Workshop on Graph-Based Educational Data Mining (G-EDM)

Graph data has become increasingly prevalent in data-mining and data analysis generally. Many types of data can be represented naturally as graphs including social network data, log traversal, and online discussions. Moreover recent work on the importance of social relationships, peer tutoring, collaboration, and argumentation has highlighted the importance of relational information in education including:

- Graphical solution representations such as argument diagrams and concept maps;
- Graph-based models of problem-solving strategies;
- User-system interaction data in online courses and open-ended tutors;
- Sub-communities of learners, peer-tutors and project teams within larger courses; and
- Class assignments within a larger knowledge space.

Our goal in this workshop was to highlight the importance of graph data and its relevance to the wider EDM community. We also sought to foster the development of an interested community of inquiry to share common problems, tools, and techniques. We solicited papers from academic and industry professionals focusing on: common problems, analytical tools, and established research. We also particularly welcomed new researchers and students seeking collaboration and guidance on future directions. It is our hope that the papers published here will serve as a foundation for ongoing research in this area and as a basis for future discussions.

The papers included here cover a range of topics. Kovanovic, Joksimovic, Gasevic & Hatala focus on evaluating social networks, and specifically on the development of social capital and high-status individuals in a course context while Catete, Hicks, Barnes, & Lynch describe an online tool designed to promote social network formation in new students. Similar work is also described by Jiang, Fitzhugh & Warschauer who focus on the identification of high-connection users in MOOCs.

Other authors turned to the extraction of plan and hint information from course materials and user logs. Belacel, Durand, & Laplante define a graph-based algorithm for identifying the best path through a set of learning objects. Kumar describes an algorithm for the automatic construction of behavior graphs for example-tracing tutors based upon expert solutions and Dekel & Gal in turn consider plan identification to support automatic guidance. Two further papers by Vaculík, Nezvalová & Popelínský, and by Mostafavi & Barnes, apply graph analysis techniques to the specific domain of logic tutoring and, in particular, on the classification of student solutions and to the evaluation of problem quality.

And finally several authors chose to present general tools for the evaluation of graphical data. Lynch describes Augmented Graph Grammars, a formal rule representation for the analysis of rich graph data such as argument diagrams and interconnected student assignments, and details an implementation of it. Sheshadri, Lynch, & Barnes present InVis a visualization and analysis platform for student interaction data designed to support the types of research described above. And

McTavish describes a general technique to support graph analysis and visualization particularly for student materials through the use of interactive hierarchical edges. We thank the included authors for their contributions to the discussion and look forward to continued research.

The G-EDM workshop organizers
Collin F. Lynch
Tiffany M. Barnes

Table of Contents G-EDM

A Binary Integer Programming Model for Global Optimization of Learning Path Discovery

Nabil Belacel
National Research Council Canada
100, des Aboiteaux St.
Moncton, E1A 7R1,Canada
+1.506.861.0963
nabil.belacel@NRC.gc.ca

Guillaume Durand
National Research Council Canada
100, des Aboiteaux St.
Moncton, E1A 7R1,Canada
+1.506.861.0961
guillaume.durand@NRC.gc.ca

François Laplante
Université de Moncton
60, Notre-Dame-du-Sacré-Cœur St.
Moncton, E1A 3E9,Canada
+1.506.381.6220
francois.laplante@umoncton.ca

ABSTRACT

This paper introduces a method based on graph theory and operations research techniques to optimize learning path discovery. In this method, learning objects are considered as nodes and competencies as vertices of a learning graph. A first step consists in reducing the solution space by obtaining an induced subgraph H. In a second step, the search of an optimal learning path in H is considered as a binary integer programming problem which we propose to solve using an exact method based on the well-known branch-and-bound algorithm. The method detailed in the paper takes into account the prerequisite and gained competencies as constraints of the optimization problem by minimizing the total competencies needed to reach the learning objective.

Keywords

Learning path, learning object recommendation, graph theory, clique, mathematical programming, binary integer programming, branch-and-bound algorithm.

1. INTRODUCTION

Global Positioning System (GPS) is a Global Navigation Satellite System (GNSS) that is massively used by car drivers. This large acceptance is easily understandable by the benefits that such a system can offer. Car navigation systems can dynamically calculate an itinerary between two points taking into account, depending on the system, several constraints like duration, distance, closed roads, traffic jams, etc....Drivers can focus exclusively on their driving limiting risks of accidents, stress, and losing their way.

To some extent, the learning path followed by a student could be seen as an itinerary between several learning objects [9]. In this context, constraints on learning objects are not distance or time duration to go from one learning object to the other but rather prerequisite and gained competencies. As a result the itinerary or path between learning objects is regulated by competency dependencies that lead a learner from an initial to a targeted competency state. For example, a learner with solid grounds in integer arithmetic (starting location) willing to learn the solving of systems with multiple variables (destination) should be advised to previously learn to solve one variable linear equations (next step of the itinerary).

Over the years, educational data mining and recommendation technologies have proposed significant contributions to provide learners with adequate learning material by recommending educational papers [18] or internet links [10], using collaborative and/or content-based filtering. These approaches usually aim at recommending learning material satisfying an immediate interest rather than fitting in the learner's sequential learning process.

Sequential pattern [28] and process mining [19] technologies have also been investigated. However, these technologies have been used to understand the learner's interaction with content to discover general patterns and trends rather than to recommend adapted learning paths to learners.

Other approaches, in the course generation research community, address the need for recommending not only the learning objects themselves, but sequences of learning objects. Sicilia et al. [17] or Ulrich and Melis [20] addressed learning design concepts and requirements through Course Generation. Though numerous solutions have been proposed, using statistical methods [13], decision rules [23], production rules [11], Markov processes [8] and Hierarchical Task Network Planning [17, 21, 22], most of them do not take into account eventual competency dependencies among learning objects and/or are not designed for large repositories of interdependent learning objects[1].

Therefore, we detailed in [7] a dynamic graph based model and a heuristic approach tailored to find a learning path in a graph containing millions of learning object nodes.

This paper is an extension of this previous work and summarizes the model, the heuristic presented in [7], and proposes a major optimization to calculate a global optimum learning path. In the previous work [7], we applied a greedy heuristic algorithm to obtain a pseudo-optimal learning path from a set of cliques. Greedy heuristics are efficient, but they sometimes get stuck in a local solution and fail to find a global optimum [26]. They are based on an intimate knowledge of the problem structure and have no scope of incremental improvement.

[1] A more complete discussion can be found in [7].

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

Therefore, in this work we slightly reformulate our model in order to fit as an integer programming problem and we propose an exact method based on the branch-and-bound algorithm.

2. PROBLEM CONSIDERED

In order to facilitate the understanding of the presented model, several key elements and assumptions need to be clearly defined.

A competency can be seen as a knowledge component being part of a "model that decomposes learning into individual knowledge components (KCs)" [16]. In this paper, a competency is "an observable or measurable ability of an actor to perform a necessary action(s) in given context(s) to achieve a specific outcome(s)" [12]. A competency in our situation can be a prerequisite to the efficient completion of a learning object. According to Wiley [25], a learning object is "any digital resource that can be reused to support learning". In the rest of the paper we define the learning object as any digital resource that can be reused to provide a competency gain.

A learner is a dynamic user interacting with learning objects in order to increase his/her competencies from an initial set to a targeted set of competencies. We assume that a learner completing a learning object will gain the competencies targeted to be transmitted by the interaction with the learning object. We also assume that a learner who would not possess the prerequisite set of competencies required by a learning object should not attempt this learning object since this would result in a limited competency gain.

Last but not least, we assume that the number of learning objects available is very large (millions to billions of learning objects) and that each learning object cannot provide the gain of a competency that is a pre-requisite to itself.

2.1 Graph Theory Contribution

Graph theory aims at studying mathematical structures composed of elements having relationships or connection between them. The use of directed graphs is not a novelty in e-learning systems [1, 3, 24, 25]; however, we were unable to find a formal model for discussing learning path problems based on graph theory, especially one taking into account the dynamic nature of a learning environment.

A directed graph, or digraph, $G = (V, E)$ consists of:

- A non-empty finite set V of elements called vertices or nodes,
- A finite set E of distinct ordered pairs of vertices called arcs, directed edges, or arrows.

Let $G = (V, E)$ be a directed graph for a personalized learning path. Each vertex or node in G corresponds to a learning object. Two vertices are connected if there exists a dependency relation, such that one vertex satisfies the prerequisites of the other. So, each edge between two vertices $Arc\{u, v\}$ means that the learning object v is accessible from u. The accessibility property required to define edges between vertices relies on post and pre-requisite competencies associated to each learning object. Considering $Arc\{u, v\}$, this edge means that after having completed the learning object u, the learner should have the required competencies to undertake resource v. By extension, each vertex v is represented by a pair (C_{pre}, C_{post}) where:

- C_{pre} is a set of the competencies required by vertex v
- C_{post} is a set of competencies offered by vertex v

The relationship between learning objects and competencies is multidimensional [6]: a learning object can require several competencies and transmit more than one competency to the learner as well. The existence of an edge between two learning objects u and v can be formalized by the following formula:

$$C_{pre}(v) \subseteq C_{post}(u) \Rightarrow Arc\{u, v\}$$

$$(Condition\ 1)$$

where $C_{pre}(v) \subseteq C_{post}(u)$ means that the competencies required by v are provided by learning object u. *Condition 1* is sufficient but not necessary. For example, before having completed u, the learner might already have some or the totality of the competencies required by v. This means that we may have an arc between u and v even though none the competencies required by v are provided by u. In other words, edge set E also depends on the learner's competency set at time t: $E = E(\mathrm{C}_{\mathrm{learner}}(t))$ and $\mathrm{C}_{\mathrm{learner}}(\mathrm{t}) = \{c_1, \ldots, c_n\}$ where $c_1 \ldots c_n$ are competencies which the learner possesses. As a result, graph G is a dynamic directed graph and *condition 1* can be strengthened by the necessary and sufficient *condition 2*:

$$Arc\{u, v\} \Leftrightarrow C_{pre}(v) \subseteq C_{post}(u) \cup C_{learner}(t)$$

$$(Condition\ 2)$$

2.2 Model Dynamicity

The dynamicity of our model is due to the fact that a learning object can bring competencies that could be among the prerequisites of future learning objects.

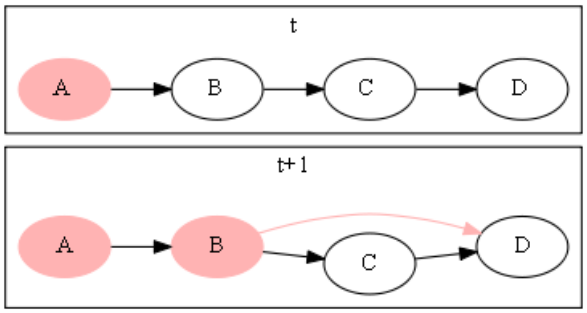


Figure 1. Edge dynamicity.

For example, as shown in Figure 1, a learning object D could be accessible to a learner if he has acquired the competencies c1 and c2. Assuming that competency c1 is provided by learning objects A and C and competency c2 is provided by learning objects B and C; D is reachable if learning objects A and B are completed or if learning object C is completed. If a learner completes learning object A at time t and learning object B at time t+1, the learner will have the competencies required to reach D and according to the condition 2, a new edge between B and D will be created (red edge on Figure 1).

3. INVESTIGATED SOLUTION

3.1 Reducing the solution space

Eliminating irrelevant learning objects is generally the first step of a course generation tool [1, 15]. In our case, as the learning object repository is supposed to be very large, the learning objects

cannot all be checked individually. The approach we chose consists in reducing the considered solution space by obtaining an induced subgraph H which consists of all the vertices and edges between the vertices in G that could be used in the learning path.

The algorithm can be seen as a loop generating complete sub-graphs, or cliques, until one such clique is generated whose prerequisites are a subset of the learner's competencies. Cliques are generated in a top-down fashion where we begin with the target clique, which is composed of a single learning object (we create a fictitious learning object, β, whose prerequisite competencies correspond to the list of the learner's target competencies). Cliques are then generated by finding every vertex where at least one output competency is found in the prerequisite competencies of the clique (the union of all prerequisite competencies of every learning object within the clique) to which it is prerequisite. As such, cliques contain the largest possible subset of vertices which satisfies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible". We simplify the stopping condition by adding a second fictitious object, α, into the dataset with no prerequisite competencies and with the learner's current competencies as its output competencies. If a clique contains this object, the stopping condition is true.

