Cross-Domain Performance of Automatic Tutor Modeling Algorithms

Rohit Kumar
Raytheon BBN Technologies
Cambridge, MA, USA
rkumar @ bbn.com

ABSTRACT

In our recent work, we have proposed the use of multiple solution demonstrations of a learning task to automatically generate a tutor model. We have developed a number of algorithms for this automation. This paper describes the application of these domain-independent algorithms to three datasets from different learning domains (Mathematics, Physics, French). Besides verifying the applicability of our approach across domains, we report several domain specific performance characteristics of these algorithms which can be used to choose appropriate algorithms in a principled manner. While the Heuristic Alignment based algorithm (*Algorithm 2*) may be the default choice for automatic tutor modeling, our empirical finding suggest that the Path Pruning based algorithm (*Algorithm 4*) may be favored for language learning domains.

Keywords

Tutor Modeling, Automation, Domain Independence, STEM domains, Language Learning

1. INTRODUCTION

Wide-scale transition of Intelligent Tutoring Systems (ITS) to the real world demands a scalable ability to develop such systems. The past decade has seen the first instantiations of industrialization of ITS development in the form of commercial products for different learning domains as well as diverse user populations. In addition to addressing non-technical challenges such as designing robust production processes around multidisciplinary teams of domain and pedagogical experts [1], the industrialization of this technology is enabled by technical advancements such as the development of general purpose authoring tools [2] which has allowed a scalable workforce to contribute to ITS development.

In this paper, we extend our recent work [3][4] on automatically developing Example-Tracing Tutors (ETTs) [5] using multiple behavior demonstrations. Conventionally, ETTs are developed in three stages by trained domain experts: (1) User Interface (UI) development, (2) Behavior demonstration, (3) Generalization and annotation of the behavior graph. As ITS are being deployed to a large active user pool, it is now possible to pilot the UI with a small sample of learners to collect multiple behavior demonstrations. We can significantly reduce the Stage 3 effort of ITS developers by using algorithms that can automatically create a generalized behavior graph from multiple demonstrations. Several algorithms to address this challenge have been proposed and evaluated [4].

In this paper, we will study the applicability and performance of these algorithms on publicly available datasets from three different learning domains. Section 3 summarizes the key characteristics of the four algorithms used in our study. Section 4 describes learning domains and the corresponding datasets used in this work. Results and Analysis from our experiments are presented in Section 5. Before diving into the algorithms, the next section reviews related work on automation of tutor model development.

2. RELATED WORK

Automation of tutor model development process has been explored in different contexts using completely automated methods as well as augmentation of authoring tools [6][7]. For example, motivated by application in language learning, a series of workshops on the problem of automatic question generation [8] explored a number of information extraction and NLP techniques that employ existing linguistic resources. Barnes and Stamper [9] proposed a method that uses existing student solutions to generate hint messages for the Logic Proof tutor. Recently, Eagle et al. [10] have used clustering of interaction network states as an approach to the same problem.

In the context of knowledge-tracing and example-tracing tutors, McLaren et al. [11] proposed the use of activity logs from novice users to bootstrap tutor model development. They developed software tools that integrate access to novice activity logs with authoring tools. The baseline algorithm (Interaction Networks) used in our work is similar to the integrated data view used in this prior work. Furthermore, the algorithms used in our work address some of the shortcomings of their work (e.g. inability to identify "buggy" paths).

In addition to tutor modeling, recent work has investigated automated methods for improving domain and student models [12] [13]. Sudol et al. [14] aggregated solution paths taken by different learners to develop a probabilistic solution assessment metric. Johnson et al. [15] are creating visualization tools for interaction networks that combine learner traces from open-ended problem solving environments. They have developed an algorithm for reducing the complexity of combined networks to make them more readable/navigable. In a similar spirit, work by Ritter et al. [16] used clustering techniques to reduce the large feature space of student models to assist in qualitative model interpretation.

3. GENERATING BEHAVIOR GRAPHS

Automatic Behavior Graph Generation (ABGG) algorithms analyze the similarities and difference between multiple solution demonstrations of a problem to induce a behavior graph that can serve as a tutor model for the problem.

3.1 Behavior Graphs

Behavior graphs [5] are directed graphs. The nodes in this graph correspond to valid solution states. Non-terminal nodes represent partial solutions. Edges in the graph represent solution paths some of which are correct and lead to the next state while other are incorrect and usually lead back to the same state. Edges are annotated with the conditions that a behavior event must meet to traverse the path.

Behavior graphs may contain multiple paths between two nodes. Multiple paths are useful to facilitate learner's exploration of alternate solutions to a problem especially in ill-defined learning domains. Behavior graphs may also include unordered groups. As the name suggests, states within an unordered group may be traversed in any order.

Well-constructed behavior graphs have several desirable characteristics which motivate the design of metrics we use to evaluate ABGG algorithms.

3.1.1 Effective

Since the purpose of the behavior graphs is to serve as a tutor model, the primary metric for evaluating these models is their learning efficacy measured via use of the models by a relevant sample of learners. However, in this paper we focus only on the use of automated metrics that do not require access to a learner pool. Further, as we in section 5, the automatically generated behavior graphs are not perfect. They require checking and refinement by ITS developers before they can be used with learners.

3.1.2 Readable

One of the key characteristics of behavior graphs that makes them a popular model is that they are readable by ITS developers without requiring a deep understanding of computational or cognitive sciences. Automatically created behavior graphs should be editable with existing authoring tools to facilitate necessary manual annotation and modifications. Ideally, ABGG algorithms should create concise graphs without losing other desirable characteristics. This may involve collapsing redundant paths and even pruning spurious or infrequent edges.

