Graph-based Educational Data Mining (G-EDM 2015)

Collin F. Lynch
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
cflynch@ncsu.edu

Dr. Tiffany Barnes
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
tmbarnes@ncsu.edu

Dr. Jennifer Albert
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
jlsharp@ncsu.edu

Michael Eagle
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
mjeagle@ncsu.edu

1. INTRODUCTION

Fundamentally, a graph is a simple concept. At a basic level a graph is a set of relationships $\{e(n_0,n_2),e(n_0,n_j),...,e(n_{j-1},n_j)\}$ between elements. This simple concept, however, has afforded the development of a complex theory of graphs [1] and rich algorithms for combinatorics [7] and clustering [4]. This has, in turn, made graphs a fundamental part of educational data mining.

Many types of data can be naturally represented as graphs such as social network data, user-system interaction logs, argument diagrams, logical proofs, and forum discussions. Such data has grown exponentially in volume as courses have moved online and educational technology has been incorporated into the traditional classroom. Analyzing it can help to answer a range of important questions such as:

- What path(s) do high-performing students take through online educational materials?
- What social networks can foster or inhibit learning?
- Do users of online learning tools behave as the system designers expect?
- What diagnostic substructures are commonly found in student-produced diagrams?
- Can we use prior student data to identify students' solution plan, if any?
- Can we use prior student data to provide meaningful hints in complex domains?
- Can we identify students who are particularly helpful based upon their social interactions?

Thus, graphs are simple in concept, general in structure, and have wide applications for Educational Data Mining (EDM). Despite the importance of graphs to data mining and data analysis there exists no strong community of researchers focused on Graph-Based Educational Data Mining. Such a community is important to foster useful interactions, share tools and techniques, and to explore common problems.

2. GEDM 2014

This is the second workshop on Graph-Based Educational Data Mining. The first was held in conjunction with EDM 2014 in London [17]. The focus of that workshop was on seeding an initial community of researchers, and on identifying shared problems, and avenues for research. The papers presented covered a range of topics including unique visualizations [13], social capital in educational networks [8], graph mining [19, 11], and tutor construction [9].

The group discussion sections at that workshop focused on the distinct uses of graph data. Some of the work presented focused on student-produced graphs as solution representations (e.g. [14, 3]) while others focused more on the use of graphs for large-scale analysis to support instructors or administrators (e.g. [18, 13]). These differing uses motivate different analytical techniques and, as participants noted, change our underlying assumptions about the graph structures in important ways.

3. GEDM 2015

Our goal in this second workshop was to build upon this nascent community structure and to explore the following questions:

1. What common goals exist for graph analysis in EDM?
2. What shared resources such as tools and repositories are required to support the community?
3. How do the structures of the graphs and the analytical methods change with the applications?

The papers that we include here fall into four broad categories: interaction, induction, assessment, and MOOCs.

Work by Poulovassilis et al. [15] and Lynch et al. [12] focuses on analyzing user-system interactions in state based learning environments. Poulovassilis et al. focuses on the analyses of individual users' solution paths and presents a novel mechanism to query solution paths and identify general solution strategies. Lynch et al. by contrast, examined user-system interactions from existing model-based tutors to examine the impact of specific design decisions on student performance.

Price & Barnes [16] and Hicks et al. [6] focus on applying these same analyses in the open-ended domain of programming. Unlike more discrete tutoring domains where users enter single equations or select actions, programming tutors allow users to make drastic changes to their code on each step. This can pose challenges for data-driven methods as the student states are frequently unique and admit no easy single-step advice. Price and Barnes present a novel method for addressing the data sparsity problem by focusing on minimal-distance changes between users [16] while in related work Hicks et al. focuses on the use of path weighting to select actionable advice in a complex state space [6].

The goal in much of this work is to identify rules that can be used to characterize good and poor interactions or good and poor graphs. Xue at al. sought address this challenge in part via the automatic induction of graph rules for student-produced diagrams [22]. In their ongoing work they are applying evolutionary computation to the induction of Augmented Graph Grammars, a graph-based formalism for rules about graphs.

The work described by Leo-John et al. [10], Guerra [5] and Weber & Vas [21], takes a different tack and focuses not on graphs representing solutions or interactions but on relationships. Leo-John et al. present a novel approach for identifying closely-related word problems via semantic networks. This work is designed to support content developers and educators in examining a set of questions and in giving appropriate assignments. Guerra takes a similar approach to the assessment of users' conceptual changes when learning programming. He argues that the conceptual relationship graph affords a better mechanism for automatic assessment than individual component models. This approach is also taken up by Weber and Vas who present a toolkit for graph-based self-assessment that is designed to bring these conceptual structures under students' direct control.

And finally, Vigentini & Clayphan [20], and Brown et al. [2] focus on the unique problems posed by MOOCs. Vigentini and Clayphan present work on the use of graph-based metrics to assess students' on-line behaviors. Brown et al., by contrast, focus not on local behaviors but on social networks with the goal of identifying stable sub-communities of users and of assessing the impact of social relationships on users' class performance.

4. REFERENCES

[1] B. Bollobás. *Modern Graph Theory*. Springer Science+Business Media Inc. New York, New York, U.S.A., 1998.

[2] R. Brown, C. F. Lynch, Y. Wang, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, and D. McNamara. Communities of performance & communities of preference. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[3] R. Dekel and K. Gal. On-line plan recognition in exploratory learning environments. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[4] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[5] J. Guerra. Graph analysis of student model networks. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[6] A. Hicks, V. Catete, R. Zhi, Y. Dong, and T. Barnes. Bots: Selecting next-steps from player traces in a puzzle game. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[7] D. E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*, volume 4A. Addison-Wesley, 1st edition, 2011.

[8] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[9] R. Kumar. Cross-domain performance of automatic tutor modeling algorithms. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[10] R.-J. Leo-John, T. McTavish, and R. Passonneau. Semantic graphs for mathematics word problems based on mathematics terminology. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[11] C. F. Lynch. AGG: augmented graph grammars for complex heterogeneous data. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[12] C. F. Lynch, T. W. Price, M. Chi, and T. Barnes. Using the hint factory to analyze model-based tutoring systems. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[13] T. McTavish. Facilitating graph interpretation via interactive hierarchical edges. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[14] B. Mostafavi and T. Barnes. Evaluation of logic proof problem difficulty through student performance data. In S. G. Santos

and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[15] A. Poulovassilis, S. G. Santos, and M. Mavrikis. Graph-based modelling of students' interaction data from exploratory learning environments. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[16] T. Price and T. Barnes. An exploration of data-driven hint generation in an open-ended programming problem. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[17] S. G. Santos and O. C. Santos, editors. *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[18] V. Sheshadri, C. Lynch, and T. Barnes. Invis: An EDM tool for graphical rendering and analysis of student interaction data. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[19] K. Vaculík, L. Nezvalová, and L. Popelínsky. Graph mining and outlier detection meet logic proof tutoring. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[20] L. Vigentini and A. Clayphan. Exploring the function of discussion forums in moocs: comparing data mining and graph-based approaches. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[21] C. Weber and R. Vas. Studio: Ontology-based educational self-assessment. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[22] L. Xue, C. F. Lynch, and M. Chi. Graph grammar induction by genetic programming. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

Graph Grammar Induction via Evolutionary Computation

Linting Xue
Electrical Engineering Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
lxue3@ncsu.edu

Collin F.Lynch
Computer Science Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
cflynch@ncsu.edu

Min Chi
Computer Science Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
mchi@ncsu.edu

ABSTRACT

Augmented Graph Grammars provide a robust formalism for representing and evaluating graph structures. With the advent of robust graph libraries such as AGG, it has become possible to use graph grammars to analyze realistic data. Prior studies have shown that graph rules can be used to evaluate student work and to identify empirically-valid substructures using hand-authored rules. In this paper we describe proposed work on the automatic induction of graph grammars for student data using evolutionary computation via the pyEC system.

Keywords

Augmented Graph Grammars, Graph Data, Evolutionary Computation

1. INTRODUCTION

Graph Grammars are logical rule representations for graph structures. They can be designed to encode classes of suitable graphs to recognize complex sub-features. They were introduced by Rosenfeld and Pfaltz in 1969 as “Context-free web grammars” [1]. Since then Graph grammars have been applied to a wide range of areas, including pattern recognition [2, 3, 4]; visual programming languages [5]; biological development [6]; classification of chemical compounds [7, 8]; and social network analysis [9, 10, 11]. Simple graph grammars are, like string grammars, composed of a set of production rules that map from one structure to another. In this case the rules map from a simple subgraph, typically a single node or arc, to a more complex structure. As with string grammars the node and arc types are drawn from finite alphabets. Despite their utility, however, graph grammars are very difficult to construct. The data structures required are complex [5]. Moreover, development of suitable graph grammars generally requires considerable domain expertise. Most existing uses of graph grammars have relied on hand-authored rules.

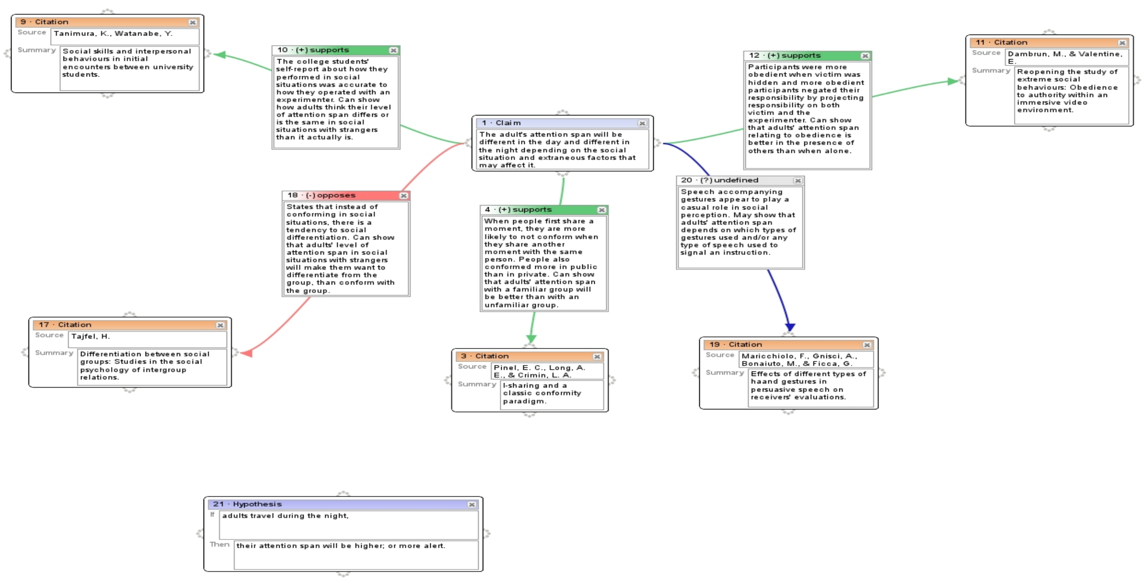
In this paper we describe our ongoing work on the automatic induction of Augmented Graph Grammars via Evolutionary Computation (EC). Our long-term goal in this work is to develop automated techniques that can extract empirically-valid graph rules which can, in turn, be used to classify student-produced argument diagrams and to provide the basis for automated student guidance and evaluation. This will build upon our prior on the evaluation of *a-priori* rules for student arguments. We will begin with background material on Augmented Graph Grammars and discuss prior work on grammar induction. We will then present an overview of our planned work.

2. AUGMENTED GRAPH GRAMMARS & ARGUMENT DIAGRAMS

Classical graph grammars are designed to deal with fixed graphs that are composed from a finite set of static node and arc types. Augmented graph grammars are an extension of simple graph grammars that allow for complex node and arc types, optional substructures, and complex rule expressions [12]. Rather than using a fixed alphabet of graph components they are defined by a complex ontology that allows for subsidiary types such as textual fields, access functions, and directional information. They can also be used to evaluate negated elements as well as quantified expressions. As such they are better suited to rich graph data such as user-system interaction logs and student-produced argument diagrams.

Augmented Graph Grammars have previously been used for the detection of empirically-valid substructures in student-produced argument diagrams [13, 14]. In that work *a-priori* rules were used to represent key discussion features and argumentative flaws. Argument diagrams are graphical argument representations that reify key features of arguments such as hypothesis statements, claims, and citations as nodes and the supporting, opposing, and informational relationships as arcs between them.

A sample diagram collected in that work is shown in Figure 1 The diagram includes a central research *claim* node, which has a single text field indicating the content of the research claim. A set of *citation* nodes are connected to the *claim* node via a set of *supporting, opposing* and *undefined* arcs colored with green, red and blue respectively. Each citation node contains two fields: one for the citation information, and the other for a summary of the cited work; each arc has a single text field explaining why the relationship holds. At



9 · Citation
Source: Tanimura, K., Watanabe, Y.
Summary: Social skills and interpersonal behaviours in initial encounters between university students.

10 · (+) supports
The college students' self-report about how they performed in social situations was accurate to how they operated with an experimenter. Can show how adults think their level of attention span differs or is the same in social situations with strangers than it actually is.

1 · Claim
The adult's attention span will be different in the day and different in the night depending on the social situation and extraneous factors that may affect it.

12 · (+) supports
Participants were more obedient when victim was hidden and more obedient participants negated their responsibility by projecting responsibility on both victim and the experimenter. Can show that adults' attention span relating to obedience is better in the presence of others than when alone.

11 · Citation
Source: Dambrun, M., & Valentine, E.
Summary: Reopening the study of extreme social behaviours: Obedience to authority within an immersive video environment.

18 · (-) opposes
States that instead of conforming in social situations, there is a tendency to social differentiation. Can show that adults' level of attention span in social situations with strangers will make them want to differentiate from the group, than conform with the group.

4 · (+) supports
When people first share a moment, they are more likely to not conform when they share another moment with the same person. People also conformed more in public than in private. Can show that adults' attention span with a familiar group will be better than with an unfamiliar group.

20 · (?) undefined
Speech accompanying gestures appear to play a casual role in social perception. May show that adults' attention span depends on which types of gestures used and/or any type of speech used to signal an instruction.

17 · Citation
Source: Tajfel, H.
Summary: Differentiation between social groups: Studies in the social psychology of intergroup relations.

3 · Citation
Source: Pinel, E. C., Long, A. E., & Crimin, L. A.
Summary: I-sharing and a classic conformity paradigm.

19 · Citation
Source: Maricchiolo, F., Gnisci, A., Bonaiuto, M., & Ficca, G.
Summary: Effects of different types of haand gestures in persuasive speech on receivers' evaluations.

21 · Hypothesis
If: adults travel during the night,
Then: their attention span will be higher; or more alert.

Figure 1: A student-produced LASAD argument diagram representing an introductory argument.

the bottom of the diagram, there is a single isolated *hypothesis* node that contains two text fields, one for a *conditional* or IF field, and the other for *conditional* or THEN field. We expect the induced graph grammars from a set of argument diagrams can be used to evaluate the student thesis work.

Figure 2 shows an *a-priori* rule that was defined as part of that work. This rule is designed to identify a subgraph where a single target node t is connected to two separate citation nodes a and b such that: a is connected to t via an opposing path; b is connected via a supporting path; and there exists no comparison arc between a and b. The rules in that study were implemented using AGG an augmented graph grammar library built in Python [12]. AGG matches the graphs using recursive stack-based algorithm. The code first matches the ground nodes at the top-level of the class (t, a, & b). It then tests for the recursive productions O, and S, before finally testing for the negated comparison arc c. This rule does not make use of the full range of potential capacity for Augmented Graph Grammars. However it is illustrative of the type of rules we plan to induce here, rules that generalize beyond basic types and draw on existing production classes but not, at least in the immediate term, use complex textual elements or functional features.

$$(ParedWcomp)$$

$$\begin{tikzcd} & \text{t} & \\ O \arrow[ur] & & S \arrow[ul] \\ \text{a} \arrow[u, dash] \arrow[r, phantom, "\vdash"] & \neg\, c & \text{b} \arrow[u, dash] \arrow[l, phantom, "\dashv"] \end{tikzcd}$$

$$\left\{ \begin{array}{c} t.Tpye = “claim''or“hypothesis'' \\ a.Type = “Citation'' \\ b.Type = “Citation'' \\ c.Type = “Comparison'' \end{array} \right\}$$

Figure 2: A simple augmented graph grammar rule that detects compared counterarguments. The rule shows a two citation nodes (a, & b) that have opposing relationships with a shared claim node (t) and do not have a comparison arc (c) drawn between them. The arcs S and O represent recursive supporting and opposing path.

3. GRAMMAR INDUCTION

Graph and relational data has grown increasingly prevalent and graph analysis algorithms have been applied in a wide range of domains from social network analysis [15] to bioinformatics [16]. Most of this work falls into one of two categories of algorithms: frequent subgraph matching, and graph compression.

A number of algorithms have been developed to discover frequent subgraphs. These include the gSpan algorithm developed by Yan and Han [17]; the AGM algorithm developed by Inokuchi et al [18]; and the FSG algorithm developed by Kuramochi and Karypis which is based on the previous Apriori algorithm [19]. They are based upon controlled graph walks coupled with indexing. While these algorithms are effective, particularly on smaller graphs, with low vertex degree they can also overfit simpler graph structures and they do not scale well to larger, denser graph data [20].

The SUBDUE system takes a greedy-compression approach to graph mining. SUBDUE searches for candidate subgraphs that can best compress the input graphs by replacing a candidate subgraph with a single vertex. Then nodes and arcs are added to the vertices to form new candidate subgraphs. The process is recursive and relies on the Minimum-Description-Length (MDL) principle to evaluate the candidates. SUBDUE has been applied successfully to extract structure from visual programming [5], web search [21], and analyzing user behaviors in games [22].

While these methods are successful they have practical and theoretical limitations. Both classes of approaches are limited to static graphs composed from a finite alphabet of node and arc types. The frequent subgraph approaches are based upon iterative graph walks and can be computationally expensive and are limited to finding exact matches. They do not generalize beyond the exact graphs matched nor do they allow for recursive typing. SUBDUE, by contrast is a greedy algorithm that finds the single most descriptive grammar and does not allow for weighted matches.

For our present purposes, however, our goal is to identify multiple heirarchical classes of the type shown in Figure 2 that can: generalize beyond exact node and arc types; can draw on recursive rule productions; and can be weighted based upon the graph quality. Moreover our long-term goal with this work is to explore graph rule induction mechanisms that can be expanded to include textual rules and complex constraints. For that reason we have elected to apply evolutionary computation. This is a general-purpose machine learning mechanism that can be tunes to explore a range of possible induction mechanisms.

4. METHODS

4.1 Evolutionary Computation

Evolutionary Computation (EC) is a general class of machine learning and optimization methods that are inspired by the process of Darwinian evolution through natural selection [23] such as Genetic Algorithms [24] or Genetic Programming [25]. EC algorithms begin with a population of randomly generated candidate solutions such as snippets of random code, strings representing a target function, or formal rules. Each of these solutions is ranked by a *fitness function* that is used to evaluate the quality of the individuals. These functions can be defined by absolute measures of success such as a suite of test cases, or by relative measures such as a competition between chess-playing systems.

Once the individuals have been ranked a new generation of individuals is produced through a combination of crossover and mutation operations. *Crossover* operations combine two or more parents to produce one or more candidate children. In Genetic Algorithms where the candidate solutions are represented as strings this can be accomplished by splitting two parents at a given index and then exchanging the substrings to produce novel children. In Genetic Programming the parents exchange blocks of code, functions, or subtrees. *Mutation* operations alter randomly-selected parts of a candidate solution by swapping out one symbol or instruction for another, adding new sub-solutions, or deleting components. This process of ranking and regeneration will iterate until a target performance threshold is reached or a maximum number of generations has passed.

EC methods are highly general algorithms that can be readily adapted to novel domains by selecting an appropriate solution representation and modification operations. Thus, in contrast to more specific methods such as SUBDUE, the EC algorithm allows us to tune the inductive bias of our search and to explore alternative ways of traversing the solution space. Therefore it is well suited to our present needs. This flexibility is costly, however, as EC is far more computationally expensive than more specialized algorithms, and applications of EC can require a great deal of tuning for each use. In the subsections below we will describe the fitness function and the operators that we will use in this work. For this work we will rely on *pyEC* a general purpose evolutionary computation engine that we have developed [26].

4.2 Dataset

Our initial analysis will be based upon a corpus of expert graded student produced argument diagrams and essays previously described in [13, 14]. That dataset was collected as part of a study on students' use of argument diagrams for writing that was conducted at the University of Pittsburgh in 2011. For that study we selected a set of students in an undergraduate-level course on Psychological Research Methods. As part of the course the students were tasked with planning and executing an empirical research study and then drafting a written report. The students were permitted to work individually or in teams. This report was structured as a standard empirical workshop paper. Prior to drafting the report the students were tasked with diagramming the argument that they planned to make using LASAD an online tool for argument diagramming and annotation.

Subsequent to this data collection process the diagrams and essays were graded by an experienced TA using a set of parallel grading rubrics. These rubrics focused on the quality of the arguments in the diagrams and essays and were used to demonstrate that the structure and quality of the diagrams can be used to predict the students' subsequent essay performance. These grades will be used as the weighting metric for the diagrams and will be correlated with performance as part of the fitness function we describe below. After completion of the data collection, grading, and testing phases and accounting for student dropout and incomplete assignments we collected 105 graded diagram-essay pairs 74 of which were authored by teams.

4.3 Solution Representation

For the purposes of our present experiments we will use a restricted solution representation that relies on a subset of the augmented graph grammar formalism exemplified by the rule shown in Figure 2. This will include only element types and recursive productions. In future work we plan to support the induction of more complex rules defined by multiple graph classes, novel productions, and expressions. However for the present study we will focus on the simple case of individual classes coupled with predefined productions.

4.4 Fitness Function

We plan to use the frequency correlation metric previously employed in [13, 14]. In that study the authors assessed the *empirical validity* of a set of *a-priori* diagram rules. The validity of each individual rule was assessed by testing the correlation between the frequency of the class in the existing graph and the graph grade. The strength of that correlation was estimated using Spearman's ρ a non-parametic measure of correlation [27]. In that work the authors demonstrated that the *a-priori* rule frequency was correlated with students' subsequent essay grades and showed that the frequencies could be used to predict students' future performance.

4.5 Mutation

Our mutation operator will draw on the predefined graph ontology to make atomic changes to an existing graph class. The change will be one of the following operations:

Change Node change an existing node's type.

Change Arc Change an existing arc's type or orientation.

Delete Node Delete a node and its associated arcs.

Delete Arc Delete an existing arc.

Add Node Add a novel node with a specified type.

Add Arc Add an arc between existing nodes or add with new nodes.

4.6 Crossover

By design the crossover operation should, like genetic crossover, be conservative. Two very similar parents should produce similar offspring. Crossover operations should therefore preserve good building blocks and sub-solutions or *introns* through random behavior [25]. Arbitrary graph alignment and crossover is a challenging problem that risks causing unsustainable changes on each iteration. We therefore treat graph crossover as a matrix problem.

For each pair of parent classes we will define a pair of diagonal matricies of the type illustrated in Figure 3. The letter indicies on the top and right indicate nodes while the numerical indicies internally indicate arcs, and the $\emptyset$ symbol indicates that no arc is present. The matricies are generated in a canonical order based upon the order in which the nodes were added to the class. Thus on each iteration of the crossover process the corresponding elements will obtain the same index. As a consequence good subsolutions will obtain the same location and will tend to be preserved over time.

Once a set of parent matricies has been generated we then generate two child matricies of the same size as the parents and then randomly select the node and arc members. In the example shown in figures 3 and 4 the parents have nodes $\{A,B,C,D\}$ and $\{E,F,G\}$ while the children have $\{E,B,G,D\}$ and $\{A,F,C\}$. Thus we align the nodes in canonical order and, for each node pair, we flip a coin to decide where they are copied. If one parent is larger than the other than any additional nodes, in this case D, will be copied to the larger child. We then perform a comparable exchange process for the arcs. Each arc or potential arc is defined uniquely in the matricies by its endpoints. We thus align the lists of arcs in a comparable manner and then decide randomly which arc, or empty arc, to copy. As with the nodes, extra arcs from the larger parent, in this case *3,5,* and one $\emptyset$ are copied directly into the larger of the two children.

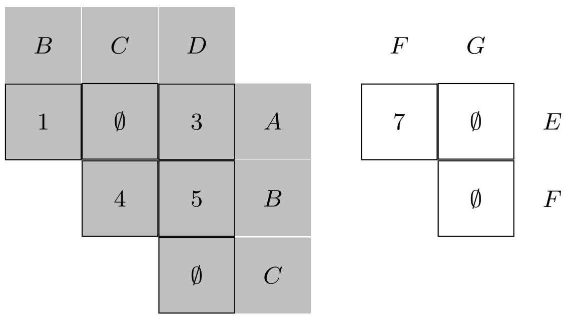


Figure 3: Canonical matricies for crossover parents.

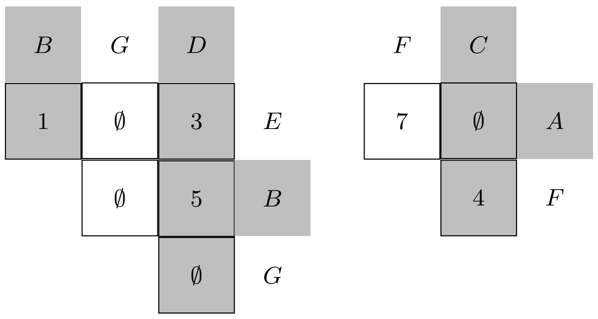


Figure 4: Canonical matricies for crossover children.

5. FUTURE WORK

In this paper we presented a method for the induction of augmented graph grammars through evolutionary computation. We are presently applying this work to the automatic induction of empirically-valid rules for student-produced argument diagrams. This work will serve to extend our prior efforts on the use of augmented graph grammars for student grading and feedback. This work represents an improvement over prior graph grammar induction algorithms which are limited to classical graph grammars and greedy extraction. This work also represents an extension for evolutionary computation by shifting it into a new domain. As part of this work we also plan to explore additional extensions to the standard evolutionary computation algorithm to address problems of over-fitting such as χ^2 reduction.

6. REFERENCES

[1] John L Pfaltz and Azriel Rosenfeld. Web grammars. In *Proceedings of the 1st international joint conference on Artificial intelligence*, pages 609–619. Morgan Kaufmann Publishers Inc., 1969.

[2] John L Pfaltz. Web grammars and picture description. *Computer Graphics and Image Processing*, 1(2):193–220, 1972.

[3] Horst Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):574–582, 1982.

[4] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph*. *Data mining and knowledge discovery*, 11(3):243–271, 2005.

[5] Keven Ates, Jacek Kukluk, Lawrence Holder, Diane Cook, and Kang Zhang. Graph grammar induction on structural data for visual programming. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 232–242. IEEE, 2006.

[6] Francesc Rosselló and Gabriel Valiente. Graph transformation in molecular biology. In *Formal Methods in Software and Systems Modeling*, pages 116–133. Springer, 2005.

[7] Luc Dehaspe, Hannu Toivonen, and Ross D King. Finding frequent substructures in chemical compounds. In *KDD*, volume 98, page 1998, 1998.

[8] Stefan Kramer, Luc De Raedt, and Christoph Helma. Molecular feature mining in hiv data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 136–143. ACM, 2001.

[9] Wenke Lee and Salvatore J Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)*, 3(4):227–261, 2000.

[10] Calvin Ko. Logic induction of valid behavior specifications for intrusion detection. In *Proceedings of the IEEE Symposium on Security and Privacy. (S&P 2000)*, pages 142–153. IEEE, 2000.

[11] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[12] Collin F Lynch. Agg: Augmented graph grammars for complex heterogeneous data. In *Proceedings of the first international workshop on Graph-Based Educational Data Mining (GEDM 2014)*.

[13] Collin F. Lynch and Kevin D. Ashley. Empirically valid rules for ill-defined domains. In John Stamper and Zachary Pardos, editors, *Proceedings of The* 7^{th} *International Conference on Educational Data Mining (EDM 2014)*. International Educational Datamining Society IEDMS, 2014.

[14] Collin F. Lynch, Kevin D. Ashley, and Min Chi. Can diagrams predict essay grades? In Stefan Trausan-Matu, Kristy Elizabeth Boyer, Martha E. Crosby, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, Lecture Notes in Computer Science, pages 260–265. Springer, 2014.

[15] Sherry E. Marcus, Melanie Moy, and Thayne Coffman. Social network analysis. In Diane J. Cook and Lawrence B. Holder, editors, *Mining Graph Data*, chapter 17, pages 443–468. John Wiley & Sons, 2006.

[16] Chang Hun You, Lawrence B. Holder, and Diane J. Cook. Dynamic graph-based relational learning of temporal patterns in biological networks changing over time. In Hamid R. Arabnia, Mary Qu Yang, and Jack Y. Yang, editors, *BIOCOMP*, pages 984–990. CSREA Press, 2008.

[17] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2002)*, pages 721–724. IEEE, 2002.

[18] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[19] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings IEEE International Conference on Data Mining. (ICDM 2001)*, pages 313–320. IEEE, 2001.

[20] MICHIHIRO Kuramochi and George Karypis. Finding topological frequent patterns from graph datasets. *Mining Graph Data*, pages 117–158, 2006.

[21] Nitish Manocha, Diane J Cook, and Lawrence B Holder. Cover story: structural web search using a graph-based discovery system. *Intelligence*, 12(1):20–29, 2001.

[22] Diane J. Cook, Lawrence B. Holder, and G. Michael Youngblood. Graph-based analysis of human transfer learning using a game testbed. *IEEE Trans. on Knowl. and Data Eng.*, 19:1465–1478, November 2007.

[23] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray: Albermarle Street: London, United Kingdom, 6 edition, 1872.

[24] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press: Cambridge, Massachusetts, 1999.

[25] Wolfgang Banzhaf. *Genetic programming: an introduction on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers ; Heidelburg : Dpunkt-verlag; San Francisco, California, 1998.

[26] Collin F. Lynch, Kevin D. Ashley, Niels Pinkwart, and Vincent Aleven. Argument graph classification with genetic programming and c4.5. In Ryan Shaun Joazeiro de Baker, Tiffany Barnes, and Joseph E. Beck, editors, *The 1st International Conference on Educational Data Mining, Montreal, Québec, Canada, June 20-21, 2008. Proceedings*, pages 137–146, 2008.

[27] Wikipedia. Spearman's rank correlation coefficient — wikipedia, the free encyclopedia, 2013. [Online; accessed 27-February-2013].

Communities of Performance & Communities of Preference

Rebecca Brown
North Carolina State University
Raleigh, NC
rabrown7@ncsu.edu

Collin Lynch
North Carolina State University
Raleigh, NC
cflynch@ncsu.edu

Yuan Wang
Teachers College, Columbia University
New York, NY
elle.wang@columbia.edu

Michael Eagle
North Carolina State University
Raleigh, NC
mjeagle@ncsu.edu

Jennifer Albert
North Carolina State University
Raleigh, NC
jennifer_albert@ncsu.edu

Tiffany Barnes
North Carolina State University
Raleigh, NC
tmbarnes@ncsu.edu

Ryan Baker
Teachers College, Columbia University
New York, NY
ryanshaunbaker@gmail.com

Yoav Bergner
Educational Testing Service
Princeton, NJ
ybergner@gmail.com

Danielle McNamara
Arizona State University
Phoenix, AZ
dsmcnamara1@gmail.com

ABSTRACT

The current generation of Massive Open Online Courses (MOOCs) operate under the assumption that good students will help poor students, thus alleviating the burden on instructors and Teaching Assistants (TAs) of having thousands of students to teach. In practice, this may not be the case. In this paper, we examine social network graphs drawn from forum interactions in a MOOC to identify natural student communities and characterize them based on student performance and stated preferences. We examine the community structure of the entire course, students only, and students minus low performers and hubs. The presence of these communities and the fact that they are homogeneous with respect to grade but not motivations has important implications for planning in MOOCs.

Keywords

MOOC, social network, online forum, community detection

1. INTRODUCTION

The current generation of Massive Open Online Courses (MOOCs) is designed to leverage student interactions to augment instructor guidance. The activity in courses on sites such as Coursera and edX is centered around user forums that, while curated and updated by instructors and TAs, are primarily constructed by students. When planning and building these courses, it is hoped that students will help one another through the course and that interacting with stronger students will help to improve the performance of weaker ones. It has not yet been shown, however, that this type of support occurs in practice.

Prior research on social networks has shown that social groups, even those that gather face-to-face, can fragment into disjoint sub-communities [37]. This small-group separation, if it takes place in an online course, can be considered negative or positive, depending on one's perspective. If poor students communicate only with similarly-floundering peers, then they run the risk of perpetuating misunderstandings and of missing insights discussed by better-performing peers and teaching staff. An instructor may wish to avoid this fragmentation to encourage poor students to connect with better ones.

These enduring subgroups may be beneficial, however, by helping students to form enduring supportive relationships. Research by Li et al. has shown that such enduring relationships can enhance students' social commitment to a course [18]. We believe that this social commitment will in turn help to reduce feelings of isolation and alienation among students in a course. Eckles and Stradley [9] have shown that such isolation is a key predictor of student dropout.

We have previously shown that students can form stable communities and that those communities are homogeneous with respect to performance [3]. However that work did not: show whether these results are consistent with prior work on immediate peer relationships; address the impact of hub students on these results; or discuss whether students' varying goals and preferences motivate the community structure. Our goal in this paper is to build upon our prior work by addressing these issues. In the remainder of this paper we will survey prior educational literature on community formation in traditional and online classrooms. We will then build upon our prior work by examining the impact of hub users. And we will look at the impact of user motivations on community formation.

2. RELATED WORK

2.1 MOOCs, Forums, & Student Performance

A survey of the literature on MOOCs shows the beginnings of a research base generating an abundance of data that has not yet been completely analyzed [19]. According to Seaton et al. [29], most of the time students spend on a MOOC is spent in discussion forums, making them a rich and important data source. Stahl et al. [30] illustrates how through this online interaction students collaborate to create knowledge. Thus students' forum activity is good not only for the individual student posting content or receiving answers, but for the class as a whole. Huang et al. [14] investigated the behavior of the highest-volume posters in 44 MOOC-related forums. These "superposters" tended to enroll in more courses and do better in those courses than the average. Their activity also added to the overall volume of forum content and they left fewer questions unanswered in the forums. Huang et al. also found that these superposters did not suppress the activity of less-active users. Rienties et al. [25] examined the way in which user interaction in MOOCs is structured. They found that allowing students to self-select collaborators is more conducive to learning than randomly assigning partners. Further, Van Dijk et al. [31] found that simple peer instruction is significantly less effective in the absence of a group discussion step, pointing again to the importance of a class discussion forum.

More recently Rosé et al. [27] examined students' evolving interactions in MOOCs using a Mixed-Membership Stochastic Block model which seeks to detect partially overlapping communities. They found that the likelihood that students would drop out of the course is strongly correlated with their community membership. Students who actively participated in forums early in the course were less likely to drop out later. Furthermore, they found one forum sub-community that was much more prone to dropout than the rest of the class, suggesting that MOOC communities are made up of students who behave in similar ways. This community can in turn reflect or impact a student's level of motivation and their overall experience in a course much like the "emotional contagion" model used in the Facebook mood manipulation study by Kramer, Guillroy, and Hancock [16].