	β_6	
v_1	$A^6_5 \quad E^6_{3,5}$	↑ **6**
v_2	$T^{3,2,4}_7 \; U^5_0$	↑ **3,5**
v_3	$L^{0,7}_{8,9} \; I^7_9 \; K^0_8$	↑ **0, 7**
	$A^{8,9}$	↑ **8, 9**

α: Fictitious LO with initial learner competency state

β: Fictitious LO with targeted learner competency state

$LO^{\text{list of gained competencies}}\ LO_{\text{list of prerequisite competencies}}$

Figure 2. Induced sub-graph generation.

Considering the target competency β as shown in Figure 2, all the vertices leading to those competencies (competency 6 in Figure 2) are selected in a set v_1, then the learning objects leading to the prerequisites of set v_1 (competencies 3 and 5) are selected from graph G to create the set v_2. This mechanism continues until the prerequisite competencies of the set v_n are all competencies which the learner has already acquired.

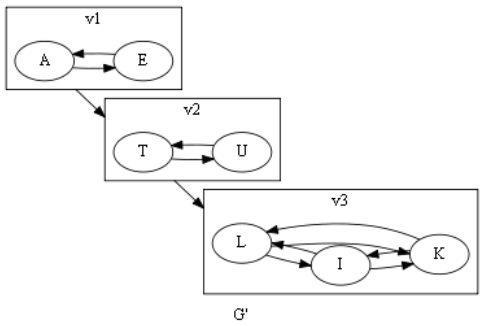


Figure 3. G' consists of connected cliques.

As shown in Figure 3, G', consisting of the vertices E of sets $v_1,\ldots,v_n$, is an induced sub-graph of G. If the learner has completed all the vertices of v_i, he/she will have access to all the vertices of v_{i+1}, thus all subsets of vertices of v_i can be considered to be a clique.

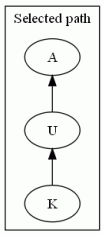
In addition to reducing the solution space, clique generation is also an efficient way to check whether a solution learning path exists between α and β. If the algorithm is not able to generate cliques linking α and β, there is no need to proceed forward with an algorithm aiming at finding one of the possible solutions.

3.2 Greedy Algorithm

Once the induced sub-graph is obtained, we use a greedy algorithm that searches for a local optimum within each clique. The definition of such a local optimum, depending on the dataset and the results pursued, has to follow a specific heuristic or strategy.

The shortest path strategy seems to be widely accepted in the literature [1, 27]. This strategy is not necessarily the best to adopt in any situation since the proposed learning path might lead to the learning of non–essential competencies and potentially cognitive overloads. For example a learning object could lead to competency gains that would not be required to reach the targeted learner competency state; there is no need to understand the proof of the Landau Damping to learn about the history of theoretical physics. Considering a learning object presenting an introduction to the perturbation theory and a second one introducing the theory and the proof of the Landau Dumping, it might make sense to choose the first one in order to minimize the cognitive load to the learner. Some might argue that using such "straight to the point" heuristic might limit too drastically the natural curiosity of the learner. As any heuristic, we agree that it is discussable but this is not the purpose of this paper.

The greedy algorithm considered attempts to find a path by considering each clique one after the other and reducing it to a minimal subset of itself which still verifies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible".



	β_6	
v_1	$\boldsymbol{A^6_5}$ $E^6_{3,5}$	$\uparrow$ **6**
v_2	$T^{3,2,4}_7$ $\boldsymbol{U^5_0}$	$\uparrow$ **3,5**
v_3	$L^{0,7}_{8,9}$ I^7_9 $\boldsymbol{K^0_8}$	$\uparrow$ **0,7**
	$A^{8,9}$	$\uparrow$ **8, 9**

α: Fictitious LO with initial learner competency state

β: Fictitious LO with targeted learner competency state

$LO^{\text{list of gained competencies}}\ LO_{\text{list of prerequisite competencies}}$

Figure 4. Illustration of the greedy algorithm execution

The first clique considered will be the one leading to the targeted competencies (the clique satisfying the prerequisites of β). In the case of the three cliques v_1 to v_3 as illustrated by Figure 3, v_1 will be considered first followed by v_2 then by v_3.

For each clique, the local optimum is considered obtained when the minimum subset of vertices with a minimum "degree", being the sum of the number of prerequisite competencies and output competencies of the vertex, are found. In other words, the greedy algorithm select in each clique a set of learning objects minimizing the number of competencies required and gained in order to locally limit the cognitive load of the selected material. The greedy algorithm locally optimizes a function called "deg" (for degree) detailed in the following section.

For clique v_1, the selected learning object is A since its number of prerequisites is smaller than that of E while they share the same competency gain. As A has been chosen in v_1, only the objects in v_2 respecting the new v_1's prerequisites is chosen. As a result, the algorithm chooses U in v_2. In v_3, K and L lead to v_2's prerequisite but K requires fewer prerequisites than L, therefore K is selected and the proposed learning path is $K \rightarrow U \rightarrow A$.

4. OPTIMIZATION

In this section we present our mathematical model for learning path discovery and then we introduce the algorithm for solving our mathematical model.

After eliminating irrelevant learning objects in the first step, we generate the optimal solution from the obtained induced sub-graph as presented in Figure 4. For this purpose, we applied in [7] a greedy algorithm to obtain an optimal or pseudo-optimal learning path from a set of cliques. Greedy heuristics are computationally efficient, but they sometimes fail to find a global optimum as we explain in the following section.

4.1 Notation and limits of the Greedy heuristic

Let $Q_{n,m}$, $G_{n,m}$, $C_{n,v}$ the matrices representing the distribution of the m competencies that are prerequisite to the n items contained in the v cliques, the m competencies that are gained when the n items of the v cliques are performed, and the clique distribution of the n items. Note that the matrix $Q_{n,m}$ could be considered as a Q-Matrix [5].

Considering our example (*Example 1*):

- $N = \{A, E, T, U, L, I, K\}$,
- $M = \{0,2,3,4,5,6,7,8,9\}$,
- $V = \{v_1, v_2, v_3\}$.

$$Q_{n=7,m=9} = \begin{pmatrix} & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ A & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ T & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ U & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ K & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{pmatrix}$$

$$G_{n=7,m=9} = \begin{pmatrix} & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ A & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ T & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ U & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ L & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ K & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C_{n=7,v=3} = \begin{pmatrix} & \mathrm{v}_1 & \mathrm{v}_2 & \mathrm{v}_3 \\ \mathrm{A} & \mathbf{1} & 0 & 0 \\ \mathrm{E} & 1 & 0 & 0 \\ \mathrm{T} & 0 & 1 & 0 \\ \mathrm{U} & 0 & \mathbf{1} & 0 \\ \mathrm{L} & 0 & 0 & 1 \\ \mathrm{I} & 0 & 0 & 1 \\ \mathrm{K} & 0 & 0 & \mathbf{1} \end{pmatrix}$$

From this example the solution sequence using the greedy algorithm is $K \rightarrow U \rightarrow A$.

To check if we get an optimal solution or not, we have to calculate the objective function called deg. The objective function deg returns the total number of prerequisite and gained competencies of a set of learning objects.

We can draw from the previous example the following conditions to check if we have an optimal solution or not.

Let $S = \{s_0, s_1, \dots, s_v, s_{v+1}\}$ a solution set (s_i contains at least one learning object as in example 3).

$$\forall s_{i=1..v} \in S, \quad s_0 = \alpha,\ s_{v+1} = \beta,\ Q_{s_i} \subseteq G_{s_{i-1}} \quad (i)$$

$$\forall\, j = 1 \dots v \neq i = 1 \dots v,\ C_{s_i} \cap C_{s_j} = \emptyset \quad (ii)$$

$$\deg(S = \{s_0, s_1, \dots, s_v, s_{v+1}\}) = \sum_{i=0}^{v+1} \sum_{j=1}^{m} (Q_{s_i,j} + G_{s_i,j}) \quad (iii)$$

$$\forall s_{i=1..v+1} \in S;\ \exists\, s^*_{i=1..v+1} \in S^*$$

$$\deg(S^* = \{s^*_0, s^*_1, \dots, s^*_v, s^*_{v+1}\}) \leq \deg(S = \{s_0, s_1, \dots, s_v, s_{v+1}\})$$

$$(iv)$$

Condition (i) and (ii) mean that the competencies required by a clique set have to be covered by the gains of the previous clique set and two different clique sets cannot share the same clique. While condition (iii) defines the deg function, condition (iv) introduces the optimality condition. A learning path is optimal if

no other path with a lower degree exists. However this doesn't apply at the clique level since the optimal s_i^* is not necessary the set of clique i having the lowest degree. The global optimum is not the sum of the local optima calculated by the greedy algorithm.

The following example highlights this case where local optima obtained by the greedy algorithm lead to non-optimal solution.

Example 2:

	β_6	
v_1	$M^6{}_5$ $N^{6,7}{}_4$	↑ **6**
v_2	$O^5{}_{3,4}$ $P^4{}_8$	↑ **4,5**
v_3	$T^8{}_7$ $R^{3.4}{}_7$	↑ **3, 4, 8**
	α^7	↑ *7*

$$\deg(\alpha, R, O, M, \beta) = 1 + 3 + 3 + 2 + 1 = 10$$
$$\deg(\alpha, Q, P, N, \beta) = 1 + 2 + 2 + 3 + 1 = 9$$

The solution obtained by the greedy algorithm is $S_a = \alpha \to R \to O \to M \to \beta$ and the associated value of the objective function deg (S_a) is equal to 10. As the algorithm starts from β, it chooses in each clique the learning object with the lowest degree which is M and keeps going until it reaches α.

The path $S_b = \alpha \to T \to P \to N \to \beta$ is an alternative that the algorithm did not find. It's even a better alternative since $\deg(S_b) = 9 \leq \deg(S_a) = 10$ and the optimal solution.

The following example highlights another case where local optima obtained by the greedy algorithm lead to a non-optimum solution. In this example, two learning objects are selected in one of the generated cliques.

Example 3:

	β_6	
v_1	$M^6{}_5$ $N^{6,7}{}_4$	↑ **6**
v_2	$O^5{}_{3,9}$ $P^4{}_8$	↑ **4,5**
v_3	$T^8{}_7$ $Y^9{}_7$, $Z^3{}_7$	↑ **3, 9, 8**
	α^7	↑ *7*

$$\deg(\alpha, Y, Z, O, M, \beta) = 1 + 2 + 2 + 3 + 2 + 1 = 11$$

The objective function of the path $(\alpha \to T \to P \to N \to \beta)$ is 9, which means that the path $(\alpha \to T \to P \to N \to \beta)$ is the optimal solution.

In the following section, we use the notation introduced here to propose a mathematical formulation of our learning path optimization problem as an integer programming problem.

4.2 Formulating the integer programming problem

Let us consider n items or learning objects and m competencies; $Q_{n,m}$ is the matrix representing m prerequisite competencies for the n items and $G_{n,m}$ is the matrix representing the m competencies that are gained when the n items are performed. In other words, if $Q_{ij} = 1$ means that the item i has competency j as one of its prerequisite competencies; and $G_{i,j} = 1$, means that the competency j is gained when the item i is performed. The personalized learning path may then be formulated as a binary integer programming (BIP) as follows:

Minimize:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\left(Q_{i,j} + G_{i,j}\right)x_i\right) = \deg(X) \qquad (1)$$

Subject to:

$$Q_{i,j}x_i - \left(\sum_{k=1}^{i-1} G_{k,j}x_k\right) \times Q_{i,j} \leq 0 \qquad (2)$$

$$for \quad i = 2, \dots, n-1; \quad for \quad j = 1, \dots, m; \quad x_i \in \{0,1\};$$

X = $\{x_i, i=1,\dots,n\}$, are the decision variables such that:

$$x_i = \begin{cases} 1 & if\ the\ item\ i\ is\ selected; \\ 0 & otherwise, \end{cases} \qquad (3)$$

We suppose that $x_1 = 1$ and $x_n = 1$, knowing that:

$$x_1 = 1\ presenting\ the\ initial\ item\ \alpha$$
$$and\ x_n = 1\ presenting\ the\ resulting\ item\ \beta$$

The function (1) represents the total number of prerequisite and gained competencies to be minimized. The constraints (2) states that if the item i has competency j as one of its prerequisite competencies; the competency j should be gained from the items on the learning path (1,…, i-1). Our problem is to minimize the objective function (1) subject to (2) and (3).