The conciseness of a graph can be measured using the number of nodes and edges in the graph. Our primary readability metric, *Compression Ratio* measures the rate at which an algorithm is able to reduce behavior events into behavior states (i.e. nodes) by finding similarities between events.

3.1.3 Complete

In order to minimize author effort, generated behaviors graphs should be as complete for creating an ETT as possible. As a minimal criterion, at least one valid path to the final solution should be included[♦]. Additionally, complete behaviors graphs are annotated with all the expected inputs by the learner. We use the *Rate of Unseen Events* in held out demonstrations as the primary metric to measure the completeness of our automatically generated behavior graphs.

3.1.4 Accurate

Behavior graphs should be error free. This includes being able to accurately capture the correct and incorrect events by learners depending on the current solution state. Edge accuracy measures the percentage of Correct & Incorrect edges that were accurately generated by the algorithm. *Error Rate* is a frequency weighted combination of edge accuracy that measures the fraction of learner events that will be inaccurately classified by the automatically generated behavior graph. We use the error rate of an automatically generate behavior graph on held out demonstrations as the primary accuracy metric.

3.1.5 Robust

One of the reasons for the success of expertly crafted ETTs is the ability to use them with a wide range of learners under different deployment conditions. Automatically generated behavior graphs should retain this characteristic; e.g., by identifying alternate paths and unordered groups. It is not unforeseeable that the use of a data-driven approach could contribute to creating behavior graphs that are more robust than those authored by a human expert.

Branching factor is the average number of data values available at each UI element. A large branching factor indicates the capability to process a large variety of learner inputs at each state. Also, the number and size of unordered groups is indicative of flexibility a graph affords to learners to explore the solution paths of a problem.

Note that readability and robustness are complementary characteristics of a behavior graph. For example, a highly complex behavior graph may be very robust but may not be very readable.

3.2 ABGG Algorithms

We use four algorithms, introduced in our previous work [4], to generate behavior graphs using multiple solution traces of a problem. The first algorithm (*Algorithm 1*) generates interaction networks by sequentially collapsing identical events in solution traces into a shared node and creating a branch whenever two different events are found. Interaction networks have been used in prior work [10][15].

Algorithm 2 uses a heuristic alignment technique [3] to align similar events across multiple solution traces. The alignment is used to obtain a sequence of traversal through the problem's steps. Furthermore, this algorithm is able to use the positional entropy of a sequence of elements while obtaining the optimal sequence to identify unordered groups.

Similar to the above algorithm, *Algorithm 3* finds the optimal sequence between aligned events. However, this algorithm uses the Center Star Algorithm [17] to align the multiple solution traces instead of the heuristic used by *Algorithm 2*. The Center Star Algorithm is a foundational algorithm used for aligning more than two sequences of symbols. It is particularly suited for our application because it is polynomial time in computational complexity and it does not make any assumptions about the space and relationship of symbols comprising the sequence.

First order transition matrix computed from solution traces can be used to represent a directed graph. *Algorithm 4* considers ABGG as the process of finding multiple paths in a directed graph. Specifically, the longest (non-repeating) path in this directed graph represents the most likely path through the solution steps. Since, the problem of finding longest paths in general graphs is known to be NP-hard, we employ a combination of bounded longest path finding and an algorithm for finding multiple shortest paths [18] in a transformed transition matrix to obtain a number of different paths through the directed graph. These paths are merged to construct a behavior graph similar to the process of constructing an interaction network.

Algorithm 2, 3 and 4 assume that if two or more events within a trace were generated by the same UI element, the latter event corresponds to a correction of the data value input at the former events. In this case, we refer to the former events as *retracted events* and data values entered at these events are assumed to be incorrect values. Using this assumption, these three algorithms are able to automatically generate incorrect paths in behavior graphs unlike *Algorithm 1*. This assumption is not applied to *Algorithm 1* to compare our work against prior work [11] on extracting tutor models from multiple demonstrations.

3.3 Discussion

Table 1 characterizes the four algorithms described above based on their capabilities. Incremental addition of demonstrations to generate interaction networks does not identify incorrect input data values. However, using the assumption about retracted events, the other three algorithms are able to identify incorrect inputs. Johnson et al. [15] used a similar assumption in their work on reducing the visual complexity of interaction networks. We notice that the *Algorithms 2 and 3* are complementary in terms of their ability to find alternate paths and unordered groups. *Algorithm 4* on the other hand offers both of these abilities.

Table 1. Comparison of Algorithm Capabilities

Capability▼ Algorithm▶	**1**	**2**	**3**	**4**
Identifies incorrect answers	N	Y	Y	Y
Generates alternate paths	N	N	Y	Y
Finds unordered groups	N	Y	N	Y
Generalizes beyond training demonstrations	N	Y	Y	Y
Guarantees all training demnstrs. will pass	Y	N	N	N
Finds atleast one path to final solution♦	Y	Y	Y	N
Discovers new/unseen data values	N	N	N	N

None of the algorithms discussed in this paper are capable of discovering unseen inputs beyond those seen in the solution traces. This type of generative ability is particularly useful for learning tasks, such as language learning, where a large number of different inputs may be expected from the learners. In our ongoing work, we use a number of heuristics [7] as well as grammar induction techniques [6] to generate unseen inputs for certain nodes in the behavior graphs.