Yang et al. [36] also notes that unlike traditional courses students can join MOOCs at different times and observed that students who join a course early are more likely to be active and connected in the forums, and less likely to drop out, than those who join later. MOOCs also attract users with a range of individual motivations. In a standard classroom setting students are constrained by availability, convention, and goals. Few students enroll in a traditional course without seeking to complete it and to get formal credit for doing so. MOOCs by virtue of their openness and flexibility attract a wide range of students with unique personal motivations [10]. Some join the course with the intent of completing it. Others may seek only to brush up on existing knowledge, obtain specific skills, or just watch the videos. These distinct motivations in turn lend themselves to different in-class behaviors including assignment viewing and forum access. The impact of user motivations in online courses has been previously discussed by Wang et al. [32, 33]; we will build upon that work here. Thus it is an open question whether these motivations affect students' community behaviors or not.

2.2 Communities, Hubs, & Peers

Kovanovic et al. [15] examined the relationship between social network position or centrality, and social capital formation in courses. Their work is specifically informed by the Community of Inquiry (COI) framework. the COI framework is focused on distance education and is particularly suited to online courses of the type that we study here. The model views course behavior through three *presences* which mediate performance: cognitive, teaching, and social.

This *social presence* considers the nature and persistence of student interactions and the extent to which they reinforce students' behaviors. In their analysis, the authors sought to test whether network relationships, specifically students' centrality in their social graph, is related to their social performance as measured by the nature and type of their interactions. To that end, they examined a set of course logs taken from a series of online courses offered within a public university. They found that students' position within their social graph was positively correlated with the nature and type of their interactions, thus indicating that central players also engaged in more useful social interactions. They did not extend this work to groups, however, focusing solely on individual hub students.

Other authors have also examined the relationship between network centrality, neighbor relationships, network density, and student performance factors. Eckles and Stradley [9] applied network analysis to student attrition, finding that students with strong social relationships with other students who drop out are significantly more likely to drop out themselves. Rizzuto et al. [26] studied the impact of social network density on student performance. Network density is defined as the fraction of possible edges that are present in a given graph. Thus it is a measure of how "clique-like" the graph is. The authors examined self-reported social networks for students in a large traditional undergraduate psychology course. They found that denser social networks were significantly correlated with performance. However, a dominance analysis [1] showed that this factor was less predictive than pure academic ability. These results serve to motivate a focus on the role of social relationships in student behavior. Their analysis is complicated, however, by their reliance on self-report data which will skew the strength and recency of the reported relationships.

Fire et al. [11] studied student interaction in traditional classrooms, constructing a social network based on cooperation on class assignments. Students were linked based on partnership on group work as well as inferred cooperation based on assignment submission times and IP addresses. The authors found that a student's grade was significantly correlated with the grade of the student with the strongest links to that student in the social network. We perform similar analysis in this paper to examine whether the same correlation exists in MOOCs.

Online student interaction in blended courses has also been linked to course performance. Dawson [8] extracted student and instructor social networks from a blended course's online discussion forums and found that students in the 90th grade percentile had larger social networks than those in the 10th percentile. The study also found that high-performing students primarily associated with other high-performing students and were more likely to be connected to the course instructor, while low-performing students tended to associate with other low-performers. In a blended course, this effect may be offset by face-to-face interaction not captured in the online social network, but if the same separation happens in MOOC communities, low-performing students are less likely to have other chances to learn from high-performing ones.

2.3 Community Detection

One of the primary activities students engage in on forums is question answering. Zhang et al. [38] conducted a social network analysis on an online question-and-answer forum about Java programming. Using vertex in-degree and out-degree, they were able to identify a relatively small number of active users who answered many questions. This allowed the researchers to develop various algorithms for calculating a user's Java expertise. Dedicated question-and-answer forums are more structured than MOOC forums, with question and answer posts identified, but a similar approach might help identify which students in a MOOC ask or answer the most questions.

Choo et al. [5] studied community detection in Amazon product-review forums. Based on which users replied to each other most often, they found communities of book and movie reviewers who had similar tastes in these products. As in MOOC forums, users did not declare any explicit social relationships represented in the system, but they could still be grouped by implicit connections.

In the context of complex networks, a community structure is a subgraph which is more densely connected internally than it is to the rest of the network. We chose to apply the Girvan-Newman edge-betweenness algorithm (GN) [13]. This algorithm takes as input a weighted graph and a target number of communities. It then ranks the edges in the graph by their edge-betweenness value and removes the highest ranking edge. To calculate Edge-betweenness we identify the shortest path $p(a,b)$ between each pair of nodes a and b in the graph. The edge-betweenness of an arc is defined as the number of shortest paths that it participates in. This is one of the centrality measures explored by Kovanovic et al. above [15]. The algorithm then recalculates the edge-betweenness values and iterates until the desired number of disjoint community subgraphs has been produced. Thus the algorithm operates by iteratively finding and removing the highest-value communications channel between communities until the graph is fully segmented. For this analysis, we used the iGraph library [7] implementation of G-N within R [24].

The strength of a candidate community can be estimated by modularity. The *modularity score* of a given subgraph is defined as a ratio of its intra-connectedness (edges within the subgraph) to the inter-connectedness with the rest of the graph minus the fraction of such edges expected if they were distributed at random [13, 35]. A graph with a high modularity score represents a dense sub-community within the graph.

3. DATA SET

This study used data collected from the "Big Data in Education" MOOC hosted on the Coursera platform as one of the inaugural courses offered by Columbia University [32]. It was created in response to the increasing interest in the learning sciences and educational technology communities in using EDM methods with fine-grained log data. The overall goal of this course was to enable students to apply each method to answer education research questions and to drive intervention and improvement in educational software and systems. The course covered roughly the same material as a graduate-level course, Core Methods in Educational Data Mining, at Teachers College Columbia University. The MOOC spanned from October 24, 2013 to December 26, 2013. The weekly course was composed of lecture videos and 8 weekly assignments. Most of the videos contained in-video quizzes (that did not count toward the final grade).

All of the weekly assignments were structured as numeric input or multiple-choice questions. The assignments were graded automatically. In each assignment, students were asked to conduct analyses on a data set provided to them and answer questions about it. In order to receive a grade, students had to complete this assignment within two weeks of its release with up to three attempts for each assignment, and the best score out of the three attempts was counted. The course had a total enrollment of over 48,000, but a much smaller number actively participated. 13,314 students watched at least one video, 1,242 students watched all the videos, 1,380 students completed at least one assignment,and 778 made a post or comment in the weekly discussion sections. Of those with posts, 426 completed at least one class assignment. 638 students completed the online course and received a certificate (meaning that some students could earn a certificate without participating in forums at all).

In addition to the weekly assignments the students were sent a survey that was designed to assess their personal motivations for enrolling in the course. This survey consisted of 3 sets of questions: MOOC-specific motivational items; two PALS (Patterns of Adaptive Learning Survey) sub-scales [21], Academic Efficacy and Mastery-Goal Orientation; and an item focused on confidence in course completion. It was distributed to students through the course's E-mail messaging system to students who enrolled in the course prior to the official start date. Data on whether participants successfully completed the course was downloaded from the same course system after the course concluded. The survey received 2,792 responses; 38% of the participants were female and 62% of the participants were male. All of the respondents were over 18 years of age.

The MOOC-specific items consisted of 10 questions drawn from previous MOOC research studies (cf. [2, 22]) asking respondents to rate their reasons for enrollment. These 10 items address traits of MOOCs as a novel online learning platform. Specifically, these 10 items included questions on both the learning content and features of MOOCs as a new platform. Two PALS Survey scales [21] measuring mastery-goal orientation and academic efficacy were used to study standard motivational constructs. PALS scales have been widely used to investigate the relation between a learning environment and a student's motivation (cf. [6, 20, 28]). Altogether ten items with five under each scale were included. The participants were asked to select a number from 1 to 5 with 1 meaning least relevant and 5 most relevant. Respondents were also asked to self-rate their confidence on a scale of 1 to 10 as to whether they could complete the course according to the pace set by the course instructor. All three groups of items were domain-general.

4. METHODS

For our analysis, we extracted a social network from the online forum associated with the course. We assigned a node to each student, instructor, or TA in the course who added to it. Nodes representing students were labeled with their final course grade out of 100 points. The Coursera forums operate as standard threaded forums. Course participants could start a new thread with an initial post, add a post to an existing thread, and add a comment or child element below an existing post. We added a directed edge from the author of each post or comment to the parent post and to all posts or comments that preceded it on the thread based upon their timestamp. We made a conscious decision to omit the textual content of the replies with the goal of isolating the impact of the structure alone.

We thus treat each reply or followup in the graph as an implicit social connection and thus a possible relationship. Such implicit social relationships have been explored in the context of recommender systems to detect strong communities of researchers [5]. This is, by design, a permissive definition that is based upon the assumption that individuals generally add to a thread after viewing the prior content within it and that individual threads can be treated as group conversations with each reply being a conscious statement for everyone who has already spoken. The resulting network forms a multigraph with each edge representing a single implicit social interaction. We removed self loops from this graph as they indicate general forum activity but not any meaningful interaction with another person. We also removed vertices with a degree of 0, and collapsed the parallel edges to form a simple weighted graph for analysis.

In the analyses below we will focus on isolating student performance and assessing the impact of the faculty and hub students. We will therefore consider four classes of graphs: *ALL* the complete graph; *Student* the graph with the instructor and TAs removed; *NoHub* the graph with the instructor and hub users removed; and *Survey* which includes only students who completed the motivation survey. We will also consider versions of the above graphs without students who obtained a score of 0, and without the isolated individuals who connect with at most one other person. As we will discuss below, a number of students received a zero grade in the course. Because this is an at-will course, however, we cannot readily determine why these scores were obtained. They may reflect a lack of engagement with the course, differential motivations for taking the course, a desire to see the course materials without assignments, or genuinely poor performance.

4.1 Best-Friend Regression & Assortativity

Fire et al. [11] applied a similar social network approach to traditional classrooms and found a correlation between a student's most highly connected neighbor ("best friend") and the student's grade. The links in that graph included cooperation on assignments as well as partnership on group assignments. To examine whether the same correlation existed in a massive online course in which students were less likely to know each other beforehand and there were no group assignments, we calculated each student's best friend in the same manner and performed a similar correlation.

The simple best friends analysis gives a straightforward mechanism for correlating individual students. However it is also worthwhile to ask about students who are one-step removed from their peers. Therefore we will also calculate the grade assortativity (r_G) of the graphs. Assortativity describes the correlation of values between vertices and their neighbors [23]. The assortativity metric r ranges between -1 and 1, and is essentially the Pearson correlation between vertex and their neighbors [23]. A network with $r = 1$ would have each vertex only sharing edges with vertices of the same score. Likewise, if $r = -1$ vertices in the network would only share edges with vertices of different scores. Thus grade assortativity allows us to measure whether individuals are not just connected directly to individuals with similar scores but whether they correlate with individuals who are one step removed.

Several commonly studied classes of networks tend to have patterns in their assortativity. Social networks tend to have high assortativity, while biological and technological networks tend to have negative values (dissortativity) [23]. In a homogeneous course or one where students only form stratified communities we would expect the assortativity to be very high while in a heterogeneous class with no distinct communities we would expect it to be quite low.

4.2 Community Detection

The process of community detection we employed is briefly described here [3]. As noted there we elected to ignore the edge direction when making our graph. Our goal in doing so was to focus on communities of learners who shared the same threads, even when they were not directly replying to one-another. We believe this to be a reasonable assumption given the role of class forums as a knowledge-building environment in which students exchange information with the group. Individuals who participate in a thread generally review prior posts before submitting their contribution and are likely to return to view the followups. Homogeneity in this context would mean that students gathered and communicated primarily with equally-performing peers and thus that they did not consistently draw from better-performing classmates and help lower-performing ones *or* that the at-will communities served to homogenize performance, with the students in a given cluster evening out over time.

While algorithms such as GN are useful for finding clusters they do not, in and of themselves, determine the *right* number of communities. Rather, when given a target number they will seek to identify the *best* possible set of communities. In some implementations the algorithm can be applied to iteratively select the maximum modularity value over a possible range. Determining the correct number of communities to detect, however, is a non-trivial task especially in large and densely connected graphs where changes to smaller communities will have comparatively small effects on the global modularity score. As a consequence we cannot simply optimize for the best modularity score as we would risk missing small but important communities [12].

Therefore, rather than select the clusterings based solely on the highest modularity, we have opted to estimate the correct number of clusters visually. To that end we plotted a series of modularity curves over the set of graphs. For each graph G we applied the GN algorithm iteratively to produce all clusters in the range $(2,|G_N|)$. For each clustering, we then calculated the global modularity score. We examined the resulting scores to identify a *crest* where the modularity gain leveled off or began to decrease thus indicating that future subdivisions added no meaningful information or created schisms in existing high-quality communities. This is a necessarily heuristic process that is similar to the use of Scree plots in Exploratory Factor Analysis [4]. We define the number identified as the *natural* cluster number.

5. RESULTS AND DISCUSSION

Before removing self-loops and collapsing the edges, the network contained 754 nodes and 49,896 edges. The final social network contained 754 nodes and 17,004 edges. 751 of the participants were students, with 1 instructor and 2 TAs. One individual was incorrectly labeled as a student when they were acting as the Chief Community TA. Since this person's posts clearly indicated that he or she was acting in a TA capacity with regard to the forums, we relabeled him/her as a TA. Of the 751 students 304 obtained a zero grade in the course leaving 447 nonzero students. 215 of the 751 students responded to the motivation survey.

There were a total of 55,179 registered users, so the set of 754 forum participants is a small fraction of the entire course audience. However, forum users are not necessarily those who will make an effort or succeed in the course. Forum users did not all participate in the course, and some students who participated in the course did not use the forums: 1,381 students in the course got a grade greater than 0, and 934 of those did not post or comment on the forums, while 304 of the 751 students who did participate in the forums received a grade of 0. Clearly students who go to the trouble of posting forum content are in some respect making an effort in the course beyond those who don't, but this does not necessarily correspond to course success.

5.1 Best-Friend Regression & Assortativity

We followed Fire et al.'s methodology for identifying Best Friends in a weighted graph and calculated a simple linear regression over the pairs. This correlation did not include the instructor or TAs in the analysis. We calculated the correlation between the students' grades to their best friends' grades in the set using Spearman's Rank Correlation Coefficient (ρ) [34]. The two variables were strongly correlated, $\rho(748) = 0.44$, $p < 0.001$. However, the correlation was also affected by the dense clusters of students with 0 grades. After removing the 0 grade students we found an additional moderate correlation, $\rho(444) = 0.29$, $p < 0.001$.

Thus the significant correlation between best-friend grade and grade holds over the transition from the traditional classroom to a MOOC. This suggests that students in a MOOC, excluding the many who drop out or do not submit assignments, behave similarly to those in a traditional classroom in this respect. These results are also consistent with our calculations for assortativity. There we found a small assortative trend for the grades as shown in Table 1. These values reflect that a student was frequently communicating with students who in turn communicated with students at a similar performance level. This in turn supports our belief that homogeneous communities may be found. As Table 1 also illustrates, the zero-score students contribute substantially to the assortativity correlation as well with the correlation dropping by as much as a third when they were removed.

Table 1: The grade assortativity for each network.

Users	Zeros	V	E	r_G
All	Yes	754	17004	0.29
All	No	447	5678	0.20
Students	Yes	751	15989	0.32
Students	No	447	5678	0.20
Non-Hub	Yes	716	9441	0.37
Non-Hub	No	422	3119	0.24

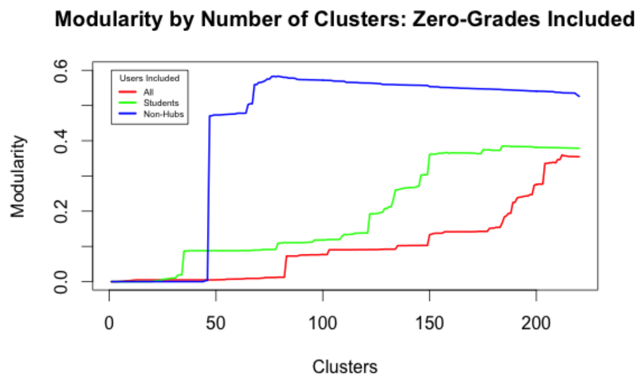


Figure 1: Modularity for each number of clusters, including students with zeros.

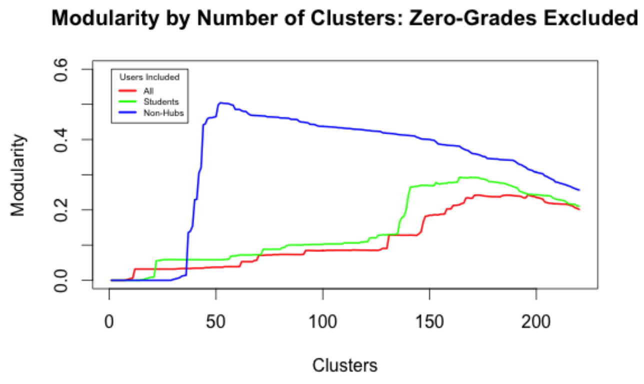


Figure 2: Modularity for each number of clusters, excluding students with zeros.

5.2 Community Structure

The modularity curves for the graphs both with and without zero-score students are shown in Figures 1 and 2. We examined these plots to select the natural cluster numbers which are shown in Table 2. As the values illustrate the instructor, TAs, and hub students have a disproportionate impact on the graph structure. The largest hub student in our graph connects to 444 out of 447 students in the network. The graph with all users had lower modularity and required more clusters than the graphs with only students or only non-hubs (see Table 2), with

Table 2: Graph sizes and natural number of clusters for each graph.

Users	Zeros	V	E	Clusters
All	Yes	754	17004	212
All	No	447	5678	173
Students	Yes	751	15989	184
Students	No	447	5678	169
Non-Hub	Yes	716	9441	79
Non-Hub	No	422	3119	52
Survey	Yes	215	1679	58

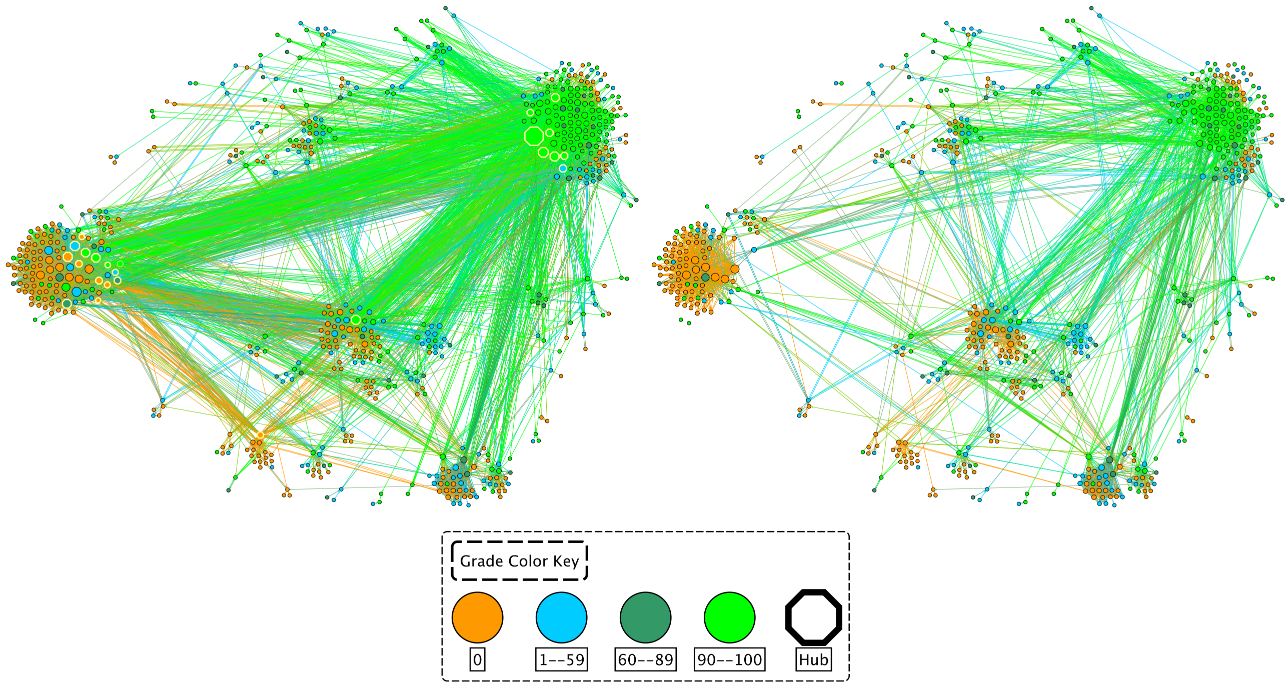


Figure 3: View of the student communities with edges of frequency <2 removed. The Student network with (left) and without (right) hub-students, with each vertex representing a student and grade represented as color.

the non-hub graph having the highest modularity. This suggests that non-hub students formed more isolated communities, while teaching staff and hubs communicated across these communities and connected them.

This largely consistent with the intent of the forums and the active role played by the instructor and TAs in monitoring and replying to all relevant posts in the forums. It is particularly interesting how closely the curves for the ALL and Student graphs mirror one another. This may indicate that the hub students are also those that followed the instructor and TAs closely, thus giving them isomorphic relationships, or it may indicate that they are more connected than even the instructors and thus came to bind the forums together on their own. This impact is further illustrated by the cluster plots shown in Figure 3. Here the absence of the hub students results in a noticeable thinning of the graph which in turn highlights the frequency of communication that can be attributed to this, comparatively small, group.

The difference between the full plots and those with zero values are also notable as the zero grade students were clearly a major factor in community formation. A direct examination of the user graph showed that many of the zero students were only connected to other zero students or were not connected at all. This is also highlighted in Figure 3. In both graphs the bulk of the zero score students are clustered in a tight network of communities on the left-hand side. That super-community consists primarily of zero score students communicating with other zero-score students, a structure we have nick-named the 'deathball.'

5.3 Student Performance & Motivation

As the color coding in Figure 3 illustrates, the students did cluster by performance. Table 3 shows the average grade and standard deviation for a small selection of the communities in the ALL reply network including zero-grades, hub students, and teaching staff. Several of the communities, particularly the larger ones, do show a blend of good and poor students, with a high standard deviation. However many if not most of the communities are more homogeneous with good and poor students sharing a community with similarly-performing peers. These clusters have markedly lower standard deviation.

Table 3: Grade statistics by community, selected to show examples of more and less homogeneous communities.

Members	Average Grade	Standard Deviation
118	21.62	36.58
41	22.00	32.45
34	25.41	40.44
31	56.13	47.69
20	49.05	45.64
16	12.44	31.13
14	88.43	22.47
12	96.08	6.36
11	96.45	7.38
4	3.00	6.00
4	8.50	9.81
4	4.25	8.50
4	96.25	3.50

An examination of the grade distribution for each of the clusters showed that the scores within each cluster were non-normal. Therefore we opted to apply the Kruskal-Wallis (KW) test to assess the correlation between cluster membership and perfor-

Table 4: Kruskal-Wallis test of student grade by community, for each graph.

Users	Zeros	Chi-Squared	df	p-value
All	Yes	349.0273	211	< 0.005
All	No	216.1534	172	< 0.02
Students	Yes	202.0814	78	< 0.005
Students	No	80.93076	51	< 0.005
Non-Hub	Yes	309.8525	183	< 0.005
Non-Hub	No	218.9603	168	< 0.01
Survey	Yes	99.99840	577	< 0.005

mance. The KW test is a nonparametric rank-based analogue to the common Analysis of Variance [17]. Here we tested grade by community number with the community being treated as a categorical variable. The results of this comparison are shown in Table 4. As that illustrates, cluster membership was a significant predictor of student performance for all of the graphs with the non-zero graphs having markedly lower p-values than those with zero students included. These results are consistent with our hypothesis that students would form clusters of equal-performers and we find that those results hold even when the highly-connected instructors, TAs and hub students are included.

We performed a similar KW analysis for the questions on the motivation survey and for a binary variable indicating whether or not the student completed the survey at all. For this analysis we evaluated the clusters on all of the graphs. We found no significant relationship between the community structure on any of the graphs and the survey question results or the survey completion variable. Thus while the clusters may be driven by separate factors they are not reflected in the survey content.

6. CONCLUSIONS AND FUTURE WORK

Our goal in this paper was to expand upon our prior community detection work with the goal of aligning that work with prior research on peer impacts, notably the work of Fire et al. [11]. We also sought to examine the impact of hub students and student motivations on our prior results.

To that end we performed a novel community clustering analysis of student performance data and forum communications taken from a single well-structured MOOC. As part of this analysis we described a novel heuristic method for selecting natural numbers of clusters, and replicated the results of prior studies of both immediate neighbors and second-order assortativity.

Consistent with prior work, we found that students' grades were significantly correlated with their most closely associated peers in the new networks. We also found that this correlation extended out to their second-order neighborhood. This is consistent with our prior work showing that students form stable user communities that are homogeneous by performance. We found that those results were stable even if instructors, hub players, students with 0 scores, and students who did not fill out the survey were removed from consideration. This suggests that either the students are forming communities that are homogeneous or that the effect of those individual and network features on the communities and on performance is minimal.

We also found that community membership was not a significant predictor of whether students would complete the motivation survey or of students' motivations. We were surprised by the fact that even when we focused solely on individuals who had completed the survey, the students did not connect by stated goals. This suggests to us that the students are more likely coalescing around the pragmatic needs of the class or conceptual challenges rather than on the winding paths that brought them there. One limitation of this work is that by relying on the forum data we were focused solely on the comparatively small proportion of enrolled students (6%) who actively participated in the forums. This group is, by definition a smaller set of more actively-involved participants.

In addition to addressing our primary questions this study also raised a number of open issues for further exploration. Firstly, this work focused solely on the final course structure, grades, and motivations. We have not yet addressed whether these communities are stable over time or how they might change as students drop in our out. Secondly, while we ruled out motivations as a basis for the community this work we were not able to identify what mechanisms do support the communities. And finally this study raises the question of generality and whether or not these results can be applied to MOOCs offered on different topics or whether the results apply to traditional and blended courses.

In subsequent studies we plan to examine both the evolution of the networks over time as well as additional demographic data with the goal of assessing both the stability of these networks and the role of other potential latent factors. We will also examine other potential clustering mechanisms that control for other user features such as frequency of involvement and thread structure. We also plan to examine other similar datasets to determine if these features transition across classes and class types. We believe that these results may change somewhat once students can coordinate face to face far more easily than online.

7. ACKNOWLEDGMENTS

This work was supported by NSF grant #1418269: "Modeling Social Interaction & Performance in STEM Learning" Yoav Bergner, Ryan Baker, Danielle S. McNamara, & Tiffany Barnes Co-PIs.

8. REFERENCES

[1] R. Azen and D. Budescu. The dominance analysis approach for comparing predictors in multiple regression. *Psychological Methods*, 8(2):129–48, 2003.

[2] Y. Belanger and J. Thornton. Bioelectricity: A quantitative approach Duke University's first MOOC. *Journal of Learning Analytics*, 2013.

[3] R. Brown, C. F. Lynch, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, and D. McNamara. Good communities and bad communities: Does membership affect performance? In C. Romero and M. Pechenizkiy, editors, *Proceedings of the 8th International Conference on Educational Data Mining*, 2015. submitted.

[4] R. B. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

[5] E. Choo, T. Yu, M. Chi, and Y. Sun. Revealing and incorporating implicit communities to improve recommender systems. In M. Babaioff, V. Conitzer, and D. Easley, editors, *ACM Conference*

on Economics and Computation, EC '14, Stanford, CA, USA, June 8-12, 2014, pages 489–506. ACM, 2014.

[6] K. Clayton, F. Blumberg, and D. P. Auld. The relationship between motivation learning strategies and choice of environment whether traditional or including an online component. *British Journal of Educational Technology*, 41(3):349–364, 2010.

[7] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[8] S. Dawson. 'Seeing' the learning community: An exploration of the development of a resource for monitoring online student networking. *British Journal of Educational Technology*, 41(5):736–752, 2010.

[9] J. Eckles and E. Stradley. A social network analysis of student retention using archival data. *Social Psychology of Education*, 15(2):165–180, 2012.

[10] A. Fini. The technological dimension of a massive open online course: The case of the CCK08 course tools. *The International Review Of Research In Open And Distance Learning*, 10(5), 2009.

[11] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach. Predicting student exam's scores by analyzing social network data. In *Active Media Technology*, pages 584–595. Springer, 2012.

[12] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. of the National Academy of Sciences*, 104(1):36–41, 2007.

[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[14] J. Huang, A. Dasgupta, A. Ghosh, J. Manning, and M. Sanders. Superposter behavior in MOOC forums. In *Proc. of the first ACM conference on Learning@ scale conference*, pages 117–126. ACM, 2014.

[15] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. In S. G. Santos and O. C. Santos, editors, *Proc. of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proc.* CEUR-WS.org, 2014.

[16] A. D. I. Kramer, J. E. Guillory, and J. T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proc. of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[17] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[18] N. Li, H. Verma, A. Skevi, G. Zufferey, J. Blom, and P. Dillenbourg. Watching MOOCs together: investigating co-located MOOC study groups. *Distance Education*, 35(2):217–233, 2014.

[19] T. R. Liyanagunawardena, A. A. Adams, and S. A. Williams. MOOCs: A systematic study of the published literature 2008-2012. *The International Review of Research in Open and Distributed Learning*, 14(3):202–227, 2013.

[20] J. L. Meece, E. M. Anderman, and L. H. Anderman. Classroom goal structure, student motivation, and academic achievement. *Annual Review of Psychology*, 57:487–503, 2006.

[21] C. Midgley, M. L. Maehr, L. Hruda, E. Anderinan, L. Anderman, and K. E. Freeman. *Manual for the Patterns of Adaptive Learning Scales (PALS).* University of Michigan, Ann Arbor, 2000.

[22] MOOC @ Edinburgh 2013. MOOC @ Edinburgh 2013 - report #1. *Journal of Learning Analytics*, 2013.

[23] M. E. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20):208701, Oct. 2002.

[24] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

[25] B. Rienties, P. Alcott, and D. Jindal-Snape. To let students self-select or not: That is the question for teachers of culturally diverse groups. *Journal of Studies in International Education*, 18(1):64–83, 2014.

[26] T. Rizzuto, J. LeDoux, and J. Hatala. It's not just what you know, it's who you know: Testing a model of the relative importance of social networks to academic performance. *Social Psychology of Education*, 12(2):175–189, 2009.

[27] C. P. Rosé, R. Carlson, D. Yang, M. Wen, L. Resnick, P. Goldman, and J. Sherer. Social factors that contribute to attrition in MOOCs. In *Proc. of the first ACM conference on Learning@ scale conference*, pages 197–198. ACM, 2014.

[28] A. M. Ryan and H. Patrick. The classroom social environment and changes in adolescents' motivation and engagement during middle school. *American Educational Research Journal*, 38(2):437–460, 2001.

[29] D. Seaton, Y. Bergner, I. Chuang, P. Mitros, and D. Pritchard. Who does what in a massive open online course? *Communications of the ACM*, 57(4):58–65, 2014.

[30] G. Stahl, T. Koschmann, and D. Suthers. Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, 2006:409–426, 2006.

[31] L. Van Dijk, G. Van Der Berg, and H. Van Keulen. Interactive lectures in engineering education. *European Journal of Engineering Education*, 26(1):15–28, 2001.

[32] Y. Wang and R. Baker. Content or platform: Why do students complete MOOCs? *MERLOT Journal of Online Learning and Teaching*, 11(1):191–218, 2015.

[33] Y. Wang, L. Paquette, and R. Baker. A longitudinal study on learner career advancement in MOOCs. *Journal of Learning Analytics*, 1(3), 2014.

[34] Wikipedia. Spearman's rank correlation coefficient — Wikipedia, the free encyclopedia, 2013. [Online; accessed 27-February-2013].

[35] Wikipedia. Modularity (networks) — Wikipedia, the free encyclopedia, 2014. [Online; accessed 5-February-2015].

[36] D. Yang, T. Sinha, D. Adamson, and C. P. Rose. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proc. of the 2013 NIPS Data-Driven Education Workshop*, volume 10, page 13, 2013.

[37] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[38] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proc. of the 16th international conference on World Wide Web*, pages 221–230. ACM, 2007.

Using the Hint Factory to Compare Model-Based Tutoring Systems

Collin Lynch
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
cflynch@ncsu.edu

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
twprice@ncsu.edu

Min Chi
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
mchi@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT
Model-based tutoring systems are driven by an abstract domain model and solver that is used for solution validation and student guidance. Such models are robust but costly to produce and are not always adaptive to specific students' needs. Data-driven methods such as the Hint Factory are comparatively cheaper and can be used to generate individualized hints without a complete domain model. In this paper we explore the application of data-driven hint analysis of the type used in the Hint Factory to existing model-based systems. We present an analysis of two probability tutors Andes and Pyrenees. The former allows for flexible problem-solving while the latter scaffolds students' solution path. We argue that the state-space analysis can be used to better understand students' problem-solving strategies and can be used to highlight the impact of different design decisions. We also demonstrate the potential for data-driven hint generation across systems.

1. INTRODUCTION
Developers of model-based tutoring systems draw on domain experts to develop ideal models for student guidance. Studies of such systems have traditionally been focused on their overall impact on students' performance and not on the students' user-system interaction. The Hint Factory, by contrast, takes a data-driven approach to extract advice based upon students' problem solving paths. In this paper we will apply the Hint Factory analytically to evaluate the impact of user interface changes and solution constraints between two closely-related tutoring systems for probability.

Model-based tutoring systems are based upon classical expert systems, which represent relevant domain knowledge via static rule bases or sets of constraints [9]. These knowledge bases are generally designed by domain experts or with their active involvement. They are then paired with classical search algorithms or heuristic satisfaction algorithms to automatically solve domain problems, identify errors in student solutions, and to provide pedagogical guidance. The goal of the design process is to produce expert models that give the same procedural advice as a human expert. Classical model-based tutors have been quite successful in field trials, with systems such as the ACT Programming Tutor helping students achieve almost two standard deviations higher than those receiving conventional instruction [5].

Data-driven hint generation methods such as those used in Hint Factory [17] take a different approach. Rather than using a strong domain model to generate *a-priori* advice, data-driven systems examine prior student solution attempts to identify likely paths and common errors. This prior data can then be used to provide guidance by directing students towards successful paths and away from likely pitfalls. In contrast to the expert systems approach, these models are primarily guided not by what experts consider to be *ideal* but by what students *do*.