To find the optimal learning path we have to solve the BIP problem with $(n+m)$ constraints and n decision variables $x_{i=1,\dots n} \in \{0,1\}$.

Considering example 3, the prerequisite and gain matrices Q and G can be written as follows:

The competencies that are required by the items are represented by the matrix Q (9x7).

$$Q = \begin{pmatrix} \frac{Comp.}{LO} & C1 \equiv 7 & C2 \equiv 8 & C3 \equiv 9 & C4 \equiv 3 & C5 \equiv 5 & C6 \equiv 4 & C7 \equiv 6 \\ \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ Y & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ Z & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ O & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ P & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ M & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ N & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The competencies that are gained by the items are represented by the matrix G (9x7).

$$G = \begin{pmatrix} \frac{Comp.}{LO} & C1 \equiv 7 & C2 \equiv 8 & C3 \equiv 9 & C4 \equiv 3 & C5 \equiv 5 & C6 \equiv 4 & C7 \equiv 6 \\ \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ T & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ Y & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ Z & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ O & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ P & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ M & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ N & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The BIP formulation of example 3 is given as follows:

Minimize :

$$\deg(X) = x_1+2x_2+2x_3+2x_4+3x_5+2x_6+2x_7+3x_8+x_9$$

Subject to:

$$x_2 - x_1 \leq 0$$
$$x_3 - x_1 \leq 0$$
$$x_4 - x_1 \leq 0$$
$$x_5 - x_3 \leq 0$$
$$x_5 - x_4 \leq 0$$
$$x_6 - x_2 \leq 0$$
$$x_7 - x_5 \leq 0$$
$$x_8 - x_6 \leq 0$$
$$x_9 - x_7 - x_8 \leq 0$$
$$x_i \in \{0,1\}, i = 1, \dots, 9$$

x_1 *is* the fictitious learning object α with initial learner competency state.

x_9 is the fictitious learning object β with targeted learner competency state.

Since $x_1 = x_9 = 1$, then our BIP becomes:

Minimize :

$$\deg(X) = 2x_2+2x_3+2x_4+3x_5+2x_6+2x_7+3x_8$$

Subject to:

$$x_5 - x_3 \leq 0$$
$$x_5 - x_4 \leq 0$$
$$x_6 - x_2 \leq 0$$
$$x_7 - x_5 \leq 0$$
$$x_8 - x_6 \leq 0$$
$$-x_7 - x_8 \leq -1$$
$$x_i \in \{0,1\}, i = 2, \dots, 8$$

4.3 The Branch-and-Bound (B&B) method for solving the BIP problem

Since the BIP problem is bounded, it has only a finite number of feasible solutions. It is then natural to consider using an enumeration procedure to find an optimal solution. However, in the case of large learning object repositories (millions of items), an enumeration procedure might be ill adapted (even after reducing the solution space); therefore, it is imperative to cleverly structure the enumeration procedure so that only a tiny fraction of feasible solutions need to be explored.

A well-known approach called branch-and-bound technique (B&B) provides such a procedure. B&B traces back to the 1960s' and the work of Land and Doig [14]. Since then, B&B algorithms have been applied with success to a variety of operations research problems. B&B is a divide and conquer method. It divides a large problem into a few smaller ones (This is the "Branch" part). The conquering part estimates the goodness of the solution that is obtained from each of the sub-problems; the problem is divided until solvable sub-problems are obtained (this is the "bound" part).

For the bounding part we use a linear programming relaxation to estimate the optimal solution [26]. For an integer programming model P; the linear programming model obtained by dropping the requirement that "*all variables must be integers*" is called the linear programming relaxation of P.

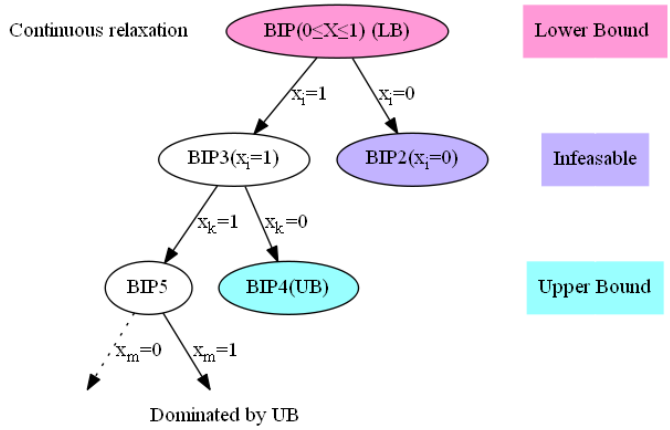


Figure 5. Branch and bound algorithm that traverses the tree by solving BIPs at every node of the tree.

The general approach of a BIP B&B algorithm [26] is presented in the following steps (see also Figure 5):

Initialization: Set deg* = $+\infty$.

The initial step represents the root node of the B&B search tree. The root node corresponds to the continuous relaxation of the BIP($0 \leq X \leq 1$), the solution value provides lower bound.

Apply the bounding step, fathoming step, and optimality test described below. If not fathomed, classify this problem as the one remaining "subproblems" for performing the first full iteration below.

Steps for each iteration:

1. Branching: Among the remaining (unfathomed) subproblems, select the one that was created most recently (break ties by selecting the subproblem with the larger bound). Branch from the node for this subproblem to create two new subproblems by fixing the next variable (the branching variable) at either 0 or 1 (see Figure 5).
2. Bounding For each new subproblem, obtain its bound by applying the simplex method to its LP-relaxation and rounding down the value of deg for the resulting optimal solution.
3. Fathoming (Pruning rules): The pruning rules for B&B BIP are based on optimality and feasibility of BIP. For

each new sub-problem, apply the fathoming tests and discard those sub-problems that are fathomed by any of the tests.

Optimality test: Stop when there are no remaining sub-problems:

- The current incumbent is optimal,
- Otherwise, return to perform another iteration.

A sub-problem is fathomed (dismissed from further consideration) if it verifies one of the following tests:

1. The relaxation of the sub-problem has an optimal solution with deg < deg∗ where deg* is the current best solution (The solution is dominated by upper bound);
2. The relaxation of the sub-problem (LP-relaxation) has no feasible solution;
3. The relaxation of the sub-problem has an optimal solution that has all binary values. (If this solution is better than the incumbent, it becomes the new incumbent, and test1 is reapplied to all unfathomed sub-problems with the new larger deg*).

For example, the *example 3* solved by B&B produces an optimal solution with deg* = 9 and x_2=1, x_6=1, x_8=1 where the number of nodes explored is 1 because the first LP-relaxation at node 1 gives an integer optimal solution with deg*=9 and the 3rd fathomed test is true, so we do not need to branch anymore.

Decision Variables	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
LO	α	T	Y	Z	O	P	M	N	β
X*	1	1	0	0	0	1	0	1	1

Figure 6. Solution of example 3in the B&B algorithm.

As illustrated in Figure 6, the optimal solution of the B&B algorithm is X*={1, 1, 0, 0, 0, 1, 0, 1, 1} and the optimal path is: $\alpha \rightarrow T \rightarrow P \rightarrow N \rightarrow \beta$.

5. CONCLUSION

The clique based approach is an asset since it offers an efficient way to reduce the solution space and check the existence of a solution. However, a greedy search within the cliques to find a leaning path does not lead, in many cases, to the best learning path according to the criteria considered.

Binary integer programming is a well-known mathematical optimization approach. While reformulating the conditions an optimal learning path should meet, we realised how we could benefit from expressing the constraints as a binary programming problem.

Our preliminary implementation of the proposed optimization using the *bintprog* function (*MATLAB*), a function based on the branch- and-bound (B&B) algorithm, shows the accuracy of the proposed integer program model.

In future work, we will apply the proposed binary integer model in order to build a learning design recommendation system in the case where learning objects are stored in very large repositories. Even though the B&B algorithm is highly accurate and somehow computationally efficient, it is not efficient enough to deal with very large size problem instances. In some cases, the bounding step of B&B is not invoked, and the branch and bound algorithm can then generate a huge number of sub-problems.

Moreover, as mentioned in [7], the efficiency of reducing the solution space with the cliques' mechanism is highly dependent on the dataset topology (average number of gain and prerequisite competencies per learning object). The solution space may remain large after the reduction

Therefore, to deal with very large problems, we will implement a variant of the B&B algorithm such as Branch & Cut [2] or Branch & Price [4]. Applegate et al. [2] showed how Branch & Cut could get a global optimal for extremely large binary optimization problems. It will be then interesting to measure both in terms of computational time and accuracy how the greedy search compares to the B&B-like approach.

6. ACKNOWLEDGMENTS

This work is part of the National Research Council of Canada's Learning and Performance Support Systems (NRC LPSS) program. The LPSS program addresses training, development and performance support in all industry sectors, including education, oil and gas, policing, military and medical devices.

7. REFERENCES

[1] Alian, M. Jabri, R. 2009. A shortest adaptive learning path in e-learning systems: Mathematical view, *Journal of American Science* 5(6) (2009) 32-42.

[2] Applegate, D., Bixby, R., Chvatal, V. and Cook, W. 1998. On The solution of traveling salesman problems, *in: Proc. Int. Congress of Mathematicians, Doc. Math. J. DMV*, Vol. 645.

[3] Atif, Y., Benlarmi, R., and Berri, J. 2003. Learning Objects Based Framework for Self-Adaptive Learning, *Education and Information Technologies, IFIP Journal, Kluwer Academic Publishers* 8(4) (2003) 345-368.

[4] Bamhart, C, Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. 1998. Branch-and-price: column generation for huge integer programs, *Operations Research* 46:316.

[5] Barnes, T. 2005. The Q-matrix Method: Mining Student Response Data for Knowledge. *Proceedings of the Workshop on Educational Data Mining at the Annual Meeting of the American Association for Artificial Intelligence*.

[6] Carchiolo, V., Longheu, A., and Malgeri, M. 2010. Reliable peers and useful resources: Searching for the best personalised learning path in a trust- and recommendation-aware environment, *Information Sciences* 180(10) (2010) 1893-1907.

[7] Durand, G., Belacel, N., and Laplante, F. 2013. Graph theory based model for learning path recommendation. *Information Sciences.* 251(10) (2013) 10-21.

[8] Durand, G., Laplante, F. and Kop, R. 2011. A learning Design Recommendation System Based On Markov Decision Processes, Proc. *17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD) Workshop on Knowledge Discovery in Educational Data*, San Diego, CA.

[9] Durand, G., Downes, S. 2009. Toward Simple Learning Design 2.0. *In: 4th Int. Conf. on Computer Science &Education 2009*, Nanning, China, 894-897.

[10] Godoy, D., Amandi, A. 2010. Link Recommendation in E-learning Systems based on Content-based Student Profiles, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining*

and Knowledge Discovery Series, Chapman & Hall/CRC Press, 273-286.

[11] Huang, Y.M., Chen, J.N., Huang, T.C., Jeng, Y.L., and Kuo, Y.H. 2008. Standardized course generation process using Dynamic Fuzzy Petri Nets, *Expert Systems with Applications*, 34 (2008) 72-86.

[12] ISO 24763/final version: Conceptual Reference Model for Competencies and Related Objects, 2011.

[13] Karampiperis, P., Sampson, D. 2005.Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society* 8 (4) (2005) 128-147.

[14] Land, A. H., Doig, A. G. 1960. An automatic method of solving discrete programming problems. *Econometrica* 28(3), 497–520.

[15] Liu, J., Greer J. 2004. Individualized Selection of Learning Object, *In: Workshop on Applications of Semantic Web Technologies for e-Learning*, Maceió, Brazil.

[16] Pavlik, P. I. Jr., Presson, N., and Koedinger K. R. 2007. Optimizing knowledge component learning using a dynamic structural model of practice, *Proc. 8th International Conference on Cognitive Modeling.* Ann Arbor, MI.

[17] Sicilia, M.-A., Sánchez-Alonso, S. and García-Barriocanal, E. 2006. On supporting the process of learning design through planners, Proc. *Virtual Campus Post-Selected and Extended*, 81–89.