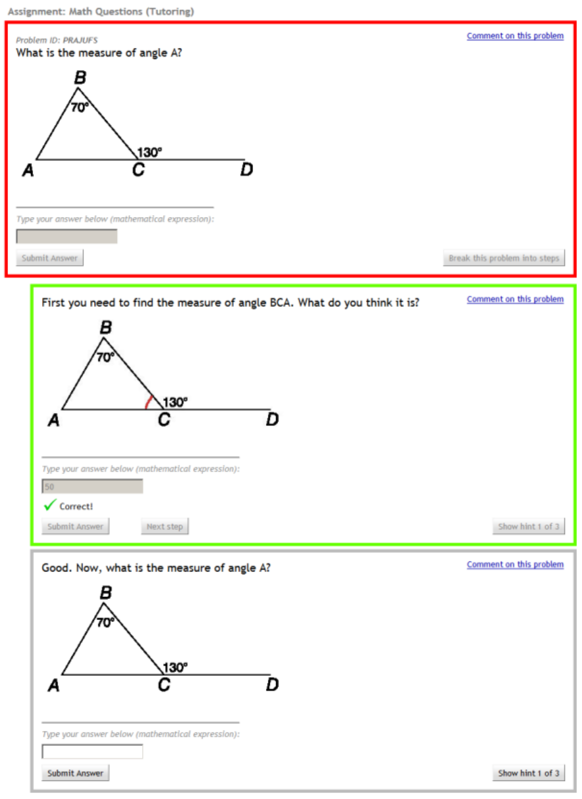
4. DATASETS

We use three datasets, accessed via DataShop[1] [19], to study the cross-domain applicability of ABGG algorithms. These datasets were filtered to use only problems that had six or more traces and had at least two UI elements. Also, we eliminated all events, such as help requests, that did not correspond to user input at a solution step. In this way, the datasets were transformed into solution traces. As discussed in Kumar et al. [4], a solution trace/demonstration comprises of a sequence of user interface (UI) events. Each event is represented as a 2-tuple $e = (u, d)$ that includes an identifier u of the UI element and data d associated with the event. A UI element may be visited any number of times within a trace. In general, data can include one or more attributes of the event such as the event type, user input, event duration, etc. In this paper, we assume single data attribute events where the data captures the learner input at the UI element.

[1] PSLC DataShop is available at http://pslcdatashop.org

Table 2. Problems & Traces for the three learning domains

	Math.	**Physics**	**French**
#Problems	1013	497	71
Max. #Unique Elements	33	62	10
Avg. #Unique Elements	4.6	9.7	2.5
Avg. #Training Traces	76.0	26.6	12.1
Avg. #Heldout Traces	38.0	13.3	6.1
Avg. #Events Per Trace	5.3	8.9	4.7


Assignment: Math Questions (Tutoring)

Problem ID: PRAJUFS

What is the measure of angle A?

Type your answer below (mathematical expression):

First you need to find the measure of angle BCA. What do you think it is?

Type your answer below (mathematical expression):

50

Correct!

Good. Now, what is the measure of angle A?

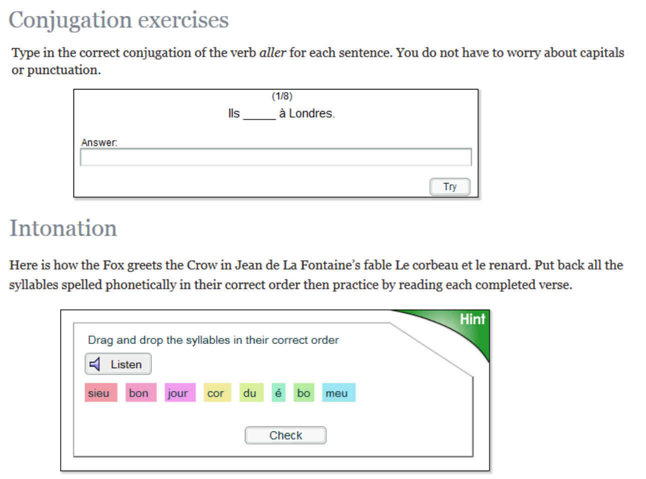
Type your answer below (mathematical expression):

Figure 1. Example Math Problem from *Assistments*
Source: www.assistments.org, April 2014

Table 2 provides some statistics about the problem and traces for each of learning domains used in this work. The Mathematics traces were derived from three *Assistments* [20] datasets. *Assistments* is a web-based learning platform, developed by Worcester Polytechnic Institute (WPI), that includes a Mathematics intelligent tutoring system for middle & high school grades. Figure 1 shows an example math problem from the *Assistments* system. Together, these datasets are the largest of the three domains we use. Prior to filtering, these dataset comprised a total of 683,197 traces and 1,905,672 events from 3,140 problems. For our experiments, we treat the three datasets to be independent of each other to account for change in UI designs of the problems common to the three datasets.

We used 10 (out of 20) of the largest datasets released under the *Andes2* project [22] to build the collection of Physics problems and traces. *Andes2* is an intelligent tutoring system that includes pedagogical content for a two-semester long college and advanced high-school level Physics course. These ten datasets are based on logs from several semesters of use of the *Andes2* system at the United States Naval Academy. Prior to filtering, these dataset comprised a total of 81,173 traces and 1,162,581 events from 2,187 different problems. Note that, as is case with the Math dataset, we treat the ten *Andes2* datasets independently. Note that, unlike typical domain independent example-tracing based tutor, the *Andes2* systems uses a model-tracing approach for tracking learner's solution of a problem and to provide feedback. The domain knowledge dependent model tracer is able to match highly inflected learner inputs (e.g. variable names) to its solution graph. Despite this difference in tutoring approach used by the *Andes2* system, we decided to include this domain in our experiments to study the performance of our algorithms on such solution traces.

Finally, the French traces are based on two dataset from the "French Course" project on DataShop. These datasets were collected from logs of student's use of the "French Online" course hosted by the Open Learning Initiative (OLI) [22] at Carnegie Mellon University. Figure 2 shows steps from couple of example problems from this course. These datasets comprised a total of 37,439 traces and 253,744 events from 1,246 different problems. Note that a significantly larger fraction of French problems were eliminated due to the filtering criterion compared to Mathematics or Physics.