Model-based systems such as Andes [18] are advantageous as they can provide appropriate procedural guidance to students *at any point* in the process. Such models can also be designed to reinforce key meta-cognitive concepts and explicit solution strategies [4]. They can also scale up rapidly to include new problems or even new domain concepts which can be incorporated into the existing system and will be available to all future users. Rich domain models, however, are comparatively expensive to construct and require the long-term involvement of domain experts to design and evaluate them.

Data-driven methods for generating feedback, by contrast, require much lower initial investment and can readily adapt to individual student behaviors. Systems such as the Hint Factory are designed to extract solutions from prior student data, to evaluate the quality of those solutions, and to compile solution-specific hints [17]. While this avoids the need for a strong domain model, it is limited to the space of solutions explored by prior students. In order to incorporate new problems or concepts it is necessary to collect additional data. Additionally, such methods are not generally designed to incorporate or reinforce higher-level solution strategies.

We believe that both of these approaches have inherent advantages and are not necessarily mutually exclusive. Our goal in this paper is to explore what potential data-driven methods have to inform and augment model-based systems. We argue that data-driven methods can be used to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems. To that end we will survey relevant prior work on model-based and data-driven tutoring. We will describe two closely-related tutoring systems and data collected from them. We will then present a series of analyses using state-based methods and discuss the conclusions that we drew from them.

2. BACKGROUND

2.1 Model-Based Tutoring

Model-based tutoring systems take a classical expert-systems approach to tutoring. They are typically based upon a strong domain model composed of declarative rules and facts representing domain principles and problem-solving actions coupled with an automatic problem solver. This knowledge base is used to structure domain knowledge, define individual problems, evaluate candidate solutions, and to provide student guidance. Novices typically interact with the system through problem solving with the system providing solution validation, automatic feedback, pedagogical guidance, and additional problem-solving tasks. The Sherlock 2 system, for example, was designed to teach avionics technicians about appropriate diagnostic procedures [11]. The system relies on a domain model that represents the avionics devices being tested, the behavior of the test equipment, and rules about expert diagnostic methods. Sherlock 2 uses these models to pose dynamic challenges to problem solvers, to simulate responses to their actions, and to provide solution guidance.

Andes [19, 18, 20] and Pyrenees [4] are closely-related model-driven ITSs in the domains of physics and probability. They were originally developed at the University of Pittsburgh under the Direction of Dr. Kurt VanLehn. Like other model-based systems, they rely on a rule-based domain model and automatic problem solvers that treat the domain rules as problem-solving steps. They distinguish between higher-level domain concepts such as Bayes' Rule, and atomic steps such as variable definitions. Principles are defined by a central equation (e.g. $p(A|B) = (p(B|A) * p(A))/p(B)$) and encapsulate a set of atomic problem-solving steps such as writing the equation and defining the variables within it.

The systems are designed to function as homework-helpers, with students logging into the system and being assigned or selecting one of a set of predefined problems. Each problem

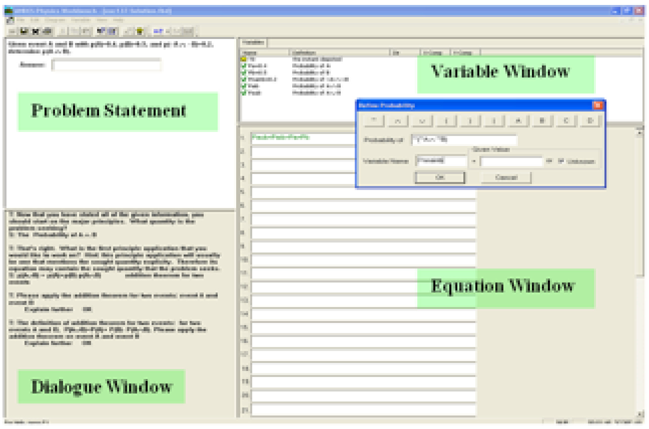
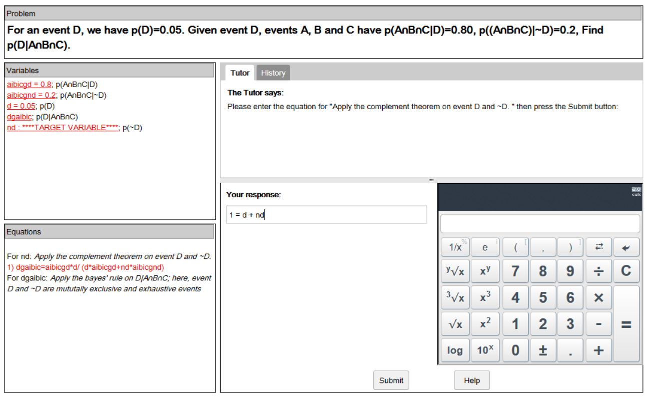


Figure 1: The Andes user interface showing the problem statement window with workspace on the upper left hand side, the variable and equation windows on the right hand side, and the dialogue window on the lower left.

is associated with a pre-compiled solution graph that defines the set of possible solutions and problem-solving steps. The system uses a principle-driven automated problem solver to compile these graphs and to identify the complete solution paths. The solver is designed to implement the Target Variable Strategy (TVS), a backward-chaining problem solving strategy that proceeds from a goal variable (in this case the answer to the problem) via principle applications to the given information. The TVS was designed with the help of domain experts and guides solvers to define basic solution information (e.g. given variables) and then to proceed from the goal variable and use principles to define it in terms of the given variables.

Students working with Andes use a multi-modal user interface to write equations, define variables and engage in other atomic problem-solving steps. A screenshot of the Andes UI can be seen in Figure 1. Andes allows students to solve problems flexibly, completing steps in any order so long as they are valid [20]. A step is considered to be valid if it matches one or more entries in the saved solution paths and all necessary prerequisites have been completed. Invalid steps are marked in red, but no other immediate feedback is given. Andes does not force students to delete or fix incorrect entries as they do not affect the solution process. In addition to validating entries, the Andes system also uses the precompiled solution graphs to provide procedural guidance (*next-step-help*). When students request help, the system will map their work to the saved solution paths. It will then select the most complete solution and prompt them to work on the next available step.

One of the original goals of the Andes system was to develop a tutor that operated as an "intelligent worksheet." The system was designed to give students the freedom to solve problems in any order and to apply their preferred solution strategy. The system extends this freedom by allowing invalid steps in an otherwise valid solution and by allowing students to make additional correct steps that do not advance the solution state or are drawn from multiple



Problem

For an event D, we have p(D)=0.05. Given event D, events A, B and C have p(AnBnC|D)=0.80, p((AnBnC)|~D)=0.2, Find p(D|AnBnC).

Variables

aibicgd = 0.8; p(AnBnC|D)
aibicgnd = 0.2; p(AnBnC|~D)
d = 0.05; p(D)
dgaibic; p(D|AnBnC)
nd : ****TARGET VARIABLE****; p(~D)

Equations

For nd: *Apply the complement theorem on event D and ~D.*
1) dgaibic=aibicgd*d/ (d*aibicgd+nd*aibicgnd)
For dgaibic: *Apply the bayes' rule on D|AnBnC; here, event D and ~D are mututally exclusive and exhaustive events*

The Tutor says:

Please enter the equation for "Apply the complement theorem on event D and ~D. " then press the Submit button:

Your response:

1 = d + nd

Figure 2: The Pyrenees user interface showing the problem statement at the top, the variable and equation lists on the left, and the tutor interaction window with calculator on the lower right.

solution paths. This was motivated in part by a desire to make the system work in many different educational contexts where instructors have their own preferred methods [20]. The designers of Andes also consciously chose only to provide advice upon demand when the students would be most willing to accept it. For the students however, particularly those with poor problem-solving skills, this passive guidance and comparative freedom can be problematic as it does not force them to adhere to a strategy.

This problem motivated the development of Pyrenees. Pyrenees, like Andes acts as a homework helper and supports students with on-demand procedural and remediation help. It uses an isomorphic domain model with the same principles, basic steps, problems, and solution paths. Unlike Andes, however, Pyrenees forces students to applying the target-variable-strategy during problem solving. It also requires them to repair incorrect entries immediately before moving on. Students are guided through the solution process with a menu-driven interface, shown in Figure 2. At each step, the system asks students what they want to work on next and permits them to make any valid step that is consistent with the TVS. Chi and VanLehn [3] conducted a study of the two systems and found that scaffolding the TVS in Pyrenees helped to eliminate the gap between high and low learners. This effect was observed both in the original domain where it was taught (in their case probability) and it transferred to a new domain (physics), where students used Andes alone.

2.2 Data-Extraction and Data-Driven Tutoring.

One of the longstanding goals of educational data-miners is to support the development of data-driven tutoring systems. Such systems use past student data to structure pedagogical and domain knowledge, administer conceptual and pedagogical advice, or evaluate student performance and needs. A number of attempts have been made to address these goals. One of the most successful data-driven systems is the Hint Factory [1, 2, 17]. The Hint Factory takes an MDP-based approach to hint generation. It takes as input a set of prior student logs for a given problem, represented as a network of interactions [6, 7]. Each vertex in this network represents the state of a student's partial solution at some point during the problem solving process, and each edge represents an action that takes the student from one state to another. A complete solution is represented as a path from the initial state to a goal state. Each state in the interaction network is assigned a weight via a value-iteration algorithm. A new student requesting a hint is matched to a previously observed state and given context-sensitive advice. If, for example, the student is working on a problem that requires Bayes' Rule and has already defined $p(A)$, $p(B)$, and $p(B|A)$ then the Hint Factory would first prompt them to consider defining $p(A|B)$, then it would point them to Bayes Rule, before finally showing them the equation $p(A|B) = (p(B|A) * p(A))/p(B)$.

These hints are incorporated into existing tutoring systems in the form of a lookup table that provides state-specific advice. When a user asks for help the tutor will match their current state to an index state in the lookup table and will prompt them to take the action that will lead them to the highest value neighboring state. If their current state is not found then the tutor will look for a known prior state or will give up. The Hint Factory has been applied successfully in a number of domains including logic proofs [17], data structures [8], and programming [15, 10, 13]. Researchers have also explored other related methods for providing data-driven hints. These include alternative state representations [13], path construction algorithms [16, 14], and example-based model-construction [12].

The primary goal of the Hint Factory is to leverage prior data to provide optimal state-specific advice. By calculating advice on a per-state basis, the system is able to adapt to students' specific needs by taking into account both their current state and the paths that they can take to reach the goal. As a consequence the authors of the Hint Factory argue that this advice is more likely to be in the students' Zone of Proximal Development and thus more responsive to their needs than a less-sensitive algorithm.

3. METHODS

In order to investigate the application of data-driven methods to model-based tutoring systems, we collected data from two studies conducted with Andes and Pyrenees in the domain of probability. We then transformed these datasets into interaction networks, consisting of states linked with actions. We used this representation to perform a variety of quantitative and qualitative analyses with the goal of evaluating the differences between the two systems and the impact of the specific design decisions that were made in each.

3.1 The Andes and Pyrenees Datasets

The Andes dataset was drawn from an experiment conducted at the University of Pittsburgh [3]. This study was designed to assess the differential impact of instruction in Andes and Pyrenees on students' meta-cognitive and problem-solving skills. Participants in this study were college undergraduates who were required to have taken high-school level algebra and physics but not to have taken a course in probability or statistics. The participants were volunteers and were paid by time not performance.

Forty-four students completed the entire study. However for the purposes of the present analysis, we drew on all

66 students who completed at least one problem in Andes-Probability. This is consistent with prior uses of the Hint Factory which draw from all students including those who did not complete the problem. The Pyrenees-Probability logs from this study were not used due to problems with the data format that prevented us from completing our analysis. From this dataset we drew 394 problem attempts covering 11 problems. The average number of steps required to solve the problems was 17.6. For each problem we analyzed between 25 and 72 problem attempts, with an average of 35.8 attempts per problem. Some attempts were from the same student, with at most two successful attempts per student. Over all problems, 81.7% of the attempts were successful, with the remainder being incomplete attempts.

The Pyrenees dataset was drawn from a study of 137 students conducted in the 200-level Discrete Mathematics course in the Department of Computer Science at North Carolina State University. This study used the same probability textbook and pre-training materials as those used in the Andes study. The students used Pyrenees as part of a homework assignment, in which they completed 12 problems using the tutoring system. One of these problems was not represented in the Andes dataset. We therefore excluded it from our analysis, leaving 11 shared problems.

Unlike the Andes students, however, the Pyrenees students were not always required to solve every problem. In this study the system was configured to randomly select some problems or problem steps to present as worked examples rather than as steps to be completed. In order to ensure that the results were equivalent we excluded the problem-level worked examples and any attempt with a step-level worked example from our analysis. As a consequence, each problem included a different subset of these students. For each problem we analyzed between 83 and 102 problem attempts, with an average of 90.8 attempts per problem. Some attempts were from the same student, with at most one successful attempt per student. Over all problems, 83.4% of the attempts were successful.

3.2 State and Action Representations

In order to compare the data from both tutors, we represented each problem as an interaction network, a representation used originally in the Hint Factory [7]. In the network a vertex, or state, represents the sum total of a students' current problem solving steps at a given time during a problem-solving attempt. Because Andes permits flexible step ordering while Pyrenees does not, we chose to represent the problem solving state s_t as the set of valid variables and equations defined by the student at time t.

A variable is a probabilistic expression, such as $P(A \cup B)$, that the student has identified as important to solving the problem, for which the probability is known or sought. An equation represents the application of a principle of probability, which relates the values of defined variables, such as the Complement Theorem, $P(A) + P(\neg A) = 1$. Because such equations can be written in many algebraically equivalent ways, we represent each equation as a 2-tuple, consisting of the set of variables included in the equation (e.g. $\{P(A), P(\neg A)\}$) and the principle being applied (e.g. Complement Theorem). Because we only represent valid equations, this representation uniquely identifies any equation for a given problem. Because we used the same state representation for both tutors, we were able to compare states directly across tutors.

Additionally, we opted to ignore incorrect entries. Pyrenees prevents students from applying the principles of probability improperly and forces them to correct any mistakes made immediately therefore any errors in the student logs are immediately removed making the paths uninformative. Andes, by contrast, gives students free reign when writing equations and making other entries. This freedom resulted in syntactic errors and improper rule application errors arising in our dataset. The meaning of these invalid equations is inherently ambiguous and therefore difficult to incorporate into a state definition. However such errors are immediately flagged by the system and may be ignored by the student without consequence as they do not affect the answer validity therefore they may be safely ignored as well.

An edge, or action, in our network represents the correct application of a rule or a correct variable definition and leads to a transition from one state to another. For the present dataset and state representation, the possible actions were the definition or deletion of variables or equations. Each of these actions was possible in both tutors.

4. ANALYSIS

In order to develop a broader understanding of our datasets, we first visualized the interaction network for each problem as a weighted, directed graph. We included attempts from both Andes and Pyrenees in the network, and weighted the edges and verticies by the frequency with which it appeared in the logs. We annotated each state and edge with the weight contributed by each tutor. Two examples of these graphs are given in Figure 3.

Throughout this section, we will use these graphs to address the points we outlined at the end of Section 1. We begin with a case study from one problem and will explore the student problem solving strategies using our graph representation. We will then compare the Andes and Pyrenees Systems with a variety of metrics based on this representation. We will relate our observations back to the design decisions of each system and identify evidence that may support or question these decisions. Finally, we will show how the analysis methods associated with data-driven hint generation can be used to validate some of these findings.

4.1 Case Study: Problem Ex242

The graphical representation of a problem is very helpful for giving a high-level overview of a problem and performing qualitative analysis. Problem *Ex242*, shown in Figure 3, presents an interesting scenario for a number of reasons. The problem was the 10th in a series of 12 practice problems, and asked the following:

> Events A, B and C are mutually exclusive and exhaustive events with $p(A) = 0.2$ and $p(B) = 0.3$. For an event D, we know $p(D|A) = 0.04$, $p(D|B) = 0.03$, and $p(C|D) = 0.3$. Determine $p(B|D)$.

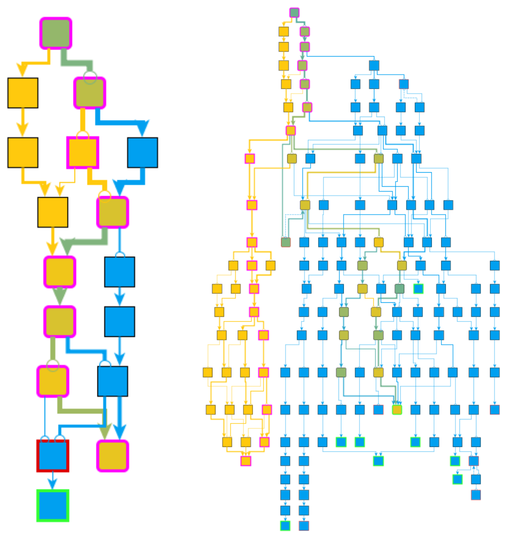


Figure 3: A graph representation of problems Ex252a, left, and Ex242, right. States and edges are colored on a gradient from blue to yellow, indicating the number of students who reached that state in the Andes and Pyrenees tutors respectively. Rounded edges indicate that at least one student from both tutors is present in a state. A green border indicates a solution state, and a pink border indicates that a state is contained in the pedagogically "ideal" solution. Edge thickness corresponds to the natural log of the number of problem attempts which included the given edge.

The problem is notable in the Pyrenees dataset because it was the only one in which the students were split almost evenly among two solution paths. For most of the problems in the dataset the vast majority of students followed the optimal solution path with only a few finding alternatives. The ideal solution path, as suggested by Pyrenees' domain model, employed repeated applications of the Conditional Probability Theorem: $P(A \cap B) = P(A|B)P(B)$, which the problem was designed to teach. The students had been previously exposed to Bayes' Theorem however, and over half of them chose to apply it instead. This allowed them to circumvent one variable definition and two applications of the Conditional Probability Theorem, achieving a slightly shorter solution path. We make no argument here which path the tutor should encourage students to take, but it is worth noting that the Hint Factory, when trained on the Pyrenees data for this problem, recommends the shorter, more popular path.

The Andes dataset gives us a very different set of insights into this problem. Because Andes lacks the strong process scaffolding of Pyrenees, students were able to make a wider variety of choices, leading to a graph with many more, less populous states. While almost every state and edge in the Pyrenees graph represents multiple students, the Andes graph contains a number of paths, including solutions, that were reached by only one student. In some state-based analyses the authors choose to omit these singleton states, for instance when generating hints. We have chosen to include them as they represent a fair proportion of the Andes data. For instance, 62 of the 126 Andes states for Ex242 were singleton states.

Interestingly, while there were small variations among their solutions, all of the Andes students choose to apply Bayes' Rule rather than relying solely on the Conditional Probability Theorem as suggested by Pyrenees. This, coupled with the strong proportion of Pyrenees students who also chose the Bayes' Rule solution, indicates that the solution offered by Pyrenees may be unintuitive for students, especially if they have recently learned Bayes' Rule. Again, this can be interpreted as evidence that Pyrenees' strong guidance did have an impact on students' problem solving strategies, but it also raises concerns about how reasonable this guidance will appear to the students. Regardless of one's interpretation, an awareness of a trend like this can help inform the evolution of model-based tutors like Andes and Pyrenees.

4.2 Comparing datasets

We now turn to qualitatively comparing the datasets. While it is not a common practice to directly compare data from different tutors, we argue that it is appropriate, especially in this context. In longstanding tutoring projects it is common for developers and researchers to make many substantive changes. The Andes system itself has undergone substantial interface changes over the course of its development [20]. These changes can alter student behavior in substantial ways, and it is important for researchers to consider how they affect not just learning outcomes but also problem solving strategies, as was investigated by Chi et al. [4].

In many respects the close relationship between Andes and Pyrenees makes them analogous to different versions of the same tutor and the presence of an isomorphic knowledge base and problem set makes it possible for us to draw meaningful comparisons between students. In this section we will inspect how the scaffolding design decisions made when constructing the tutors affected the problem solving strategies exhibited by the students.

A visual inspection of the state graphs for each problem revealed significant portions of each graph were shared between the two datasets and portions that were represented in only one of the two. Despite the fact that students using Andes were capable of reaching any of the states available to students in Pyrenees, many Pyrenees states were never discovered by Andes students. As noted in Section 4.1, this suggests that guidance from the Pyrenees tutor is successful in leading students down solution paths that they would not otherwise have discovered, possibly applying skills that they would otherwise not have used.

To quantify these findings, we calculated the relative similarity of students in each tutor. For a given problem, we defined the state-similarity between datasets A and B as the probability that a randomly selected state from a student in A will be passed through by a randomly selected student in B. Recall from Section 3.2 that our state representation allows us to directly compare states across tutors. By this definition, the self-similarity of a dataset is a measure of how closely its students overlap each other while the cross-similarity is a measure of how closely its students overlap

States	Andes	Pyrenees	Solution
Andes	0.551 (0.134)	0.494 (0.153)	0.478 (0.186)
Pyrenees	0.460 (0.141)	0.688 (0.106)	0.669 (0.146)
Actions	Andes	Pyrenees	Solution
Andes	0.878 (0.085)	0.874 (0.118)	0.851 (0.140)
Pyrenees	0.828 (0.117)	0.936 (0.021)	0.923 (0.036)

Table 1: Pairwise similarity across tutors and the ideal solution path. Similarities were calculated for each problem, and each cell lists the mean (and standard deviation) over all problems. The top half covers the state similarity metrics while the bottom half of each table covers action similarity.

with the other dataset. Similarity measures for the datasets can be found at the top of Table 1.

Predictably, both datasets have higher self-similarity than cross-similarity, with Pyrenees showing higher self-similarity than Andes. This indicates that Pyrenees students chose more homogeneous paths to the goal. This is reasonable and consistent with the heavy scaffolding that is built into the system. It is important to note that our similarity metrics are not symmetric. The cross-similarity of Pyrenees with Andes is higher than the reverse. This indicates that the path taken by Pyrenees students are more likely to have been observed by Andes students than vice-versa. This has important implications for designers who are interested in collecting data from a system that is undergoing modifications. If a system becomes increasingly scaffolded and restrictive over time, past data will remain more relevant than in a system that is relaxed. In many ways this simply reflects the intuition that allowing students to explore a state space more fully will produce more broadly useful data, and restricting students will produce data that is more narrowly useful. Note that here we are only observing trends, and we make no claims of statistical significance.

In our analysis, we found that, within both datasets, many solution paths or sub-paths differed only in the order that actions were performed. In our domain, many actions do not have ordering constraints. It is possible, for example, to define the variables A and B in either order, and the resulting solution paths would deviate from one another. We thus sought to determine how much of the observed difference between our two datasets was due to these ordering effects. To that end we define the *action-similarity* between datasets A and P as the probability that a randomly selected action performed by a student in A will have been performed by a by a randomly selected student in B. These values are shown in the bottom of Table 1, and each of the trends observed for state-similarity hold, with predictably higher similarity values.

It is notable that the similarity between Pyrenees and Andes is almost as high as Andes' self-similarity, indicating that the actions taken by Pyrenees students are almost as likely to be observed in Andes students as Pyrenees students. This suggests that, for the most part, the Andes students performed a superset of the actions performed by the Pyrenees students. Thus the impact of Pyrenees is most visible in the *order* of execution, not the actions chosen. This is consistent

States	Andes	Pyrenees	Solution
Andes	0.419 (0.150)	0.327 (0.139)	0.253 (0.172)
Pyrenees	0.372 (0.193)	0.709 (0.145)	0.601 (0.214)
Actions	Andes	Pyrenees	Solution
Andes	0.818 (0.151)	0.582 (0.168)	0.636 (0.205)
Pyrenees	0.678 (0.212)	0.899 (0.038)	0.879 (0.055)

Table 2: Pairwise similarity across tutors and the ideal solution path calculated using a variable-free state representation. Rows and columns are the same as in Table 1.

Problem	**Andes**	**Pyrenees**
ex132	26 (47.5%)	2 (98.7%)
ex132a	13 (33.3%)	1 (100.0%)
ex144	2 (96.6%)	1 (100.0%)
ex152	11 (0.0%)	4 (0.0%)
ex152a	8 (59.0%)	3 (97.4%)
ex152b	12 (0.0%)	1 (100.0%)
ex212	8 (71.4%)	1 (100.0%)
ex242	9 (0.0%)	2 (49.38%)
ex252	7 (76.9%)	2 (98.4%)
ex252a	4 (81.8%)	2 (98.6%)
exc137	19 (0.0%)	2 (98.75%)

Table 3: For each problem, the tables gives the number of unique solution states represented in each tutor's dataset. Note that there may exist many solution paths which reach a given solution. The following percent (in parentheses) represents the percent solution paths that ended in the pedagogically ideal solution.

with the design goals of Pyrenees which was set up to guide students along the otherwise unfamiliar path of the TVS.

We also opted to examine the impact of the variable definitions on our evaluation. As noted above, variable definitions are an atomic action. They do not depend upon any event assertion and thus have no ordering constraints unlike the principles. We did so with the hypothesis that this would increase the similarity metrics for the datasets by eliminating the least constrained decisions from consideration. Our results are shown in Table 2 below. Contrary to our expectations, this actually reduced the similarity both within and across the datasets, with the exception of Pyrenees' self-similarity. Thus the unconstrained variable definitions did not substantially contribute to the dissimilarity. Rather, most of the variation lay in the order of principle applications.

4.3 Similarity to an "ideal" solution

We now turn to exploring how well the ideal solution was represented in the datasets. For both tutors the ideal solution is the pedagogically-desirable path constructed via the TVS. Our measure of cross-similarity between two datasets can also be applied between a single solution path and a dataset by treating the single solution as a set of one. We can thus measure the average likelihood of an ideal solution state appearing in a student's solution from each dataset. The results of this calculation are shown in tables 1 and 2, using both the state- and action-similarities explained earlier. Predictably, the solution has a high similarity with Pyrenees students, as these students are scaffolded tightly and offered few chances to deviate from the path.

As Table 3 shows, Pyrenees students were funneled almost exclusively to the ideal solution on the majority of problems, even if their paths to the solution were variable. We found only one problem, Ex152, where the Pyrenees students missed the ideal path. That was traced to a programming error that forced students along a similar path. Otherwise, there was only one problem, Ex242 (discussed in Section 4.1), where a meaningful percentage of students chose a different solution. The Andes students, by contrast, were much less likely to finish in the ideal solution state, but this was also problem-dependent.

4.4 Applications of the Hint Factory

Finally, having shown that the datasets differ, and that these differences are consistent with the differing design choices of the two tutors, we sought to determine what effect those differences would have on data-driven hint generation. Our goal was to determine how applicable a hint model of the type produced by the Hint Factory would be for one dataset if it was trained on another. To that end we performed a modified version of the Cold Start Experiment [1], which is designed to measure the number of state-specific hints that Hint Factory can provide given a randomly selected dataset. The Cold Start experiment functions like leave-one-out cross-validation for state-based hint generation. In the original Cold Start experiment, one student was selected at random and removed from the dataset, to represent a "new" student using the tutor. Each remaining student in the dataset was then added, one at a time, in a random order to the Hint Factory's model. On each iteration, the model is updated and the percentage of states on the 'new' student's path for which a hint is available is calculated. This is repeated a desired number of times with new students to account for ordering effects.

For the present study we calculated cold-start curves for both the Pyrenees and Andes datasets. We also calculated curves using the opposing dataset to illustrate the growth rate for cross-tutor hints. For these modified curves we selected the hint-generating students from the opposing dataset. All four curves are shown in Figure 4. Here *AvA* and *PvP* designate the within tutor curves for Andes and Pyrenees respectively while *PvA* and *AvP* designate the cross-tutor curves for hints from Pyrenees provided to Andes users and vice-versa. Figure 4 represents an average over all problems, and therefore the x-axis extends only as far as the minimum number of students to complete a problem. As the curves illustrate, the within-tutor curves reach high rates of coverage relatively quickly with *PvP* reaching a plateau above 95% after 21 students and *AvA* reaching 85%.

The cross-tutor curves, by contrast, reach much lower limits. *AvP* reaches a plateau of over 75% coverage, while *PvA* reaches a plateau of 60% coverage. This reflects the same trends observed in Tables 1 and 2, where Andes better explains the Pyrenees data than vice-versa; however, neither dataset completely covers the other. On the one hand this result is somewhat problematic as it indicates that prior data has a limited threshold for novel tutors or novel versions of a

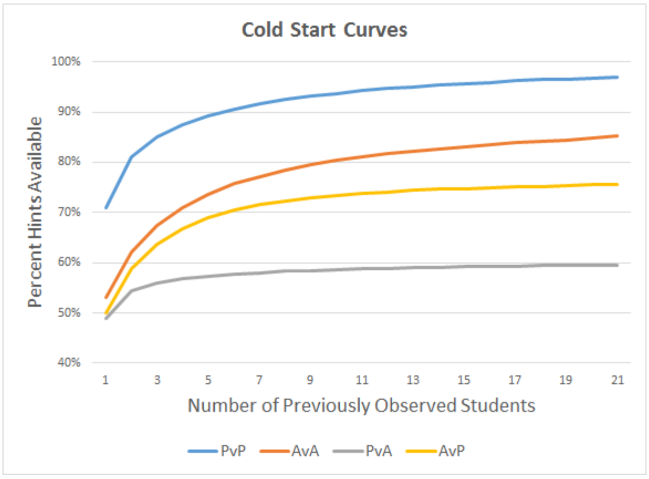


Figure 4: The four Cold Start curves, averaged across all problems. The x-axis shows the number of students used to train the model, and the y-access shows the percentage of a new student's path that has available hints. The curve labeled "XvY" indicates training on the X dataset and selecting a new student from the Y dataset (A = Andes; P = Pyrenees).

system in the same domain. Clearly a substantive interface and scaffolding change of the type made in Pyrenees can change the state space sufficiently that we cannot trivially rely on our prior data. On the other hand, while the cross-application of data does have upper limits, those limits are comparatively high. Clearly data from a prior system can be reused and can serve as a reliable baseline for novel system, with the caveat that additional exploratory data is required.

5. DISCUSSION AND CONCLUSION

Our goal in this paper was to evaluate the application of data-driven methods such as the Hint Factory to model-based tutoring systems. To that end we analyzed and compared datasets collected from two closely-related tutoring systems: Andes and Pyrenees. Through our analysis we sought to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems.

We found that, while the systems shared isomorphic domain models, problems, and ideal solutions, the observed user behaviors differed substantially. Students using the Andes system explored the space more widely, were more prone to identify novel solutions, and rarely followed the ideal solution path. Students in Pyrenees, by contrast, were far more homogeneous in their solution process and were limited in the alternative routes they explored. Contrary to our expectations, we found that this variation was not due to simple ordering variations in the simplest of steps but of alternative strategy selection for the higher-level domain principles. This is largely consistent with the design decisions that motivated both systems and with the results of prior studies.

We also found that the state-based hint generation method used in the Hint Factory can be applied to the Andes and Pyrenees data given a suitable state representation. For this analysis we opted for a set-based representation given the absence of strong ordering constraints across the principles. We then completed a cold-start analysis to show that the cross-tutor data could be used to bootstrap the construction of hints for a novel system but does not provide for complete coverage.

Ultimately we believe that the techniques used for data-driven hint generation have direct application to model-based systems. Data-driven analysis can be used to identify the behavioral differences between closely related systems and, we would argue, changes from one version of a system to another. We also found that these changes can be connected to the specific design decisions made during development. Further, we found that data-driven methods can be applied to model-based tutoring data to generate state-based hints. We believe that hint information of this type may be used to supplement the existing domain models to produce more user-adaptive systems. In future work we plan to apply these analyses to other appropriate datasets and to test the incorporation of state-driven hints or hint refinement to existing domain models.

6. ACKNOWLEDGMENTS

Work supported by NSF Grant #1432156 “Educational Data Mining for Individualized Instruction in STEM Learning Environments” Min Chi & Tiffany Barnes Co-PIs.

7. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.

[2] T. Barnes and J. C. Stamper. Automatic hint generation for logic proof tutoring using historical data. *Educational Technology & Society*, 13(1):3–12, 2010.

[3] M. Chi and K. VanLehn. Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In *Intelligent Tutoring Systems (ITS)*, volume 5091, pages 603–613, 2008.

[4] M. Chi and K. VanLehn. Meta-Cognitive Strategy Instruction in Intelligent Tutoring Systems: How, When, and Why. *Educational Technology & Society*, 3(1):25–39, 2010.

[5] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, pages 137–147, 2001.

[6] M. Eagle and T. Barnes. Exploring Differences in Problem Solving with Data-Driven Approach Maps. In *Educational Data Mining (EDM)*, 2014.

[7] M. Eagle and T. Barnes. Exploring Networks of Problem-Solving Interactions. In *Learning Analytics (LAK)*, 2015.

[8] D. Fossati, B. D. Eugenio, and S. Ohlsson. I learn from you, you learn from me: How to make iList learn from students. In *Artificial Intelligence in Education (AIED)*, 2009.

[9] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building Expert Systems*. Addison-Wesley Publishing Company Inc., Reading, Massachusetts, U.S.A., 1983.

[10] A. Hicks, B. Peddycord III, and T. Barnes. Building Games to Learn from Their Players: Generating Hints in a Serious Game. In *Intelligent Tutoring Systems (ITS)*, pages 312–317, 2014.

[11] S. Katz, A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin, and L. Greenberg. Sherlock 2: An intelligent tutoring system built on the lrdc framework. In C. P. Bloom and R. B. Loftin, editors, *Facilitating the Development and Use of Interactive Learning Environments*, Computers, Cognition, and Work, chapter 10, pages 227 – 258. Lawrence Erlbaum Associates, Mawah New Jersey, 1998.

[12] R. Kumar, M. E. Roy, B. Roberts, and J. I. Makhoul. Toward automatically building tutor models using multiple behavior demonstrations. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, editors, *Proceedings of the* 12^{th} *International Conference on Intelligent Tutoring Systems*, LNCS 8474, pages 535 – 544. Springer Verlag, 2014.

[13] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.

[14] C. Piech, M. Sahami, J. Huang, and L. Guibas. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Learning at Scale (LAS)*, 2015.

[15] K. Rivers and K. Koedinger. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013.

[16] K. Rivers and K. Koedinger. Automating Hint Generation with Solution Space Path Construction. In *Intelligent Tutoring Systems (ITS)*, 2014.

[17] J. C. Stamper, M. Eagle, T. Barnes, and M. J. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *I. J. Artificial Intelligence in Education*, 22(1-2):3–17, 2013.

[18] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Five years of evaluations. In C. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker, editors, *Artificial Intelligence in Education - Supporting Learning through Intelligent and Socially Informed Technology, Proceedings of the 12th International Conference on Artificial Intelligence in Education, AIED 2005, July 18-22, 2005, Amsterdam, The Netherlands*, volume 125 of *Frontiers in Artificial Intelligence and Applications*, pages 678–685. IOS Press, 2005.

[19] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Lessons learned. *I. J. Artificial Intelligence in Education*, 15(3):147–204, 2005.

[20] K. VanLehn and B. van de Sande. The Andes physics tutoring system: An experiment in freedom. *Advances in intelligent tutoring systems.*, pages 421–443, 2010.