[18] Tang, T.Y., Mccalla, G.G. 2010. Data Mining for Contextual Educational Recommendation and Evaluation Strategies, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 18,257-271.

[19] Trcka, N., Pechenizkiy, M. and Van-Deraalst, W. 2010. Process Mining from Educational Data, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 9, 123-141.

[20] Ullrich, C., Melis, E. 2010. Complex Course Generation Adapted to Pedagogical Scenarios and its Evaluation, *Educational Technology & Society*, 13 (2) (2010) 102–115.

[21] Ullrich, C., Melis, E. 2009. Pedagogically founded courseware generation based on HTN-planning, *Expert Systems with Applications* 36(5) (2009) 9319-9332.

[22] Ullrich C. 2005. Course Generation Based on HTN Planning, Proc. *13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, Saarbrucken, Germany,74-79.

[23] Vassileva, J., Deters, R. 1998, Dynamic courseware generation on the www, *British Journal of Educational Technology*, 29(1) (1998) 5–14.

[24] Viet, A., Si, D.H. 2006. ACGs: Adaptive Course Generation System - An efficient approach to Build E-learning, *Proc. 6th IEEE International Conference on Computer and Information Technology*, Jeju Island, Korea, 259-265.

[25] Wiley, D.A. 2002. Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy, *In: The Instructional Use of Learning Objects, D. A. WILEY (Ed.)*, 3-23.

[26] Winston, W.L., Venkataramanan, M. 2003. *Operations Research: Introduction to Mathematical Programming*. Thompson, 4th Edition.

[27] Zhao, C., Wan, L. 2006. A Shortest Learning Path Selection Algorithm in E-learning, *Proc. 6th IEEE International Conference on Advanced Learning Technologies*, Kerkrade, The Netherlands, 94-95.

[28] Zhou, M., Xu, Y., Nesbit, J.C. and Winne, P.H. 2010. Sequential pattern analysis of learning logs: Methodology and applications, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 8, 107-120.

On-Line Plan Recognition in Exploratory Learning Environments

Reuth Dekel and Ya'akov (Kobi) Gal
Dept. of Information Systems Engineering
Ben-Gurion University
Beer-Sheva 84105, Israel

ABSTRACT

Exploratory Learning Environments (ELE) are open-ended and flexible software, supporting interaction styles by students that include exogenous actions and trial-and-error. ELEs provide a rich educational environment for students and are becoming increasingly prevalent in schools and colleges, but challenge conventional plan recognition algorithms for inferring students' activities with the software. This paper presents a new algorithm for recognizing students' activities in ELEs that works on-line during the student's interaction with the software. Our approach, called CRADLE, reduces the amount of explanations that is maintained by the plan recognition in a way that is informed by how people execute plans. We provide an extensive empirical analysis of our approach using an ELE for chemistry education that is used in hundreds of colleges worldwide. Our empirical results show that CRADLE was able to output plans exponentially more quickly than the state-of-the-art without compromising correctness. This result was confirmed in a user study that included a domain expert who preferred the plans outputted by CRADLE to those outputted by the state-of-the-art approach for the majority of the logs presented.

1. INTRODUCTION

This paper focuses on inferring students' activities in educational environments in which students engage widely in exploratory behavior, and present new approaches for plan recognition in such settings that can outperform the state-of-the-art.

Our empirical analysis is based on students' interactions with an Exploratory Learning Environment (ELE) in which students build scientific models and examine properties of the models by running them and analyzing the results[1, 6]. Such software is open-ended and flexible and is generally used in classes too large for teachers to monitor all students and provide assistance when needed. The open-ended nature of ELEs affords a rich spectrum of interaction for students: they can solve problems in many different ways, engage in exploratory activities involving trial-and-error, they can repeat activities indefinitely, and they can interleave between activities.

These aspects significantly hinder the possibilities of making sense of students' activities without some sort of support. This paper presents a new algorithm for recognizing students' interactions with ELEs in real time, which can support both teachers and students. For teachers, this support takes the form of visualizing students' activities during their interaction in a way that faciliteates their understanding of students' learning. For students, this support can take the form of machine generated intervention that guides their learning and adapts to individual students' needs based on their inferred behavior.

The focus of this paper is on-line recognition that occurs during the students' actual interaction with the ELE, and outputs a hierarchy of interdependent activities that best describe the student's work at a given point in time. Recognizing students' activities this way is challenging because the algorithm needs to reason about and maintain possible explanations for future (yet unseen) student activities. The number of possible explanations grows exponentially with the number of observations. As we show in the empirical section of this paper, this significantly hinders the performance of the state-of-the-art, even for very short interaction sequences.

Our algorithm, called CRADLE (Cumulative Recognition of Activities and Decreasing Load of Explanations) builds on an existing approach for on-line plan recognition, but filters the space of possible explanations in a way that reflects the style of students' interactions in ELEs. The filtering aim is to produce complete, parsimonious and coherent explanations of students' interactions that can be easily understood by teachers and education researchers.

Our empirical evaluations were based on comparing CRADLE to the state-of-the-art approach for recognizing logs of students' interactions with a widely used ELE for chemistry education. We evaluated both of the approahes in terms of computation speed and correctness of the outputted explanation, as determined by a domain expert. Succeeding in both of these measures is criticial for an on-line plan recognition approach to work successfully.

Our empirical results show that CRADLE was able to outperform the state-of-the-art without compromising correctness. Specifically, although the state of the art approach is (in theory) complete, it was not able to terminate within an allocated time frame on many logs. In contrast, CRADLE was able to produce correct explanations for such logs. In addition, CRADLE significantly outperformed the state-of-the-art both in terms of correctness and speed of recognition.

These results demonstrate the benefit of applying novel plan recog-

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

nition technologies towards intelligent analysis of students' interactions in open-ended and flexible software. Such technologies can potentially support teachers in their understanding of student behavior as well as students in their problem solving, and lead to advances in automatic recognition in other exploratory domains.

2. RELATED WORK

Our work relates to two strands of research, inferring students' activities in educational software, and on-line planning algorithms in artificial intelligence. We relate to each of these in turn.

2.1 Inferring Students' Activities in ELEs and ITS systems

We first describe works that infer students' plans from their interactions with pedagogical software that assume the complete interaction sequence is known in advance. Gal et al. [11] and Reddy et al. [10] used plan recognition to infer students' plans from their interactions with TinkerPlots, an exploratory learning environment for statistics. Both of these approaches take as input a complete interaction sequence of a student as well as recipes for ideal solutions to TinkerPlots problems, and infer the plan used by the student retrospectively. Reddy et al. [10] proposed a complete algorithm which modeled the plan recognition task as a Constraint Satisfaction Problem (CSP). The complexity of the CSP algorithm is exponential in the size of both the interaction sequence and the data set containing the recipes. This approach requires that all possible plans can be explicitly represented, and therefore does not support recursive grammars which are needed to understand students' activities in VirtualLabs.

Other works have implemented plan recognition techniques to model students' activities in Intelligent Tutoring Systems (ITS) during their interactions. In contrast to exploratory learning environments, in intelligent tutoring systems the system takes an active role in students' interactions, as it tutors the student by providing feedback and hints. As an example, in the Andes physics tutor wrong steps are marked by the tutor and the students may ask for a "what's wrong here?" hint from the tutor. In addition, students can ask for a "what next?" hint to receive instruction when uncertain about how to proceed [20]. These systems are typically more closed-ended and less exploratory than ELEs. In the Andes physics tutor a probabilistic algorithm was used to infer the solutions plan followed by the student. For each Andes problem, a solution graph representing the possible correct solutions to the problem was automatically generated and were modeled using a dynamic Bayesian network. The algorithm observes students' actions and updates the probabilities of the different possible plans. The inferred plans were used to generate hints and to update students' cognitive models.

The tutors developed by the ACT-R group for teaching LISP, geometry and algebra, performed plan recognition using a model-tracing algorithm that tracked students' solution plans [2, 9]. These tutors maintained a list of production rules that can be triggered to accomplish the goal and sub-goals for solving a problem. The algorithm infers students' plans by identifying the production rules that were triggered according to the actions students had taken. After each observed action, the algorithm commits to a production rule that it infers the student triggered to perform the action. The system constrained students to remain on "correct paths" throughout their session by providing feedback after each action taken by the student. Moreover, ambiguities regarding the production rules being used by students were resolved by querying the student. By committing to one production rule at a time and enforcing students to remain on correct solution paths, the complexity of the plan recognition task in intelligent tutoring systems is substantially reduced.

Lastly, we mention works that use recognition techniques to model students' activities in Intelligent Tutoring Systems [20, 7, 21]. Our work is distinct from works on plan recognition in intelligent tutoring systems in several ways. First, ITS are more closed-ended from ELEs. Thus, students' activities with such software more constrained and less exploratory, and are easier to model and recognize. In addition, the tutoring systems described above provided constant feedback to students which helped them remain on correct solution paths that are recognizable by the model used. Second, the tutoring systems described above explicitly modeled all possible solution plans for solving a specific problem. This is not possible in the VirtualLabs domain, as there may be an infinite number of possible plans for solving a problem.

2.2 On-line Plan Recognition in Artificial Intelligence

We now discuss general work from Artificial Intelligence that is concerned with plan recognition in general, rather than recognizing students' activities in pedagogical software. On-line plan recognition is a significantly more difficult task than its off-line variant. The fact that the interaction sequence is not observed ahead of time raises additional challenges to on-line plan recognition. Blaylock et al. [4] developed an algorithm to infer the goal of users from their actions in a Linux shell environment. Pynadath [19] proposes a probabilistic inference of plan, but requires the observations to be fully ordered. The approach by Bui [5] used particle filtering to provide approximate solutions to on-line plan recognition problems. Avrahami and Kaminka [3] presented a symbolic on-line plan recognition algorithm which keeps history of observations and commits to the set of possible plans only when it is explicitly needed for querying. Geib and Goldman presented PHATT [14], a probabilistic on-line plan recognition algorithm that builds all possible plans incrementally with each new observation. This algorithm was applied to recognizing users' strategies in real-time video games [17].

All of these works have been evaluated on simulated, synthesized problems [19, 3, 14] or on toy problems [4, 17]. These approaches do not scale to the complexities of real-world domains. An exception is the work of Conati et al. [8, 18] who used on-line plan recognition algorithms to infer students' plans to solve a problem in an educational software for teaching physics, by comparing their actions to a set of predefined possible plans. Unfortunately, the number of possible plans grow exponentially in the types of domains we consider, making it unfeasible to apply this approach.

3. PLANS AND EXPLANATIONS

In this section we provide the basic definitions that are required for formalizing the on-line plan recognition problems in ELEs. Throughout the paper we will use an existing ELE for chemical education called VirtualLabs to demonstrate our approach which is actively used by students worldwide as part of their introductory chemistry courses. VirtualLabs allows students to design and carry out their own experiments for investigating chemical processes by simulating the conditions and effects that characterize scientific inquiry in the physical laboratory [22]. We use the following problem called "Oracle", which is given to students:

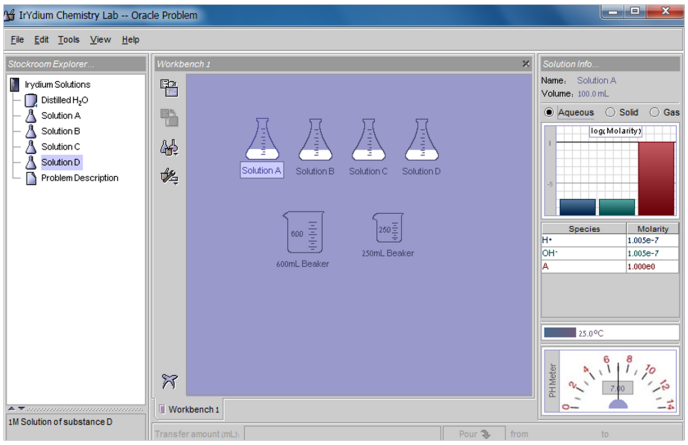


Figure 1: Snapshot of VirtualLabs

(a) $\underline{\mathsf{MSD}}[s_1 + s_2, d] \rightarrow \underline{\mathsf{MSD}}[s_1, d], \underline{\mathsf{MSD}}[s_2, d]$
(b) $\underline{\mathsf{MIF}}[s_1, d_2] \rightarrow \underline{\mathsf{MSD}}[s_1, d_1], \underline{\mathsf{MSD}}[d_1, d_2]$
(c) $\underline{\mathsf{MSD}}[s, d] \rightarrow \underline{\mathsf{MIF}}[s, d]$
(d) $\underline{\mathsf{MSD}}[s, d] \rightarrow \mathsf{MS}[s, d]$

Figure 2: Recipes for VirtualLabs

> Given four substances A, B, C, and D that react in a way that is unknown, design and perform virtual lab experiments to determine the correct reaction between these substances.

