$.

(M) Move: $5x - 2 = -6 + 3x \Rightarrow_M 5x = -6 + 2 + 3x$.

(R) Rearrange: $x - 8 + 7x = 10 \Rightarrow_R x + 7x - 8 = 10$.

Each such action type is further decomposed into subtypes according to the parameters it is applied over. For example, Unite (U) is decomposed into 8 subtypes according to 3 binary arguments: the type of terms united (variable or constant), the sign of the first term ('+' or '-'), and the sign of the second. In a similar manner each action type is decomposed, giving us a total of $|A| = 18$ abilities: 8 (U), 2 (O), 2 (D), 4 (M), 2 (R).

Question set generation

For this algebraic domain we devised a question generating algorithm. It starts with an equation representing the desired solution, and repetitively applies complicating operations while retaining equation equivalence.

The algorithm receives two parameters: depth (d) and width (w). It begins with a solution equation of the sort $x = c$ and manipulates it by applying a short random sequence of basic operations, resulting in an equation of the form $a_1x+b_1 = a_2x+b_2$, where the parameters a_1, a_2, b_1, b_2 may be 0, 1 or any other value. The algorithm then iteratively performs d "deepening" manipulations, transforming expressions of the sort $ax + b$ to $a'x + b' + c(a''x + b'')$ while maintaining that $a = a' + ca''$ and $b = b' + cb''$. These deepening manipulations are performed on random levels of the equation tree structure, and so may be applied to an inner part created by a previous iteration.

The algorithm continues with w "widening" iterations where a random term is split to two, i.e, $b \Rightarrow b' + b''$ or $ax \Rightarrow a'x + a''x$ (where $b = b' + b''$, $a = a' + a''$). Finally, all terms are shuffled recursively to produce a random permutation. In all manipulations, the algorithm ensures that the newly formed coefficients are bounded by some constant (100).

A set of 160 questions used for evaluation, Q, was produced by applying the described procedure 10 times with each (w, d) value pair in $\{0, 1, 2, 3\}^2$.

4.2 Empirical Methodology

Ideally we would have liked to evaluate the exams by measuring their fitness over the entire model population. Since this is computationally infeasible we use an "oracle" sample for evaluation. Two things are important with regard to this oracle sample: first, it is taken independently from the utility sample, and second, it is considerably larger. The resulting evaluation function is therefore (1) unbiased in evaluating the algorithm's produced exams, and (2) a better approximation of the entire distribution. For completeness, we present the values of both the guiding utility and the oracle evaluation in some graphs of this section.

Four independent variables were experimented with: the utility sample size, the exam size k_e, and two parameters controlling the student population ($\mathcal{P}_\mathcal{M}$) and ability weights ($\bar{w}$): ϵ_p, and $\epsilon_{\bar{w}}$ respectively. The ability probabilities $\{p_i\}$ were independently sampled from $Uni(0.75 \pm \epsilon_p)$ and the $\{w_i\}$ values were sampled from $Uni(1 \pm \epsilon_{\bar{w}})$. Default values, used unless stated otherwise, are sample Size $= 400, k_e = 10, \epsilon_p = 0.15, \epsilon_w = 0.5$.

A sample of size 1000 was used for the oracle. All results presented are based on 50 independent experiment runs using the same question set Q. The derivation of their action landmarks was also performed once[1].

4.3 Experiments

We tested the performance of the MOEG algorithm and compared it to three baseline algorithms we defined[2]. *Uniform generate & test* generates random uni-weight exams and evaluates them using the search utility, maintaining the best exam found yet. An improved variation is the *weighted generate & test*, which makes a biased selection of exam questions inspired by the *Item Response Theory* concept of *item-information* [De Ayala, 2009], reflecting a question's usefulness in exams. It selects questions with probability proportional to their information level defined as $p(1 - p)$, where p is the proportion of utility sample students answering the question correctly. Note that these two baseline algorithms use a key component of our framework — the utility function.