An Exploration of Data-Driven Hint Generation in an Open-Ended Programming Problem

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
twprice@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
tmbarnes@ncsu.edu

ABSTRACT

Data-driven systems can provide automated feedback in the form of hints to students working in problem solving environments. Programming problems present a unique challenge to these systems, in part because of the many ways in which a single program can be written. This paper reviews current strategies for generating data-driven hints for programming problems and examines their applicability to larger, more open-ended problems, with multiple, loosely ordered goals. We use this analysis to suggest directions for future work to generate hints for these problems.

1. INTRODUCTION

Adaptive feedback is one of the hallmarks of an Intelligent Tutoring System. This feedback often takes the form of hints, pointing a student to the next step in solving a problem. While hints can be authored by experts or generated by a solver, more recent data-driven approaches have shown that this feedback can be automatically generated from previous students' solutions to a problem. The Hint Factory [18] has successfully generated data-driven hints in a number of problem solving domains, including logic proofs [2], linked list problems [5] and a programming game [12]. The Hint Factory operates on a representation of a problem called an interaction network [4], a directed graph where each vertex represents a student's state at some point in the problem solving process, and each edge represents a student's action that alters that state. A solution is represented as a path from the initial state to a goal state. A student requesting a hint is matched to a previously observed state and directed on a path to a goal state. The Hint Factory takes advantage of the intuition that students with the same initial state and objective will follow similar paths, producing a well-connected interaction network. When this occurs, even a relatively small sample of student solutions can be enough to provide hints to the majority of new students [1].

While problems in many domains result in well-connected networks, this is not always the case. Programming problems have a large, often infinite, space of possible states. Even a relatively simple programming problem may have many unique goal states, each with multiple possible solution paths, leaving little overlap among student solutions. Despite this challenge, a number of attempts have been made to adapt the Hint Factory to programming problems [9, 12, 16]. While these approaches have been generally successful, they are most effective on small, well structured programming problems, where the state space cannot grow too large. This paper explores the opposite type of problem: one that has a large state space, multiple loosely ordered goals, unstructured output and involves creative design. Each of these attributes poses a challenge to current data-driven hint generation techniques, but they are also the attributes that make such problems interesting, useful and realistic. In this paper, we will refer to these as open-ended programming problems. We investigate the applicability of current techniques to these problems and suggest areas for future research.

The primary contributions of this paper are 1) a review of current data-driven hint generation methods for programming problems, 2) an analysis of those methods' applicability to an open-ended problem and 3) a discussion of the challenges that need to be addressed before we can expect to generate hints for similar problems.

2. CURRENT APPROACHES

Current approaches to generating data-driven programming hints, including alternatives to the Hint Factory [10, 13, 17], can be broken down into three primary components:

1. A representation of a student's state and a method for determining when one state can be reached from another, meaning they are connected in the network
2. An algorithm that, given a student's current state, constructs an optimal path to a goal state. Often we simplify this to the problem of picking the first state on that path
3. A method to present this path, or next state, to the student in the form of a hint

For the purposes of this paper, we limit our discussion to the first step in this process. (For a good discussion of the second step, see an analysis by Piech et al. [13], comparing path selection algorithms.) While in some domains this first step is straightforward, in programming tasks, especially open-ended problems, it is likely the most challenging. The simplest approach is to take periodic snapshots of a student's code and treat these as states, connecting consecutive snapshots in the network. However, because two students' programs are unlikely to match *exactly*, this approach is likely to produce a very sparse, poorly connected network, making it difficult to match new students to prior solution attempts. A variety of techniques have been presented to address this problem, which can be grouped into three main strategies: canonicalization, connecting states and alternative state definitions.

2.1 Canonicalization

Canonicalization is the process of putting code states into a standardized form, often by removing semantically unimportant information, so that trivial difference do not prevent two states from matching. Rivers and Koedinger [15] present a method for canonicalizing student code by first representing it as an Abstract Syntax Tree (AST). Once in this form, they apply a number of functions to canonicalize the code, including normalizing arithmetic and boolean operators, removing unreachable and unused code, and inlining helper functions. After performing this canonicalization on a set of introductory programming problems, they found that a median 70% of states had at least one match in the network. Jin et al. [9] represent a program's state as a Linkage Graph, where each vertex is a code statement, and each directed edge represents an ordering dependency, determined by which variables are read and assigned to in each statement. This state representation allows the Hint Factory to ignore statement orderings which are not important to the execution of the program. Lazar and Bratko [10] use the actual text of Prolog code to represent a student's state, and then canonicalize the code by removing whitespace and normalizing variable names.

2.2 Connecting States

Even with canonicalization, a student requesting a hint may not match any existing state in the network. In this case, we can look for a similar state and create a connection between them. Rivers and Koedinger [16] use normalized string edit distance as a similarity metric between two program states. They connect any two states in the network which have at least 90% similarity, even if no historical data connect these states. Additionally, they use a technique called path construction to generate new solution paths from a given state to a nearby, unconnected goal state by searching for a series of insertions, deletions and edits to their AST that will transform it into the goal state [17]. They also use this method to discover new goal states which may be closer to the student's current state. Jin et al. [9] use a similar technique to transform their Linkage Graphs to better match the current state of a student when no direct matches can be found in the interaction network. Piech et al. [13] use path construction to interpolate between two consecutive states on a solution path which differ by more than one edit. This is useful to smooth data when student code is recorded in shapshots that are too far apart.

2.3 Alternate State Definitions

Another approach is to forego the traditional code-based representation of a student's state, and use an alternate definition. Hicks and Peddycord [7, 12] used the Hint Factory to generate hints for a programming game called Bots, in which the player writes a program to direct a robot through various tasks in a 3D level. They chose to represent the state of a player's program as the final state of the game world after the program was executed. They compared the availability of hints when using this "world state" model with a traditional "code state" model, and found that using world states significantly reduced the total number of states and increased the availability of hints. The challenge with this approach, as noted by the authors, is the generation of actionable hints. A student may be more capable of making a specific change to her code than determining how to effect a specific change in the code's output.

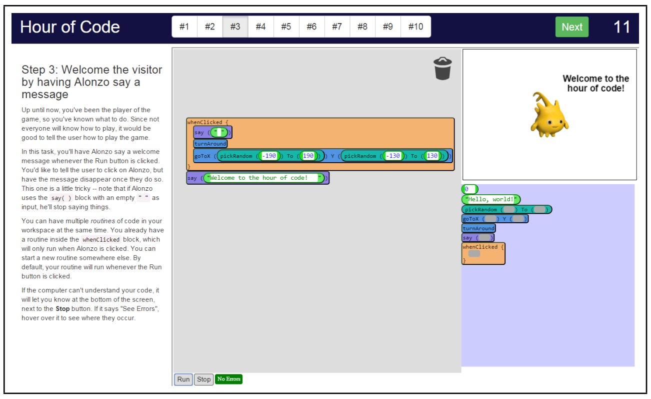
3. AN OPEN-ENDED PROBLEM

The above techniques have all shown success on smaller, well-structured problems, with ample data. We want to investigate their applicability to an open-ended problem, as described in Section 1, where this is not the case. The purpose of this paper is not to create actionable hints, nor are we attempting to show the failures of current methods by applying them to an overly challenging task. Rather, our purpose is exploratory, using a small dataset to identify areas of possible future work, and challenges of which to be mindful when moving forward with hint generation research.

We collected data from a programming activity completed by 6th grade students in a STEM outreach program called SPARCS [3]. The program, which meets for half-day sessions approximately once a month during the school year, consists of lessons designed and taught by undergraduate and graduate students to promote technical literacy. The class consisted of 17 students, 12 male and 5 female.

The activity was a programming exercise based on an Hour of Code activity from the Beauty and Joy of Computing curriculum [6]. It was a tutorial designed to introduce novices to programming for the first time. The exercise had users create a simple web-based game, similar to whack-a-mole, in which players attempt to click on a sprite as it jumps around the screen to win points. The exercise was split into 9 objectives, with tutorial text at each stage. Students were not required to finish an objective before proceeding. A finished project required the use of various programming concepts, including events, loops, variables and conditionals. The students used a drag-and-drop, block-based programming language called Tiled Grace [8], which is similar to Scratch [14]. The user writes a program by arranging code blocks, which correspond directly to constructs in the Grace programming language. The editor also supports switching to textual coding, but this feature was disabled. A screenshot of the activity can be seen in Figure 1.

During the activity, the students were allowed to go through the exercise at their own pace. If they had questions, the students were allowed to ask for help from the student volunteers. Students were stopped after 45 minutes of work. Snapshots of a student's code were saved each time it was run and periodically throughout the session. Occasional technical issues did occur in both groups. One student had severe technical issues, and this student's data was not analyzed (and is not reflected in the counts above). Students


Step 3: Welcome the visitor by having Alonzo say a message

Up until now, you've been the player of the game, so you've known what to do. Since not everyone will know how to play, it would be good to tell the user how to play the game.

In this task, you'll have Alonzo say a welcome message whenever the Run button is clicked. You'd like to tell the user to click on Alonzo, but have the message disappear once they do so. This one is a little tricky -- note that if Alonzo uses the `say( )` block with an empty `" "` as input, he'll stop saying things.

You can have multiple *routines* of code in your workspace at the same time. You already have a routine inside the `whenClicked` block, which will only run when Alonzo is clicked. You can start a new routine somewhere else. By default, your routine will run whenever the Run button is clicked.

If the computer can't understand your code, it will let you know at the bottom of the screen, next to the **Stop** button. If it says "See Errors", hover over it to see where they occur.

```
whenClicked {
  say ("")
  turnAround
  goToX (pickRandom (-190) To (190)) Y (pickRandom (-130) To (130))
}
say ("Welcome to the hour of code!")
```

Figure 1: A screenshot of the Hour of Code activity that students completed. Students received instructions on the left panel. The center panel was a work area, and students could drag blocks in from the bottom-right panel. The top-right panel allowed students to test their games.

produced on average 148.5 unique code states and accomplished between 1 and 6 of the activity's objectives, averaging 3.2 objectives per student.

4. ANALYSIS

We attempted to understand the applicability of each of the techniques discussed in Section 2 to our dataset. However, since the output of our program was a game that involved nondeterminism, we felt it would be inappropriate to attempt to represent a program's state as the result of its execution. We therefore focused on the first two strategies, canonicalization and connecting states.

4.1 Canonicalization

Our initial representation of a student's code state was a tree, where each code block was a node, and its children included any blocks that were nested inside of it. In this way, our representation was similar to Rivers and Koedinger's ASTs. To get a baseline for the sparsity of our dataset, we first analyzed the code states without performing any canonicalization. We calculated the total number of states in the interaction network and the percentage which were only reached by one student. Of those states reached by multiple students, we calculated the mean and median number of students who reached them. We also calculated the percentage of each student's states that were unreached by any other student in the dataset.

We then canonicalized the data by removing variable names and the values of number and string literals. Our problem featured very few arithmetic or logical operators, and these were generally not nested, so we did not normalize them, as suggested by Rivers and Koedinger [15]. We reran our analyses on the canonicalized data. To ensure that we had effectively removed all unimportant ordering information, we recursively sorted the children of each node in the tree. This effectively removed any ordering information from a student's code state, and kept only hierarchical information. This is somewhat more extreme than the Linkage Graphs of Jin et al. [9], and it does allow two meaningfully different code states to be merged in the process. We therefore see this as an upper bound on the value of removing unimportant orderings from a code state. We recomputed our metrics for the ordered-canonicalized interaction network as well. The results can be seen in Table 1. Our later analyses use the unsorted, canonicalized code representation.

	Raw	Canonical	Ordered
Total States	2380	1781	1656
% Unique	97.5%	94.8%	92.8%
Mean NU Count	3.44	3.95	2.82
Median NU Count	2	2	2
Mean % Path Unique Standard Deviation	89.9% (6.67)	83.0% (10.5)	78.9% (13.3)

Table 1: Various measures of the sparseness of the interaction network for the raw, canonicalized, and ordered-canonicalized state representations. Mean and median NU counts refer to the number of students who reached each non-unique state.

These results indicate that canonicalization does little to reduce the sparsity of the state space, with students spending most of their time in states that no other student has seen. For comparison, recall Rivers and Koedinger found 70% of states in a simple programming problem had a match after canonicalization [15], though they were using a much larger dataset. In our dataset, it is unlikely that we would be able to find a direct path from a new student's state to a goal state in order to suggest a hint.

4.2 Connecting States

To address this, we explored the feasibility of connecting a new student's state to a similar, existing state in the network. It is unclear how close two code states should be before it is appropriate to connect them as in [16], or to generate a path between them as in [17]. It certainly depends on the state representation and distance metric used. Rather than identifying a cutoff and measuring how often these techniques could be applied, we chose to visualize the distance between two students and make qualitative observations. Because our code states were already represented as trees, we used Tree Edit Distance (TED) as a distance metric. While Rivers and Koedinger reported better success with Levenshtein distance [16], we believe that TED is the most appropriate distance metric for block code, where tree edit operations correspond directly to user actions.

For each pair of students, A and B, we created an N by M distance matrix, D, where N is the number of states in A's solution path, and M is the number of states in B's solution path. $D_{i,j} = d(A_i, B_j)$, where d is the TED distance function, A_i is the ith state of A and B_j is the jth state of B. We used the RTED algorithm [11] to calculate the distance function, putting a weight of 1.0 on insertions, deletions and replacements. We also omitted any state which was identical to its predecessor state. We normalized these values by dividing by the maximum value of $D_{i,j}$, and plotted the result as an image. Three such images can be seen in Figure 2.

We also calculated the "path" through this matrix that passes through the least total distance. This does not represent a

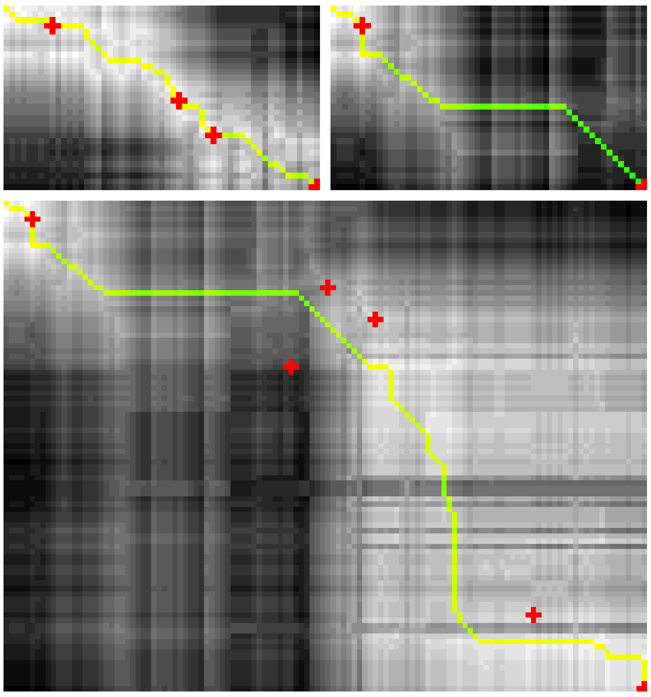


Figure 2: Three distance matrices, each comparing two students, where each pixel represents the TED between two states. Lighter shades of gray indicate smaller distances. The green/yellow line shows the path through the matrix with minimized total distance, with yellow shades indicating smaller distances. Red crosses indicate where both students met an objective. The top-left figure compares two students as they completed objectives 1-4 in the exercise. In this example, the minimum-distance line crosses each objective. The top right figure also depicts two students completing objective 4, but the darker colors and straighter line indicate less alignment. The bottom figure depicts two students completing objectives 1-6, with high alignment.

path through the interaction network, but rather an alignment between the states of student A and those of student B. Each pixel of the line represents a pairing of a state from A with a state from B, such that these pairings are contiguous and represent the smallest total distance. While we do not suggest applying this directly as a strategy for hint generation, it serves an a useful visual indicator of the compatibility of two students for hinting purposes.

An alternate approach would have been to pair each state in the interaction network with its closest pair from any other student, and use this as a measure of how sparse the network was. We chose to compare whole students, rather than individual states, because we felt that the former could lead to strange hinting behavior. Imagine, for instance, that a student requests a hint, which initially points to a state from student B, but at the very next step requests a hint that points instead to student C. Perhaps the attributes that make the student's state similar to that of B are different from those that make the state similar to C. The resulting hints would be at best confusing, and at worst conflicting.

	Mean	Median	Max	Farthest
1	0.25 (0.27)	0.15 (0.29)	0.76 (0.56)	2.23 (0.75)
2	4.88 (3.93)	4.95 (4.34)	9.18 (5.74)	12.73 (6.10)
4	4.92 (2.77)	4.83 (2.78)	10.11 (3.69)	14.67 (4.77)
5	7.79 (1.32)	7.75 (1.41)	13.17 (1.72)	18.17 (1.72)
6	7.49 (1.11)	7.76 (1.37)	13.17 (0.98)	18.67 (1.75)

Table 2: For each objective, average distances (and standard deviations) of minimum-distance student pairs, using the mean, median and max metrics. For reference, the final column represents the average maximum distance each student moved from the start state while completing the objective.

A visual inspection of these matrices reveals that while many student pairs are quite divergent, some show a notable closeness throughout the exercise. In order to quantify these results, we developed a set of distance metrics between *students*. First, the distance matrix and the minimum-distance "path" were calculated for the two students. The path is comprised of pairs of states, and for each pair, we recorded the tree edit distance between the states. From this list of distances, we calculated the mean, median and maximum distances between the two students. We looked at each objective in the exercise, and isolated the relevant subpath of each student who completed that objective. We paired each of these subpaths with the most similar subpath in the set, using the mean, median and max distance metrics. Table 2 shows the average values of these minimized pairs of students, using each metric. Objective 3, and objectives 7-9 were omitted, as too few students completed them.

5. DISCUSSION

We have attempted to apply a meaningful canonicalization to our state space, which did serve to reduce the number of states by 30.4%. However, as seen in Table 1, even after the strongest canonicalization, over 90% of the states in the interaction network had only been reached by one student, with an average 78.9% of the states in each student's solution path being unique to that student. It seems that our approach to canonicalization is insufficient to produce a meaningful reduction of the state space, though it is possible a more stringent canonicalization would be more effective.

Connecting existing states seems to be a more promising approach, giving us the ability to link new states to previously observed states, even when they do not match exactly. Our distance matrices indicate that some students take parallel, or slowly diverging solution paths, which suggests that they may be useful to each other in the context of hinting. As shown in Figure 2, students are often closest together when completing the same objective. This may seem self-evident, but it does indicate that our distance metric is meaningful. It is more difficult to put the actual TED values into context. Students get, on average, farther away from their closest paired student as they complete more objectives, but this average distance does not exceed 8 tree edits during the first 6 objectives. To put that number into context, during this same time students do not, on average, get more than 19 tree edits away from the start state. This suggest that there is certainly hint-relevant knowledge in these paired students, but that it may be difficult to harness this knowledge to generate a hint.

5.1 Limitations

It is important to note that this analysis is an exploratory case study, and makes no strong claims, only observations. We studied data from only 17 novice programmers, and the problem we analyzed was highly complex, involving multiple control structures, loosely ordered objectives, and unstructured output. This makes the problem quite dissimilar from previous problems that have been been studied in the context of hint generation, making it difficult to determine what observations should be generalized.

5.2 Future Work

Rivers and Koedinger [16], as well as Jin et al. [9] note the limitations of their methods for larger problems, and each suggest that breaking a problem down into subproblems would help to address this. Lazar and Bratko [10] attempted this in their Prolog tutor by constructing hints for individual lines of code, which were treated as independent subproblems. Similarly, we may be able to isolate the subsection of a student's code that is currently relevant, and treat this as an independent problem. The hierarchical nature of block-based coding environments lends itself to this practice, making it an appealing direction for future work.

Our work makes the simplifying assumption that unweighted TED is a reliable distance metric for code, but future work should investigate alternative metrics. This might include a weighted TED metric, which assigns different costs to insertions, deletions and replacements, or even to different types of nodes (e.g. deleting a for-loop node might cost more than a function call node). Regardless of the metric used, once two proximate states are identified, it is still an open question how this information can be best used for hint generation. It is possible to construct a path between the states and direct a student along this path. However, future work might also investigate how to extract hint-relevant information from one state and apply it to a similar state directly.

Because of the nature of our problem's output, we did not explore non-code-based state representations, as described in Section 2.3. It would still be worth investigating how this might be applied to open-ended problems. For instance, a code state could be represented as a boolean vector, indicating whether the code has passed a series of Unit Tests, and hints could direct the student to the current flaw in their program. However, creating actionable hints from this information would pose a significant challenge.

6. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.

[2] T. Barnes, J. Stamper, L. Lehman, and M. Croy. A pilot study on logic proof tutoring using hints generated from historical student data. In *Proceedings of the 1st Annual International Conference on Educational Data Mining (EDM)*, pages 1–5, 2008.

[3] V. Catete, K. Wassell, and T. Barnes. Use and development of entertainment technologies in after school STEM program. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 163–168, 2014.

[4] M. Eagle, M. Johnson, and T. Barnes. Interaction Networks: Generating High Level Hints Based on Network Community Clustering. In *International Educational Data Mining Society*, pages 164–167, 2012.

[5] D. Fossati, B. D. Eugenio, and S. Ohlsson. I learn from you, you learn from me: How to make iList learn from students. In *Artificial Intelligence in Education (AIED)*, 2009.

[6] D. Garcia, B. Harvey, L. Segars, and C. How. AP CS Principles Pilot at University of California, Berkeley. *ACM Inroads*, 3(2), 2012.

[7] A. Hicks, B. Peddycord III, and T. Barnes. Building Games to Learn from Their Players: Generating Hints in a Serious Game. In *Intelligent Tutoring Systems (ITS)*, pages 312–317, 2014.

[8] M. Homer and J. Noble. Combining Tiled and Textual Views of Code. In *Proceedings of 2nd IEEE Working Conference on Software Visualization*, 2014.

[9] W. Jin, T. Barnes, and J. Stamper. Program representation for automatic hint generation for a data-driven novice programming tutor. In *Intelligent Tutoring Systems (ITS)*, 2012.

[10] T. Lazar and I. Bratko. Data-Driven Program Synthesis for Hint Generation in Programming Tutors. In *Intelligent Tutoring Systems (ITS)*. Springer, 2014.

[11] M. Pawlik and N. Augsten. RTED: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[12] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.

[13] C. Piech, M. Sahami, J. Huang, and L. Guibas. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Learning at Scale (LAS)*, 2015.

[14] M. Resnick, J. Maloney, H. Andrés, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[15] K. Rivers and K. Koedinger. A canonicalizing model for building programming tutors. In *Intelligent Tutoring Systems (ITS)*, 2012.

[16] K. Rivers and K. Koedinger. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013.

[17] K. Rivers and K. Koedinger. Automating Hint Generation with Solution Space Path Construction. In *Intelligent Tutoring Systems (ITS)*, pages 329–339, 2014.

[18] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *Artificial Intelligence in Education (AIED)*, 22(1):3–17, 2013.

Studio: Ontology-Based Educational Self-Assessment

Christian Weber
Corvinno Technology Transfer Center
Budapest, Hungary
cweber@corvinno.com

Réka Vas
Corvinus University of Budapest
Budapest, Hungary
reka.vas@uni-corvinus.hu

ABSTRACT

Students, through all stages of education, grasp new knowledge in the context of knowledge memorized all through their previous education. To self-predict personal proficiency in education, self-assessment acts as an important learning feedback. The in-house developed Studio suit for educational self-assessment enables to model the educational domain as an ontology-based knowledge structure, connecting assessment questions and learning material to each element in the ontology. Self-assessment tests are then created by utilizing a sub-ontology, which frames a tailored testing environment fitting to the targeted educational field. In this paper we give an overview of how the educational data is modeled as a domain ontology and present the concepts of different relations used in the Studio system. We will deduct how the presented self-assessment makes use of the knowledge structure for online testing and how it adapts the test to the performance of the student. Further we highlight where potentials are for the next stages of development.

Keywords

Education, adaptive test, self-assessment, educational ontology

1. INTRODUCTION

Students exploring new fields of education are always confronted with questions regarding their individual progress: how much do they know after iterations of learning, in which directions should they progress to fill the field most effectively, how to grasp the outline and details of the field and how much of their time should they invest in learning? Especially in higher education, where learning becomes a self-moderated, personalized process, students are in need of continuous self-assessment to capture their current state of proficiency. At the same time, the unframed, informal self-prediction of students regarding their personal skills is often substantive and systematically flawed [1]. Here a systematic and objective solution for self-assessment is substantial to prevent a wrong or biased self-evaluation and to support the self-prediction of the personal proficiency.

Following Jonassen, knowledge in education could be split into nine types across three categories to capture the human's cognitive behavior. In his discussion, eight out of nine knowledge types underline that knowledge in the scope of learning is interrelated and strongly associated with previous experiences [2]. As such, a supporting solution for self-assessment should grasp and formalize the knowledge to assess in the context of related knowledge.

The Studio suit for educational self-assessment, presented in this paper, provides here a software solution for testing the personal proficiency in the context of related knowledge. It enables to model areas of education as a substantial source for assessment and narrows the gap between a potentially flawed self-prediction and the real proficiency, by offering an objective and adaptive online knowledge-test. To follow the natural learning process and enable an easy extension, the software embeds the assessed knowledge into a network of contextual knowledge, which enables to adapt the assessment to the responses of the students.

This paper will give an overview of the Studio educational domain ontology and the aspects of the system supporting personalized self-assessment. Further it will highlight potentials for data mining on the gathered educational data with an outlook on the next stages of evaluation.

2. THE STUDIO APPROACH FOR SELF-ASSESSMENT

The basic concept of Studio is to model the focused education as an interrelated knowledge structure, which divides the education into sub-areas and knowledge items to know. The managed structure formalizes the relation between knowledge areas as a learning context and models the requirements to master specific parts of the education. This structure is used to create and support knowledge tests for students. Through this combination of assessment and knowledge structure, the student gains the freedom to explore not only single knowledge items but the education in the context of related knowledge areas, while the embedded requirements are used to map the modeled knowledge against the expected educational outcome.

The assessment-system is designed to be accompanied by phases of learning within the system, where the student gets access to learning material, based on and supported by the test feedback. This combined approach offers a unique self-assessment to the students, where the backing knowledge context is used to adapt the assessment in dependency of the test performance of the student.

Before any regular examination students may use Studio to assess their knowledge on their own. It is the tutor's responsibility to set the course of self-assessment test in Studio system by selecting knowledge areas and sub-knowledge areas which are relevant for the target education from the domain ontology. Then the frame will be automatically completed with elements from the ontology which detail the selected knowledge areas and are modeled as required for this part of the education.

As the system stores assessment questions for each knowledge element, Studio will then automatically prepare an assessment test, based on the defined selection and the domain ontology. The resulting knowledge-test is then accessible as a self-assessment test for the student, who explores the backed knowledge structure, which pictures the expected learning outcome, in cycles of testing, reflection and learning. The process of test definition and assessment is shown in Figure 1, while the result preparation for reflection and learning is discussed in section 2.5.

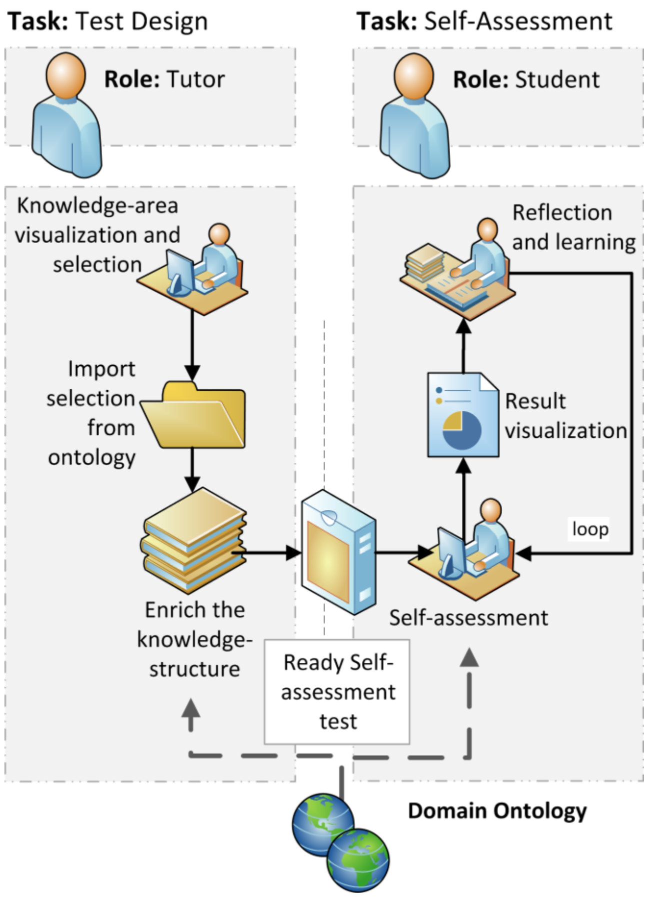


Figure 1: The overall design, assess and reflection cyle of the system.

2.1 The Educational Domain Ontology

The Studio system is based on a predesigned educational ontology, explained in detail by Vas in [3]. Domain ontology is a frequently used term in the field of semantic technologies and underlines the storage and conceptualization of domain knowledge and is often used in a number of projects and solutions [4][5][6] and could address a variety of domains with different characteristics in their creation, structure and granularity, depending on the aim and the modeling person [7]. A specialization in terms of the field is the educational domain ontology which is a domain ontology adapted to the area and concepts of education. They could target to model different aspects of education as the curriculum or aspects relevant for the task of learning and course creation [8][9][10] or describe the design, use and retrieval of learning materials till creating courses [11], as well as directly the learner within the education [12].

Within the area of educational ontologies, domain ontologies tend to model too specific details of the education, in an attempt to model the specific field as complete as possible. This enables a comprehensive view on the field but it comes at the cost of generality, with the potential to be inflexible to handle changes. Other concepts model the education across different ontologies, matching concepts like the learner, the education and the course description, introducing a broad horizon but with additional overhead to combine modelled insights and reason on new instances.

The appeal of the Studio educational ontology is the size and focus of the main classes and their relationships between each other. The knowledge to learn is the main connecting concept in the core of education. It enables a great flexibility to be resourceful for different education related questions. An example is here the business process management extension PROKEX, which maps process requirements against knowledge areas to create assessment test, reflecting the requirements of attached processes [13].

An important factor in learning is the distance between the expectation of the tutor and the learning performance of the student. Here a short cycle of repeated assessment and learning is a major factor for a better personal learning performance [14]. This aspect directly benefits from the focused concentration on knowledge-areas as the main exchange concept between students and tutors. As even further the close connections between learners and educators via direct tutoring is one major enabler for computer aided systems [15], each step towards a more direct interaction through focused concepts is an additional supporter.

The class structure fuses the idea of interrelated knowledge with a model of the basic types of educational concepts, involved in situations of individual learning. Figure 2 visualizes the class concepts as knowledge elements, together with the relation types, used to model the dependencies between different aspects of knowledge and learning within the educational ontology.

The Knowledge Area is the super-class and core-concept of the ontology. The ontology defines two qualities of main relations between knowledge areas: Knowledge areas could be a sub-knowledge area of other knowledge areas with the “has_sub-knowledge_area” relation or be required for another knowledge area with the “requires_knowledge_of” relation. A knowledge area may have multiple connected knowledge areas, linked as a requirement or sub-area. The “requires_knowledge_of” relation defines that a node is required to complete the knowledge of a parent knowledge area. This strict concept models a requirement dependency between fields of knowledge in education and yields the potential to assess perquisites of learning, analog to the basic idea of perquisites within knowledge spaces, developed by Falmagne [16].

Education is a structured process which splits the knowledge to learn into different sub-aspects of learning. Knowledge areas in the ontology are extended by an additional sub-layer of knowledge elements in order to effectively support educational and testing requirements. Figure 2 visualizes the sub-elements and their relations. By splitting the assessed knowledge into sub-concepts, the coherence and correlation of self-assessment questions could be expressed more efficiently and with the potential of a more detailed educational feedback.

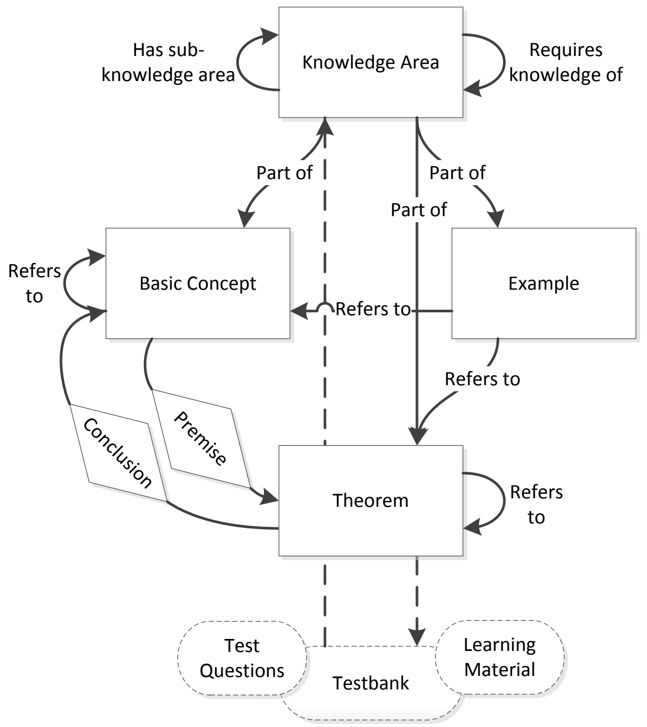


Figure 2: Model of the educational ontology.

Theorems express in a condensed and structured way the fundamental insights within knowledge areas. They fuse and explain the basic concepts of the depicted knowledge and set them in relation to the environment of learning with examples. Multiple theorems could be "part_of" a knowledge area. Each theorem may define multiple Basic Concepts as a "premise" or "conclusion", to structure how the parts of the knowledge area are related. Examples enhance this parts as a strong anchor for self-assessment questions and "refer_to" the theorems and basic concepts as a "part_of" one or more knowledge areas.

2.2 The Testbank

In order to connect the task of self-assessment with the model of the educational domain, the system integrates a repository of assessment questions. Each question addresses one element of the overall knowledge and is directly associated with one knowledge area or knowledge element instance within the ontology. The domain ontology provides here the structure for the online self-assessment while the repository of questions supplements the areas as a test bank. The target of the self-assessment is to continuously improve the personal knowledge within the assessed educational areas, by providing feedback on the performance after each phase of testing. To do so, the Studio system includes Learning Material connected to the test bank and the knowledge areas, analog to the test questions. The learning material is organized into sections as a structured text with mixed media, as pictures and videos, and is based on a wiki-engine to maintain the content, including external links.