Conjugation exercises

Type in the correct conjugation of the verb *aller* for each sentence. You do not have to worry about capitals or punctuation.

(1/8)

Ils _____ à Londres.

Answer:

Intonation

Here is how the Fox greets the Crow in Jean de La Fontaine's fable Le corbeau et le renard. Put back all the syllables spelled phonetically in their correct order then practice by reading each completed verse.

Drag and drop the syllables in their correct order

Figure 2. Example Steps from Problem from the French Online Course *Source: oli.cmu.edu, April 2014*

The datasets used in our experiments contain solution traces. Traces are paths through an existing behavior graph, unlike behavior demonstrations which are unconstrained by existing tutor models. In addition to the fact that these are the only available large scale collection of solution paths, we use these datasets in our experiments because these traces have been

Table 3. Averaged Metrics for the Graphs Generated by ABGG Algorithms
*indicates significant ($p < 0.05$) difference with the other algorithms (within the same dataset)

	Mathematics (*Assistments*)				Physics (*Andes2*)				French (OLI)			
Algorithm ▶	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
#Nodes	79.2	**5.4***	6.0*	6.6*	147.8	**7.9***	11.5*	11.7*	25.6	**3.8***	4.5*	4.5*
#Correct Edges	148.0	**12.9***	18.3*	17.5*	182.2	43.5*	76.4	**34.5***	37.2	**6.9**	9.8	9.5
#Incorrect Edges		23.9	33.5	**19.5***		35.1	53.0	**13.4***		**4.2**	11.0	8.0
Compression Ratio	6.7	**76.8***	66.8	60.2	2.3	**31.6***	21.9	21.7	2.2	**14.6**	12.8	12.8
% Accurate Correct Edges	39.1	41.9	42.5*	**44.1***	61.4	80.2*	58.9	**80.8***	22.5	27.7*	26.9*	**29.8***
% Accurate Incorrect Edges		**99.9***	97.2	99.5*		**92.5***	67.3	85.5		**97.8***	86.1	87.2
Training Error Rate	51.4	25.4	17.7*	**17.5***	33.6	**17.2***	25.8	24.3	75.2	56.1	**22.3***	25.3*
Heldout Error Rate	42.8	23.5	16.1*	**15.7***	29.1	**25.5***	33.3	30.8	45.3	35.9	19.9*	**18.5***
% Training Unseen Events	**0.0***	10.7	2.2	6.8	**0.0***	14.1	12.2	24.6	**0.0***	13.4	5.2	4.5
% Heldout Unseen Events	**10.2***	19.1	11.5*	13.9	**35.9***	41.7	38.4*	42.6	**31.7***	40.7	34.4*	34.3*
Branching Factor	2.2	10.9	**12.6***	8.5	1.5	**13.4***	12.9*	6.0	1.6	6.7*	**9.4***	7.8*
#Groups		**0.5***		0.0		0.8		**1.4***		**0.3***		0.1
Avg. Group Size		**1.9***		0.0		**2.0**		**2.0**		**0.6***		0.3
% Group Coverage		**31.8***		0.5		27.2		**30.6***		**15.4***		6.1

collected from a large set of real users. They contain realistic variations in learner inputs similar to demonstrations.

5. EXPERIMENTS

We use a three-fold cross validation design that splits the available traces into three different training and held out sets. The readability metrics (i.e. number of nodes, number of edges and compression ratio) as well as the robustness metrics (branching factor, number of unordered groups, average group size and coverage of graph within groups) are reported on the behavior graphs generated by the algorithms. On the other hand, some accuracy metrics such as the accuracy of correct and incorrect edges are measured on generated graphs whereas others such as error rate are measured on event sequences which could be the training traces; i.e., sequences used to generate the graphs, or held out traces. Similarly, our completeness metrics, i.e. the rate of unseen events in a sequence, can be measured on both training as well as held out traces. Note that the metrics computed on training traces used to generate the graphs may not accurately indicate the performance of an algorithm due to over-fitting. This is the motivation for choosing the cross validation based experimental design.

5.1 Results

Table 3 shows our results along 14 metrics for each of the four algorithms applied to the three learning domains under consideration. Reported metrics are averaged over three cross validation splits as well as over all the problems for each domain. The metrics are organized by the four desirable characteristics discussed earlier. Primary metric for each characteristic is highlighted.