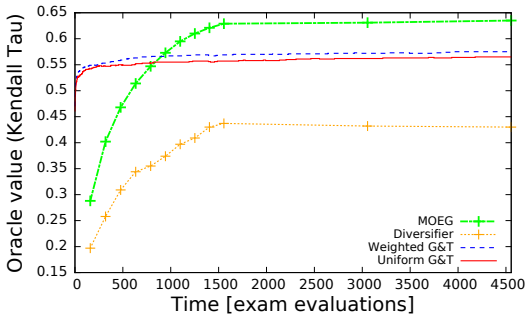

Figure 3: Performance over time

The third baseline algorithm, the *diversifier*, attempts to maximize the diversity of exam questions in terms of syntactical features, because we assume that this will better differentiate between students. We defined six question features: number of *constant terms, variable terms, positive coefficients, negative coefficients, parentheses,* and *overall terms*. The feature values are normalized as Z-scores to account for the different scales. The algorithm starts with a random question, then iteratively selects a question to add, maximizing the sum over pairwise Euclidean distances between questions. This is followed by a similar swapping phase.

Figure 3 displays MOEG's improvement over time during the question selection and swapping phases, compared to the baseline competitors. We can see that the MOEG curve surpasses the others, even with a partial exam of 6 questions out of 10. The *weighted generate and test* performs better than the *uniform* version as expected, & the *diversifier* performs surprisingly poor.

Another algorithmic variation we tested allows question swapping during all phases, i.e. question addition and weight perturbation. It might have been reasonable to expect better performance due to the additional flexibility we allow the algorithm. However, results show that this is not the case and the proposed alternative algorithm yields nearly equivalent results. The runtime it requires, however, is significantly longer, as may be expected due to the larger branching factor.

Longer exams are expected to allow a better ordering of the student population. Figure 4 presents how the exam length, k_e, affects algorithm performance. As expected, both curves increase monotonically with a diminishing slope. As expected, the utility is higher than the oracle since the search algorithm tries to optimize it.

Increasing the size of the utility sample is expected to improve the quality of the utility function. Figure 5 shows the

[1]parameters: $D_{lim} = 40, SOL_{lim} = 100, T_{lim} = 300$ sec.

[2]We could not compare MOEG to testsheet composition methods such as [Hwang *et al.*, 2006], as they work with completely different input and cannot be applied to the setup we use.

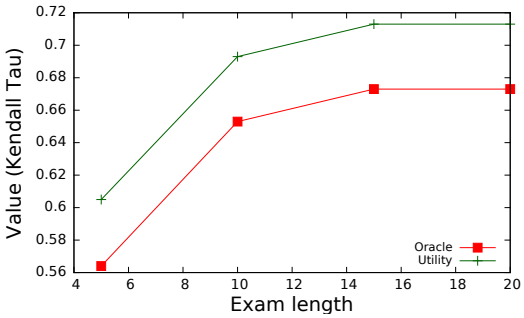

Figure 4: The effect of exam length on performance

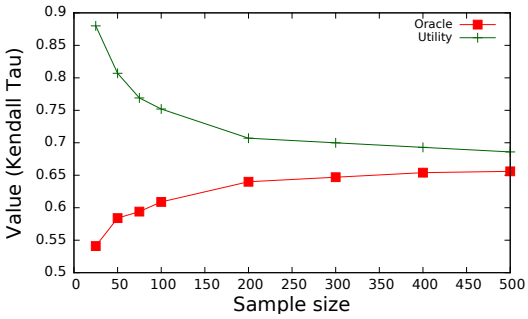

Figure 5: The effect of sample size on performance

effect of the utility sample size on performance. Indeed, we can see that the performance of the algorithm improves as sample size increases. The difference between the value of the oracle and that of the search utility can be viewed as the estimation error of the latter. We can see that this error decreases as larger sample sizes are used.