2.3 Creating and Maintaining Tests

The creation and continues maintenance of the domain ontology is a task of ontology engineering. The ontology engineer (the ontologist), creates, uses and evaluates the ontology [17], with a strong focus on maintaining the structure and content. Within Studio, this process is guided and supported by a specialized administration workflow and splits in three consecutive task areas, in line with decreasing access rights:

- **Ontology engineering (instance level):** The creation and linking of instances of the existing knowledge-area classes into the overall domain ontology.
- **Test definition:** Knowledge areas, which are relevant to a target self-assessment test, are selected and grouped into specialized containers called Concept Groups (CG). These concept groups are organized into a tree of groups, in line with the target of the assessment. The final tree in this regards captures a sub-ontology. Concept groups are internally organized based on the overall ontology and include all relations between knowledge elements, as defined within the domain ontology.
- **Question and learning material creation:** Questions and learning materials alike are directly connected to single knowledge areas within the designed test frame and get imported, if already existing, from the domain ontology. More questions and learning materials are defined now, in line with the additional need of the targeted education and are available for future tests.

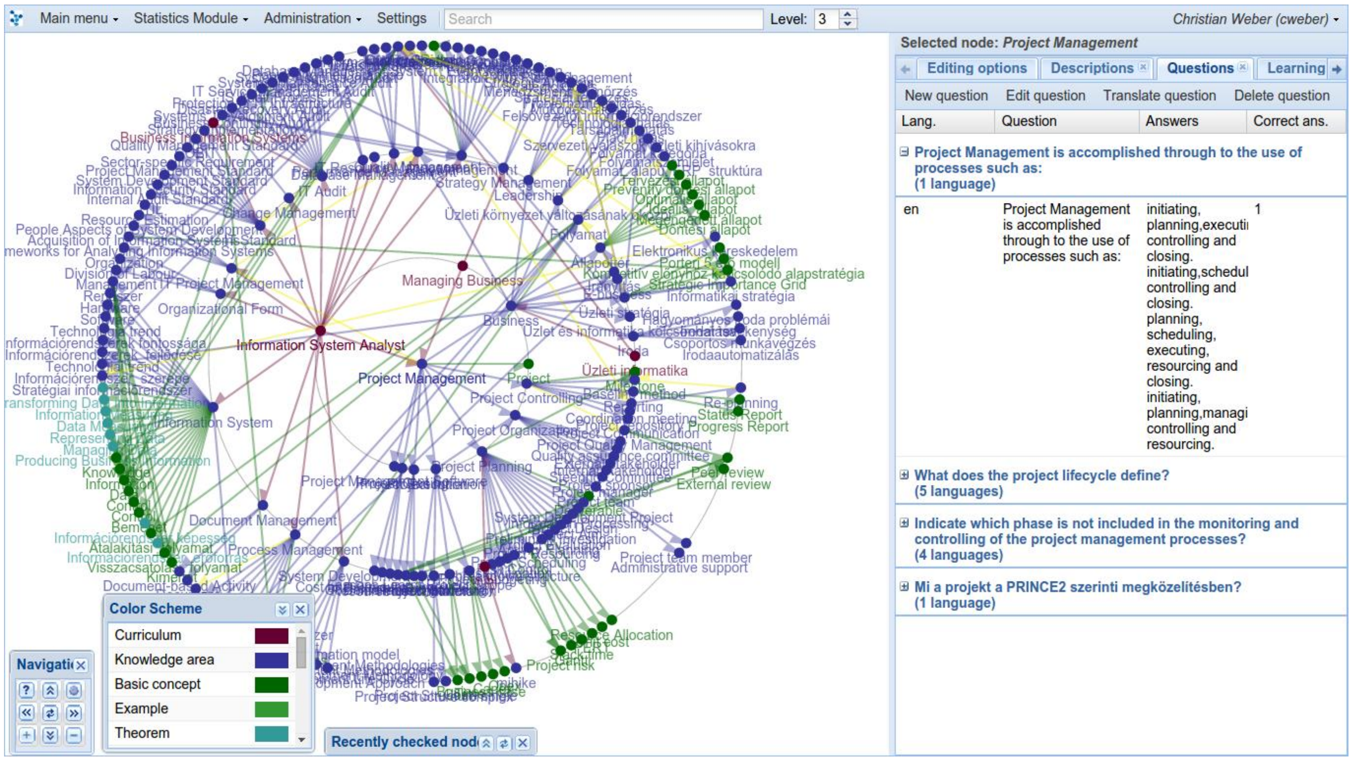
The pre-developed structure of classes and relations is fixed as the central and integral design of the system. A view of the system interface for administration is provided in Figure 3. The left area shows the visualization of the current ontology section in revision and the right area shows the question overview with editing options. Tabs give access to additional editing views, including the learning material management and interfaces to modify relations between nodes and node descriptions.

2.4 Adaptive Self-Assessment

To prepare an online self-assessment test, the system has to load the relevant educational areas from the domain ontology and extract the questions and relations of the filtered knowledge areas.

The internal test algorithm makes use of two assumptions:

- **Knowledge-area ordering:** As the main knowledge areas are connected through "requires_knowledge_of" and "part_of" relations, every path, starting with the start-element, will develop on average from general concepts to detailed concepts - given that the concept groups in the test definition are also selected and ordered to lead from general to more detailed groups.
- **Knowledge evaluation dependency:** If a person, taking the test, fails on general concepts he or she will potentially also fail on more detailed concepts. Further, if a high number of detailed concepts are failed, the parent knowledge isn't sufficiently covered and will be derived as failed, too.



Selected node: *Project Management*

Editing options | Descriptions | Questions | Learning

New question | Edit question | Translate question | Delete question

Lang.	Question	Answers	Correct ans.
Project Management is accomplished through to the use of processes such as: (1 language)			
en	Project Management is accomplished through to the use of processes such as:	initiating, planning,executi controlling and closing. initiating,schedul controlling and closing. planning, scheduling, executing, resourcing and closing. initiating, planning,managi controlling and resourcing.	1
What does the project lifecycle define? (5 languages)			
Indicate which phase is not included in the monitoring and controlling of the project management processes? (4 languages)			
Mi a projekt a PRINCE2 szerinti megközelítésben? (1 language)			

Figure 3: The main ontology maintenance and administration interface, showing a part of the domain ontology.

The filtering is done based on the selection of a tutor, acting as an expert for the target educational area. The tutor chooses related areas, which are then created as a Test Definition, containing Concept Groups, as described in section 2.3. The system then uses the test definition as a filtering list to extract knowledge areas. After the extraction, the structure is cached as a directed graph, while the top element of the initial concept group is set as a start element. Beginning with the start-element, the test will move then through the graph, while administering the questions connected to knowledge areas and knowledge elements.

The loading of knowledge-elements follows three steps:

1. Each type of relation between two knowledge-elements implements a direction for the connection. Assuming the system loads all relations, starting with the start-element and ending on a knowledge-element, this creates a two level structure where the start-node is a parent-element and all related, loaded elements are child-elements, as seen below in Figure 4.
2. The loading algorithm then selects one child-element and assumes it as a start-element and repeat the loading process of knowledge-elements.
3. When no knowledge-elements for a parent-element could be loaded, the sub-process stops. When all sub-processes have stopped, the knowledge structure is fully covered.

The test algorithm will now activate the child knowledge areas of the start element and select the first knowledge area to the left and draw a random question from the selected knowledge area. If the learner fails the question, the algorithm will mark the element as failed and selects the next knowledge area from the same level. If the learner's answer is correct, the system will activate the child elements of the current node and draw a random question from the first left child.

Based on the tree shaped knowledge structure, the assessment now follows these steps to run the self-assessment, supported by the extracted knowledge structure:

1. Starting from the start-element, the test algorithm will activate the child knowledge-areas of the start element.
2. The algorithm now selects the first child-knowledge area and draws a random question out of the pool of available questions for this specific knowledge-element from the test bank.
3. If the learner fails the question, the algorithm will mark the element as failed and select the next knowledge area from the same level. If the learner's answer is correct, the system will activate the child elements of the current node and trigger the process for each child-element.

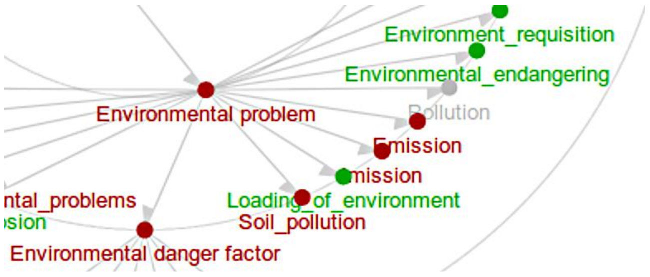
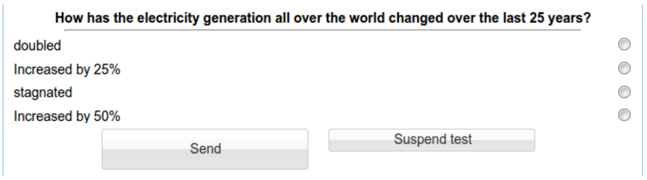


Figure 4: Excerpt from the sub-ontology visualization, with the visible parent-child relationship, as used in the data-loading for preparing the self-assessment.

An example question is shown below in Figure 5. Further following the testing algorithm, the system dives down within the domain ontology and triggers questions depending on the learner's answers and the extracted model of the relevant education. In this regards the Studio system adapts the test on the fly to the performance of the learner. Correlating to the idea of adaptation, the learner will later gain access to learning material for each mastered knowledge area. As the learner continues to use the self assessment to evaluate the personal knowledge, he or she will thus explore different areas of the target education, following their individual pace of learning.



How has the electricity generation all over the world changed over the last 25 years?

- doubled ○
- Increased by 25% ○
- stagnated ○
- Increased by 50% ○

Figure 5: Test interface with a random drawn test question.

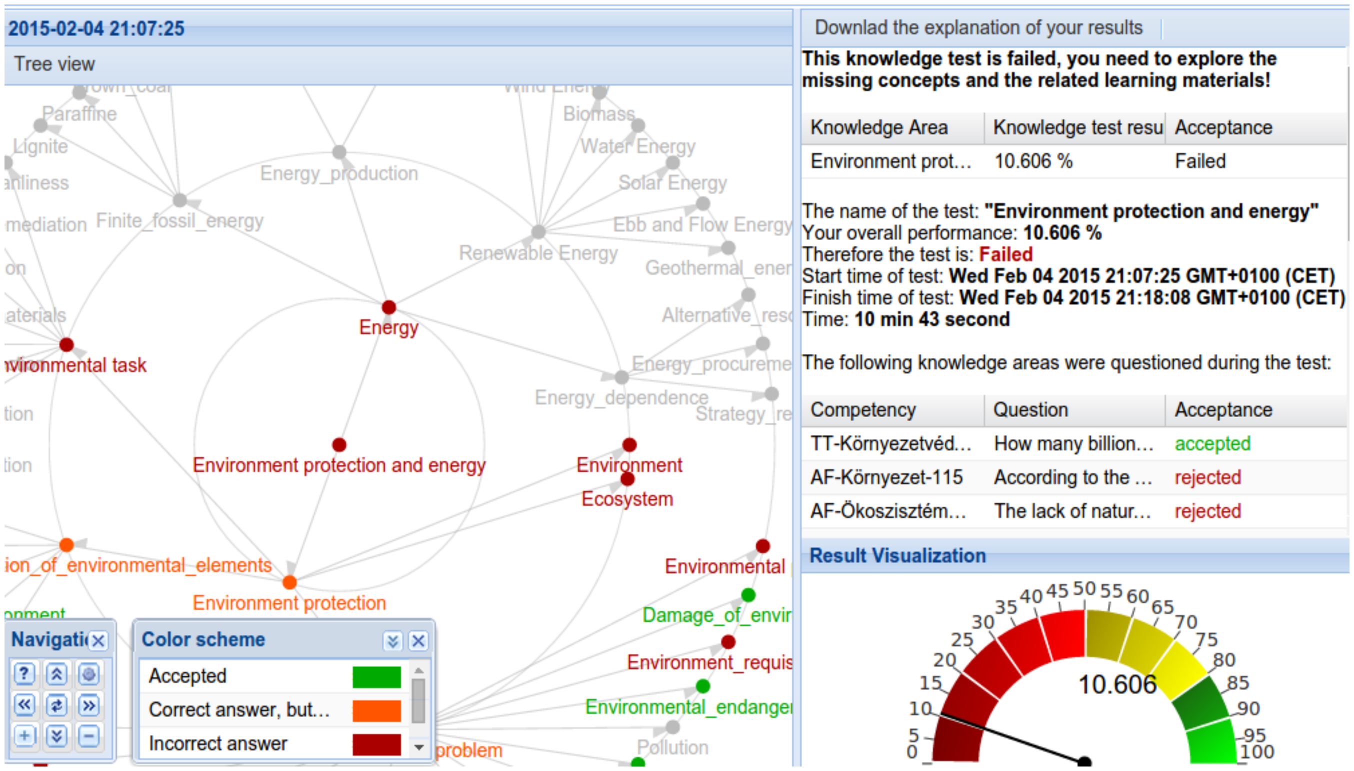
2.5 Test Feedback and Result Visualization

An important aspect of the system is the test feedback and evaluation interface. The educational feedback is one of the main enabler for the student to grasp the current state and extend of the personal education. The domain ontology models the structure and the dependencies of the educational domain, and the grouped test definition extracts the relevant knowledge for the target area or education. As such, the visualization of the ontology structure extracted for the test, together with the indication of correct and incorrect answers, represents a map of the knowledge of the learner.

Throughout each view onto the ontology, the system uses the same basic visualization, making use of the Sencha Ext JS JavaScript framework [18]. The visualization itself is a custom build, similar to the Ext JS graph function "Radar" and based on the idea of Ka-Ping, Fisher, Dhamija and Hearst [19]. All views are able to zoom in and out of the graph, move the current excerpt and offer a color code legend, explaining the meaning of the colored nodes. In comparison with state of the art, the interface offers no special grouping or additional visualization features like coding information into the size of nodes. Each interface offers an additional textual tree view to explore the knowledge-elements or concept groups in a hierarchical listing. This simple, straightforward approach for visualization correlates with the goal of a direct and easy to grasp feedback through interfaces which have a flat learning curve and enable to catch the functionality in a small amount of time.

While this simple visualization is sufficient for the reasonable amount of knowledge-elments within the result view, this alone is not suitable for the domain ontology administration interface, as seen in Figure 3. Here Studio realizes methodologies to filter and transform the data to visualize. To do so it makes use of two supporting mechanisms:

- The **maximum-level-selector** defines the maximum level the system extracts from the domain ontology for full screen visualization.
- In combination with the maximum level, the ontologist could select single elements within the domain ontology. This triggers an **on-demand re-extraction of the visualized data**, setting the selected knowledge-element as the centre element. The system then loads the connected nodes, based on their relations into the orientation circles till the maximum defined level is reached. More details about the transformation are in [19].



Color scheme	
Accepted	🟩
Correct answer, but...	🟧
Incorrect answer	🟥

This knowledge test is failed, you need to explore the missing concepts and the related learning materials!

Knowledge Area	Knowledge test resu	Acceptance
Environment prot...	10.606 %	Failed

The name of the test: **"Environment protection and energy"**
Your overall performance: **10.606 %**
Therefore the test is: **Failed**
Start time of test: **Wed Feb 04 2015 21:07:25 GMT+0100 (CET)**
Finish time of test: **Wed Feb 04 2015 21:18:08 GMT+0100 (CET)**
Time: **10 min 43 second**

The following knowledge areas were questioned during the test:

Competency	Question	Acceptance
TT-Környezetvéd...	How many billion...	accepted
AF-Környezet-115	According to the ...	rejected
AF-Ökoszisztém...	The lack of natur...	rejected

Figure 6: Result visualization as educational feedback for the learner.

Together, this selection and transformation mechanism enables the fluent navigation within the complete domain ontology structure, while re-using the same visualization interface.

Figure 6 shows the main view of the result interface. The left area shows the sub-ontology extracted for the test, while the colored nodes represent the answers to the administered questions. A red node visualizes wrong answers, while orange nodes are rejected nodes with correct answers but with an insufficient number of correctly answered child nodes, indicating a lack of the underlying knowledge. Green nodes represent accepted nodes with correct answers and a sufficient amount of correctly answered questions for child nodes. Grey nodes are not administered nodes, which were not yet reached by the learner, as higher order nodes had no adequate acceptance.

Even though the target of the system is not a strict evaluation in number, the evaluation of the percentage of solved and accepted knowledge elements helps the learner to track the personal progress and could additionally be saved as a report for further consultation. Besides providing an overview of the self-assessment result, the result interface gives access to the integrated learning material. For every passed node, the learner can now open the correlated material and intensify the knowledge for successful tested areas.

Retaking the test in cycles of testing and learning, while adapting the educational interaction, is the central concept of the Studio approach for self-assessment. As a consequence the system will not disclose the right answers to questions or learning material for not yet administered knowledge areas, to promote an individual reflection on the educational content outside of a flat memorization of content.

3. SYSTEM EVALUATION

The system has been used, extended and evaluated in a number of European and nationally funded research projects, including applications in business process management and innovation-transfer [20], medical education [21] and job market competency matching [22].

Currently the system is being evaluated based on a running study with 200 university students in the field of business informatics. The study will conclude on two current research streams which are improving the systems testing and analysis capability. The first direction looks into potentials for the integration of learning styles into adaptive learning systems to offer valuable advice and instructions to teachers and students [23]. Within the second direction the question is challenged on how to adapt the presented self-assessment further towards the performance of the students, based on extracting assessment paths from the knowledge structure [24].

For each running test, Studio collects basic quantitative data about the number of assigned questions, how often tests are taken and how many students open which test and when. This is completed by qualitative measures, collecting which questions and knowledge elements the students passed or failed. To conclude further on the mechanisms and impacts of Studio within the current study, a new logging system was developed, collecting the interaction with the system and detailed information about the feedback as detailed events.

Each event stores information about the system in 7 dimensions, as described in Table 1 below:

Table 1: Event blueprint to store events concerning system interaction.

Attribute	Description
Event description code	Which type of event and what factors are relevant.
Location code	On which part of the assessment-process or interface the event has occurred.
Session identifier	Each access of the system is one session for one user.
Numerical value storage	Multi-purpose field, filled depending on the event type.
String value storage	Multi-purpose field, filled depending on the event type.
Event-time	The time of the start of the event.
Item reference	A unique reference code, identifying the correlated item within the ontology. E.g. a question or a knowledge-element ID.

All events are stored in order of their occurrence, so if no explicit end event is defined, the next event for the same session and user is acting as the implicit end date. Extending the existing storage of information within Studio, the new logging system stores the additional events, as shown in Table 2 below:

Table 2: Assessment events and descriptions.

Event type	Description
START_TEST	Marks the start of a test.
END_TEST	Marks the end of a test.
OPEN_WELCOME_LM	The user opened the welcome page.
OPEN_LM_BLOCK	The student opened a learning material block on the test interface.
OPEN_LM	The student opened the learning material tab on the test interface.
RATE_LM	The student rated the learning material.
CHECK_RESULT	The student opened a result page.
CONTINUE_TEST	The student submitted an answer.
FINISH_TEST	The test has been finished.
SUSPEND_TEST	The user suspended the test.
RESUME_TEST	The user has restarted a previously suspended test.
SELECT_TEST_ALGO-RITHM	The algorithm used to actually test the student is selected.

TEST_ALGORITHM_-EVENT	The behavior of the current test algorithm changes, e.g. entering another stage of testing.
ASK_TESTQUESTION	Sends out a test question to the user to answer.
STUDIO_LOGOUT	The user logs out of the Studio system.

To store the events, the system implements an additional logging database, splitting the concepts of the logging to a star-schema for efficient extraction, transformation and loading. The logging system is modular and easy to extend with new concepts and easy to attach to potential event positions within the Studio runtime. Together with the existing logging of the assessment evaluation feedback, this new extension tracks the exploration of the sub-ontology within the assessment and enriches the feedback data with context information of the students behavior on the system.

4. NEXT STEPS

The domain ontology offers a functional and semantically rich core for supporting learning and education. Yet not all the semantic potentials are fully leveraged to support and test the learner's progress. The "requires_knowledge_of" relation-requirement is a potential start-concept to model sub-areas as groups which together compose the dependency. This could act as an additional input for the assessment, where the system derives more complex decision how to further explore the related parts of the structure [25]. This could also be visualized, enabling the learner to grasp the personal knowledge as a visible group of concepts.

Besides giving colors to the different types of relations, the visualizing of edges between knowledge areas is yet unfiltered, offering no further support for navigation. A next stage of implementation could be the introduction of a visual ordering and grouping of knowledge areas and relations. Underlying relations of sub-nodes could be interpreted visually through the thickness of relations between nodes, easing the perception of complex parts of the domain ontology, especially within administration and maintenance tasks.

The feedback of the current evaluation study of Studio will provide additional insights into the usage of the system by the students. Based on this new data it is possible to mine profiles over time on the knowledge structure. One major application is here the creation of behavior profiles, as proposed in [23].

5. REFERENCES

[1] D. Dunning, C. Heath, and J. M. Suls, "Flawed self-assessment implications for health, education, and the workplace," *Psychological science in the public interest*, vol. 5, no. 3, pp. 69–106, 2004.

[2] D. Jonassen, "Reconciling a Human Cognitive Architecture," in *Constructivist Instruction: Success Or Failure?*, S. Tobias and T. M. Duffy, Eds. Routledge, 2009, pp. 13–33.

[3] R. Vas, "Educational Ontology and Knowledge Testing," *Electronic Journal of Knowledge Management*, vol. 5, no. 1, pp. 123 – 130, 2007.

[4] M. Y. Dahab, H. A. Hassan, and A. Rafea, "TextOntoEx: Automatic ontology construction from natural English text," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1474–1480, Feb. 2008.

[5] S.-H. Wu and W.-L. Hsu, "SOAT: a semi-automatic domain ontology acquisition tool from Chinese corpus," in *Proceedings of the 19th international conference on Computational linguistics-Volume 2*, 2002, pp. 1–5.

[6] M. Missikoff, R. Navigli, and P. Velardi, "The Usable Ontology: An Environment for Building and Assessing a Domain Ontology," in *The Semantic Web — ISWC 2002*, I. Horrocks and J. Hendler, Eds. Springer Berlin Heidelberg, 2002, pp. 39–53.

[7] T. A. Gavrilova and I. A. Leshcheva, "Ontology design and individual cognitive peculiarities: A pilot study," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3883–3892, May 2015.

[8] S. Sosnovsky and T. Gavrilova, "Development of Educational Ontology for C-programming," 2006.

[9] V. Psyché, J. Bourdeau, R. Nkambou, and R. Mizoguchi, "Making learning design standards work with an ontology of educational theories," in *12th Artificial Intelligence in Education (AIED2005)*, 2005, pp. 539–546.

[10] T. Nodenot, C. Marquesuzaá, P. Laforcade, and C. Sallaberry, "Model Based Engineering of Learning Situations for Adaptive Web Based Educational Systems," in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, New York, NY, USA, 2004, pp. 94–103.

[11] A. Bouzeghoub, C. Carpentier, B. Defude, and F. Duitama, "A model of reusable educational components for the generation of adaptive courses," in *Proc. First International Workshop on Semantic Web for Web-Based Learning in conjunction with CAISE*, 2003, vol. 3.

[12] W. Chen and R. Mizoguchi, "Learner model ontology and learner model agent," *Cognitive Support for Learning-Imagining the Unknown*, pp. 189–200, 2004.

[13] G. Neusch and A. Gábor, "PROKEX – INTEGRATED PLATFORM FOR PROCESS-BASED KNOWLEDGE EXTRACTION," *ICERI2014 Proceedings*, pp. 3972–3977, 2014.

[14] H. L. Roediger III, "Relativity of Remembering: Why the Laws of Memory Vanished," *Annual Review of Psychology*, vol. 59, no. 1, pp. 225–254, 2008.

[15] J. D. Fletcher, "Evidence for learning from technology-assisted instruction," *Technology applications in education: A learning view*, pp. 79–99, 2003.

[16] J.-C. Falmagne, M. Koppen, M. Villano, J.-P. Doignon, and L. Johannesen, "Introduction to knowledge spaces: How to build, test, and search them," *Psychological Review*, vol. 97(2), pp. 201–224, 1990.

[17] F. Neuhaus, E. Florescu, A. Galton, M. Gruninger, N. Guarino, L. Obrst, A. Sanchez, A. Vizedom, P. Yim, and B. Smith, "Creating the ontologists of the future," *Applied Ontology*, vol. 6, no. 1, pp. 91–98, 2011.

[18] "Ext JS API." [Online]. Available: http://www.objis.com/formationextjs/lib/extjs-4.0.0/docs/index.html. [Accessed: 04-May-2015].

[19] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," in *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001*, 2001, pp. 43–50.

[20] M. Arru, “Application of Process Ontology to improve the funding allocation process at the European Institute of Innovation and Technology.,” in *3rd International Conference on Electronic Government and the Information Systems Perspective (EGOVIS).*, Munich, Germany, 2014.

[21] M., M., Ansari, F., Dornhöfer Khobreh and M. Fathi, “An ontology-based Recommender System to Support Nursing Education and Training,” in *LWA 2013*, 2013.

[22] V. Castello, L. Mahajan, E. Flores, M. Gabor, G. Neusch, I. B. Szabó, J. G. Caballero, L. Vettraino, J. M. Luna, C. Blackburn, and F. J. Ramos, “THE SKILL MATCH CHALLENGE. EVIDENCES FROM THE SMART PROJECT,” *ICERI2014 Proceedings*, pp. 1182–1189, 2014.

[23] H. M. Truong, “Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities,” *Computers in Human Behavior*, 2015.

[24] C. Weber, “Enabling a Context Aware Knowledge-Intense Computerized Adaptive Test through Complex Event Processing,” *Journal of the Sientific and Educational on Forum on Business Information Systems*, vol. 9, no. 9, pp. 66–74, 2014.

[25] C. Weber and R. Vas, “Extending Computerized Adaptive Testing to Multiple Objectives: Envisioned on a Case from the Health Care,” in *Electronic Government and the Information Systems Perspective*, vol. 8650, A. Kő and E. Francesconi, Eds. Springer International Publishing, 2014, pp. 148–162.

Graph Analysis of Student Model Networks

Julio Guerra
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA
jdg60@pitt.edu

Yun Huang
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA, USA
yuh43@pitt.edu

Roya Hosseini
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA, USA
roh38@pitt.edu

Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA
peterb@pitt.edu

ABSTRACT

This paper explores the feasibility of a graph-based approach to model student knowledge in the domain of programming. The key idea of this approach is that programming concepts are truly learned not in isolation, but rather in combination with other concepts. Following this idea, we represent a student model as a graph where links are gradually added when the student's ability to work with connected pairs of concepts in the same context is confirmed. We also hypothesize that with this graph-based approach a number of traditional graph metrics could be used to better measure student knowledge than using more traditional scalar models of student knowledge. To collect some early evidence in favor of this idea, we used data from several classroom studies to correlate graph metrics with various performance and motivation metrics.

1. INTRODUCTION

Student modeling is widely used in adaptive educational systems and tutoring systems to keep track of student knowledge, detect misconceptions, provide targeted support and give feedback to the student [2]. The most typical *overlay* student model dynamically represents the inferred knowledge level of the student for each *knowledge element* (KE) (also called *knowledge component* or KC) defined in a domain model. These knowledge levels are computed as the student answers questions or solves problems that are mapped to the domain KEs. Student models are frequently built over *networked* domain models where KEs are connected by prerequisite, is-a, and other ontological relationships that are used to propagate the knowledge levels and produce a more accurate representation of the knowledge of the learner. Since these connections belong to domain models, they stay the same for all students and at all times. In this work we explore the idea that it might be beneficial for a student model to include connections between domain KEs that represent some aspects of individual student knowledge rather than domain knowledge. This idea is motivated by the recognition that the mastery in many domains is reached as the student practices connecting different KEs, i.e., each KE is practiced in conjunction with other KEs. To address this, we build a model represented as a network of KEs that get progressively connected as the student successfully works with problems and assessment items containing multiple KEs. As the student succeeds in more diverse items mapped to different KEs, her model gets better connected.

To explore the value of this graph-based representation of student knowledge, we compute different graph metrics (e.g., density, diameter) for each student and analyze them in relation to student performance metrics and attitudinal orientations drawn from a motivational theory. This analysis was performed using data collected from 3 cohorts of a Java programming course using the same system and the same content materials. In the remaining part of the paper, we describe related work, introduce and illustrate the suggested approach, describe graph and performance metrics, and report the results of the correlation analysis.

2. RELATED WORK

Graph representation of student activity is not new. The 2014 version of the Graph-Based Educational Data Mining Workshop [1] contains two broad types of related work: the analysis of the networking interaction among students, for example work on social capital [14] and social networking in MOOCs [3, 12]; and analyses of learning paths over graph representation of student traces while performing activities in the system [1, 5]. Our work fits in the second type since we model traces of each student interacting with the system. However, our approach is different as it attempts to combine an underlying conceptual model with the traces of the student learning.

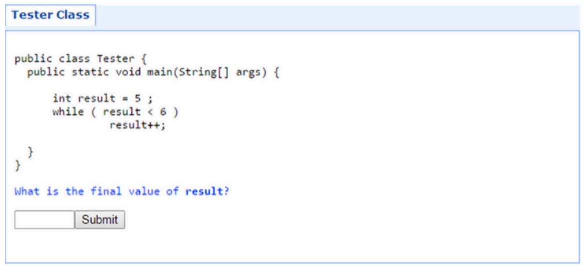
[1]`http://ceur-ws.org/Vol-1183/gedm2014_proceedings.pdf`

A considerable amount of work focused on graph representation of domain models that serve as a basis for overlay student models. The majority of this work focused on constructing the prerequisite relationships between domain knowledge components (concept, skills) [6, 13]. In this case links established between a pair of concepts represent prerequisite - outcome relationship. Another considerable stream of work explored the use of formal ontologies with such relationships as is-a and part-of for connecting domain knowledge components [7]. Ontological representation, in turn, relates to another stream of work that applies graph techniques to structural knowledge representation, for example by analyzing the network properties of ontologies [9].

The research on graph-based domain models also leads to a stream of work on using Bayesian networks to model the relationships between domain concepts for knowledge propagation in the process of student modeling [15, 4]. Yet, in both cases mentioned above links between knowledge components were not parts of individual student model, but either parts of the domain model or student modeling process and thus remain the same for all students. In contrast, the approach suggested in this paper adds links between knowledge components to *individual student models* to express combinations of knowledge components that the given student explored in a problem solving or assessment process. This approach is motivated by our belief that in the programming domain, student knowledge is more effectively modeled by capturing student progress when students needed to apply multiple concepts at the same time.

3. THE APPROACH

The idea behind our approach is that knowledge is likely to be stronger for concepts which are practiced together with a larger variety of other concepts. We hypothesize, for example, that a student who solves exercises, in which the concept *for-loop* is used with *post-incremental operator* and *post-decremental operator* will have a better understanding of *for-loop* than another student who practices (even the same amount of times) the *for loops* concept in a more narrow context, i.e., only with *post-incremental operator*. To represent our approach, for each student we build a graph-based student model as a network of concepts where the edges are created as the student succeeds in exercises containing both concepts to be connected. The Domain Model defining the concept space and the mapping between the concepts and programming exercises is explained in the next section. The weight of the edges in the graph is computed as the overall success rate on exercises performed by the student which contain the pair of concepts. Pairs of concepts that do not co-occur in exercises succeeded by the student are not connected in her graph. In this representation, highly connected nodes are concepts successfully practiced with different other concepts. We also compute a measure of *weight* for each node by taking the average weight among edges connecting the node. This measure of the success rate on concepts favors exercises that connect more concepts because each exercise containing n concepts produce or affects $n(n-1)/2$ edges. For example, a success on an exercise having 10 concepts contributes to 45 edges, but a successful attempt to an exercise connecting 5 concepts only contributes to 10 edges. We hypothesize that in a graph built following this approach, metrics like average degree, density, average path length, and average node weight can be good indicators of student knowledge compared to the amount of activities done or overall measures of assessment like success rate on exercises. We further explore these graph metrics in relation with motivational factors drawn from a learning motivation theory.



```
public class Tester {
  public static void main(String[] args) {

      int result = 5 ;
      while ( result < 6 )
              result++;

  }
}
```

What is the final value of result?

Figure 1: Exercise jwhile1

3.1 Domain Model and Content

Our content corpus is composed by a set of 112 interactive parameterized exercises (i.e., questions or problems) in the domain of Java programming from our system QuizJet [11]. Parameterized exercises are generated from a template by substituting a parameter variable with a randomly generated value. As a result each exercise can be attempted multiple times. To answer the exercise the student has to mentally execute a fragment of Java code to determine the value of a specific variable or the content printed on a console. When the student answers, the system evaluates the correctness, reports to the student whether the answer was correct or wrong, shows the correct response, and invites the student to "try again". As a result, students may still try the same exercises even after several correct attempts. An example of parameterized java exercise can be seen in Figure 1.

In order to find the concepts inside all of the exercises, we used a parser [10] that extracts concepts from the exercise's template code, analyzes its abstract syntax tree (AST), and maps the nodes of the AST (concepts extracted) to the nodes in a Java ontology [2]. This ontology is a hierarchy of programming concepts in the java domain and the parser uses only the concepts in the leaf nodes of the hierarchy.

In total there are 138 concepts extracted and mapped to QuizJet exercises. Examples of concepts are: *Int Data Type*, *Less Expression*, *Return Statement*, *For Statement*, *Subtract Expression*, *Constant*, *Constant Initialization Statement*, *If Statement*, *Array Data Type*, *Constructor Definition*, etc. We excluded 8 concepts which appear in all exercise templates (for example "Class Definition" or "Public Class Specifier" appear in the first line of all exercises). Each concept appears in one or more Java exercises. Each of the 112 exercises maps to 2 to 47 Java concepts. For example, the exercise "jwhile1", shown in Figure 1, is mapped to 5 concepts: *Int Data Type*, *Simple Assignment Expression*, *Less Expression*, *While Statement*, *Post Increment Expression*.

[2]`http://www.sis.pitt.edu/~paws/ont/java.owl`

3.2 Graph Metrics

To characterize the student knowledge graph we computed standard graph metrics listed below.

- **Graph Density** (density): the ratio of the number of edges and the number of possible edges.
- **Graph Diameter** (diameter): length of the longest shortest path between every pair of nodes.
- **Average Path Length** (avg.path.len): average among the shortest paths between all pairs of nodes.
- **Average Degree** (avg.degree): average among the degree of all nodes in an undirected graph.
- **Average Node Weight** (avg.node.weight): the weight of a node is the average of the weight of its edges. We then average the weights of all nodes in the graph.

3.3 Measures of Activity

To measure student activity so that it could be correlated with the graph metrics we collected and calculated the following success measures:

- **Correct Attempts to Exercises** (correct.attempts): total number of correct attempts to exercises. It includes repetition of exercises as well.
- **Distinct Correct Exercises** (dist.correct.attempts): number of distinct exercise attempted successfully.
- **Overall Success Rate** (success.rate): the number of correct attempts to exercises divided by the total number of attempts.
- **Average Success Rate on Concepts** (avg.concept.succ.rate): we compute the success rate of each concept as the average success rate of the exercises containing the concept. Then we average this among all concepts in the domain model.