The flexibility of VirtualLabs affords two classes of solution strategies to this problem (and many variations within each). In the first strategy, a student mixes all four solutions together, and infers the reactants by inspecting the resulting solution. In the second strategy, a student mixes pairs of solutions until a reaction is obtained. A snapshot of a student's interaction with VirtualLabs when solving the Oracle problem is shown in Figure 1.

3.1 Definitions

We make the following definitions taken from the classical planning literature [16]. We use the term *basic actions* to define rudimentary operations that cannot be decomposed. These serve as the input to our plan recognition algorithm. For example, the basic "Mix Solution" action ($\mathsf{MS}_1[s = 1, d = 3]$) describes a pour from flask ID 1 to flask ID 3. A *log* is the output of a student's interaction. It is a sequence of basic level actions representing students' activities'. This is also the input to the plan recognition algorithm described in the next section.

Complex actions describe higher-level, more abstract activities that can be decomposed into sub-actions, which can be basic actions or complex actions themselves. For example, the complex action $\underline{\mathsf{MSD}}[s = 1 + 5, d = 3]$ (as shown in Figure 3) represents separate pours from flask ID 1 and 5 to flask ID 3.

A *recipe* for a complex action specifies the sequence of actions required for fulfilling the complex action. Figure 2 presents a set of basic recipes for VirtualLabs . In our notation, complex actions are underlined, while basic actions are not. Actions are associated with parameters that bind to recipe parameters. Recipe (a) in the figure, called Mix to Same Destination ($\underline{\mathsf{MSD}}$), represents the activity of pouring from two source flasks (s_1 and s_2) to the same destination flask (d). Recipe (b), called Mix via Intermediate Flask ($\underline{\mathsf{MIF}}$), represents the activity of pouring from one source flask (s_1) to a destination flask (d_2) via an intermediate flask (d_1).

Recipes can be recursive, capturing activities that students can repeat indefinitely. Indeed, this is a main characteristic of students' use of ELEs. For example, the constituent actions of the complex action $\underline{\mathsf{MSD}}$ in recipe (a) decompose into two separate $\underline{\mathsf{MSD}}$ actions. In turn each of these actions can itself represent a Mix to Same-Destination action, an intermediate-flask pour (by applying recipe (c)) or a basic action mix which is the base-case recipe for the recursion (recipe (d)). Recipe parameters also specify the type and volume of the chemicals in the mix, as well as temporal constraints between constituents, which we omit for brevity.

More generally, the four basic recipes in the figure can be permuted to create new recipes, by replacing $\underline{\mathsf{MSD}}$ on the right side of the first two recipes with $\underline{\mathsf{MIF}}$ or MS. An example of a derivation is the following recipe for creating an intermediate flask out of a complex Mix to Same Destination action and basic Mix Solution action.

$$\underline{\mathsf{MIF}}[s_1, d_2] \rightarrow \mathsf{MSD}[s_1, d_1], \mathsf{MS}[d_1, d_2] \quad (1)$$

These recipes may be combined to describe the different solution strategies by which students solve problems in VirtualLabs (e.g., capturing students mixing all possible substance pairs versus mixing all four pairs together).

A set of nodes N *fulfills* a recipe R if there exists a one-to-one matching between the constituent actions in R and their parameters to nodes in N. For example, the nodes $\mathsf{MS}_3[s = 5, d = 4]$ and $\mathsf{MS}_5[s = 4, d = 3]$ fulfill the Mixing via an Intermediate Flask recipe shown in Equation 1.

3.2 Planning

Planning is the process by which students use recipes to compose basic and complex actions towards completing tasks using VirtualLabs . Formally, a *plan* is an ordered set of basic and complex actions, such that each complex action is decomposed into sub-actions that fulfill a recipe for the complex action. Each time a recipe for a complex action is fulfilled in a plan, there is an edge from the complex action to its sub-actions, representing the recipe constituents.

Figure 3 shows part of a plan describing part of a student's interaction when solving the Oracle problem. The leaves of the trees are the actions from the student's log, and are labeled by their order of appearance in the log. For example, the node labeled with the complex action $\underline{\mathsf{MSD}}[s = 1 + 5, d = 3]$ includes the activities for pouring two solutions from flask ID 1 and ID 5 to flask ID 3. The pour from flask ID 5 to 3 is an intermediate flask pour ($\underline{\mathsf{MIF}}[s = 5, d = 3]$) from flask ID 5 to ID 3 via flask ID 4. The root of the plan represents te complex action of pouring three substances from flasks ID 1, 5 and 6 to flask ID 3.

In a plan, the constituent sub-actions of complex actions may interleave with other actions. This way, the plan combines the free-order nature of VirtualLabs recipes with the exploratory nature of students' learning strategies. Formally, we say that two ordered complex actions *interleave* if at least one of the sub-actions of the first action occurs after some sub-action of the second action. For

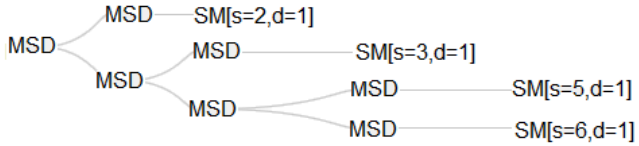


Figure 4: Example of an Explanation containing a Single Plan

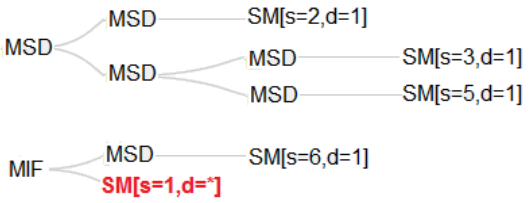


Figure 5: Example of an Explanation containing Two Plans, one of which has an open frontier

example, the nodes $\mathsf{MS}_3[s=5,d=4]$ and $\mathsf{MS}_5[s=4,d=3]$ and $\mathsf{MS}_2[s=6,d=8]$ and $\mathsf{MS}_4[s=8,d=3]$ both fulfill the Mixing via an Intermediate Flask recipe shown in Equation 1, but they are interleaved in the log. This interleaving quality makes the plan recognition task even more challenging.

4. ONLINE PLAN RECOGNITION

In this section we address the problem of on-line recognition in which agents' plans need to be inferred in real-time during execution. On-line recognition is essential for settings in which it is necessary to generate interventions to users. In ELEs, such intervention can provide feedback to students about their progress, alerting them to recurring mistakes or giving them hints about next steps during their exploration.

The fact that the interaction sequence is not known in advance requires to maintain the set of all plans that can explain the observations, including leaving place-holders for actions in the plan that relate to unseen future activities. Following Geib and Goldman [12], we define an *explanation* of actions O at time t a set of plans, such that there is an injective mapping from each action in O to a leaf in one of the plan instances. Each plan in an explanation describes a non-overlapping subset of the actions O. Some leaves in an explanation may not be included in O, and describe actions that are expected to appear in the future. These leaves are called the *open frontier* of the plan.

To illustrate, consider the recipes for VirtualLabs and the following explanations: Figure 4 shows a possible explanation for the observation sequence $SM[s=2,d=1], SM[s=3,d=1], SM[s=5,d=1], SM[s=6,d=1]$ in which all of the actions are constituents of the complex action MSD.[1] The explanation consists of a single plan.

Figure 5 shows a possible explanation for the same observation sequence, but in this case, the explanation consists of two plans. Here, the bold action $SM[s=1,d=*]$ represents a future (unseen) observation and is in the plan frontier. If the fifth observation turns out to be an SM action with $s=1$ (the parameter d does not hold any constraints), then the algorithm will incrementally combine this observation into the explanation. Otherwise, a third plan instance will be added to the explanation that matches the new observation, leaving $SM[s=1,d=*]$ in (and possibly adding new actions to) the plan frontier. We note that the plan frontier may also include complex actions, allowing to reason about future higher-level activities for which none of the constituents have been observed. The fact that the algorithm needs to maintain explanations for unseen observations is a significant computational challenge, as the possible number of explanations grows exponentially with the number of observations.

[1]For expository purposes we have omitted the parameters from nodes above the leaves.

5. CRADLE AND PHATT

The purpose of this section is to describe the state-of-the art in online plan recognition approach called PHATT, and our proposed extension to this approach for recognizing students' activities in ELEs.

We define the on-line plan recognition as follows: Given a set of observation at time t, output a set of explanations such that each explanation in the set can be used to derive the observations. PHATT is a top-down probabilistic algorithm that incrementally builds the set of possible explanations for explaining an observation sequence. PHATT works as follows: For each observation o^{t+1}, it takes the set of the possible explanations for the previous observations O^t, and tries to incorporate the new observation into each of the explanations in the set. This can be done either by integrating the new observation into one of the existing plans of the explanation, or by adding the observation as the first step of a new plan that will be added to the forest of plans in the explanation.

5.1 Using Filters

We now describe the basis for our proposed extension to PHATT, which is constraining the space of possible explanations in a way that reflect students' use of educational software. Our approach is called CRADLE (Cumulative Recognition of Activities and Decreasing Load of Explanations).[2]

Cradle extents the PHATT algorithm by constraining the space of possible explanations. We designed several "filters" that reduce the size of the explanation set in a way that reflects the intended use of plan recognition in ELEs. Specifically, the filters aim to produce complete, parsimonious and coherent explanations of students' interactions that can be easily understood by teachers and education researchers. We detail these filters below:

Explanation size This filter prefers explanations with smaller number of plans. Specifically, we discard explanations in which the number of plans is larger than a pre-computed threshold (the average number of plans per explanation).

Aging This filter prefers explanations in which successive observations extend existing sub-plans in the explanation rather than generate new plans. We discard explanations in which observations have not extended an existing plan for a given number of iterations.

[2]Also, cradle is the name of the mechanical contrivance used in placer mining, consisting of a box on rockers and moved by hand, used for washing out the gold-bearing soil, leaving only nuggets of gold.

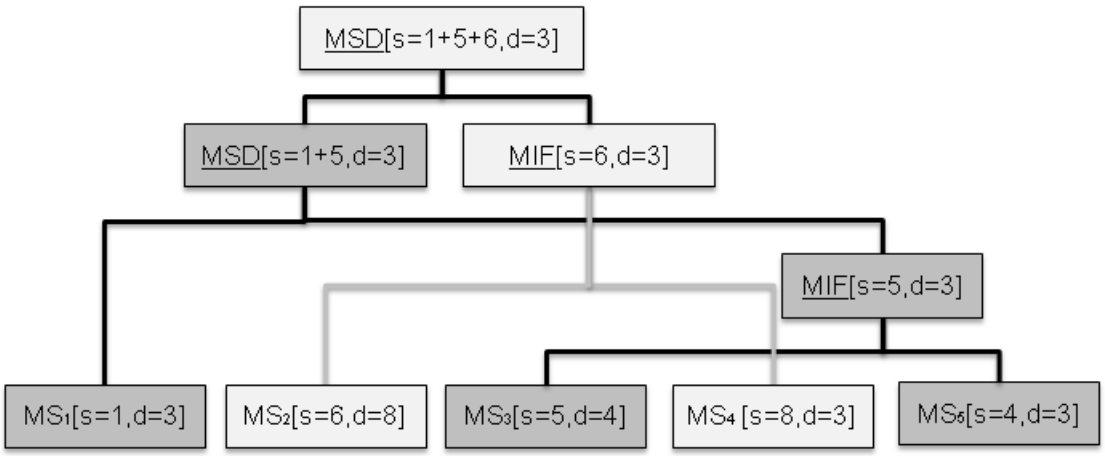


Figure 3: A partial plan for a student's log

Frontier size This filter prefers explanations which makes fewer commitments about future observations. It measure the amount of actions in the frontier that exist in each explanation, and discard explanations where this amount is above the average.