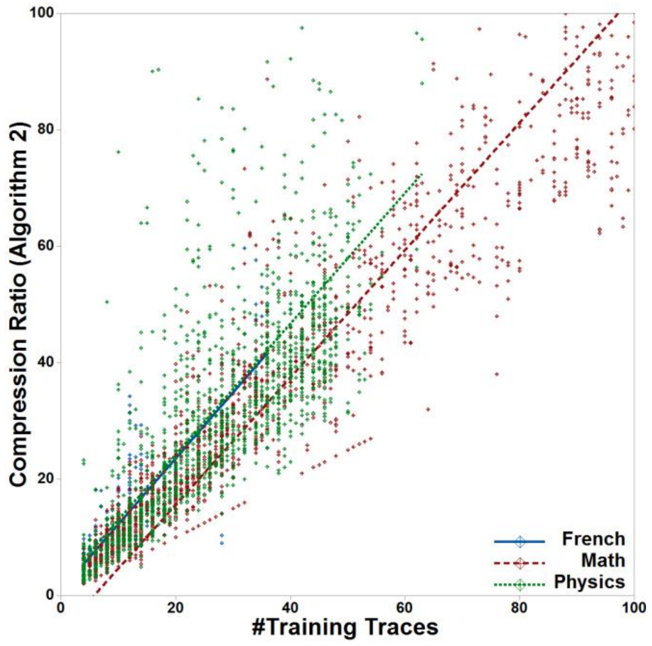


Figure 3. Compression Ratio of *Algorithm 2*

5.1.1 Mathematics

As expected, the interaction networks comprise of a large number of nodes and edges that lead them to have significantly smaller compression ratio. *Algorithm 2* (Heuristic Alignment) outperforms all other algorithms on three of the readability metrics. On the other hand, *Algorithm 4* (Path Pruning) significantly outperforms the other algorithms on three of the accuracy metrics for this dataset and is not significantly worse on the fourth metric. Because of their lossless nature, *Algorithm 1* (Interaction Network) performs the best on Completeness metrics (% unseen events). However, it is not significantly better than *Algorithm 3* (Center-Star Alignment). We find evidence of over-fitting of the algorithms to training traces on this metric as indicated by the approximately 9% higher rate of unseen events for held out traces for all the algorithms. *Algorithm 3* significantly outperforms the other algorithms on the primary robustness metric (Branching Factor) for this domain. *Algorithm 2* is better than *Algorithm 4* for the metrics based on unordered groups.

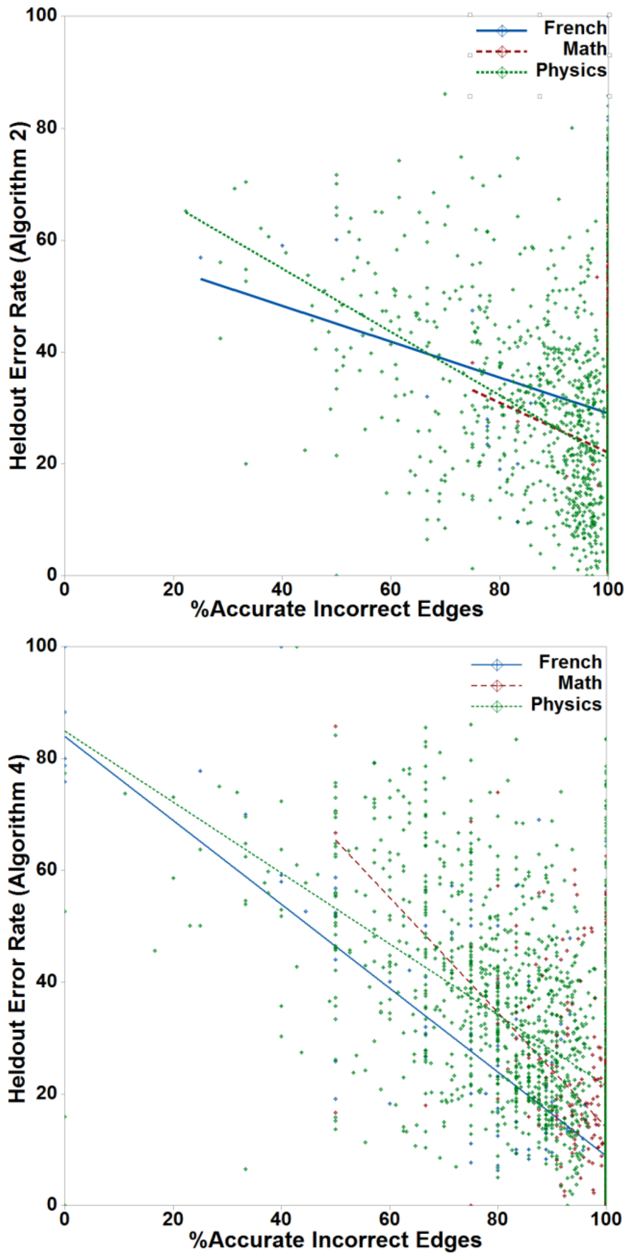


Figure 4. Heldout Error Rate of *Algorithms 2 and 4*

5.1.2 Physics

On the primary readability metric (Compression Ratio), *Algorithm 2* outperforms the others on the Physics dataset as was the case with Mathematics. This is consistent with prior conclusion [4] on the use of *Algorithm 2* for readability. We note that the Physics dataset has significantly lower compression ratio than the previous dataset. Figure 3 shows a scatter plot and domain-specific regression fits for the compression ratio of *Algorithm 2* for different problems with different number of training traces and UI elements. We see that for equivalent number of training traces, the compression ratio for Physics is actually slightly better than Mathematics. However, as we know from Table 2, fewer training traces are available for the Physics problems on average.

On the primary accuracy metric, we find that *Algorithm 2* works best for Physics unlike the case with the Mathematics domain. We can note that the *Algorithm 2* is significantly better on the accuracy of incorrect edges. Figure 4 shows the relationship between the error rate in heldout traces and the accuracy of incorrect edges. We also see that the percentage of unseen events in heldout traces is significantly higher for Physics. The lower incorrect edge accuracy and higher percentage of unseen events can be attributed to the differences in the tutoring approach underlying the *Andes2* system which uses domain-specific knowledge to match a large variety of inputs from the learner at each step of the solution. Because of this, *Andes2* elicits significantly diverse (& hence novel) inputs across traces. *Algorithms 2 and 3* are not significantly different in terms of the primary robustness metric.