3.4 Motivational Factors

We use the revised Achievement-Goal Orientation questionnaire [8] which contains 12 questions in a 7-point Likert scale. There are 3 questions for each of the 4 factors of the *Achievement-Goal Orientation framework*: Mastery - Approach, Mastery-Avoidance, Performance-Approach and Performance-Avoidance. **Mastery-Approach** goal orientation relates to intrinsic motivation: "I want to learn this because it is interesting for me", "I want to master this subject"; **Mastery-Avoidance** relates to the attitude of avoid to fail or avoid learning less than the minimum; **Performance -Approach** goal orientation stresses the idea of having a good performance and relates well with social comparison: "I want to perform good in this subject", "I want to be better than others here"; and **Performance-Avoidance** oriented students avoid to get lower grades or avoid to perform worse than other students. The goal orientation of a student helps to explain the behavior that the student exposes when facing difficulty, but does not label the final achievement of the student. For example, if a student is Mastery-Approach oriented, it does not necessarily mean that the student reached the mastery level of the skill or knowledge. In our case, we believe the achievement-goal orientation of the student can convey the tendency to pursue (or avoid) to solve more diverse (and more difficult) exercises, which contain more heterogeneous space of concepts, thus contribute to form better connected graphs.

Table 1: Correlation between activity measures and grade and between graph metrics and grade. * significance at 0.1, ** significance at 0.05.

Measure of Activity	Corr. Coeff.	Sig. (p)
Correct Attempts to Exercises	.046	.553
Distinct Correct Exercises	.114	.147
Overall Success Rate	.298	.000**
Avg. Success Rate on Concepts	.188	.016**

Graph Metric	Corr. Coeff.	Sig. (p)
Average Degree	.150	.055*
Graph Density	.102	.190
Graph Diameter	.147	.081
Average Path Length	.152	.052*
Average Node Weight	.201	.010**

4. EXPERIMENTS AND RESULTS

4.1 Dataset

We collected student data over three terms of a Java Programming course using the system: Fall 2013, Spring 2014, and Fall 2014. Since the system usage was not mandatory, we want to exclude students who just tried the system while likely using other activities (not captured by the system) for practicing Java programming. For this we looked at the distribution of distinct exercises attempted and we exclude all student below the 1st quartile (14.5 distinct exercises attempted). This left 83 students for our analysis. In total these students made 8,915 attempts to exercises. On average, students have attempted about 55 (Standard Deviation=22) distinct exercises while performing an average of 107 (SD=92) exercises attempts. On average, students have *covered* about 63 concepts with SD=25 (i.e., succeeded in at least one exercise containing the concept), and have covered about 773 concept pairs with SD=772 (i.e., succeeded in at least one exercise covering the concept pair.) The average success rate ($\frac{\#\text{correct attempts}}{\#\text{total attempts}}$) across students is about 69% (SD=11%).

4.2 Graph Metrics and Learning

We compare graph metrics (Avg. Degree, Graph Density, Graph Diameter, Avg. Path Length and Avg Node Weight) and measures of activity (Correct Attempts to Exercises, Distinct Correct Exercises, Overall Success Rate and Avg. Success Rate on Concepts) by computing the Kendall's τ_B correlation of these metrics with respect to the students' grade on the programming course. Results are displayed in Table 1.

Surprisingly, the plain **Overall Success Rate** (which does not consider concepts disaggregation, nor graph information) is better correlated with course grade than any other measure. Students who succeed more frequently, get in general better grades. Interestingly, both the **Average Success Rate on Concepts** and the **Average Node Weight** are both significantly correlated with grade. This last measure uses the graph information and presents a slightly better correlation than the former, which does not consider the graph information.

Among the other graph metrics, **Average Degree** and **Average Path Length** are marginally correlated with course

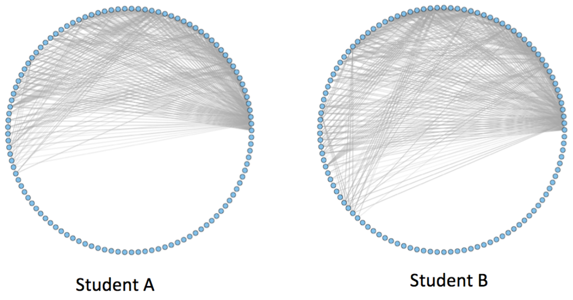


Figure 2: Graph representation of two students.

Table 2: Graph metrics, measures of activity and motivational scores of 2 students.

	Student A	Student B
Graph Density	0.077	0.094
Graph Diameter	2.85	2.00
Avg. Path Length	1.77	1.78
Avg. Degree	8.64	10.55
Avg. Node Weight	0.49	0.51
Correct Attempts	71	83
Dist. Correct Exercises	66	61
Overall Succ. Rate	0.82	0.76
Avg. Succ.Rate on Concepts	0.50	0.53
Mastery-Approach	0.83	0.78
Mastery-Avoidance	0.83	0.56
Performance-Approach	1.0	0.17
Performance-Avoidance	1.0	0.0
Grade (%)	100	97

grade (p values less than 0.1). Although this is a weak evidence, we believe that we are in the good track. A higher **Average Degree** means a better connected graph, thus it follows our idea that highly connected nodes signal more knowledge. **Average Path Length** is more difficult to interpret. A higher **Average Path Length** means a less connected graph (which contradicts our assumption), but also, it can express students reaching more "rear" concepts which appear in few more-difficult-exercises and generally have longer shortest paths. We think that further exploration of metrics among sub-graphs (e.g. a graph for an specific topic), and further refinement of the approach to build edges (e.g. connecting concepts that co-occur close to each other in the exercise) could help to clarify these results

Figure 2 shows the graphs of 2 students who have similar amount of distinct exercises solved correctly but present different graph metrics and motivational profile. See metrics in Table 2. Student B has more edges, lower diameter, higher density, higher degree, solved less questions more times. Student A presents a less connected graph although she he/she solved more distinct questions (66 compared to 61 on Student B). Student B has lower Mastery-Avoidance orientation score and lower Performance orientation scores than Student A, which could explain why Student B work result in a better connected graph. Analyses of Motivational factors are described in the following section.

4.3 Metrics and Motivation

We now explore the relationship between motivational factors and the graphs of the students. The idea is to see to which extent the motivational profile of the student explains the graph's shape. Step-wise regression models were used where the dependent variables are the graph metrics and the independent variables are the motivational factors. We found a significant model of the diameter of the graph ($R_2 = 0.161$, $F = 6.523$, $p = 0.006$) with the factors *Mastery-Avoidance* ($B = 0.952$, $p = 0.001$) and *Mastery-Approach* ($B = -0.938$, $p = 0.006$). Note the negative coefficient for Mastery-Approach and the positive coefficient for Mastery-Avoidance. As the Achievement-Goal Orientation framework suggests, Mastery-Approach oriented students are motivated to learn more, tend to explore more content and do not give up easily when facing difficulties; Mastery-Avoidance students, in the other hand, do not cope well with difficulties and tend to give up. Then, a possible explanation of the results is that, in one hand, students with higher Mastery-Approach orientation are more likely to solve difficult questions which connects more and more distant concepts which decreases the graph diameter; and on the other hand, Mastery-Avoidance students avoid difficult exercises containing many concepts, thus making less connections and producing graphs with higher diameters. Correlations between graph metrics and motivational factors confirmed the relation between Mastery-Avoidance and Graph Diameter (Kendall's $\tau_B = 0.197$, $p = 0.030$). Although these results are encouraging, they are not conclusive. For example, Mastery-Approach students might just do more work, not necessarily targeting difficult questions. More analysis is needed to deeply explore these issues.

5. DISCUSSIONS AND CONCLUSIONS

In this paper we proposed a novel approach to represent student model in the form of a dynamic graph of concepts that become connected when the student succeed in assessment item containing a pair of concepts to be connected. The idea behind this approach is to strengthen the model for those concepts that are applied in more different contexts, i.e., in assessment items containing other different concepts. We applied this approach to data of assessment items answered by real students and analyzed the graph properties comparing them to several performance measures such as course grade as well as motivational factors. Results showed that this idea is potentially a good indicator of knowledge of the students, but further refinement of the approach is needed. We used several measures of the built graphs as descriptors of student knowledge level, and we found that a metric aggregating the success rates of the edges to the level of concepts (nodes) is highly correlated to course grade, although it does not beat the plain overall success rate of the student in assessment items.

In the future work, we plan to repeat our analysis using more reliable approaches to construct the knowledge graph. One idea is to use rich information provided by the parser (mapping between exercises and concepts) to ensure that each new link connects concepts that interact considerably in the program code. This could be done by controlling the concepts proximity in the question code (e.g. only consider co-occurrence when concepts are close to each other in the parser tree.) Another approach to keep more reliable edges is to consider only a subset of important concepts for each problem using feature selection techniques. Also we plan to perform analyses of sub-graphs targeting specific "zones" of knowledge. For example, a partial graph with only concepts that belongs to a specific topic, or concepts that are prerequisites of a specific concept. Another interesting idea relates to recommendation of content: guide the student to questions that will connect the isolated parts of the knowledge graph or minimize the average path length of the graph. Along the same lines, the analysis of the graph shortest paths and overall connectivity can help in designing assessment items that better connect distant concepts.

6. REFERENCES

[1] N. Belacel, G. Durand, and F. Laplante. A binary integer programming model for global optimization of learning path discovery. In *Workshop on Graph-Based Educational Data Mining*.

[2] P. Brusilovsky and E. Millán. User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 1, pages 3–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[3] V. Catet́é, D. Hicks, C. Lynch, and T. Barnes. Snag'em: Graph data mining for a social networking game. In *Workshop on Graph-Based Educational Data Mining*, volume 6, page 10.

[4] C. Conati, A. Gertner, and K. Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, 12(4):371–417, 2002.

[5] R. Dekel and Y. Gal. On-line plan recognition in exploratory learning environments. In *Workshop on Graph-Based Educational Data Mining*.

[6] M. C. Desmarais and M. Gagnon. *Bayesian student models based on item to item knowledge structures*. Springer, 2006.

[7] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. The personal reader: Personalizing and enriching learning resources using semantic web technologies. In P. De Bra and W. Nejdl, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137 of *Lecture Notes in Computer Science*, pages 85–94. Springer Berlin Heidelberg, 2004.

[8] A. J. Elliot and K. Murayama. On the measurement of achievement goals: Critique, illustration, and application. *Journal of Educational Psychology*, 100(3):613, 2008.

[9] B. Hoser, A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. *Semantic network analysis of ontologies*. Springer, 2006.

[10] R. Hosseini and P. Brusilovsky. Javaparser: A fine-grain concept indexing tool for java exercises. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, pages 60–63, 2013.

[11] I.-H. Hsiao, S. Sosnovsky, and P. Brusilovsky. Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming. *Journal of Computer Assisted Learning*, 26(4):270–283, 2010.

[12] S. Jiang, S. M. Fitzhugh, and M. Warschauer. Social positioning and performance in moocs. In *Workshop on Graph-Based Educational Data Mining*, page 14.

[13] T. Käser, S. Klingler, A. G. Schwing, and M. Gross. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *Intelligent Tutoring Systems*, pages 188–198. Springer, 2014.

[14] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. 2014.

[15] E. Millán, T. Loboda, and J. L. Pérez-de-la Cruz. Bayesian networks for student model engineering. *Computers & Education*, 55(4):1663–1683, 2010.

Graph-based Modelling of Students' Interaction Data from Exploratory Learning Environments

Alexandra Poulovassilis
London Knowledge Lab
Birkbeck, Univ. of London
ap@dcs.bbk.ac.uk

Sergio Gutierrez-Santos
London Knowledge Lab
Birkbeck, Univ. of London
sergut@dcs.bbk.ac.uk

Manolis Mavrikis
London Knowledge Lab
UCL Institute of Education
m.mavrikis@lkl.ac.uk

ABSTRACT

Students' interaction data from learning environments has an inherent temporal dimension, with successive events being related through the "next event" relationship. Exploratory learning environments (ELEs), in particular, can generate very large volumes of such data, making their interpretation a challenging task. Using two mathematical microworlds as exemplars, we illustrate how modelling students' event-based interaction data as a graph can open up new querying and analysis opportunities. We demonstrate the possibilities that graph-based modelling can provide for querying and analysing the data, enabling investigation of student-system interactions and leading to the improvement of future versions of the ELEs under investigation.

Keywords

Exploratory Learning Environments, Interaction Data, Graph Modelling

1. INTRODUCTION

Much recent research has focussed on *Exploratory Learning Environments* (ELEs) which encourage students' open-ended interaction within a knowledge domain, coupled with intelligent techniques that aim to provide pedagogical support to ensure students' productive interaction [9]. The data gathered from students' interactions with such ELEs provides a rich source of information for both pedagogical and technical research, to help understand how students are using the ELE and how the intelligent support that it provides may be enhanced to better support students' learning.

In this paper, we consider how modelling students' event-based interaction data as a *graph* makes possible graph-based queries and analyses that can provide insights into the ways that students are using the affordances of the system and the effects of system interventions on students' behaviour. Our case studies are two intelligent ELEs: the MiGen system, that aims to foster 11-14 year old students' learning of algebraic generalisation [15]; and the iTalk2Learn system that aims to support 8-10 year old students' learning of fractions [7]. Both systems provide students with mathematical microworlds in which they undertake construction tasks: in MiGen creating 2-dimensional tiled models using a tool called *eXpresser* and in iTalk2learn creating fractions using the *FractionsLab* tool. In eXpresser, tasks typically require the construction of several models, moving from specific models involving specific numeric values to a general model involving the use of one or more variables; in parallel, students are asked to formulate algebraic rules specifying the number of tiles of each colour that are needed to fully colour their models. In FractionsLab, tasks require the construction, comparison and manipulation of fractions, and students are encouraged to talk aloud about aspects of their constructions, such as whether two fractions are equivalent.

Both systems include intelligent components that provide different levels of feedback to students, ranging from unsolicited prompts and nudges, to low-interruption feedback that students can choose to view if they wish. The aim of this feedback is to balance students' freedom to explore while at the same time providing sufficient support to ensure that learning is being achieved [9]. The intelligent support is designed through detailed cognitive task analysis and Wizard-of-Oz studies [13], and it relies on meaningful *indicators* being detected as students are undertaking construction tasks. Examples of such indicators in MiGen are 'student has made a building block' (part of a model), 'student has unlocked a number' (i.e. has created a variable), 'student has unlocked too many numbers for this task'; while examples of such indicators in FractionsLab are 'student has created a fraction', 'student has changed a fraction' (numerator or denominator), 'student has released a fraction' (i.e. has finished changing it).

Teacher Assistance tools can subscribe to receive real-time information relating to occurrences of indicators for each student, and can present aspects of this information visually to the teacher [8]. Indicators are either *task independent* (TI) or *task dependent* (TD). The former refer to aspects of the student's interaction that are related to the microworld itself and do not depend on the specific task the student is working on, while the latter require knowledge of the task the student is working on, may relate to combinations of student actions, and their detection requires intelligent reasoning to be applied (a mixture of case-based, rule-based and probablistic techniques). Detailed discussions of MiGen's TI and TD indicators and how the latter are inferred may be found in [8].

In this paper we explore how graph-based representation of event-based interaction data arising from ELEs such as MiGen and FractionsLab can aid in the querying and analysis of such data, with the aim of exploring both the behaviours of the students in undertaking the exploratory learning tasks set and the effectiveness of the intelligent support being provided by the system to the students. Data relating to learning environments has often been modelled as a graph in previous work, for example in [10] for providing support to moderators in e-discussion environments; in [16, 18] for supporting learning of argumentation; in [17] for modelling data and metadata relating to episodes of work and learning in a lifelong learning setting; in [1] for learning path discovery as students "navigate" through learning objects; in [3] for recognising students' activity planning in ELEs; and in [23] for gaining better understanding of learners' interactions and ties in professional networks.

Previous work that is close to ours is the work on interaction networks and hint generation [6, 21, 20, 4, 5], in which the graphs used consist of nodes representing states within a problem-solving space and edges representing students' actions in transitioning between states. This approach targets learning environments where students are required to select and apply rules, and the interaction network aims to represent concisely information relating to students' problem-solving sequences in moving from state to state. Our focus here differs from this in that we are using graphs to model fine-grained event-based interaction data arising from ELEs. In our graphs, nodes are used to represent indicator occurrences (i.e. events, not problem states) and edges between such nodes represent the "next event" relationship. Also, rather than using the information derived from querying and analysing this data to automatically generate hints, our focus is on investigating how students are using the system and the effects of the system's interventions in order to understand how students interact with the ELEs and improve their future versions.

2. GRAPH-BASED MODELLING

Figure 1 illustrates our Graph Data Model for ELE interaction data. We see two classes of nodes: Event — representing indicator occurrences; and EventType — representing different indicator types. The instances of the Event class are occurrences of indicators that are detected or generated by the system as each student undertakes a task. We see that instances of Event have several attributes: dateTime: the date and time of the indicator occurrence; userID: the student it relates to; sessionID: the class session that the student was participating in at the time; taskID: the taskID that the student was working on; and constrID: the construction that the student was working on[1].

[1]The model in Fig. 1 focusses on the interaction data. The full data relating to ELEs such as eXpresser and FractionsLab would also include classes relating to users, tasks, sessions and constructions; and attributes describing instances of these classes, such as a user's name and year-group, a task's name and description, a construction's content and description, and a session's description and duration.

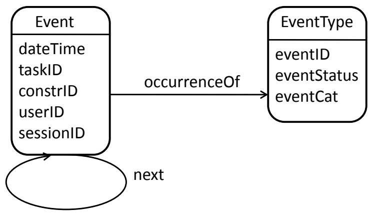


Figure 1: Core Graph Data Model

There is a relationship 'next' linking an instance of Event to the next Event that occurs for the same user, task and session. There is a relationship 'occurrenceOf' linking each instance of Event to an instance of the EventType class.

The instances of the EventType class include: startTask, endTask, numberCreated, numberUnlocked, unlockedNumberChanged, buildingBlockMade, correctModelRuleCreated, incorrectModelRuleCreated, interventionGenerated, interventionShown, in the case of eXpresser (see [8] for the full list); and startTask, endTask, fractionCreated, fractionChanged, fractionReleased, inverventionShown, in the case of FractionsLab.

We see that instances of the EventType class have several attributes, including:

- eventID: a unique numerical identifier for each type of indicator;
- eventStatus: this may be -1, 0 or 1, respectively stating that an occurrence of this type of indicator shows that the student is making negative, neutral or positive progress towards achieving the task goals; an additional status 2 is used for indicators relating to system interventions;
- eventCat: the category into which this indicator type falls; for example, startTask and endTask are task-related indicators; interventionGenerated and interventionShown are system-related ones; numberCreated, numberUnlocked, unlockedNumberChanged are number-related; and fractionCreated, fractionChanged, fractionReleased are fraction-related.

Figure 2 shows a fragment of MiGen interaction data conforming to this graph data model. Specifically, it relates to the interactions of user 5 as he/she is working on task 2 during session 9. The user makes three constructions during this task (with constrIDs 1, 2 and 3). The start and end of the task are delimited by an occurrence of the startTask and endTask indicator type, respectively — events 23041 and 33154. We see that the two events following 23041 relate to an intervention being generated and being shown to the student (this is likely to be because the student was inactive for over a minute after starting the task); following which, the student creates a number — event 24115.

There are additional attributes relating to events, not shown here for simplicity, capturing values relating to the student's

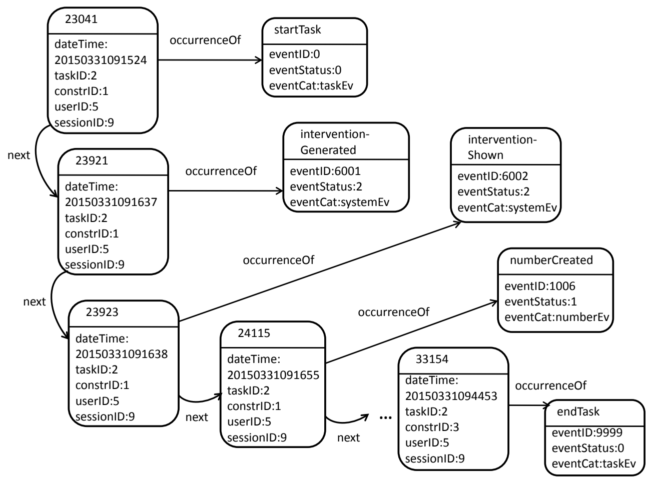


Figure 2: Fragment of Graph Data

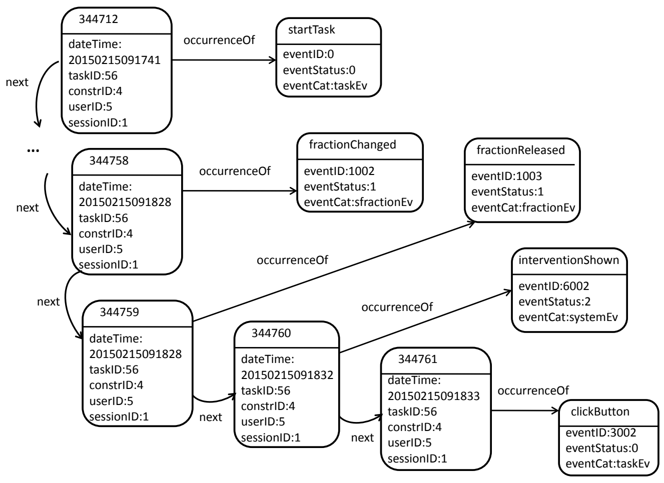


Figure 3: Fragment of Graph Data

constructions and information relating to the system's interventions. For example, for event 24115, the value of the number created, say 5; for event 23921, the feedback strategy used by the system to generate this intervention, say strategy 8; and for event 23923, the content of the message displayed to the user, say "How many green tiles do you need to make your pattern?" and whether this is a high-level interruption by the system or a low-level interruption that the student can choose to view or not. Such information can be captured through additional edges outgoing from an event instance to a literal-valued node: 24115 $\xrightarrow{value}$ 5, 23921 $\xrightarrow{strategy}$ 8, 23932 $\xrightarrow{message}$ "How many green tiles do you need to make your pattern?", 23932 $\xrightarrow{level}$ "high". Since graph data models are semi-structured (and graph data therefore does not need to strictly conform to a single schema), this kind of heterogeneity in the data is readily accommodated.

Figure 3 similarly shows a fragment of FractionsLab interaction data, relating to the interactions of user 5 working on task 56 during session 1. The user makes one construction during this task. We see events relating to the student changing and 'releasing' a fraction. Following which the system displays a message (in this case, it was a high-interruption message of encouragement "Great! Well Done").

We see from Figures 2 and 3 that the sub-graph induced by edges labelled 'next' consists of a set of paths, one path for each task undertaken by a specific user in a specific session. The entire graph is a DAG (directed acyclic graph): there are no cycles induced by the edges labelled 'next' since each links an earlier indicator occurrence to a later one; while the instances of EventType and other literal-valued nodes can have only incoming edges. The entire graph is also a bipartite graph, with the two parts comprising (i) the instances of Event, and (ii) the instances of EventType and the literal-valued nodes.

As a final observation, we note that Figures 1 – 3 adopt a "property graph" notation (e.g. as used in the Neo4J graph database, neo4j.com) in which nodes may have attributes. In a "classical" graph data model, each attribute of a node would be represented by an edge and its value by a literal-valued node. So, for example, the information that the taskID of event 23041 is 2 would be represented by an edge 23041 $\xrightarrow{taskID}$ 2. The query examples in the next section assume this "classical" graph representation.

3. GRAPH QUERIES AND ANALYSES

Because the sub-graph induced by edges labelled 'next' consists of a set of paths, the data readily lends itself to exploration using *conjunctive regular path (CRP) queries* [2]. A CRP query, Q, consisting of n conjuncts is of the form

$$(Z_1, \ldots, Z_m) \leftarrow (X_1, R_1, Y_1), \ldots, (X_n, R_n, Y_n)$$

where each X_i and Y_i is a variable or a constant, each Z_i is a variable that appears also in the right hand side of Q, and each R_i is a regular expression over the set of edge labels. In this context, a regular expression, R, has the following syntax:

$$R := \epsilon \,|\, a \,|\, \_ \,|\, (R_1.R_2) \,|\, (R_1|R_2) \,|\, R^* \,|\, R^+$$

where ϵ denotes the empty string, a denotes an edge label, _ denotes the disjunction of all edge labels, and the operators have their usual meaning. The answer to a CRP query on a graph G is obtained by finding for each $1 \leq i \leq n$ a binary relation r_i over the scheme (X_i, Y_i), where there is a tuple (x, y) in r_i if and only if there is a path from x to y in G such that: $x = X_i$ if X_i is a constant; $y = Y_i$ if Y_i is a constant; and the concatenation of the edge labels in the path satisfies the regular expression R_i. The answer is then given by forming the natural join of the binary relations $r_1, \ldots, r_n$ and finally projecting on $Z_1, \ldots, Z_m$.

To illustrate, the following CRP query returns pairs of events x, y such that x is an intervention message shown to the user by the system and y indicates that the user's next action – in eXpresser – was to create a number (note, variables in queries are distinguished by an initial question mark):

```
(?X,?Y) <- (?X,occurrenceOf,interventionShown),
           (?X,next,?Y),
           (?Y,occurrenceOf,numberCreated)
```

The result would contain pairs such as (23923,24115) from Figure 2, demonstrating that there are indeed situations where an intervention message displayed by the MiGen system leads directly to the creation of a number by the student.

The following query returns pairs of events x, y such that that x is an intervention message shown to the user by the system and y is the user's next action; the type of y is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionShown),
              (?X,next,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23923,24115,numberCreated) from Figure 2 and (344760,344761,clickButton) from Figure 3, allowing researchers to see what types of events directly follow the display of an intervention message. This would allow the confirmation or contradiction of researchers' expectations regarding the immediate effect of intervention messages on students' behaviours.