Probability This filter prefers explanations with a higher likelihood. It discards explanations whose probability of generating the observation sequence is lower than the average probability of the other explanations.

5.2 Augmenting PHATT

Figure 6 describes how CRADLE extends PHATT using the following methods, which we outline in some level of abstraction.

- Expand. Given a set of explanations that derive O^t, it is given a new observation o^{t+1}, this method creates all possible subplans in which o^{t+1} is a leaf, and tries to combine each of these subplans in all possible ways to each explanation. Each such subplan can be combined in two ways: (1) combineInExistingTrees - if the root of the subplan matches one of the plan frontier items, it replaces the frontier item with the subplan (replacing the placeholder with a concrete observation) or (2) extendWithANewTree - if the root of the subplan matches a possible goal, it is adding the subplan as the top levels of a new plan in the explanation's forest of plans.
- Filter. This function takes a set of explanations, calculates the average age, frontier size and amount of trees per explanation and filtered away all explanations with values above average. This means it prefers explanations with small frontier (less future expected observations), small age (observations continue existing plans instead of creating new ones) and small amount of trees (observations related to the same plan rather than describe different plans).
- Main. This is the main function of the new recognition process. It is made out of the two previous described stages - Extend and Filter - performed alternatly for each new observation encountered.

6. EMPIRICAL METHODOLOGY

The purpose of this section is to evaluate CRADLE to PHATT algorithm for real-world data sets of students' interactions with VirtualLabs . The PHATT approach is representative of an array of algorithms in the literature for performing on-line plan recognition by maintaining sets of observations (see for example the ELEXIR and YAPPR algorithm [13, 15]) and would behave similarly on our ELE data sets.

Specifically, we sampled 16 logs of students' interactions who solved two problems. The first was the Oracle problem described earlier. The second problem was called "Unknown Acid" and required students to determine the concentration level of an unknown solution. The length of the logs were chosen to have a wide range, between 4 to 152 actions.

6.1 Completeness and Run-time

The number of explanations maintained by the PHATT approach grows exponentially in the number of observations. It can be shown that for n observations and a set g of possible extensions for an explanation, the number of possible explanations is bounded by $n * |g|^n$. To illustrate, a 4 observation log outputted 142 different explanations, and a log of 12 observations generated more than 10,000 explanations. Most of these explanations included an abundance of plan instances with extremely large frontiers, clearly not the most coherent descriptions of the students' work.

Figure 7 shows the performance obtained using PHATT, augmentation of PHATT with single filter, and CRADLE. The x-axis in the figure corresponds to ranges of different log sizes. The y-axis determines the success ratio by measuring whether the algorithm was able to terminate and produce the explanations describing the student's activities within an upper bound of two hours of CPU time. As shown by the figure, PHATT was not able to terminate on logs

```
1: function EXPAND(o, Exps)  ▷ o: a new observation, Exps is the set of all explanations until o
2:     newExps = []
3:     for all explanation e ∈ Exps do
4:         newExps += e.combineInExistingTrees(o)
5:         newExps += e.extendWithANewTree(o)
6:     end for
       return newExps
7: end function

8: function FILTER(Exps)  ▷ Exps is the set of all explanations collected so far
9:     filteredExps = []
10:    for all explanation e ∈ Exps do
11:        if e.age ≤ averageAge & e.frontierSize ≤ averageFrontierSize & e.trees ≤ averageTrees then
12:            filteredExps += e
13:        end if
14:    end for
       return filteredExps
15: end function

16: function MAIN(Obs)  ▷ Obs is the set of all observations
17:    tempExps = [⟨emptyExp⟩]  ▷ Only one explanation - the empty explanation
18:    for all observation o in Obs do
19:        allExps = Expand(o, tempExps)
20:        filteredExps = Filter(allExps)
21:        tempExps = filteredExps
22:    end forreturn tempExps
23: end function
```

Figure 6: Main functions of the CRADLE algorithm

over 4 actions within this designated time frame. In contrast, CRADLE was able to significantly increase the performance of PHATT algorithm by applying the filters. Specifically, applying the different filters independently allowed to improve the success ratio for some of the logs, with the highest improvement attributed to the CRADLE approach which applied the age, frontier size and explanation size filters. Interestingly, there was not a single filter method that outperformed all of the other methods for all log size.

Next, we compare the run-time of CRADLE and PHATT on fragments of logs for which PHATT was able to terminate. Figure 8 shows the average run-time on each size of log, measured in seconds, presented in a logarithmic scale. It can be seen that the average run-time of CRADLE is exponentially better than the average run-time of PHATT for the aforementioned logs.

6.2 Domain Expert Evaluation

In this section, we show that although the CRADLE approach reduces the number of possible explanations that is maintained by the plan recognition algorithm, it does not hinder the correctness of the algorithm. To this end, we sampled 20 logs of the Oracle problem and presented the output of the PHATT and CRADLE approach to a domain expert.[3]. We ran the cut logs on PHATT and CRADLE and collected the outputted set of explanations for each log. For

[3]Logs of length greater than 6 actions were cut arbitrarily at 6,7,9,10 and 11 actions, in order to simulate incomplete interaction sequences and to allow PHATT to terminate on these logs in reasonable time.

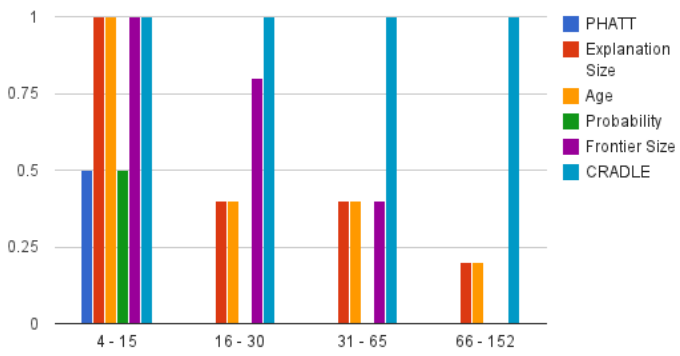


Figure 7: Performance of PHATT, CRADLE and Single Filter Variants on Various Log Sizes

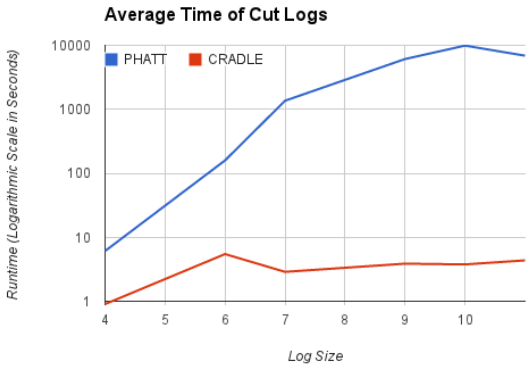


Figure 8: Runtime of PHATT and CRADLE

each of the approaches, we chose to present the domain expert with the explanation that did not include an open frontier (that is, the explanations provided a complete description of the activities of the student). If there was no explanations without an open frontier, we chose the most likely explanation as measured by its probability.

Out of the 20 examined logs, in 9 logs PHATT and CRADLE's explanations were the same (though CRADLE was able to output the solution exponentially faster). We presented the explanations for which CRADLE and PHATT different to a domain expert, who is one of the developers of the VirtualLabs software, who compared between the two explanations. We did not label the explanations with the algorithm that generated them. In 8 out of these 11 logs, the domain expert preferred explanations which were presented by CRADLE over the explanations of PHATT. In one case, the domain expert said none of the explanations describe the activities of the student correctly. To illustrate, Figure 4 shows the explanation outputted by CRADLE for a particular log which included a mix of 4 substances into a single flas. Figure 9 shows the PHATT explanation for that same log, using two plans to explains the observation sequence. In this case, the domain expert preferred the CRADLE explanation, which explained the observation sequence using a single plan.

7. DISCUSSION AND FUTURE WORK

Our results show that the CRADLE approach was able to extend the state-of-the-art (PHATT algorithm) towards successfully rec-

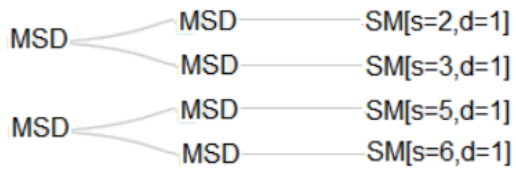


Figure 9: Example of PHATT explanation

ognizing students' activities in an ELE for chemistry education. We showed that CRADLE was able to produce better explanations than PHATT, and with exponentially faster running time. Specifically, the ouputted explanations of CRADLE were as good as or better than PHATT in 18 out of the 20 logs that we sampled, giving CRADLE a success rate of 90% at an exponentially lower runtime. The paper demonstrate that on-line plan recognition in ELEs is a challening computational problem, and show the efficacy of the CRADLE approach in addresssing these problems by reducing the number of explanations maintaind by the algorithms in an intelligent way. We are currently pursuing work with CRADLE in several directions. First, we are evaluating the scalability of the CRADLE approach by evaluating it with different ELEs for statistics education, as well as simulated data that simulates users' interactions with software. This ELE is significantly different than VirtualLabs in that student's interactions are more likely to engage in trial-and-error, which we predict will further challenge the recognition problem. Second, we are developing a formal langauge that explains students' activities with ELEs that will help us construct more accurate grammars for the recognition algorithms.

8. ACKNOWLEDGEMENTS

This work was supported in part by Israeli Science Foundation Grant no. 1276/12.

9. REFERENCES

[1] S. Amershi and C. Conati. Automatic recognition of learner groups in exploratory learning environments. In *Intelligent Tutoring Systems (ITS)*, 2006.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The Journal of Learning Sciences*, 4(2):167–207, 1995.

[3] D. Avrahami-Zilberbrand, G. Kaminka, and H. Zarosim. Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations. In *Proc. of the AAAI Workshop on Modeling Others from Observations, MOO*, 2005.

[4] N. Blaylock and J. F. Allen. Statistical goal parameter recognition. In *ICAPS*, volume 4, pages 297–304, 2004.

[5] H. H. Bui. A general model for online probabilistic plan recognition. In *IJCAI*, volume 3, pages 1309–1315, 2003.

[6] M. Cocea, S. Gutierrez-Santos, and G. Magoulas. S.: The challenge of intelligent support in exploratory learning environments: A study of the scenarios. In *Proceedings of the 1st International Workshop in Intelligent Support for Exploratory Environments on European Conference on Technology Enhanced Learning*, 2008.

[7] 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.

[8] 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.

[9] A. Corebette, M. McLaughlin, and K. C. Scarpinatto. Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interaction*, 10:81—108, 2000.

[10] Y. Gal, S. Reddy, S. Shieber, A. Rubin, and B. Grosz. Plan recognition in exploratory domains. *Artificial Intelligence*, 176(1):2270 – 2290, 2012.

[11] Y. Gal, E. Yamangil, A. Rubin, S. M. Shieber, and B. J. Grosz. Towards collaborative intelligent tutors: Automated recognition of users' strategies. In *Intelligent Tutoring Systems (ITS)*, 2008.

[12] C. Geib and R. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[13] C. W. Geib. Delaying commitment in plan recognition using combinatory categorial grammars. In *IJCAI*, pages 1702–1707, 2009.

[14] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[15] C. W. Geib, J. Maraist, and R. P. Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In *ICAPS*, pages 91–98, 2008.

[16] B. Grosz and S. Kraus. The evolution of sharedplans. *Foundations and Theories of Rational Agency*, pages 227–262, 1999.

[17] F. Kabanza, P. Bellefeuille, F. Bisson, A. R. Benaskeur, and H. Irandoust. Opponent behaviour recognition for real-time strategy games. In *Plan, Activity, and Intent Recognition*, 2010.

[18] S. Katz, J. Connelly, and C. Wilson. Out of the lab and into the classroom: An evaluation of reflective dialogue in andes. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 158:425, 2007.

[19] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 507–514. Morgan Kaufmann Publishers Inc., 2000.

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

[21] M. Vee, B. Meyer, and K. Mannock. Understanding novice errors and error paths in object-oriented programming through log analysis. In *Proceedings of Workshop on Educational Data Mining at ITS*, pages 13–20, 2006.

[22] D. Yaron, M. Karabinos, D. Lange, J. Greeno, and G. Leinhardt. The ChemCollective–Virtual Labs for Introductory Chemistry Courses. *Science*, 328(5978):584, 2010.