An additional experiment was conducted to test the stability of the framework with respect to the ϵ_p and $\epsilon_{\bar{w}}$ parameters. We ran MOEG with $\epsilon_p \in \{0, 0.05, 0.15, 0.25\}$, $\epsilon_{\bar{w}} \in \{0, 0.1, 0.3, 0.5\}$ and tested the average oracle evaluation and standard deviation over 50 runs. From the generally consistent statistics acquired, and in the absence of any clear trend, we conclude that the framework is stable with respect to the model population and the ability weights.

5 Discussion

This paper presents MOEG, a generic framework for automating exam composition, an important task in the pedagogical process. The framework is applicable to any domain where a set of cognitive abilities A and a sufficiency predicate ψ may somehow be defined. Automatically deducing ψ via action landmarks requires further that solutions be represented as paths in search graphs, with A as the operator set. Several such procedural domains may come to mind, one of which is geometry. A classic ITS paper [Anderson *et al.*, 1985] presents a tutor for teaching geometrical reasoning. The tutor utilizes a library of inference rules: triangle congruency/similarity, properties of polygons, transitivity of angle/segment congruency, etc. It is implemented as a production system and used to generate proofs, using different combinations of production rules. This solving process is naturally formalized as a graph search with inference rules as operators.

The idea of inference steps as operators is also applicable in various computational domains. Consider, for example, inferring the intersection point of two functions in *analytical geometry*, the length of a right triangle's hypotenuse in *trigonometry*, or the 2D location at time t of an object moving according to $\vec{x}(t)$. These example inferences may serve as arcs on solution paths of three different domains in which diverse sets of useful inference rules exist.

Automatically deducing ψ for questions is also possible for domain types other than search spaces. For example, domains where solutions may be obtained by automated theorem provers are also MOEG-applicable with a simple extension. The abilities A will be the axioms, lemmas, and theorems that a student should know, while solutions will be complete proof trees. Given a proof tree representing a solution, the axioms at the leaves will be considered the set of required abilities for the solution. Collecting these ability sets from different proofs found by the theorem prover results in sufficiency predicates of familiar form: *disjunctive action landmarks*.

In other domains, different methods for deducing ψ may exist. Consider the domain of *combinatorial problems* in which problems are given in text, e.g., "How many non-empty subsets does a set of size N have?" It is natural here to define domain abilities as combinatorial concepts such as *non-redundant combination* (nrc), *summation principle* (sum), *redundant permutation* (rp), or *subtraction principle* (sub). Automatically deducing ψ from the question text alone is beyond the state of the art, but given the set of possible solutions, the task becomes feasible using standard syntactic parsers. For the question above the solution set is $\left\{\sum_{i=1}^{i=n} \binom{N}{i}, 2^N - 1\right\}$, and the resulting sufficiency predicate is $\psi = (nrc \wedge sum) \bigvee (rp \wedge sub)$.

Item Response Theory (*IRT*) [De Ayala, 2009] is another paradigm that addresses test design by mapping ability levels to performance on questions. *IRT* assumes the existence of a (typically unidimensional) *latent trait* per student indicating ability level. Furthermore, every question (item) is assumed to have a representative *item characteristic curve* (*ICC*) that maps ability level to success probability. Several accepted parametric *ICC*s exist, e.g., the classic *3-parameter logistic model* [Birnbaum, 1957] with parameters for item difficulty, discrimination, and guessing probability. Our work differs from the above in several respects. We replace the latent trait representing students with student models, and we replace the items represented by predefined *ICC* functions, controlled by a few parameters, with formal structures, manipulated in the search for solutions. Finally, we replace the probabilistic prediction of student performance with deterministic algorithmic inference, based on the student model and question structure. Our framework produces well-balanced exams that rank students by proficiency level as defined by the educator. We believe this is an important step towards making AI techniques practical for improving education.

References

[Anderson *et al.*, 1985] John R Anderson, C Franklin Boyle, and Gregg Yost. The geometry tutor. In *IJCAI*, pages 1–7, 1985.

[Birnbaum, 1957] ALLAN Birnbaum. Efficient design and use of tests of a mental ability for various decision-making problems. *Randolph Air Force Base, Texas: Air University, School of Aviation Medicine*, 26, 1957.