5.1.3 French

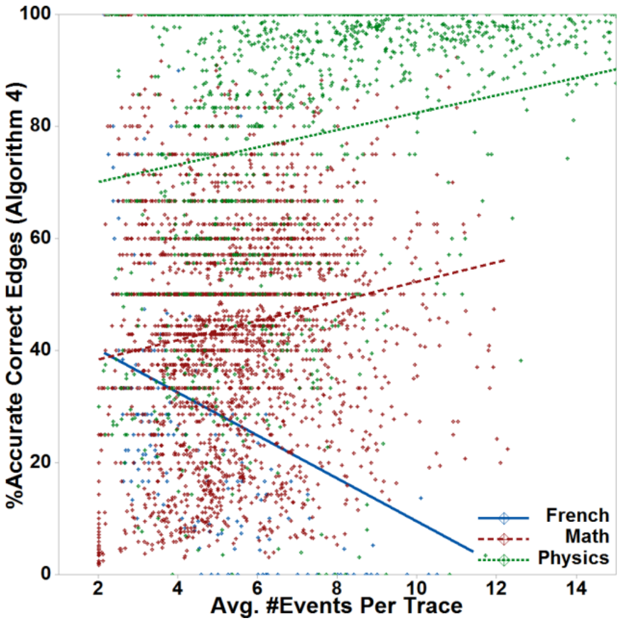


Figure 5. Accuracy of Correct Edges for *Algorithm 4*

The results for our non-STEM domain are largely consistent with the Mathematics domain. This may be attributed to the similarities of the underlying tutoring approach for the *Assistments* system and the French Online course which has been developed using the Cognitive Tutor Authoring Tools (CTAT) [2]. However, we can notice two key differences. First, the accuracy of correct edges for this domain is significantly lower. Because the French Online Course is deployed on an publicly accessible platform, its likely that a large number of the solution traces were generated by beginners as well as non-serious users leading to the dataset containing many incomplete solution traces containing no correct answers. This is evidenced in Figure 5 as we see that correct edge accuracy dramatically degrades for long traces which is contrary to the case with the other two domains.

Second, we expect the branching factor to be higher for a language learning domain, due to the high degree of linguistic variation in learner inputs. The results in Table 3 do not indicate this. However, Figure 6 verifies this intuition. Branching factor for the French behavior graphs is higher than those for the STEM domain for problems that have 10 or more traces.

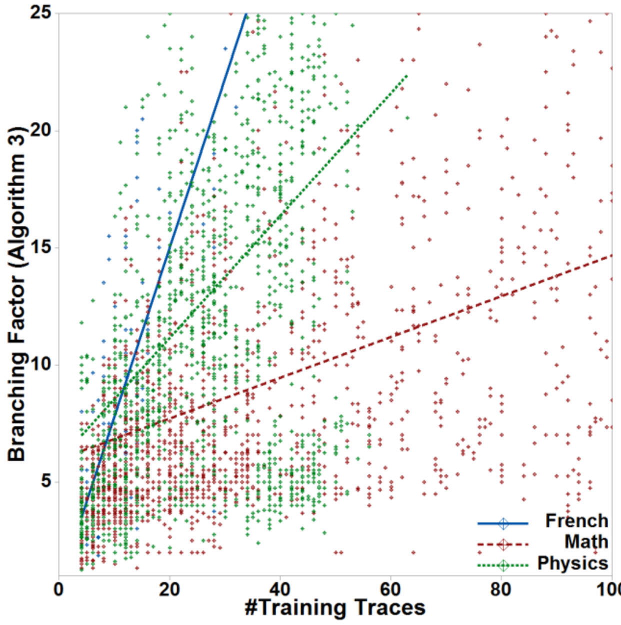


Figure 6. Branching Factor of *Algorithm 3*

5.1.4 Automatically Generated Behavior Graphs

Figures 7, 8 and 9 showcase several qualitative characteristics of automatically generated behavior graphs (truncated to fit) for the problems in the three datasets used in this work. We use the following visual convention: Circular nodes represent states and are labeled with identifiers *u* of the corresponding UI element. Edges are labeled with the data values *d*. Correct edges are labeled with green rectangles and incorrect edges are labeled with red rectangles. Unordered groups are shown using blue containers.

Figure 7 shows graphs generated by two different algorithms for the same Mathematics problem in the *Assistments* dataset using 241 solution traces by learners. The graph generated by *Algorithm 1* is dense and hardly readable due to the large number of nodes and edges in this graph. Also, as discussed in Section 3, this algorithm is unable to identify incorrect paths. Contrary to that, the graph in Figure 7b is composed of only 6 nodes. The various paths taken by learners are compressed into 46 correct and 39 incorrect edges. We can notice that not all paths are accurate. However, the accurate paths are more frequent, as indicated by the thicker arcs associated with the edge. In our ongoing work, we are extending these algorithms to use this frequency attribute to eliminate inaccurate paths (either automatically, or by providing additional controls to model developers in authoring tools).

A behavior graph from the Physics dataset is shown in Figure 8. As discussed earlier, the large variation in learner input at each state is depicted in the edge labels of this graph. We notice that for the last state (s6) which corresponds to the learners filling in the answer to a problem, many minor variations of the correct answer are accurately captured. Due to the domain independent nature of our algorithms, these answers are treated as different string. Integration of domain knowledge can lead to further compression of these answers into a single path.

The linguistic variation in the inputs to a problem in the French dataset is also noticeable in the two graphs for the same problem in Figure 9. We can see the several wrong answers are marked as correct answers (and vice versa), although the frequency-based edge notation identifies the correct answer as was the case in Figure 7b. In this problem, learners are asked to listen to an audio file and type in the French word they hear. Learners are allowed to go back and forth between these two steps. The first step has no wrong answer. We notice that our assumption to consider retracted events as incorrect fails in this case.