What is the Source of Social Capital?

The Association Between Social Network Position and Social Presence in Communities of Inquiry

Vitomir Kovanovic*
School of Interactive Arts and Technology
Simon Fraser University
250 - 13450 102nd Avenue
Surrey, BC, V3T0A3 Canada
vitomir_kovanovic@sfu.ca

Srecko Joksimovic
School of Interactive Arts and Technology
Simon Fraser University
250 - 13450 102nd Avenue
Surrey, BC, V3T0A3 Canada
sjoksimo@sfu.ca

Dragan Gasevic
School of Computing Science
Athabasca University
1 University Drive
Athabasca, AB, T9S 3A3
Canada
dgasevic@acm.org

Marek Hatala
School of Interactive Arts and Technology
Simon Fraser University
250 - 13450 102nd Avenue
Surrey, BC, V3T0A3 Canada
mhatala@sfu.ca

ABSTRACT

It is widely accepted that the social capital of students – developed through their participation in learning communities – has a significant impact on many aspects of the students' learning outcomes, such as academic performance, persistence, retention, program satisfaction and sense of community. However, the underlying social processes that contribute to the development of social capital are not well understood. By using the well-known Community of Inquiry (CoI) model of distance and online education, we looked into the nature of the underlying social processes, and how they relate to the development of the students' social capital. The results of our study indicate that the affective, cohesive and interactive facets of social presence significantly predict the network centrality measures commonly used for measurement of social capital.

General Terms

Social Network Analysis, Community of Inquiry, Social Presence

1. INTRODUCTION

Asynchronous online discussions have been frequently used both in blended and fully online learning [41]. However, with the broader adoption of social-constructivist pedagogies and the shift towards the collaborative learning [2], they are viewed as one of the important study tools for the computer-supported collaborative learning (CSCL) within the online learning environments. Their use has produced an enormous amount of data about the interactions between students and instructors [21]. The distance education and CSCL research communities have tried to use these data for gaining insights into the very complex nature of the learning phenomena. Among the different ways of researching students' social interactions *Quantitative Content Analysis* (QCA) [38, 19] and *Social Network Analysis* (SNA) [52, 46] represent two commonly used methods.

A widely accepted model of distance education which makes a use of QCA is the *Community of Inquiry* (CoI) model [28]. According to Garrison and Arbaugh [30], it is one of the leading models of distance education that describes the key constructs of the overall educational experience. The CoI model provides the in-depth assessment of teaching, cognitive and social dimensions of learning phenomena, and how those three dimensions affect: i) the overall success of the learning process, and ii) the attainment of learning objectives [28]. Empirical research showed that the social dimension of learning plays an important role in the learning communities by mediating the relationship between the teaching and cognitive dimensions [31]. Still, the CoI model does not explicitly address the question of student social networks, their structure, or the effects they have on the overall educational experience and learning outcomes. Given the amount of evidence from the studies of student social networks [46], this warrants further investigation.

One of the central aspects in the study of social networks is the idea of the *social capital* [13, 12]. Generally speaking, social capital can be defined as a value resulting from occupying a particularly advantageous position within a social network [12]. Over the years, the study of social capital has become increasingly popular in the field of education [14]. The large number of studies in the distance education field indicated an important connection between the students' social capital and many important aspects of education and learning including academic performance [33, 15, 7, 49, 43], retention [23], persistence [50], program satisfaction [7], and sense of community [17]. Still, research of the student social networks have involved mostly isolated studies that were focused on the understanding of the relationship between a particular set of constructs selected by the researchers and the students' network position. Likewise, the underlying mechanisms responsible for the observed social structure are typically not addressed, which is understandable given the lack of educational theories that explicitly

*Corresponding Author

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

take into the consideration student social networks.

In this paper, we present the results of the study which explored the links between the CoI model and the social network analysis of student networks. With the current advancement within the CoI research and most recent validations of the model [31], the model is mature enough and empirically sound to provide this missing theoretical foundation for understanding the structure of students' social networks. Likewise, the understanding of the structure of social networks can provide a more comprehensive overview of the social dimension of learning that it is already accounted for in the research of the CoI model.

Given the exploratory nature of this study, we focused on the relationship between social capital and social processes which are indicative of the student social presence development. The main question we aim to answer, in this paper is *which social processes, and to what extent, are indicative of the development of the social capital in a communities of inquiry*? Given the detailed characterization of social aspects of learning in the CoI model through the construct of *social presence*, we explored how this construct relates to the students' social capital, as characterized by their position in social networks formed around communities of inquiry. As the community of inquiry provides characterization of different sociological processes that constitute social presence, we looked how each of them contributed to the development of social capital withing students' social network.

2. THEORETICAL BACKGROUND

2.1 Social network analysis

2.1.1 Social capital

The study of social networks has attracted much attention in social and behavioral sciences [17, 14]. The focus in social network analysis is on the study of *relationships*, also known as *ties*, between a set of *actors*, or *participants* [14]. Through the relationships, members of a network engage in sharing, exchange or delivery of various resources including information [36]. Social network analysis draws much of its ideas from the mathematical graph theory and the sociometric studies of the human relationships [52].

An important concept in the study of social networks is the idea of *relation strength* [34], which is used to make a distinction between *strong social ties*, which require a substantial commitment (e.g., family, close friends), and *weak social ties* which do not obligate a strong commitment (e.g., acquaintances). Likewise, the idea of *network brokerage* builds on the fact that in a large network, the density of relationships is not uniform, which indicates the existence of smaller sub-communities within a large social network [12, 13]. In his seminal paper, Granovetter [34] stressed the tremendous importance of weak social ties, as they provide access to novel information from different parts of a social network and provide pathways of information exchange between sub-communities. An individual who possesses a large number of weak ties in many different sub-communities is able to take advantage by combining diverse information coming from different sub-communities, and to even control to a certain degree the spread of information from one sub-community to another [12]. This ability to create a value from occupying a particular position in a social network is known as *social capital* [13]. To study and assess values of different network positions, the principles of graph theory are the most commonly used [52]. The notion of *centrality* is particularly important. This notion captures the relative importance of individuals in social networks [52]. Given the complexity of measuring actors' relative importance, a large number of centrality measures were proposed over the years out of which degree, closeness and betweenness centralities are the most frequently used [26].

2.1.2 Social network analysis in education

While social network analysis has been widely adopted in social and behavioral sciences, its adoption in the field of education was initially very limited [14]. According to Carolan [14], the main reasons for this are *"overemphasis on individual explanations of educational opportunities and outcomes, a quest for scientific legitimacy, and a preference for experimental designs that estimate the causal effects of 'educational interventions' "* [14, 32]. Nevertheless, over the years, the number of studies that indicated the importance of social connections on the overall academic experience has grown considerably. A good example is the study of students' overall academic experience from early 1990s by Astin [5] in which he concluded that: i) the environment made by the instructors and students is crucial, and ii) the single most important environmental influence is *peer group*.

In the context of distance education, there have been many studies recently that looked at the connection between several important learning constructs and social capital of students. Likewise, in the fields of educational data mining (EDM) [6] and learning analytics [40], the interest in SNA has been growing. The recent review of the EDM field by Romero and Ventura [44] noted a growing interest in SNA; likewise, in the learning analytics community, SNA was recognized as one of the most important techniques of social learning analytics [11, 25].

As expected, academic performance was the focus of a large majority of the studies [33, 50, 15, 7, 49, 43] that have found positive effects of student positions in social networks on academic performance. Still, academic performance was not the only construct that was examined. The study of retention by Eckles and Stradley [23] found that for each friend that leaves an academic degree program makes a student five times more likely to leave as well, while every friend who stays makes a student 2.25 times more likely to also stay in college. The study of student persistence and integration by Thomas [50] found that students with a broader set of acquaintances are more likely to persist in the academic program of a higher education institution, and that students with a higher proportion of ties outside their peer group also perform better academically. This is aligned with the findings of Dawson [17] who showed that students' sense of community membership was positively related to their closeness and degree centrality measures. Similarly, in the study of a team-based MBA program by Baldwin et al. [7], it was found that the high embeddedness in the friendship network increased students' perception of learning and enjoyment in the program; as well, the centrality in the communication networks was found to be positively linked with the student grades.

One important thing to notice is that the majority of the studies did not draw their theoretical foundations of network formation from the established educational theories. As pointed out by Rizzuto et al. [43], there is a lack of *"theory of academic performance that combines individual characteristics as well as social and infrastructural factors"* (p180). The main exception is the use of retention theories by Tinto [51] and Bean [8] in the study of student persistence and retention. The other notable theories that are adopted, such as Feld's theory of focused choice [24], or Lin's theory of social resources [39] are general sociological theories that do not take into the account the specific of learning processes and educational contexts.

2.2 The community of inquiry (CoI) model

2.2.1 Overview

The Community of Inquiry (CoI) model is a general model of distance education which explains the constructs that contribute to the overall learning experience. It is rooted in the social constructivist

philosophy, most notably in the work of John Dewey [20], and is particularly well suited for understanding different aspects of learning within the learning communities. The main goal of the CoI model was to define the constructs that characterize a worthwhile educational experience, and a methodology for their assessment. The CoI model consists of the three interdependent constructs, also known as *presences*, that together provide a comprehensive coverage of the distance learning phenomena:

1) **Cognitive Presence** explains different phases of students' knowledge construction process through social interactions within a learning community [28].
2) **Teaching Presence** describes the instructor's role in course delivery and during course design and preparation [3].
3) **Social Presence** explains the social relationships and the social climate within a learning community that have a significant effect on the success and quality of social learning [45].

The CoI model is well-researched and widely accepted within the distance learning research community as shown by a recent two-part special issue of The Internet and Higher Education journal [1]. The model defines its own coding schemes that are used to assess the levels of the three presences through the QCA in transcripts of asynchronous online discussions. More recently, instead of relying on the QCA, a CoI survey instrument [4] was developed as an alternative way of assessing the levels of the three presences.

2.2.2 *Social presence*

Social presence is defined as the *"ability of participants in a community of inquiry to project themselves socially and emotionally, as "real" people (i.e., their full personality), through the medium of communication being used"* [28, p3]. Critical thinking, social construction of knowledge and the development of the cognitive presence are more easily developed in the cases where the appropriate levels of social presence have been established [28].

Given the form of delivery in distance education, face-to-face communication that is typical for more traditional forms of education delivery is not possible. Hence, establishing and sustaining social presence is more challenging. Distance education was often criticized as being inferior to more traditional forms of education, particularly because of the inability to create social presence between the members of a learning community [2]. However, according to Garrison et al. [28], the form of communication is not the solely factor determining the development of social presence. A key aspect of establishing social presence in face-to-face settings are visual cues, while participants in online communities use different techniques – such as emoticons – to convey the affective dimension of communication that lacks in typical text-based communications.

As described by Rourke et al. [45], the origins of social presence can be found in the work of Mehrabian [42] and his notion of *immediacy* which is defined as *"the extent to which communication behaviors enhance closeness to and nonverbal interaction with another"* [42, p203]. This, and the set of follow-up studies by communication theorists, defined the theoretical background on which the construct of social presence was based [45]. The social presence in the CoI model is defined as consisting of three different dimension of communication:

1) **Affectivity and expression of emotions**: Since emotions are strongly associated with motivation and persistence, they are indirectly connected to critical thinking and communities of inquiry. More formally, emotional expression has been indicated by the *"ability and confidence to express feelings related to the educational experience"* [28, p99].
2) **Interactivity and open communication**: In order to promote the development of higher-order critical thinking skills, the notion that the other side is listening and attending is crucial [45]. Thus, activities such as praising of the student work, actions, or comments contribute to the teacher immediacy, which in turn leads to affective, behavioral and cognitive learning [45]. Similarly, open communication is defined as *"reciprocal and respectful exchanges of messages"* [28, p100] and together with interactivity provide a basis on which productive social learning can be established.
3) **Cohesiveness**: The activities that *"build and sustain a sense of group commitment"* [28, p101] define cohesiveness. The goal is to create a group where the members possess strong bonds to both i) each other and ii) the group as a whole. This in turn stimulates productive learning and the development of critical thinking skills.