Focussing for the rest of this section on the data in Figure 2, the following query returns pairs of events x, y such that x is any type of event and y indicates that the user's next action was to unlock a number; the type of x is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,?Z),
              (?X,next,?Y),
              (?Y,occurrenceOf,numberUnlocked)
```

The result would allow researchers to see what types of events immediately precede the unlocking of a number (i.e. the creation of a variable). This would allow confirmation of researchers' expectations about the design of the MiGen system's intelligent support in guiding students towards generalising their models by changing a fixed number to an 'unlocked' one.

The following query returns pairs of events x, y such that that x is an intervention generated by the system and y is any subsequent event linked to x through a path comprising one or more 'next' edges; the type of y is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,next+,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23921, 23923, interventionShown), (23921, 24115, numberCreated), ... (23921, 33154, endTask), allowing researchers to see what types of events directly or indirectly follow the display of an intervention message by the system. This would allow the confirmation or contradiction of researchers' expectations regarding the longer-term effect of intervention messages on students' behaviours.

We can modify the query to retain only pairs x, y that relate to the same construction:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,constrID,?C), (?X,next+,?Y),
              (?Y,constrlID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain triples such as
(23921, 23923, interventionShown),
(23921, 24115, numberCreated),
(23921, 24136, numberUnlocked),
(23921, 24189, unlockedNumberChanged),
relating to construction 1 made by user 5 during session 9 for task 2 (two more events — 24136 and 24189 — relating to construction 1 have been assumed here, in addition to 23923 amd 24115 shown in Figure 2, for illustrative purposes). The results would not contain (23921,33154,endTask), since event 33154 relates to construction 3.

To show more clearly the answers to the previous query in the form of possible event *paths*, we can use *extended regular path* (ERP) queries [11], in which a regular expression can be associated with a *path variable* and path variables can appear in the left-hand-side of queries. Thus, for example, the following query returns the possible paths from x to y:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), (?X,next+:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain answers such as
(23921, [next], 23923, interventionShown),
(23921, [next, 23923, next], 24115, numberCreated),
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked),
(23921, [next, 23923, next, 24115, next, 24136, next], 24189, unlockedNumberChanged).

The use of the regular expressions `next` and `next+` in the previous queries matches precisely one edge labelled 'next', or any number of such edges (greater than or equal to 1), respectively. However, for finer control and ranking of query answers, it is possible to use *approximate* answering of CRP and ERP queries (see [11, 17]), in which edit operations such as insertion, deletion or substitution of an edge label can be applied to regular expressions.

For example, using the techniques described in [11, 17], the user can chose to allow the insertion of the label 'next' into a regular expression, at an edit cost of 1. Submitting then this query:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), APPROX(?X,next:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

would return first exact answers, such as
(23921, [next], 23923, interventionShown). The regular expression `next` in the conjunct `APPROX(?X,next:?P,?Y)` would then be automatically approximated to `next.next`, leading to answers such as

(23921, [next, 23923, next], 24115, numberCreated)
at an edit distance of 1 from the original query. Following this, the regular expression `next.next` would be automatically approximated to `next.next.next`, leading to answers such as
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked)
at distance 2. This incremental return of paths of increasing length can continue for as long as the user wishes, and allows researchers to examine increasingly longer-term effects of intervention messages on students' behaviours. It would also be possible for users to specify from the outset a minimum and maximum edit distance to be used in approximating and evaluating the query, for example to request paths encompassing between 2 and 4 edges labelled 'next'.

Queries based on evaluating regular expressions over a graph-based representation of interaction data, such as those above, can aid in the exploration of students' behaviours as they are undertaking tasks using ELEs and the effectiveness of the intelligent support being provided by the ELE. The query processing techniques employed are based on incremental query evaluation algorithms which run in polynomial time with respect to the size of the database graph and the size of the query and which return answers in order of increasing edit distance [11]. A recent paper [19] gives details of an implementation, which is based on the construction of an automaton (NFA) for each query conjunct, the incremental construction of a weighted product automaton from each conjunct's automaton and the data graph, and the use of a ranked join to combine answers being incrementally produced from the evaluation of each conjunct. The paper also presents a performance study undertaken on two data sets — lifelong learning data and metadata [17] and YAGO [22]. The first of these has rather 'linear' data, similar to the interaction data discussed here, while the second has 'bushier' connectivity. Query performance is generally better for the former than the latter, and the paper discusses several possible approaches towards query optimisation.

In addition to evaluating queries over the interaction data, by representing the data in the form of a graph it is possible to apply graph structure analyses such as the following:

- *path finding and clustering*: this would be useful for determining patterns of interest across a whole dataset, or focussing on particular students, tasks or sessions c.f. [4];
- *average path length*: this would be useful for determining the amount of student activity (i.e. the number of indicator occurrences being generated per task) across a whole dataset, or focussing on particular students, tasks or sessions;
- *graph diameter*: to determine the greatest distance between any two nodes (which, due to the nature of the data, would be event type nodes); this would be an indication the most long-running and/or most intensive task(s);
- *degree centrality*: determining the in-degree centrality of event type nodes would identify key event types occurring in students' interactions; this analysis could be applied across a whole dataset, or focussing on particular students, tasks or sessions;
- nodes that have a high probability of being visited on a randomly chosen shortest path between two randomly chosen nodes have high *betweenness centrality*; determining this measure for pairs of event type nodes (ignoring the directionality of the 'occurrenceOf' edges) would identify event types that play key mediating roles between other event types.

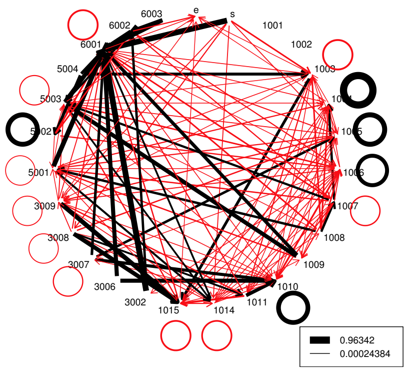


Figure 4: Transitions between Event Types

We have already undertaken some *ad hoc* analyses of interaction data arising from classroom sessions using ELEs. For example, Figure 4 shows the normalised incoming transitions for a 1-hour classroom session involving 22 students using MiGen (in the diagram, s denotes the 'startTask' and e the 'endTask' event types). Event types with an adjacent circle show transitions where this type of event occurs repeatedly in succession. The thickness of each arrow or circle indicates the value of the transition probability: the thicker the line, the higher the probability. Red (light grey) is used for probabilities < 0.2 and black for probabilities ≥ 0.2. We can observe a black arrow 3007 $\rightarrow$ 1005, indicating transitions from events of type 3007 (detection by the system that the student has made an implausible building block for this task) to events of type 1005 (modification of a rule by the student). Such an observation raises a hypothesis for more detailed analysis or further student observation, namely: "does the construction of an incorrect building block lead students to self-correct their rules?". Developing a better understanding of such complex interaction can lead to improvement of the system. For this particular example, we designed a new prompt that suggests to students to first consider the building block against the given task before proceeding unnecessarily in correcting their rules. More examples of such *ad hoc* analyses are given in [14]. Representing the interaction data in graph form will allow more systematic, flexible and scalable application of graph-structure algorithms such as those identified above.

4. CONCLUSIONS AND FUTURE WORK

We have presented a graph model for representing event-based interaction data arising from Exploratory Learning Environments, drawing on the data generated when students undertake exploratory learning tasks with the eXpresser and

FractionsLab microworlds. Although developed in the context of these systems, the model is a very general one and can easily be used or extended to model similar data from other ELEs.

We have explored the possibilities that evaluating regular path queries over this graph-based representation might provide for exploring the behaviours of students as they are working in the ELE and the effectiveness of the intelligent support that it provides to them. We have also identified additional graph algorithms that may yield further insights about learners, tasks and significant indicators.

Planned worked includes transformation and uploading of the interaction data sets gathered during trials and full classroom sessions of the two systems into an industrial-strength graph database such as Neo4J, following the graph model presented in Section 2; followed by the design, implementation and evaluation of meaningful queries, analyses and visualisations over the graph data, building on the work presented in Section 3. Equipped with an appropriate user interface, educational researchers, designers or even teachers with less technical expertise could in this way explore the data from their perspective. This has the potential to lead to an improved understanding of interaction in this context and to feed back to the design of the ELEs. We see this approach very much in the spirit of "polyglot persistence" (i.e. using different data storage methods to address different data manipulation problems), and hence being used in conjunction with other EDM resources such as DataShop [12]. Another direction of research is investigation of how the flexible querying processing techniques for graph data (including both query approximation and query relaxation) that have been developed in the context of querying lifelong learners' data and metadata [11, 17] might be applied or adapted to the much finer-granularity interaction data described here and the more challenging pedagogical setting of providing effective intelligent support to learners undertaking exploratory tasks in ELEs.

Acknowledgments

This work has been funded by the ESRC/EPSRC MiGen project, the EU FP7 projects iTalk2Learn (#318051) and M C Squared (#610467). We thank all the members of these projects for their help and insights.

5. REFERENCES

[1] N. Belacel, G. Durand, and F. LaPlante. A binary integer programming model for global optimization of learning path discovery. *G-EDM*, 2014.

[2] D. Calvanese and et al. Containment of conjunctive regular path queries with inverse. *KRR*, pages 176–185, 2015.

[3] R. Dekel and K. Gal. On-line plan recognition in exploratory learning environments. *G-EDM*, 2014.

[4] M. Eagle and T. Barnes. Exploring differences in problem solving with data-driven approach maps. *EDM*, 2014.

[5] M. Eagle, D. Hicks, B. Peddycord III, and T. Barnes. Exploring networks of problem-solving interactions. *LAK*, pages 21–30, 2015.

[6] M. Eagle, M. Johnson, and T. Barnes. Interaction networks: Generating high level hints based on network community clustering. *EDM*, 2012.

[7] B. Grawemeyer and et al. Light-bulb moment?: Towards adaptive presentation of feedback based on students' affective state. *IUI*, pages 400–404, 2015.

[8] S. Gutierrez-Santos, E. Geraniou, D. Pearce-Lazard, and A. Poulovassilis. Design of Teacher Assistance Tools in an Exploratory Learning Environment for Algebraic Generalization. *IEEE Trans. Learn. Tech.*, 5(4):366–376, 2012.

[9] S. Gutierrez-Santos, M. Mavrikis, and G. D. Magoulas. A Separation of Concerns for Engineering Intelligent Support for Exploratory Learning Environments. *J. Research and Practice in Inf. Tech.*, 44:347–360, 2013.

[10] A. Harrer, R. Hever, and S. Ziebarth. Empowering researchers to detect interaction patterns in e-collaboration. *Frontiers in Artificial Intelligence and Applications*, 158:503, 2007.

[11] C. Hurtado, A. Poulovassilis, and P. Wood. Finding top-k approximate answers to path queries. *FQAS*, pages 465–476, 2009.

[12] K. Koedinger and et al. A data repository for the EDM community: The PSLC datashop. *Handbook of Educational Data Mining*, 43, 2010.

[13] M. Mavrikis and S. Gutierrez-Santos. Not all Wizards are from Oz: Iterative design of intelligent learning environments by communication capacity tapering. *Computers and Education*, 54(3):641–651, 2010.

[14] M. Mavrikis, Z. Zheng, S. Gutierrez-Santos, and A. Poulovassilis. Visualisation and analysis of students' interaction data in exploratory learning environments. *Workshop on Web-Based Technology for Training and Education (at WWW)*, 2015.

[15] R. Noss and et al. The design of a system to support exploratory learning of algebraic generalisation. *Computers and Education*, 59(1):63–82, 2012.

[16] N. Pinkwart and et al. Graph grammars: An ITS technology for diagram representations. *FLAIRS*, pages 433–438, 2008.

[17] A. Poulovassilis, P. Selmer, and P. Wood. Flexible querying of lifelong learner metadata. *IEEE Trans. Learn. Tech.*, 5(2):117–129, 2012.

[18] O. Scheuer and B. McLaren. CASE: A configurable argumentation support engine. *IEEE Trans. Learn. Tech.*, 6(2):144–157, 2013.

[19] P. Selmer, A. Poulovassilis, and W. P.T. Implementing flexible operators for regular path queries. *GraphQ (EDBT/ICDT Workshops)*, pages 149–156, 2015.

[20] V. Sheshadri, C. Lynch, and T. Barnes. InVis: An EDM tool for graphical rendering and analysis of student interaction data. *G-EDM*, 2014.

[21] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *Artificial Intelligence in Education*, pages 345–352, 2011.

[22] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge. *WWW*, 2007.

[23] D. Suthers. From contingencies to network-level phenomena: Multilevel analysis of activity and actors in heterogeneous networked learning environments. *LAK*, 2015.

Exploring the function of discussion forums in MOOCs: comparing data mining and graph-based approaches

Lorenzo Vigentini
Learning & Teaching Unit
UNSW Australia,
Lev 4 Mathews, Kensington 2065
+61 (2) 9385 6226
l.vigentini@unsw.edu.au

Andrew Clayphan
Learning & Teaching Unit
UNSW Australia,
Lev 4 Mathews, Kensington 2065
+61 (2) 9385 6226
a.clayphan@unsw.edu.au

ABSTRACT

In this paper we present an analysis (in progress) of a dataset containing forum exchanges from three different MOOCs. The forum data is enhanced because together with the exchanges and the full text, we have a description of the design and pedagogical function of forums in these courses and a certain level of detail about the users, which includes achievement, completion, and in some instances more details such as: education; employment; age; and prior MOOC exposure.

Although a direct comparison between the datasets is not possible because the nature of the participants and the courses are different, what we hope to identify using graph-based techniques is a characterization of the patterns in the nature and development of communication between students and the impact of the 'teacher presence' in the forums. With the awareness of the differences, we hope to demonstrate that student engagement can be directed 'by-design' in MOOCs: teacher presence should therefore be planned carefully in the design of large-scale courses.

Keywords

MOOCs, Discussion forums, graph-based EDM, pedagogy.

1. INTRODUCTION

In the past couple of years MOOCs (Massive Open Online Courses) have become the center of much media hype as disruptive and transformational [1, 2]. Although the focus has been on a few characteristics of the MOOCS – i.e. free courses, massive numbers, massive dropouts and implicit quality warranted by the status of the institutions delivering these courses – a rapidly growing research interest has started to question the effectiveness of MOOCS for learning and their pedagogies. If one ignores entirely the philosophies of teaching driving the design and delivery of MOOCs going from the the socio-constructivist (cMOOC, [4, 5]) to instructivist (xMOOC, [3]), at the practical level, instructors have to make specific choices about how to use the tools available to them. One of these tools is the discussion forum. Forums are one of the most popular asynchronous tools to support students' communication and collaboration in web-based learning environments [6]. These can be deployed in a variety of ways, ranging from a tangential support resource which students can refer to when they need help, to a space for learning with others, driven by the activities students have to carry out (usually sharing work and eliciting feedback). The latter, in a sense, emulates class-time in traditional courses providing a space for structured discussions about the topics of the course. One could argue that like in face-to-face classes, the value of the interaction depends on the importance attributed to the forums by the instructors. This is an interesting point to explore teachers' presence and the value of their input in directing such conversations. Mazzolini & Maddison characterize the role of the teacher and teacher presence in online discussion forums as varying from being the 'sage on the stage', to the 'guide on the side' or even 'the ghost in the wings' [7]. Furthermore they argue that the 'ideal' degree of visibility of the instructor in discussion forums depends on the purpose of forums and their relationship to assessment. There are also a number of accounts indicating that students' learning in forums is not very effective [8, 9]. However if one looks at the data there are numerous examples indicating that behaviours in forums are good predictors of performance in the courses using them, particularly if forum activities are assessed [10,11,12,13]. Yet, forums in MOOCs tend to attract only a small portion of the student activity [14]. This is setting forums in MOOCs apart from 'tutorial-type' forums used to support students' learning in online or blended courses in higher education. Furthermore, some argue that active engagement is not the only way of benefiting from discussion forums [15] and students' characteristics and preferences could be more important than the course design in determining the way in which they take full advantage of online resources [16].

2. THE THREE MOOCS IN DETAIL

In order to investigate the way in which students use the discussion forums, we have extracted data from three MOOCS delivered by a large, research intensive Australian university. The three courses are: P2P (From Particles to Planets - Physics); LTTO (Learning to Teach Online); and INTSE (Introduction to Systems Engineering), which are broadly characterised in the top of Table 1. The courses were specifically designed in quite different ways to test hypotheses about their design, delivery and effectiveness.

In particular, P2P was designed emulating a traditional university course in a sequential manner. All content was released on a week-by-week basis dictating the pace of instruction. LTTO and INTSE, instead were designed to provide a certain level of flexibility for the students to elect their learning paths. All content was readily available at the start, however for LTTO, the delivery followed a week-on-week delivery focusing on the interaction with students and a selective attention to particular weekly topics (i.e. weekly feedback videos driven by the discussion forums as well as weekly announcements). Although announcements were used also in INTSE, the lack of weekly activities in the forums did not impose a strong pacing. In INTSE, the forums had only a tangential support value and were used mainly to respond to students' queries and to clarify specific topics emerging from the quizzes. Table 1 provides an overview of the different courses. This also shows that the forum activity in the various courses is a very small portion of all actions emerging from the logs of activity which has been reported in the literature [9].

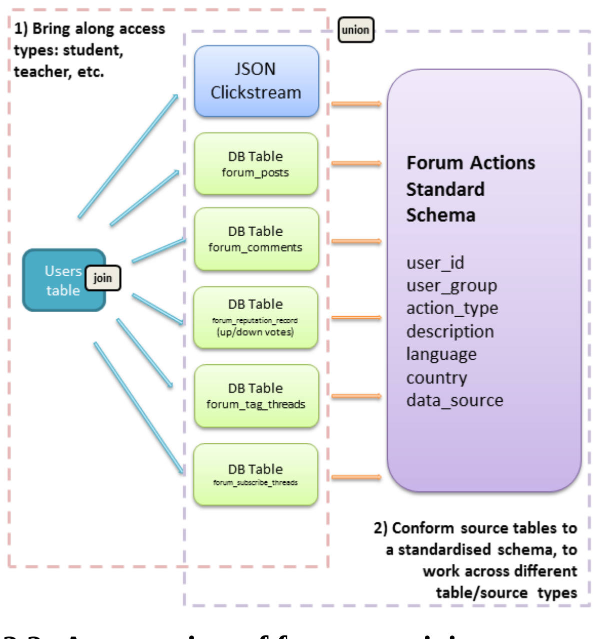
3. DETAILS OF THE DATASET

3.1 The dataset

The data under consideration is an export form the Coursera platform. Raw forum database tables (posts, comments, tags, votes) as well as a JSON based web clickstream were used. The clickstream events consist of a key which specifies action – either a 'pageview' or 'video' item. Forum clickstream events were identified by a common '/forum' prefix.

The clickstream was further classified into: browsing; profile lookups; social interaction (looking at contributions); search; tagging; and threads. From the classification it became evident the clickstream did not record all events, such as when a post or comment was made, or when votes were applied. For these, specific database tables were used. In order to manage different data sets and sources, a standardized schema was built, allowing disparate sources to feed into, but exposing a common interface to conduct analysis over forum activities. This is shown in Figure 1.

Figure 1. Forum data transformation process


1) Bring along access types: student, teacher, etc.
union
JSON Clickstream
DB Table forum_posts
DB Table forum_comments
DB Table forum_reputation_record (up/down votes)
DB Table forum_tag_threads
DB Table forum_subscribe_threads
Users table
join
Forum Actions Standard Schema
user_id
user_group
action_type
description
language
country
data_source
2) Conform source tables to a standardised schema, to work across different table/source types

3.2 An overview of forums activity

There are very interesting trends which require more detailed examination (bottom of table 1). As expected, in LTTO the forum activity is larger than in the other courses and this is probably due to the fact that students were asked to submit post in forums following the learning activities. The proportion of active students in forum is 4x in magnitude compared to the other courses. Yet, if we look at the average amount of posts or comments, the patterns are not straightforward to interpret, as the level of engagement is similar across the courses with 3 to 5 posts per student and 1 to 3 comments (i.e. replies to existing posts), but with P2P showing a higher level of engagement than the other courses. One possible explanation is the different target group of the different courses with INTSE including a majority of professional engineers with postgraduate qualifications, P2P focusing on high school student and teachers, and LTTO targeting a broad base of teachers across different educational levels.

	INTSE	LTTO	P2P
Target group	Engineers	Teachers at all levels	High school and teachers
Course length	9 weeks	8 weeks	8 weeks
Forums	54 (14 top level)	105 (17 top-level)	63 (15 top-level)
Design mode	All-at-once	All-at-once	Sequential
Delivery mode	All-at-once	Staggered	Staggered
Use of forums	**Tangential**	**Core activity**	**Support**
N in forum	422 (2.1%)	1685 (9.3%)	293 (2.8%)
Tot posts	1361 (avg=3.3)	6361 (avg=3.8)	1399 (avg=4.8)
Tot comments	285 (avg=0.7)	2728 (avg=1.7)	901 (avg=3.1)
Registrants	32705	28558	22466
Active students[1]	60%	63%	47%
Completing[2]	4.2% (0.3% D)	4.4% (2.4 D)	0.7% (0.2%)

Table 1. Summary of the courses under investigation. NOTE: 1. Active students are those appearing in the clickstream; 2. Completing students achieve the pass grade or Distinction (D)

The type of activity is summarised in Figure 2. In the chart, the five categories refer to the following: View corresponds to listing forums, threads and viewing posts; Post is the writing of a post or start of a new thread; Comment is a reply to an existing post; Social refers to all actions engaging directly with other's status (up-vote, down-vote and looking at profiles/reputation); Engage refers to the additional interaction with forums content (searching, tagging, 'watching' or subscribing to posts or threads).

The viewing behaviour is the most prominent for both the student and instructor groups and the figures are pretty much similar across the board. A two-way ANOVA (2x5, role by activity) on the percentage of distributions, shows that there is no significant difference between students and instructors, but there is an obvious difference between views and the other types of behaviour ($F(4,29) = 1656.3$, $p < .01$).

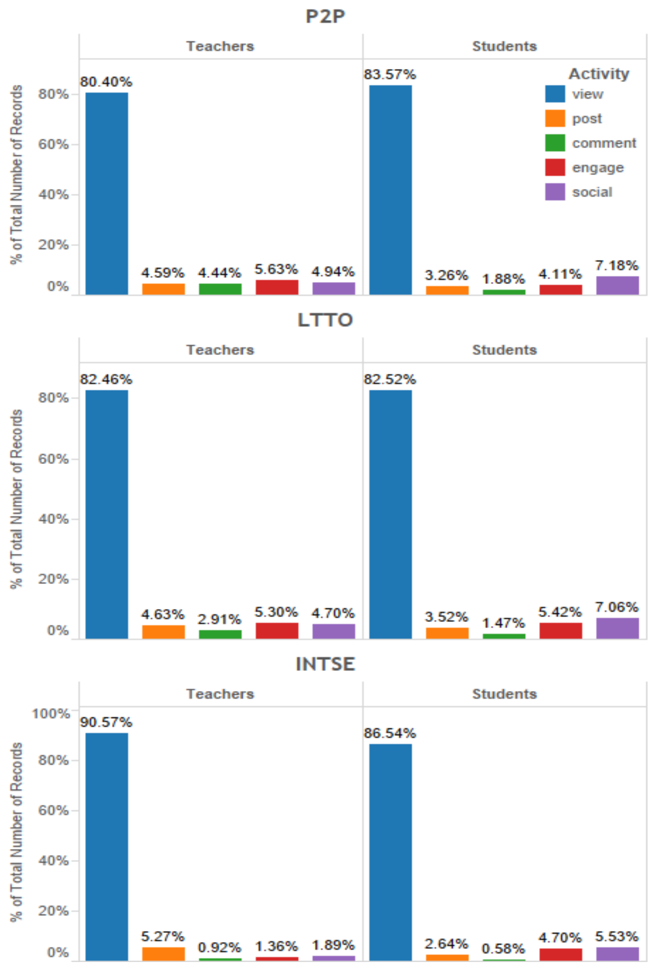
If we consider the engagement over the timeline and compare the type of activities carried out by students and instructors, Figure 3 (end of the paper) shows the patterns for the three courses. The most striking pattern is that there doesn't seem to be an obvious one. For what concerns posts and views in all the three courses there is a sense of synchronicity between the two groups, however from this chart it is not possible to understand in more detail what are the connections between what students and teachers do.

Instructors' comments are slightly offset, possibly as a reaction to students' posts. An interesting aspect is the amount of 'social' engagement in the P2P course that merits further analysis.

4. DIRECTIONS AND OPEN QUESTIONS

From this coarse analysis it is apparent that there seem to be minimal behavioural differences in the way students and instructors interact in the different courses, however more analysis is required to tackle questions about the individual differences in students' and instructors' patterns of interaction and their interrelations. Furthermore little can be said about how the nature of interactions drives the development of communication and engagement. However a number of questions like the following remain open and unanswered: how do discussions develop over time? How teacher presence affects the development of discussions? Is the number of forums affecting how students engage with them (i.e. causing disorientation)?

Figure 2. Distribution of forum activities by role



4.1 The DM and graph-based approaches

A possible way to answer the questions about the types/patterns of behaviours, the structure and development of networks and the growth of groups/communities over time might be using data mining and graph-based approaches. For example, [6] used a combination of quantitative, qualitative and social network information about forum usage to predict students' success or failure in a course by applying classification algorithms and classification via clustering algorithms. In their approach the activity of students in the forums is organized according to a set of commonly used quantitative metrics and a couple of measures borrowed from Social Network Analysis (table 2). Although this seems to be a promising approach, there are two issues with this methodology in the MOOCs: 1) only a tiny proportion of students can be considered active and 2) it is hard to scale the instructor's evaluation. The first problem is not easily resolved and it is an issue in the literature reviewed [17, 18]; non-posting behavior is considered as an index of disengagement, partly because this is easy to measure. In principle the latter could be substituted by peer evaluation (up-vote, down-vote), but there is no easy way to ensure consistency.

Indicator	Type	Description
Messages	Quantitative	Number of messages written by the student.
Threads	Quantitative	Number of new threads created by the student.
Words	Quantitative	Number of words written by the student.
Sentences	Quantitative	Number of sentences written by the student.
Reads	Quantitative	Number of messages read on the forum by the student.
Time	Quantitative	Total time, in minutes, spent on forum by the student.
AvgScoreMsg	Qualitative	Average score on the instructor's evaluation of the student's messages.
Centrality	Social	Degree centrality of the student.
Prestige	Social	Degree prestige of the student.

Table 2. Possible indicators characterising forum engagement

An alternative method that can be explored is graph-based approaches. For example, Bhattacharya et al. [19] used graph-based techniques to explore the evolution of software and source branching providing an insight in the process. Kruck et al. [20] developed GSLAP, an interactive, graph-based tool for analyzing web site traffic based on user-defined criteria.

Kobayashi et al. [18] used a method to quickly identify and track the evolution of topics in large datasets using a mix of assignment of documents to time slices and clustering to identify discussion topics. Yang et al [21] integrated graph-based clustering to characterize the emergence of communities and text-based analysis to portray the nature of exchanges. In fact, students move in the various sub-forums taking different roles or stances as they engage with different subsets of students. As the reasons to engage in these discussions are partly determined by different interests, goals, and issues, it is possible to construct a social network graph based on the post-reply-comment structure within threads. The network generated provides a possible view of a student's social participation within a MOOC, which may indicate some detail about their values, beliefs and intentions.

Furthermore, Brown et al [22] have already shown the value of exploring the communities in discussion forums in MOOCs particularly for what concerns the homogeneity of performance but dissimilarity of motivations characterizing student hubs.

4.2 Discussion points

The examples above provide evidence of the potential for using graph-based methods to obtain better insights into the process and content analysis for our dataset and to extend its applicability to MOOCs, however there are a number of contentious points to raise which will provide opportunities for discussion.

Firstly the number of students who are actively involved in discussion is a very small proportion of the *active* participants. This means that the subset may not be representative at all. One could argue that these students are already engaged or desperately need help. Previous literature [21, 22, 23] focused on the ability to predict performance and on the peer effect which can emerge from the analysis of the graphs/social networks.

Secondly, one could question the value of the communities in xMOOCs: especially when courses are designed with an instructivits approach leading to mastery, by definition this is an individualistic perspective focused on the testing of one's own skills/learning. Of course in cMOOCs -connectivists by design- the importance of the development of social support is essential. This seems to be supported by Brown et al [22]: they were not able to uncover a direct relation between stated goals and motivations with the participation in forums, and attributed this to pragmatic needs. However, as the authors suggested earlier, the instructors might play a fundamental role in shaping the communities based on the value attributed to forums in their plans/design and the level of engagement/interaction. Considering the split between cMOOCs and xMOOCs again, interesting work might come out of the experiment conducted by Rose' and colleagues in the DALMOOC in which automated agents were deployed to support students' conversations. In Coursera the deployment of 'community mentors' will be an interesting space to explore, given that the importance of design seems to be removed from instructors in the 'on-demand' model.

Lastly, more research is needed in the time-based dimension of development of forums in MOOCs. Questions like how students bond and create stable relations, how they become authoritative and what motivates them to contribute over time are all open questions which the analysis of graphs over time might be able to address.

5. REFERENCES

[1] Dirk Jan van den Berg and Edward Crawley. Why MOOCS Are Transforming the Face of Higher Education. Retrieved April 12, 2015 from http://www.huffingtonpost.co.uk/dirk-jan-van-den-berg/why-moocs-are-transforming_b_4116819.html

[2] Chris Parr. The evolution of Moocs. Retrieved April 12, 2015 from http://www.timeshighereducation.co.uk/comment/opinion/the-evolution-of-moocs/2015614.article

[3] C. Osvaldo Rodriguez. 2012. MOOCs and the AI-Stanford Like Courses: Two Successful and Distinct Course Formats for Massive Open Online Courses. *European Journal of Open, Distance and E-Learning* (January 2012).

[4] George Siemens. 2005. Connectivism: A learning theory for the digital age. *International journal of instructional technology and distance learning* 2, 1 (2005), 3–10.

[5] Stephen Downes. 2008. Places to go: Connectivism & connective knowledge, Innovate.

[6] Cristóbal Romero, Manuel-Ignacio López, Jose-María Luna, and Sebastián Ventura. 2013. Predicting students' final performance from participation in on-line discussion forums. *Computers & Education* 68 (October 2013), 458–472. DOI: http://dx.doi.org/10.1016/j.compedu.2013.06.009

[7] Margaret Mazzolini and Sarah Maddison. 2007. When to jump in: The role of the instructor in online discussion forums. *Computers & Education* 49, 2 (September 2007), 193–213. DOI:http://dx.doi.org/10.1016/j.compedu.2005.06.011

[8] M.j.w. Thomas. 2002. Learning within incoherent structures: the space of online discussion forums. *Journal of Computer Assisted Learning* 18, 3 (September 2002), 351–366. DOI:http://dx.doi.org/10.1046/j.0266-4909.2002.03800.x

[9] Daniel F.O. Onah, Jane Sinclair, and Russell Boyatt. 2014. Exploring the use of MOOC discussion forums. In *Proceedings of London International Conference on Education*. London: LICE, 1–4.

[10] Alstete, J.W. and Beutell, N.J. Performance indicators in online distance learning courses: a study of management education. *Quality Assurance in Education 12*, 1 (2004), 6–14.

[11] Cheng, C.K., Paré, D.E., Collimore, L.-M., and Joordens, S. Assessing the effectiveness of a voluntary online discussion forum on improving students' course performance. *Computers & Education 56*, 1 (2011), 253–261.

[12] Palmer, S., Holt, D., and Bray, S. Does the discussion help? The impact of a formally assessed online discussion on final student results. *British Journal of Educational Technology 39*, 5 (2008), 847–858.

[13] Patel, J. and Aghayere, A. Students' Perspective on the Impact of a Web-based Discussion Forum on Student Learning. *Frontiers in Education Conference, 36th Annual*, (2006), 26–31.

[14] Jacqueline Aundree Baxter and Jo Haycock. 2014. Roles and student identities in online large course forums: Implications for practice. *The International Review of Research in Open and Distributed Learning* 15, 1 (January 2014).

[15] Vanessa Paz Dennen. 2008. Pedagogical lurking: Student engagement in non-posting discussion behavior. *Computers in Human Behavior* 24, 4 (July 2008), 1624–1633. DOI:http://dx.doi.org/10.1016/j.chb.2007.06.003

[16] René F. Kizilcec, Chris Piech, and Emily Schneider. 2013. Deconstructing Disengagement: Analyzing Learner Subpopulations in Massive Open Online Courses. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*. LAK '13. New York, NY, USA: ACM, 170–179. DOI:http://dx.doi.org/10.1145/2460296.2460330

[17] Dennen, V.P. Pedagogical lurking: Student engagement in non-posting discussion behavior. *Computers in Human Behavior 24*, 4 (2008), 1624–1633.

[18] Kobayashi, M. and Yung, R. Tracking Topic Evolution in On-Line Postings: 2006 IBM Innovation Jam Data. In T. Washio, E. Suzuki, K.M. Ting and A. Inokuchi, eds., *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2008, 616–625.

[19] Bhattacharya, P., Iliofotou, M., Neamtiu, I., and Faloutsos, M. Graph-based Analysis and Prediction for Software Evolution. *Proceedings of the 34th International Conference on Software Engineering*, IEEE Press (2012), 419–429.

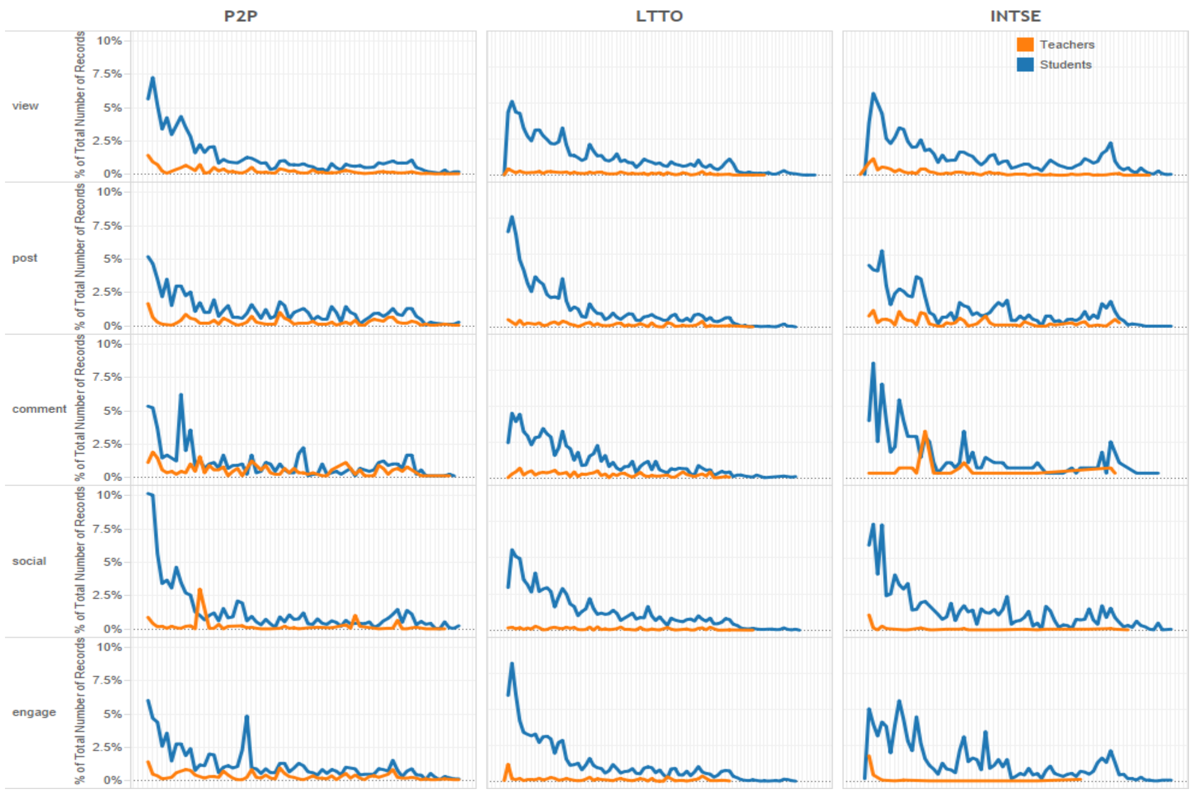
[20] Kruck, S.E., Teer, F., and Jr, W.A.C. GSLAP: a graph‐based web analysis tool. *Industrial Management & Data Systems 108*, 2 (2008), 162–172.

[21] D. Yang, M. Wen, A. Kumar, E. P. Xing, and C. P. Rose, "Towards an integration of text and graph clustering methods as a lens for studying social interaction in MOOCs," *The International Review of Research in Open and Distributed Learning*, vol. 15, no. 5, Oct. 2014.

[22] R. Brown, C Lynch, Y. Wang, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, D. McNamara. Communities of performance and communities of preference. GEDM 2015, in press.

[23] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach, "Predicting Student Exam's Scores by Analyzing Social Network Data," in *Active Media Technology*, R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, and B. Jin, Eds. Springer Berlin Heidelberg, 2012, pp. 584–595.

Figure 3. Time sequence of activity in the forums in the three courses by students and instructors grouped by activity type



BOTS: Selecting Next-Steps from Player Traces in a Puzzle Game

Drew Hicks
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
aghicks3@ncsu.edu

Yihuan Dong
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
ydong2@ncsu.edu

Rui Zhi
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
rzhi@ncsu.edu

Veronica Cateté
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
vmcatete@ncsu.edu

Tiffany Barnes
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
tmbarnes@ncsu.edu

ABSTRACT

In the field of Intelligent Tutoring Systems, data-driven methods for providing hints and feedback are becoming increasingly popular. One such method, Hint Factory, builds an interaction network out of observed player traces. This data structure is used to select the most appropriate next step from any previously observed state, which can then be used to provide guidance to future players. However, this method has previously been employed in systems in which each action a player may take requires roughly similar effort; that is, the "step cost" is constant no matter what action is taken. We hope to apply similar methods to an interaction network built from player traces in our game, BOTS; However, each edge can represent a varied amount of effort on the part of the student. Therefore, a different hint selection policy may be needed. In this paper, we discuss the problems with our current hint policy, assuming all edges are the same cost. Then, we discuss potential alternative hint selection policies we have considered.

Keywords

Hint Generation, Serious Games, Data Mining

1. INTRODUCTION

Data-driven methods for providing hints and feedback are becoming increasingly popular, and are especially useful for environments with user- or procedurally-generated content. One such method, Hint Factory, builds an interaction network out of observed player traces. An Interaction Network is a complex network of student-tutor interactions, used to model student behavior in tutors, and provide insight into problem-solving strategies and misconceptions. This data structure can be used to provide hints, by treating the Interaction Network similarly to a Markov Decision Process and selecting the most appropriate next step from the requesting user's current state. This method has successfully been employed in systems in which each action a player may take is of similar cost; for example in the Deep Thought logic tutor each action is an application of a particular axiom. Applying this method to an environment where actions are of different costs, or outcomes are of varying value will require some adaptations to be made. In this work, we discuss how we will apply Hint Factory methods to an interaction network built from player traces in a puzzle game, BOTS. In BOTS, each "Action" is the set of changes made to the program between each run. Therefore, using the current hint selection policy would result in very high-level hints comprising a great number of changes to the student's program. Since this is undesirable, a different hint selection policy may be needed.

2. DATA-DRIVEN HINTS AND FEEDBACK

In the ITS community, several methods have been proposed for generating hints/feedback from previous observations of users' solutions or behavior. Rivers et al propose a data-driven method to generate hints automatically for novice programmers based on Hint Factory[8]. They present a domain-independent algorithm, which automates hint generation. Their method relies on solution space, which utilizes graph to represent the solution states. In solution space, each node represents a candidate solution and each edge represents the action used to transfer from one state to another. Due to the existence of multiple ways to solve a programming problem, the size of the solution space is huge and thus it is impractical to use. A Canonicalizing model is used to reduce the size of the solution space. All states are transformed to canonicalized abstract syntax trees (ASTs). If the canonical form of two different states are identical, they can be combined together. After simplifying the solution space, hint generation is implemented. If the current state is incorrect and not in the solution space, the path construction algorithm will find an optimal goal state in the solution space which is closest to current state. This algorithm uses change vectors to denote the change between current state and goal state. Once a better goal state is found during enumerating all possible changes, it returns the current combination of change vectors. Each change vector can be applied to current state and then form an intermediate state. The intermediate states are measured by desirability score, which represents the value of the state. And then the path construction algorithm generates optimal next states based on the rank of the desirability scores of all the intermediate states. Thus a new path can be formed and added to the solution space, and appropriate hints can be generated.