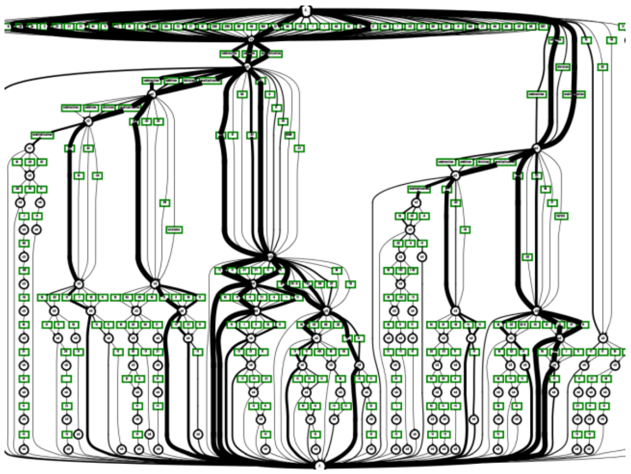


Figure 7a. Behavior Graph: Mathematics, *Algorithm 1*

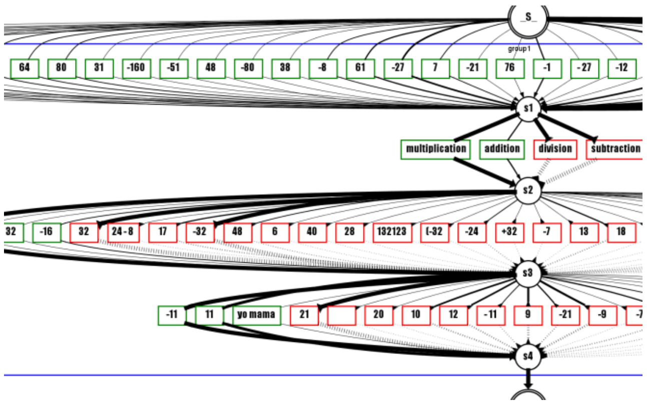


Figure 7b. Behavior Graph: Mathematics, *Algorithm 2*

It is particularly interesting to note the differences in the way *Algorithm 2* and *Algorithm 4* encode robustness into the learnt tutor model. While *Algorithm 2* identifies an unordered group containing the *listen* and *answer* nodes which allows learners to traverse these nodes in any order, *Algorithm 4* identifies that the *listen* step is optional and create two different way to reach the *answer* step based on the solution behaviors exhibited by learners in the traces.

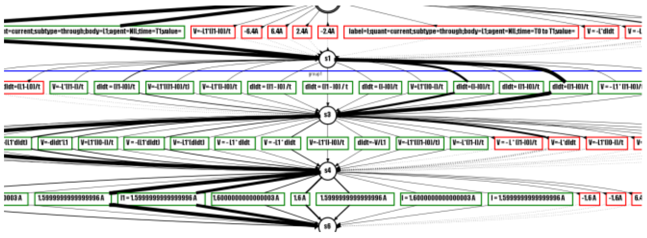


Figure 8. Behavior Graph: Physics, *Algorithm 2*

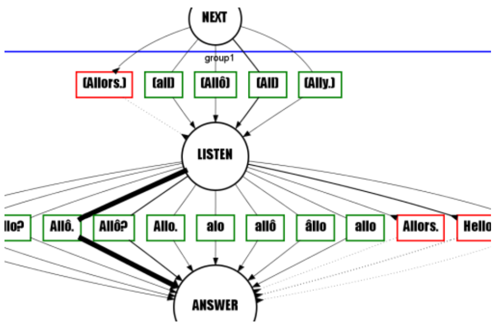


Figure 9a. Behavior Graph: French, *Algorithm 2*

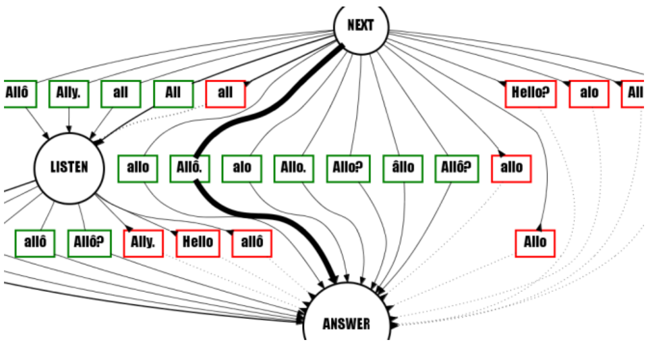


Figure 9b. Behavior Graph: French, *Algorithm 4*

6. CONCLUSIONS

In this paper, we have shared results from an empirical analysis of application of ABGG algorithms to three different learning domains. Several similarities and differences between the performances of four algorithms on problems from these three domains were discussed in the previous section.

We find that the accuracy of these algorithms suffers when they are applied to solution traces collected from a tutoring system that uses domain knowledge to process a large variety of inputs from learners. While in our previous work [4], we have recommended the use of *Algorithm 2* as the default ABGG algorithm for use within authoring tools, we find that for language learning domains, *Algorithm 4* may be preferable since it is the most accurate on the French dataset and not significantly worse than the other algorithms on the other primary metrics.

We identified multiple potential improvements to the ABGG algorithms based on these analyses. There are several domain specific nuances to the UI elements that comprise the problems in each domain. For example, in the French domain, we found steps that do not have any wrong answer. For broad use, ABGG algorithms should identify these UI elements and selectively apply the powerful assumption about retracted events. Furthermore, the algorithms can exploit additional features computed from across the multiple traces, such as the frequency of a data value at a node, to improve the accuracy of the automatically generated behavior graphs.

Finally, this paper extends our recent work on use of multiple behavior demonstrations to automatically generate tutor models using ABGG algorithms. While these algorithms can be improved in specific ways discussed above, we find evidence for their applicability to multiple domains.