[Brusilovskiy, 1994] PL Brusilovskiy. The construction and application of student models in intelligent tutoring systems. *Journal of Computer and Systems Sciences International*, 32(1):70–89, 1994.

[De Ayala, 2009] RJ De Ayala. *The Theory and Practice of Item Response Theory*. The Guilford Press, 2009.

[Domshlak *et al.*, 2011] Carmel Domshlak, Malte Helmert, Erez Karpas, Emil Keyder, Silvia Richter, Gabriele Röger, Jendrik Seipp, and Matthias Westphal. Bjolp: The big joint optimal landmarks planner. *IPC 2011 Planner Abstracts*, pages 91–95, 2011.

[Duan *et al.*, 2012] Hong Duan, Wei Zhao, Gaige Wang, and Xuehua Feng. Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm ts/bbo. *Mathematical Problems in Engineering*, 2012, 2012.

[El-Alfy and Abdel-Aal, 2008] El-Sayed M El-Alfy and Radwan E Abdel-Aal. Construction and analysis of educational tests using abductive machine learning. *Computers & Education*, 51(1):1–16, 2008.

[Geiger *et al.*, 1990] Dan Geiger, Thomas Verma, and Judea Pearl. Identifying independence in bayesian networks. *Networks*, 20(5):507–534, 1990.

[Goodman and Kruskal, 1954] Leo A Goodman and William H Kruskal. Measures of association for cross classifications*. *Journal of the American Statistical Association*, 49(268):732–764, 1954.

[Gronlund, 1998] Norman E Gronlund. *Assessment of Student Achievement*. ERIC, 1998.

[Guzmán and Conejo, 2005] Eduardo Guzmán and Ricardo Conejo. Self-assessment in a feasible, adaptive web-based testing system. *IEEE Transactions on Education*, 48(4):688–695, 2005.

[Hwang *et al.*, 2006] Gwo-Jen Hwang, Bertrand MT Lin, and Tsung-Liang Lin. An effective approach for test-sheet composition with large-scale item banks. *Computers & Education*, 46(2):122–139, 2006.

[Hwang *et al.*, 2008] Gwo-Jen Hwang, Hui-Chun Chu, Peng-Yeng Yin, and Ji-Yu Lin. An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests. *Computers & Education*, 51(3):1058–1072, 2008.

[Hwang, 2003] Gwo-Jen Hwang. A test-sheet-generating algorithm for multiple assessment requirements. *IEEE Transactions on Education*, 46(3):329–337, 2003.

[Kendall, 1938] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, pages 81–93, 1938.

[Kendall, 1945] Maurice G Kendall. The treatment of ties in ranking problems. *Biometrika*, pages 239–251, 1945.

[Kumar and Vassilvitskii, 2010] Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th International Conference on World Wide Web*, pages 571–580. ACM, 2010.

[Lin *et al.*, 2012] Huan-Yu Lin, Jun-Ming Su, and Shian-Shyong Tseng. An adaptive test sheet generation mechanism using genetic algorithm. *Mathematical Problems in Engineering*, 2012.

[Polson and Richardson, 2013] Martha C Polson and J Jeffrey Richardson. *Foundations of Intelligent Tutoring Systems*. Psychology Press, 2013.

[Singh *et al.*, 2012] Rohit Singh, Sumit Gulwani, and Sriram K Rajamani. Automatically generating algebra problems. In *AAAI*, 2012.

[Somers, 1962] Robert H Somers. A new asymmetric measure of association for ordinal variables. *American Sociological Review*, pages 799–811, 1962.

[Wang *et al.*, 2009] Feng-rui Wang, Wen-hong Wang, Quanke Pan, Feng-chao Zuo, and JJ Liang. A novel online test-sheet composition approach for web-based testing. In *IEEE International Symposium on IT in Medicine & Education*, volume 1, pages 700–705. IEEE, 2009.

[Yuan *et al.*, 2013] Li Yuan, Stephen Powell, and JISC CETIS. Moocs and open education: Implications for higher education. *Cetis White Paper*, 2013.