Jin et al propose linkage graph to generate hints for programming courses[4]. Linkage graph uses nodes to represent program statements and direct edges to indicate ordered dependencies of those statements. Jin's approach applies matrix to store linkage graph for computation. To generate linkage matrix, first, they normalize variables in programs by using instructor-provided variable specification file. After variable normalization, they sort the statement with 3 steps: (i) preprocessing, which breaks a single declaration for multiple variables (e.g. int a, b, c) into multiple declaration statements (e.g. int a; int b; int c;); (ii) creating statement sets according to variable dependencies, which put independent statements into first set, put statements depend only on statements in the first set into second set, put statements depends only on statements in the first and second set into third set, and so on; (iii) in-set statement sorting, during which the statements are sorted in decreasing order within set using their variable signatures. In hint generation, they first generate linkage graphs with a set of correct solutions, as the sources for hint generation. They also compose the intermediate steps during program development into a large linkage graph, and assign a reward value to each state and the correct solution. Then, they apply value iteration to create a Markov Decision Process (MDP). When a student requires hint, tutor will generate a linkage graph for the partial program and try to find the closest match in MDP. If a match is found in MDP, the tutor would generate hint with the next best state based on highest assigned value. If a match is not found in current MDP, which means the student is taking a different approach from existing correct solutions, the tutor will try to modify those correct solutions to fit student's program and then provide hints.

Hint Factory[9] is an automatic hint generation technique which uses Markov decision processes (MDPs) to generate contextualized hints from past student data. It mainly consists of two parts - Markov Decision Process (MDP) generator and hint provider. The MDP generator runs a process to generate MDP values for all states seen in previous students' solutions. In this process, all the students' solutions are combined together to form a single graph. Each node of the graph represents a state, and each edge represents an action one student takes to transform from current state to another state. Once the graph is built, the MDP generator uses Bellman backup to assign values for all nodes. After updating all values, a hint file is generated. The hint provider uses hint file to provide hint. When a student asks for a hint at a existing state, hint provider will retrieve current state

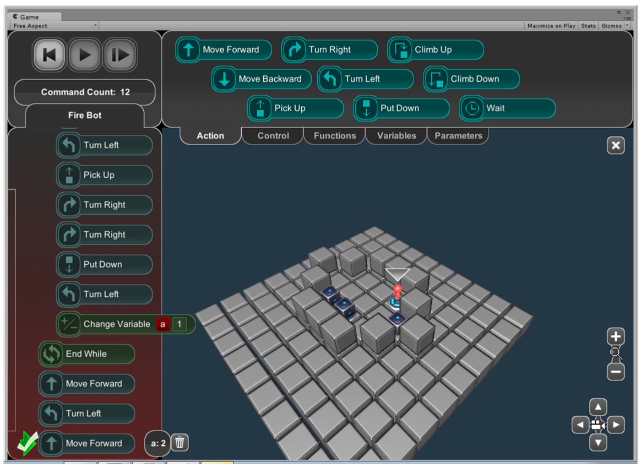


Figure 1: The BOTS interface. The robot's program is along the left side of the screen. The "toolbox" of available commands is along the top of the screen.

information and check if hints are available for the state. The action that leads to subsequent state with the highest value is used to generate a hint sequence. A hint sequence consists of four types of hints and are ordered from general hint to detailed hint. Hint provider will then show hint from top of the sequence to the student. Hint Factory has been applied in logic tutors which helps students learn logic proof. The result shows that the hint-generating function could provide hints over 80% of the time.

3. BOTS

BOTS is a programming puzzle game designed to teach fundamental ideas of programming and problem-solving to novice computer users. The goal of the BOTS project is to investigate how to best use community-authored content within serious games and educational games. BOTS was inspired by games like LightBot [10] and RoboRally [2], as well as the success of Scratch and it's online community [1] [5]. In BOTS, players take on the role of programmers writing code to navigate a simple robot around a grid-based 3D environment, as seen in Figure 1. The goal of each puzzle is to press several switches within the environment, which can be done by placing an object or the robot on them. To program the robots, players will use simple graphical pseudo-code, allowing them to move the robot, repeat sets of commands using "for" or "while" loops, and re-use chunks of code using functions. Within each puzzle, players' scores depend on the number of commands used, with lower scores being preferable. In addition, each puzzle limits the maximum number of commands, as well as the number of times each command can be used. For example, in the tutorial levels, a user may only use the "Move Forward" instruction 10 times. Therefore, if a player wants to make the robot walk down a long hallway, it will be more efficient to use a loop to repeat a single "Move Forward" instruction, rather than to simply use several "Move Forward" instructions one after the other. These constraints are meant to encourage players to re-use code and optimize their solutions.

In addition to the guided tutorial mode, BOTS also contains an extensive “Free Play” mode, with a wide selection of puzzles created by other players. The game, in line with the “Flow of Inspiration” principles outlined by Alexander Repenning [7], provides multiple ways for players to share knowledge through authoring and modifying content. Players are able to create their own puzzles to share with their peers, and can play and evaluate friends’ puzzles, improving on past solutions. Features such as peer-authored hints for difficult puzzles, and a collaborative filtering approach to rating are planned next steps for the game’s online element. We hope to create an environment where players can continually challenge their peers to find consistently better solutions for increasingly difficult problems.

User-generated content supports replayability and a sense of a community for a serious game. We believe that user-created puzzles could improve interest, encouraging students to return to the game to solidify their mastery of old skills and potentially helping them pick up new ones.

4. ANALYSIS

4.1 Dataset

Data for the BOTS studies has come from a middle school computer science enrichment program called SPARCS. In this program, the students attend class on Saturday for 4 hours where computer science undergraduates teach them about computational thinking and programming. Students attend a total of 7 sessions, each on a different topic, ranging from security and encryption to game design. The students all attend the same magnet middle school. The demographics for this club are 74.2% male, 25.8% female, 36.7% African American, and 23.3% Hispanic. The student’s grade distribution is 58% 6th grade, 36% 7th grade and 6% 8th grade.

From these sessions, we collected gameplay data for 20 tutorial puzzles as well as 13 user-created puzzles, With this data, we created an Interaction Network in order to be able to provide hints and feedback for future students [3]. However, using program edits as states, the interaction networks produced were very sparse. In order to be better able to relate similar actions, we produced another interaction network using program output as our state definition [6].

4.2 States and Transitions

Based on the data collected, we can divide the set of observed states into classes. First among these is the *start state* in which the problem begins. By definition, every player’s path must begin at this state. Next is the set of *goal states* in which all buttons on the stage are pressed. These are reported by the game as correct solutions. Any complete solution, by definition, ends at one such state. Among states which are neither start nor goal states, there are three important classifications: *Intermediate states* (states a robot moves through during a correct solution), *mistake states* (states a robot does not move through during a correct solution), and *error states* (states which result from illegal output, like attempting to move the robot out-of-bounds). Based on these types of states, we classified our hints based on the transitions they represented.

4.2.1 *Subgoal Transition*

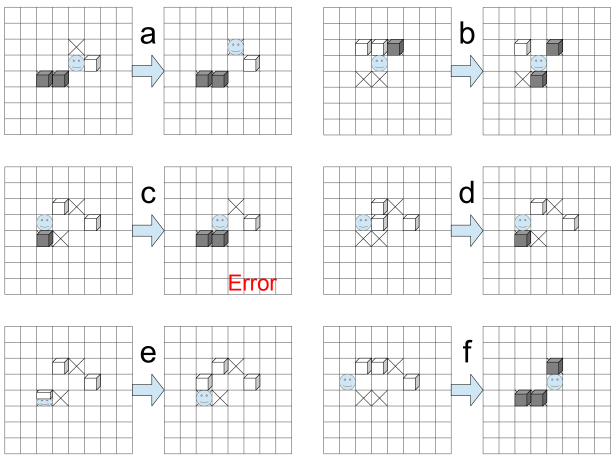


Figure 2: Several generated hints for a simple puzzle. The blue icon represents the robot. The ’X’ icon represents a goal. Shaded boxes are boxes placed on goals, while unshaded boxes are not on goals. Hint F in this figure depicts the start and most common goal state of the puzzle.

(start/intermediate) → *(intermediate/goal)* These transitions occur when a student moves the robot to an intermediate state rather than directly to the goal. Since players run their programs to produce output, we speculate that these may represent subgoals such asmoving a box onto a specific switch. After accomplishing that task, the user then appends to their program, moving towards a new objective, until they reach a goal state. Hint B in Figure 2 shows a hint generated from such a transition.

4.2.2 *Correction Transition*

(error/mistake) → *(intermediate/goal)* This transition occurs when a student makes and then corrects a mistake. These are especially useful because we can offer hints based on the type of mistake. Hints D and E in Figure 2 show hints built from this type of transition; however, hint E shows a case where a student resolved the mistake in a suboptimal way.

4.2.3 *Simple Solution Transition*

(start) → *(goal)* This occurs when a student enters an entire, correct program, and solves the puzzle in one attempt. This makes such transitions not particularly useful for generating hints, other than showing a potential solution state of the puzzle. Hint F in Figure 2 shows this type of transition.

4.2.4 *Rethinking Transition*

(intermediate) → *(intermediate/goal)* This transition occurs when rather than appending to the program as in a subgoal transition, the user deletes part or all of their program, then moving towards a new goal. As a result, the first state is unrelated to the next state the player reaches. Offering this state as a hint would likely not help guide a different user. Hint A in Figure 2 shows an example of this. Finding and recognizing these is an important direction for future work.

4.2.5 Error Transition

(start/intermediate) $\rightarrow$ *(mistake/error)* This corresponds to a program which walks the robot out of bounds, into an object, or other similar errors. While we disregarded these as hints, this type of transition may still be useful. In such a case, the last *legal* output before the error could be a valuable state. Hint C in Figure 2 is one such case.

4.3 Next Steps

While this approach was able to help us identify interesting transitions, as well as significantly reduce the sparseness of the Interaction Network by merging states with similar output, we violate several assumptions of the Hint Factory technique by using user compilation as an action. Essentially, the cost of an action can vary widely. In the most extreme examples, the best next state selected by Hint Factory will simply be the goal state.

4.4 Current Hint Policy

Our current hint selection policy is the same as the one used in the logic tutor Deep Thought with a few exceptions [9]. We combine all student solution paths into a single graph, mapping identical states to one another (comparing either the programs or the output). Then, we calculate a fitness value for each node. We assign a large positive value (100) to each goal state, a low value for dead-end states (0) and a step cost for each step taken (1). Setting a non-zero cost on actions biases us towards shorter solutions. We then calculate fitness values $V(s)$ for each state s, where $R(s)$ is the initial fitness value for the state, γ is a discount factor, and $P(s, s')$ is the observed frequency with which users in state s go to state s' next, via taking the action a. The equation for determining the fitness value of a state is as follows:

$$V(s) := R(s) + \gamma \max_a \sum_{s'} P_a(s, s')V(s') \qquad (1)$$

However, in our current representation there is only one available action from any state: "run." Different players using this action will change their programs in different ways between runs, so it is not useful to aggregate across all the possible resulting states. Instead, we want to consider each resulting state on its own. As a result, we use a simplified version of the above, essentially considering each possible resulting state s' as the definite result of its own action:

$$V(s) := R(s) + \gamma \max_{s'} P(s, s')V(s') \qquad (2)$$

Since the action "run" can encompass many changes, selecting the s' which maximizes the value may not always be the best choice for a hint. The difference between s and s' can be quite large, and this is usually the case when an expert user solves the problem in one try, forming an edge directly between the "start" state and "goal" state. These and other "short-circuits" make it difficult to assess which of the child nodes would be best to offer as a hint by simply using the calculated fitness value.

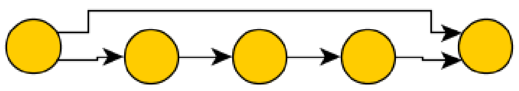


Figure 3: This subgraph shows a short-circuit where a player bypasses a chain of several steps.

Another problem which arises from this state representation is seen in Hints C and E above. These hints show states where a student traveled from a state to a worse state before ultimately solving the problem. Since we limit our search for hintable states to the immediate child states of s in s', we are unable to escape from such a situation if the path containing the error is the best or only observable path to the goal.

4.5 Proposed Hint Policies

One potential modification of the hint policy involves analyzing the programs/output on the nodes, using some distance metric $\delta(s, s')$. This measurement would be used in addition to the state's independent fitness value $R(s)$ which takes into account distance from a goal, but is irrespective of the distance from any previous state. For example in the short-circuit example above, using "difference in number of lines of code" as a distance metric we could take into account how far the "Goal" state is from the "Start" state, and potentially choose a nearer state as a hint. This also helps correct for small error-correction steps in player solutions; if the change between the current state and the target hint state is very small, we may want to consider hinting toward the next step instead, or a different solution path altogether.

$$V(s) := R(s) + \gamma \max_a \sum_{s'} \delta(s, s')P(s, s')V(s') \qquad (3)$$

One potential downside to this approach is that it requires somewhat more knowledge of the domain to be built into the model. If the distance metric used is inaccurate or flawed, there may be cases where we choose a very suboptimal hint. using difference in lines of code as our distance metric, the change between a state where a player is using no functions and a state where the user writes existing code into a function may be very small. Hints selected in these cases might guide students away from desired outcomes in our game.

Another problem we need to resolve with our current hint policy, as discussed above, is the case where the best or only path to a goal from a given state s has an error as a direct child s'. One method of resolving this could be, instead of offering s' as a hint, continuing to ask for next-step hints from s' until some s' is a hintable, non-error state. This solution requires no additional knowledge of the game domain, however it's possible that the hint produced will be very far from s, or that we may skip over important information about how to resolve the error or misconception that led the student into state s in the first place.

Other modifications to the hint selection policy may produce better results than these. We hope to look into as many possible modifications as we can, seeing which modifications produce the most suitable hints on our current dataset before settling on an implementation for the live version of the game.

5. ACKNOWLEDGMENTS

Thanks to the additional developers who have worked on this project or helped with our outreach activities so far, including Aaron Quidley, Trevor Brennan, Irena Rindos, Vincent Bugica, Victoria Cooper, Dustin Culler, Shaun Pickford, Antoine Campbell, and Javier Olaya. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 0900860 and Grant No. 1252376.

6. REFERENCES

[1] I. F. de Kereki. Scratch: Applications in computer science 1. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, pages T3B–7. IEEE, 2008.

[2] R. Garfield. Roborally. [Board Game], 1994.

[3] A. Hicks, B. Peddycord III, and T. Barnes. Building games to learn from their players: Generating hints in a serious game. In *Intelligent Tutoring Systems*, pages 312–317. Springer, 2014.

[4] W. Jin, T. Barnes, J. Stamper, M. J. Eagle, M. W. Johnson, and L. Lehmann. Program representation for automatic hint generation for a data-driven novice programming tutor. In *Intelligent Tutoring Systems*, pages 304–309. Springer, 2012.

[5] D. J. Malan and H. H. Leitner. Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39(1):223–227, 2007.

[6] B. Peddycord III, A. Hicks, and T. Barnes. Generating hints for programming problems using intermediate output.

[7] A. Repenning, A. Basawapatna, and K. H. Koh. Making university education more like middle school computer club: facilitating the flow of inspiration. In *Proceedings of the 14th Western Canadian Conference on Computing Education*, pages 9–16. ACM, 2009.

[8] K. Rivers and K. R. Koedinger. Automating hint generation with solution space path construction. In *Intelligent Tutoring Systems*, pages 329–339. Springer, 2014.

[9] J. Stamper, T. Barnes, L. Lehmann, and M. Croy. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*, pages 71–78, 2008.

[10] D. Yaroslavski. LightBot. [Video Game], 2008.

Semantic Graphs for Mathematics Word Problems based on Mathematics Terminology

Rogers Jeffrey Leo John
Center for Computational Learning Systems
Columbia University
New York, NY, USA
rl2689@columbia.edu

Thomas S. McTavish
Center for Digital Data, Analytics & Adaptive Learning
Pearson
Austin, TX, USA
tom.mctavish@pearson.com

Rebecca J. Passonneau
Center for Computational Learning Systems
Columbia University
New York, NY, USA
becky@ccls.columbia.edu

ABSTRACT

We present a graph-based approach to discover and extend semantic relationships found in a mathematics curriculum to more general network structures that can illuminate relationships within the instructional material. Using words representative of a secondary level mathematics curriculum we identified in separate work, we constructed two similarity networks of word problems in a mathematics textbook, and used analogous random walks over the two networks to discover patterns. The two graph walks provide similar global views of problem similarity within and across chapters, but are affected differently by number of math words in a problem and math word frequency.

1. INTRODUCTION

Curricula are compiled learning objects, typically presented in sequential order and arranged hierarchically, as in a book's Table of Contents. Ideally, a *domain model* captures relationships between the learning objects and the knowledge components or skills they exercise. Unfortunately, domain models are not often granular enough for optimal learning experiences. For example, prerequisite relationships may be lacking, or the knowledge components associated with an exercise may be unknown. In such cases, assessments on those learning objects will be insufficient to enable appropriate redirection unless expert (i.e. teacher) intervention is explicitly given. Domain models remain coarse because using experts to enumerate and relate the knowledge components is costly.

As a means to automatically discover relationships among learning objects and to reveal their knowledge components, we demonstrate the use of direct similarity metrics and random graph walks to relate exercises in a mathematics curriculum. We first apply a standard cosine similarity measure between pairs of exercises, based on bag-of-word vectors consisting of math terms that we identified in separate work [7]. Then, to extract less explicit relationships between exercises, we randomly walk a graph using the cosine distance as edge weights. We also recast the problem as a bipartite graph with exercises on one side and words on the other, providing an edge when an exercise contains the math word. We contrast these two different types of random walks and find somewhat similar results, which lends confidence to the analysis. The bipartite graph walks, however, are more sensitive to differences in word frequency. Casting measures of similarity as graphs and performing random walks on them affords more nuanced ways of relating objects, which can be used to build more granular domain models for analysis of prerequisites, instructional design, and adaptive learning.

2. RELATED WORK

Random walks over graphs have been used extensively to measure text similarity. Applications include similarity of web pages [15] and other documents [5], citations [1], passages [14], person names in email [12] and so on. More recently, general methods that link graph walks with external resources like WordNet have been developed to produce a single system that handles semantic similarity for words, sentences or text [16]. Very little work compares walks over graphs of the same content, where the graphs have different structure. We create two different kinds of graphs for mathematics word problem and compare the results. We find that the global results are very similar, which is good evidence for the general approach, and we find differences in detail that suggest further investigation could lead to customizable methods, depending on needs.

An initiative where elementary science and math tests are a driver for artificial intelligence has led to work on knowledge extraction from textbooks. Berant et al. [2] create a system to perform domain-specific deep semantic analysis of a 48 paragraphs from a biology textbook for question answering. Extracted relations serve as a knowledge base against which to answer questions, and answering a question is treated as finding a proof. A shallow approach to knowledge extraction from a fourth grade science curriculum is taken in [6], and the knowledge base is extended through dialog with users until a path in the knowledge network can be found that supports a known answer. In the math domain, Kushman et al. [10] generate a global representation of algebra problems in order to solve them by extracting relations from sentences and aligning them. Seo et al. [18] study text and diagrams together in order to understand the diagrams better through textual cues. We are concerned with alignment of content

Two machines produce the same type of widget. Machine A produces W widgets, X of which are damaged. Machine B produces Y widgets, Z of which are damaged. The **fraction** of damaged widgets for Machine A is $\frac{X}{W}$ or (*simplified fraction*). The **fraction** of damaged widgets for Machine B is $\frac{Z}{Y}$ or (*simplified fraction*). Write each **fraction** as a **decimal** and a **percent**. Use pencil and paper. Select a small **percent** that would allow for a small **number** of damaged widgets. Find the **number** of widgets by which each machine exceeded the acceptable **number** of widgets.

Figure 1: Sample problem; math terms are in boldface.

across rather than within problems, and our objective is finer-grained analysis of curricula.

Other work that addresses knowledge representation from text includes ontology learning [3], which often focuses on the acquisition of sets of facts from text [4]. There has been some work on linking lexical resources like WordNet or FrameNet to formal ontologies [17, 13], which could provide a foundation for reasoning over facts extracted from text. We find one work that applies relation mining to e-learning: Šimko and Bieliková [19] apply automated relation mining to extract relations to support e-course authoring in the domain of teaching functional programming. Li et al. [11] apply k-means clustering to a combination of problem features and student performance features, and propose the clusters correspond to Knowledge Components [8].

3. METHODS

3.1 Data

We used 1800 exercises from 17 chapters of a Grade 7 mathematics curriculum. Most are word problems, as illustrated in Figure 1. They can incorporate images, tables, and graphs, but for our analysis, we use only the text. The vocabulary of the resulting text consists of 3,500 distinct words. We construct graphs where math exercises are the nodes, or in a bipartite graph, math exercises are the left side nodes and words are the right side nodes. Our initial focus is on exercise similarity due to similarity of the math skills that exercises tap into, and we use mathematics terminology as an indirect proxy of skills a problem draws upon.

3.2 Math Terminology

The text of the word problems includes ordinary language expressions unrelated to the mathematics curriculum, such as the nouns *machines, widgets* shown in problem in Figure 1, or the verbs *produces, damaged*. For our purposes, mathematics terminology consists of words that expresses concepts that are needed for the mathematical competence the curriculum addresses. To identify these terms, we developed annotation guidelines for human annotators who label words in their contexts of use, and assessed the reliability of annotation by these guidelines. Words can be used in the math texts sometimes in a math sense and sometimes in a non-math sense. Annotators were instructed to label terms based on the most frequent usage.

Using a chance-adjusted agreement coefficient in [-1,1] [9], reliability among three annotators was 0.81, representing high agreement. All the non-stop words were then labeled by a trained annotator. We developed a supervised machine learning approach to classify vocabulary into math and non-math words [7] that can be applied to new mathematics curricula. For the text used here, there were 577 math terms.

3.3 Random Walks in Graphs

A random walk on a graph starts at a given node and steps with random probability to a neighboring node. The same random decision process is employed at this and every subsequent node until a termination criterion is met. Each time a node is visited, it is counted. Open random walks require that the start node and end nodes differ. Traversal methods may employ a bias to navigate toward or away from certain neighbors through edge weights or other graph attributes.

In a graph, $G = (V, E)$ with nodes V and edges E, a random walk that begins at v_x and ends at v_y can be denoted as $(v_x, ..., v_y)$. By performing several random walks, the fraction of times the node v_y is visited converges to the probability of target v_y being visited given the start node v_x, which can be expressed as $P(v_y|v_x)$ under the conditions of the walk. In the case of a random walk length of 1, $P(v_y|v_x)$ will simply measure the probability of v_y being selected as an adjacent node to v_x.

3.4 Cosine Similarity Graph

Math exercises are represented as bag-of-words vectors with boolean values to indicate whether a given math term is present. Cosine similarity quantifies the angle between the two vectors, and is given by the dot product of two vectors.

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \qquad (1)$$

Similarity values of 1 indicate that both the vectors are the same whereas a value of zero indicates orthogonality between the two vectors. Pairwise cosine similarities for all 1800 exercises were computed, yielding a cosine similarity matrix M_{cos}. The matrix corresponds to a graph where non-zero cosine similarities are edge weights between exercises.

In a graph walk, the probability that a node v_y will be reached in one step from a node v_x is given by the product of the degree centrality of v_x and the normalized edge weight (v_x, v_y). With each exercise as a starting node, we performed 100,000 random walks on the cosine-similarity graph, stepping with proportional probability to all outgoing cosine similarity weights. To measure 2nd degrees of separation, with each walk we made two steps.

For two math vectors considered as the sets A and B, cosine similarity can be conceptualized in terms of the intersection set $C = A \cup B$ and set differences $A \setminus B$ and $B \setminus A$. Cosine similarity is high when $|C| \gg A \setminus B$ and $|C| \gg B \setminus A$.

The degree of a node affects the probability of traversing any edge from that node. The two factors that affect degree centrality of a start node are the document frequencies of its math words, and the total number of math words. Here, document frequency (df) is the normalized number of exercises a word occurs in. A high df math word in a problem increase its degree centrality because there will be more

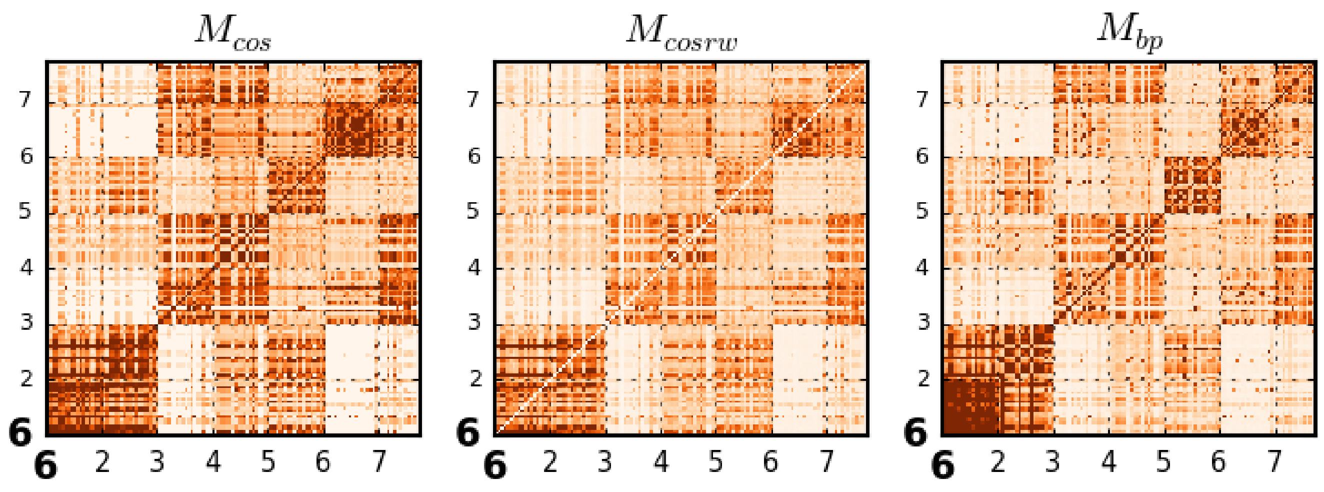


Figure 2: Exercise-to-Exercise similarity in Chapter 6. The exercises of Chapter 6 are displayed in row-columns in a square matrix. The rows represent source nodes and columns represent targets. Each row has been normalized across the book, even though only Chapter 6 is shown. The axes demarcate the sections of the chapter. M_{cos} is the cosine similarity. M_{cosrw} is the output from the random walk using cosine similarity edge weights, M_{bp} is the output from the random bipartite walk. Raw values displayed between 0 and 0.005 corresponding to light and dark pixels, respectively.

problems it can share words with, resulting in non-zero cosine values and therefore edges. The number of math words in a problem also increases its degree centrality.

3.5 Bipartite exercise and word graph

The set of exercises V_e are the left-side nodes and the math words V_w are the right-side nodes in the undirected bipartite graph $G = (V_e, V_w, E)$, where an edge exists between v_{ex} and v_{wi} if exercise x contains the math word i.

We performed open random walks on this graph to measure similarity between nodes. To measure the similarity of exercises, we walk in even steps – a step to a connected word followed by a step back to one of the exercises that shares that word. The degrees of separation between vertices on the same side of the graph (e.g. exercise-to-exercise) will be $l/2$ where l is the length of the walk. In this paper, we explored first and second degrees of separation so our bipartite graphs had a walk length of 4.

Table 1: Summary statistics of the similarity distributions

	cosine	rw$_{cos}$	rw$_{bp}$
minimum	0	0	0
maximum	6.3×10^{-2}	0.50	0.11
mean	5.55×10^{-4}	5.55×10^{-4}	5.55×10^{-4}
median	0	2.41×10^{-4}	2.06×10^{-4}
std. dev.	1.19×10^{-3}	8.57×10^{-4}	1.24×10^{-3}

Because exercise nodes are connected via word nodes, we interpret the fraction of node visits as a similarity measure between the source node and any node visited. We performed 100,000 random walks from each node. Exercise-to-exercise similarity can be visualzed as square matrices with source nodes in the rows and target nodes in the columns. To factor out the times a source may have been selected as one of the targets, we set the diagonal of the matrix to zero. We then normalized across the rows so that we could interpret the distribution across the row as a probability distribution to all other nodes for that source node.

4. RESULTS

We compare the three measures of similarity between exercises: 1) cosine similarity, 2) random walks using cosine similarity as edge weights, and 3) random walks along a bipartite graph of exercises and words.

4.1 Exercise-to-Exercise Similarity

We describe exercise-to-exercise similarity with square matrices where each exercise is represented as a row-column. A number of features of the measures are embedded in Figure 2, which shows heatmaps of color values for pairs of exercises in chapter 6 for each matrix. We find that within chapters and especially within sections of those chapters, there is a high degree of similarity between exercises regardless of the measure. This demonstrates that words within sections and chapters share a common vocabulary. We can see that M_{cos} has more extreme values than M_{cosrw}; as explained below, it has both more zero cosine values, and more very high values. This is most likely because M_{cosrw}, from doing the walk, picks up exercises that are another degree of separation away. When the row of the matrix is normalized to capture the distribution of the source node, the otherwise high values from M_{cos} are tempered in the M_{cosrw} matrix. This shift to a large number of lower scores is shown in the bottom panel of Figure 3. M$_{bp}$ and M$_{cosrw}$ are very similar, but M$_{bp}$ generally has a wider dynamic range.

4.2 Comparison of the Graph Walks

Table 1 provides summary statistics for cosine similarity and the two random walks for all pairs of problems (N=3,250,809). The cosine matrix is very sparse, as shown by the median value of 0. Of the two random walks similarities, rw$_{cos}$ has a lower standard deviation around the mean, but otherwise the two random walks produce similar distributions.

The similarity values given by cosine and the cosine random walk will increasingly differ the more that the start problem has relatively higher degree centrality due either to more words or higher frequency of words in exercises (df). For reference, the word that occurs most frequently, *number*, has a df of 0.42, and the second most frequent occurs in only 15% of the exercises. Fifty eight nodes have no edges (0 degree), the most frequent number of edges is 170, and the maximum is 1,706. Table 2 gives the summary statistics for df, number of math words, and degree centrality.

Inspection of the data shows that for pairs of problems in the two chapters for our case study, if the cosine similarity between a pair is high (≥ 0.75), the similarity values for rw_{cos} tend to go down as the number of shared word increases from 3 to between 5 and 7. For the rw_{bp}, the opposite trend occurs, where the similarity goes up as the number of words increases. This difference helps account for an observed divergence in the two graph walks for sections 5 and 6 of Chapter 6.

Table 3 illustrates two pairs of problems from section 5 that have high cosine similarities, and relatively higher rw_{bp} similarities (greater than the rw means of 0.0055) and relatively lower rw_{cos} (lower than the rw means). The reverse pattern is seen for two pairs of problems from section 6 that have high cosine similarities. These problems have higher than average rw_{cos} and lower than average rw_{bp}. What differentiates the two pairs of problems is that the section 5 problems have a relatively large number of words in common: 14 for the first pair, 12 for the second pair. In both pairs, some of the words have relatively high document frequency. As discussed above, these two properties increase the degree centrality of the start node of a step in the rw_{cos} graph, and thus lower the probability of hitting each of the start node's one-degree neighbors. This effect propagates along the two steps of the walk. For the rw_{bp} graph, however, as the number of shared math words for a pair of problems increases, the number of paths from one to the other also increases, thus raising the probability of the traversal. This effect also propagates through a two-step walk. In contrast to the section 5 problems, the two section 6 problems have relatively fewer words in common: 3 for both pairs.

For problem pairs where the cosine similarity is between 0.40 and 0.60, the mean similarity from rw_{bp} is 30% higher than for rw_{cos} for when the number of math words in common is 3 (0.0033 vs. 0.0043), 80% higher when the number of math words in common is 6 (0.0024 versus 0.0045), and three as high when the number of math words in common is 9 (0.0023 versus 0.0068). For problems pairs where the cosine similarity is less than 0.20, the two walks produce very similar results. The average similarity values for the bipartite walk are about 20% higher, and the maximum values are higher, but the two walks produce similar means, independent of the lengths of the common word vectors, or the total number of math words.

Table 2: Summary statistics of document frequency (df) of math words, number of math words in problems, and degree centrality of rw_{cos}

	df	math words	degree ctr. rw_{cos}
minimum	5.54×10^{-4}	1	0
maximum	0.424	24	1,706
mean	0.183	8.35	418.4
median	6.66×10^{-3}	8.00	340
std. dev.	0.314	3.64	317.1

Table 3: Two pairs of problems with high cosine similarity and reverse patterns of graph walk similarity. The first pair, from section 5, have lower than average rw_{cos} and higher than average rw_{bp} due to relatively many words in common (12 and 14). The second pair, from section 6, have higher than average rw_{cos} and lower than average rw_{bp} due to relatively few words in common.

Prob 1	Prob 2	cosine	rw_{cos}	rw_{bp}	N	max df
6.5.99	6.5.85	1.0000	0.0032	0.0102	12	0.42
6.5.94	6.5.83	0.8819	0.0026	0.0064	14	0.13
6.6.109	6.6.102	0.8660	0.0068	0.0037	3	0.11
6.6.104	6.6.102	0.7746	0.0068	0.0029	3	0.11

Since we normalized the matrices across rows, which are the source nodes, differences between the bipartite matrix, M_{bp}, and the cosine matrices implied that the degree of the target node had a greater impact on the variability in the bipartite matrix. To measure the impact of the edge degree on the target nodes, we considered the column sum for those targets that had 1 edge, those that had 2, etc. up to 20 edges. The results are summarized in Figure 4. As can be seen, the column sum varies linearly by the number of target edges in the bipartite matrix, whereas the cosine matrices do not. We found the cubed root of the column sum in M_{bp} approaches the distribution of column sums of the cosine matrices, which is provided in Figure 4.

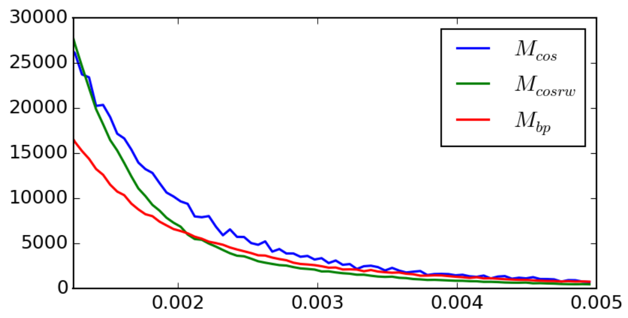


Figure 3: Tail distribution of similarity values in M_{cos}, M_{cosrw}, and M_{bp}. Because 62% of the values in M_{cos} are 0, the plot shows only non-zero values.

5. CONCLUSION

Visualization of the three similarity matrices shows they reveal the same overall patterns, thus each is confirmed by the others. However, the bipartite walk was the most sensitive to word frequency across exercises, and the number of words in problems. With our goal of automatically discovering knowledge components and identifying their relationships, the random walk that stepped in proportion to its cosine similarity performed best. It was able to discover second-degree relationships that seem reasonable as we explore by eye those matches. Future work will test these re-

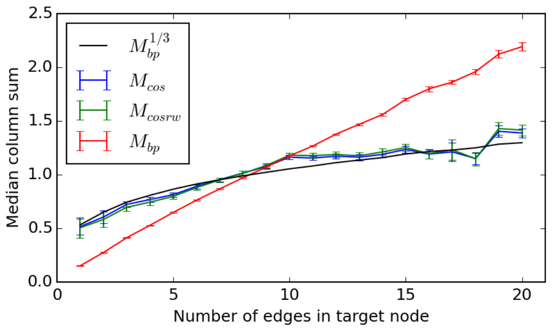


Figure 4: Distribution of column sums by number of edges in the target node represented by the column. Error plots show the mean and standard error for each type. Black line is the cubed root of the mean of the column sums of M_{bp}.

lationships with student performance data. We should find, for example, that if two exercises are conceptually similar, then student outcomes should also be similar and learning curves should reveal shared knowledge components. In this respect, such automatically constructed knowledge graphs can create more refined domain models that intelligent tutoring systems and robust assessments can be built upon.

6. REFERENCES

[1] Y. An, J. Janssen, and E. E. Milios. Characterizing and mining the citation graph of the computer science literature. *Knowledge and Information Systems*, 6(6):664–678, 2004.

[2] J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] P. Buitelaar, A. Frank, M. Hartung, and S. Racioppa. Ontology-based information extraction and integration from heterogeneous data sources, 2008.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1306–1313, 2010.

[5] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, Dec. 2004.

[6] B. Hixon, P. Clark, and H. Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, CO, May-June 2015.

[7] R. J. L. John, R. J. Passonneau, and T. S. McTavish. Semantic similarity graphs of mathematics word problems: Can terminology detection help? In *Proceedings of the Eighth International Conference on Educational Data Mining*, 2015.

[8] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction (KLI) framework: Toward bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36(5):757–798, 2012.

[9] K. Krippendorff. *Content analysis: An introduction to its methodology*. Sage Publications, Beverly Hills, CA, 1980.

[10] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[11] N. Li, W. W. Cohen, and K. R. Koedinger. Discovering student models with a clustering algorithm using problem content. In *Proceedings of the 6th International Conference on Educational Data Mining*, 2013.

[12] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 27–34, New York, NY, USA, 2006. ACM.

[13] I. Niles and A. Pease. Mapping WordNet to the SUMO ontology. In *Proceedings of the IEEE International Knowledge Engineering Conference*, pages 23–26, 2003.

[14] J. Otterbacher, G. Erkan, and D. R. Radev. Biased lexrank: Passage retrieval using random walks with question-based priors. *Information Processing and Management*, 45(1):42–54, 2009.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[16] T. M. Pilehvar, D. Jurgens, and R. Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1351. Association for Computational Linguistics, 2013.

[17] J. Scheffczyk, A. Pease, and M. Ellsworth. Linking FrameNet to the suggested upper merged ontology. In B. Hennett and C. Fellbaum, editors, *Formal Ontology in Information Systems*, pages 289–. IOS Press, 2006.

[18] M. J. Seo, H. Hajishirzi, A. Farhadi, and O. Etzioni. Diagram understanding in geometry questions. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[19] M. Šimko and M. Bieliková. Automatic concept relationships discovery for an adaptive e-course. In *Proceedings of the Second International Conference on Educational Data Mining (EDM)*, pages 171–179, 2009.