ACKNOWLEDGEMENTS

This research was funded by the US Office of Naval Research (ONR) contract N00014-12-C-0535.

7. REFERENCES

[1] Johnson, W. L., and Valente, A. 2008. Collaborative authoring of serious games for language and culture. In *Proceedings of SimTecT* (March 2008).

[2] Aleven, V., McLaren, B. M., Sewall, J., and Koedinger. K. R. 2006. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (ITS'06), Ikeda, M., Ashley, K. D., and Chan, T.W. (Eds.). Springer-Verlag, Berlin, Heidelberg, 61-70.

[3] Kumar, R., Roy, M.E, Roberts, R.B., and Makhoul, J.I. 2014. Towards Automatically Building Tutor Models Using Multiple Behavior Demonstrations. In *Proceedings of 12th Intl. Conf. on Intelligent Tutoring Systems* (ITS 2014), Honolulu, HI.

[4] Kumar, R., Roy, M.E, Roberts, R.B., and Makhoul, J.I. 2014. Comparison of Algorithms for Automatically Building Example-Tracing Tutor Models. In *Proceedings of 7th Intl. Conf. on Educational Data Mining* (EDM 2014), Honolulu, HI.

[5] Aleven, V., Mclaren, B. M., Sewall, J., and Koedinger. K. R. 2009. A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *Int. J. Artif. Intell. Ed.* 19, 2 (April 2009), 105-154.

[6] Kumar, R., Sagae, A., and Johnson, W. L. 2009. Evaluating an Authoring Tool for Mini-Dialogs. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education*, Dimitrova, V., Mizoguchi, R., du Boulay, B., and Graesser, A. (Eds.). IOS Press, Amsterdam, The Netherlands, The Netherlands, 647-649.

[7] Kumar, R., Roy, M.E, Pattison-Gordon, E. and Roberts, R.B 2014. General Purpose ITS Development Tools. In *Proceedings of Workshop on Intelligent Tutoring System Authoring Tools, 12th Intl. Conf. on Intelligent Tutoring Systems* (ITS 2014), Honolulu, HI.

[8] Question Generation Workshops. 2008-2011. http://www.questiongeneration.org/

[9] Barnes, T. and Stamper, J. 2008. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (ITS '08). Woolf, B. P., Aimeur, E., Nkambou, R., and Lajoie , S. (Eds.). Springer-Verlag, Berlin, Heidelberg, 373-382.

[10] Eagle, M., Johnson, J., and Barnes, T., 2012. Interaction Networks: Generating High Level Hints Based on Network Community Clusterings, In *Proceedings of the 5th International Conference on Educational Data Mining* (EDM 2012). Yacef, K., Zaïane, O., Hershkovitz, H., Yudelson, M., and Stamper, J. (Eds.). 164-167

[11] McLaren, B.M., Koedinger, K.R., Schneider, M., Harrer, A., and Bollen, L. 2004. Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files. In *Proceedings of the Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, 7th International Conference on Intelligent Tutoring Systems* (ITS 2004). August 2004

[12] Pavlik, P.I., Cen, H., and Koedinger, K.R. 2009. Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models, In *Proceedings of the 2nd International Conference on Educational Data Mining* (EDM 2009). Barnes, T., Desmarais, M., Romero, C., Ventura, S. (Eds.). 121-130

[13] Koedinger, K.R., Mclaughlin E.A., and Stamper, J.C. 2012. Automated student model improvement, In *Proceedings of the 5th International Conference on Educational Data Mining* (EDM 2012). Yacef, K., Zaïane, O., Hershkovitz, H., Yudelson, M., and Stamper, J. (Eds.). 17-24

[14] Sudol, L.A, Rivers, K., and Harris, T.K. 2012. Calculating Probabilistic Distance to Solution in a Complex Problem Solving Domain, In *Proceedings of the 5th International Conference on Educational Data Mining* (EDM 2012). Yacef, K., Zaïane, O., Hershkovitz, H., Yudelson, M., and Stamper, J. (Eds.). 144-147

[15] Johnson, M., Eagle, M., Stamper, J., and Barnes, T. 2013. An Algorithm for Reducing the Complexity of Interaction Networks, In *Proceedings of the 6thInternational Conference on Educational Data Mining*, (EDM 2013). D'Mello, S. K., Calvo, R. A., Olney, A. (Eds.). 248-251

[16] Ritter, R., Harris, T.K, Nixon, T., Dickison, D., Murray, R.C., and Towle, B. 2009. Reducing the Knowledge Tracing Space, In *Proceedings of the 2ndInternational Conference on Educational Data Mining* (EDM 2009). Barnes, T., Desmarais, M., Romero, C., Ventura, S. (Eds.). 151-160

[17] Gusfield, D. 1997. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, New York.

[18] Yen, J. Y. 1971. Finding the K Shortest Loopless Paths in a Network. *Management Science* 17(11). 712-716

[19] Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., and Stamper, J. 2010. A Data Repository for the EDM community: The PSLC DataShop. In *Handbook of Educational Data Mining*. Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.). Boca Raton, FL: CRC Press

[20] Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N. T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T. E., Upalekar. R, Walonoski, J.A., Macasek. M.A. and Rasmussen, K. P. 2005. The Assistment project: Blending assessment and assisting. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) IOS Press. 555-562.

[21] VanLehn, K., Lynch, C., Schulze, K. Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. 2005. The Andes physics tutoring system: Lessons Learned. In *International Journal of Artificial Intelligence and Education*, 15 (3), 1-47

[22] Strader, R. and Thille, C. 2012. The Open Learning Initiative: Enacting Instruction Online. In *Game Changers: Education and Information Technologies*. Oblinger, D.G. (Ed.) Educause. 201-213.