Given that there are three different dimensions of social presence, the coding scheme for social presence (see Table 1) defines a list of indicators for each dimension. By looking at the content and the timing of each message, it is possible to see how the social climate unfolded during the course delivery. This provides a way of understanding and evaluating the different pedagogical interventions with respect to the development of a productive social climate in a learning community which enables for the meaningful social interactions [53].

2.3 Research Question: Characterization of social capital through social presence

As indicated in the previous sections, there is a strong evidence that social capital plays an important role in the shaping of the overall learning experience. The main research question that we investigate in this paper:

> What is the relationship between the students' *social capital*, as captured by social network centrality measures, and students' *social presence*, as defined by the three categories in the Community of Inquiry model?

The higher the social capital of a learner is, the more capable the learner is in terms of learning opportunities, information exchange, or integration within the academic environment. Still, the origins of social capital are not fully understood. Why certain students occupy advantageous positions in social networks? What are the social processes that enable them to take advantage of their social relationships? As for now, not a single theory of learning addresses the question of social capital directly, even though the impact of social context on learning is widely acknowledged.

As indicated by the previous study by de Laat et al. [18], content analysis techniques can be used in combination with SNA to provide a more comprehensive view of the social learning processes. In this paper, we propose the use of the Community of Inquiry model, given its holistic view of educational experience and extensive empirical evaluation by the research community [29], with the aim to characterize the origins of social capital in communities of inquiry. The CoI model description of important behavioral indices that contribute to the development of the positive social climate could be used to interpret the observed differences among students positions in a social network.

Likewise, the synergistic effect of using those two perspectives on student interactions provide a value for the CoI model by emphasizing the effects of the theorized social processes. For example, are interactivity and open communication important for the development of social capital? Are the students who show group cohesion the ones who take brokerage positions? Recently, there have been

Table 1: Social Presence Categories and Indicators as defined by Rourke et al. [45]

Category	Code	Name	Definition
Affective	A1	Expression of emotions	Conventional expressions of emotion, or unconventional expression of emotion, includes repetitions punctuation, conspicuous capitalization, emoticons.
	A2	Use of humor	Teasing, cajoling, irony, understatements, sarcasm.
	A3	Self-disclosure	Presenting details of life outside of class, or express vulnerability.
Interactive or Open Communication	I1	Continuing a thread	Using reply feature of software rather than starting a new thread.
	I2	Quoting from others' messages	Using software features to quote others entire messages or cutting and pasting selections of others' messages.
	I3	Referring explicitly to others' messages	Direct references to contents of others' posts
	I4	Asking questions	Students ask questions of other students or the moderator.
	I5	Complementing, expressing appreciation	Complimenting others or contents of others' messages.
	I6	Expressing agreement	Expressing agreement with others or content of others' messages.
Cohesive	C1	Vocatives	Addressing or referring to participants by name.
	C2	Addresses or refers to the group using inclusive pronouns	Addresses the group as *we,us, our, group*.
	C3	Phatics, salutations	Communication that serves a purely social function: greetings, closures.

some attempts [47, 48] that make use of SNA in conjunction with the CoI model to provide insights into particular aspects of learning, such as self-regulation [9]. Still, the central question of social capital is left unexplored and that is the goal in our study.

3. METHODS

3.1 Dataset

For our study, we used the dataset consisting of six offers (Winter 2008, Fall 2008, Summer 2009, Fall 2009, Winter 2010, Winter 2011) of the masters level software-engineering course offered through the fully online instructional condition at a Canadian open public university. The course is 13 weeks long, research-intensive, and focuses on understanding of current research trends and challenges in the area of software engineering. Students were requested: i) to participate in online discussions for which they received 15% of their final grade (see details in [32]), and ii) to work on a four tutor marked assignments. Overall, 81 student created the total of 1747 discussion messages which were then used as the main data source for this study. The total number of students and messages for all six course offerings are shown in Table 2.

Table 2: Course offering statistics

	Student count	Message count	Graph density
Winter 2008	15	212	0.52
Fall 2008	22	633	0.69
Summer 2009	10	243	0.84
Fall 2009	7	63	0.58
Winter 2010	14	359	0.84
Winter 2011	13	237	0.77
Average	13	291	0.71
Total	81	1747	

3.2 Social network measures

In order to measure students' social capital we extracted student social network graphs from the interactions on the discussion boards. We extracted *directed* social graphs, so that whenever a student $X1$ responded to a message from another student $X2$, we created a direct relationship between the two of them ($X1 \Rightarrow X2$). Since two students can exchange more than one message, we extracted a *weighted* graph where the weights corresponded to the number of exchanges between a given pair of students. We created a separate social graph for each of the course offerings independently and the graph densities for each offering are shown in Table 2.

From the constructed social network graphs, we extracted the three network centrality measures which are most frequently used for the study of the educational social networks [14]:

1) **Betweenness centrality** captures brokerage opportunities of actors in a network and is the most directly related to the social capital construct [13, 12]. For a given actor A, it is mathematically defined as the number of shortest paths between any two other actors that "pass through" the actor A [26].
2) **Degree centrality** measures the total number of relationships that each participant has [26]. Given that we constructed the directed social graphs, we considered separately the in-degree and out-degree centrality measures. They represent the total number of incoming and outgoing relations for a given individual, respectively. Degree is the simplest centrality measure, very easy to calculate, as it takes into account only the direct relationships between the actors [52].
3) **Closeness centrality** represents the distance of an individual participant in the network from all the other network participants [26]. It is defined as the inverse of the sum of the distances to all other participants [14], and hence takes into account both direct and indirect relationships [52]. Much like degree centrality, given that the student graphs are directed, we calculated the in-closeness and the out-closeness centrality measures. For a given actor A, in-closeness centrality measures how many indirect steps are needed for all other actors to reach the actor A, while out-closeness measures how many indirect steps the actor A requires in order to reach all the other actors in the network.

Table 3: Descriptive statistics of social network metrics

	$Mean$	SD	Min	Max
Betweenness	9.04	14.51	0.00	74.20
In-degree	19.84	8.62	4.00	42.00
Out-degree	19.86	9.37	3.00	44.00
In-closeness	0.09	0.04	0.04	0.17
Out-closeness	0.08	0.04	0.03	0.18

Table 3 shows the descriptive statistics for all five extracted centrality measures. We can see that on average the students wrote around 20 messages, and also received on average around 20 responses. This level of activity was expected, as by the course design the students were expected to spend a significant amount of time on the online discussions. Still, from the descriptive statistics reported in Table 3, we can observe the large differences between the individual students in the case of all five centrality measures.

3.3 Message coding

In order to assess students' social presence, all messages were manually coded by two coders in accordance with the coding scheme defined by Rourke et al. [45]. As the individual messages can

Table 4: Social Presence Indicators

Category	Code	Indicator	Count	Percent Agreement
Affective	A1	Expression of emotions	288 (16.5%)	84.4
	A2	Use of humor	44 (2.52%)	93.1
	A3	Self-disclosure	322 (18.4%)	84.1
Interactive	I1	Continuing a thread	1664 (95.2%)	98.9
	I2	Quoting from others messages	65 (3.72%)	95.4
	I3	Referring explicitly to other's messages	91 (5.21%)	92.7
	I4	Asking questions	800 (45.8%)	89.4
	I5	Complementing, expressing appreciation	1391 (79.6%)	90.7
	I6	Expressing agreement	243 (13.9%)	96.6
Cohesive	C1	Vocatives	1433 (82%)	91.8
	C2	Addresses or refers to the group using inclusive pronouns	144 (8.24%)	88.8
	C3	Phatics, salutations	1281 (73.3%)	96.1

Table 5: Social Presence Categories.

Category	Count	Percent Agreement
Affective	530 (30.3%)	80.8
Interactive (Excluded I1 and I5)	1030 (59%)	86.2
Cohesive (Excluded C1)	1326 (75.9%)	93.4

be simultaneously classified into more than one category of social presence, each message was coded with three binary codes indicating whether the message belongs to a particular social presence category. However, early in the coding process, we observed an extremely high frequency of some of the indicators in the cohesive and interactive categories. Because of this, almost all of the messages could be classified as both interactive and cohesive, which would limit the discriminatory power of those two categories. Thus, to resolve this issue, instead of coding on the levels of categories, the coding was done on the levels of the individual indicators, so that each message was coded with the twelve binary codes (i.e., three indicators of the affective category, six indicators of the interactive category and three indicators of the cohesive category) each indicating an occurrence of a particular social presence indicator within a given message. This enabled us to look at the distribution of the individual indicators and to be more selective in the type of the indicators that we wanted to investigate. Overall, the coding agreement was high, with all of the indicators reaching percent agreement of at least 84%, and all the coding disagreements were resolved through discussion between the coders in a follow-up meeting, after they first coded the messages independently. The coding results are shown in Table 4. The results show that some of the indicators were recorded in a disproportionately large number of messages. Thus, in order to evaluate different aspects of social presence captured by those three categories, we omitted some of the indicators from our analysis: i) Continuing a thread, ii) Complementing, expressing appreciation, and iii) Vocatives. We intentionally kept the "Phatics, salutations indicator" as its removal would render the cohesive category in only 8.24% of the messages. By using the remaining nine indicators, we categories all of the messages in the corpus, and the final results are shown in Table 5.

3.4 Statistical analysis

In order to investigate the relationships between the three categories of social presence, as defined by the CoI model, and *social capital*, as operationalized through the five network centrality measures, we conducted backward-stepwise multiple linear regression analyses [35] for each of the five extracted network centrality measures. To evaluate different regression models for a particular centrality measure, we used the popular Akaike Information Criterion (AIC) [35]. In order to control for the inflation of the Type-I error rate due to multiple statistical significance testing, we used the Holm-Bonferroni correction [37], also known as the sequential rejective Bonferroni correction. It provides a control for Type-I errors at a prescribed significance level – in our case $\alpha = 0.05$ – while providing a substantial increase in the statistical power over the commonly used Bonferroni correction [22]. In the case of testing the family of N null-hypothesis and significance level α, the Holm-Bonferroni method proceeds as follows:

1) Hypothesis with the smallest observed p-value, is tested using the adjusted significance level $\alpha' = \alpha/N$, in the same manner as in the traditional Bonferroni procedure.
2) However, the next smallest observed p-value is tested using differently adjusted significance level $\alpha' = \alpha/(N-1)$.
3) The same process repeats up to the hypothesis with the highest observed p-value which is tested using the unadjusted significance level α.
4) The important additional rule is that if any of the hypothesis in the family gets rejected, then *all the subsequent* hypotheses are rejected as well regardless of their observed p-values.

By using differently adjusted statistical significance levels, Holm-Bonferroni method guarantees that the family-wise error rate is kept at the prescribed level, while providing a significant increase in the statistical power over the more commonly used simple Bonferroni correction [22]. We used the Holm-Bonferroni correction for testing the overall significance of the regression models, and for testing the significance of the individual predictor variables. In our case, with five hypothesis tests, the values of the adjusted statistical significance levels were $\alpha = [0.01, 0.0125, 0.0167, 0.0250, 0.05]$.

We also inspected the QQ-Plots for the signs of the severe deviation from the normality of residuals, and we assessed the multicollinearity of the three predictor variables using the variance-inflation factors (VIFs). The QQ-Plots did not reveal deviations from the normality of the residuals and VIF values were substantially lower than the typically used thresholds such as 4 or 10 [10]. Thus, we considered the use of the multiple linear regression appropriate for our study.

4. RESULTS

The results of the regression analyses are shown in Table 6. The models for betweenness, in-degree, out-degree and in-closeness centralities were significant, while the model for out-closeness was marginally significant.

In the case of betweenness centrality, the multiple regression model explained 32% of the variability in the students scores of betweenness centrality. The backwards-stepwise regression analysis selection using the (AIC) criterion resulted in a regression model consisting of the affective and interactive categories of social presence, and both variables were found to be statistically significant predictors of betweenness centrality. In terms of their relative importance, the interactive category had a slightly larger standardized β coefficient than the affective category of social presence, indicating a slightly larger effect on the students' betweenness centrality scores.

With respect to degree centrality, the regression models explained 86% and 83% of the variability in the measures of in-degree and out-degree centralities, respectively. All three predictors were positively associated with the degree centrality measures, and all three reached the statistical significance. In terms of their relative importance, in both models, the interactive category of social presence

Table 6: Regression results for selected centrality measures after stepwise model selection using AIC criterion.

	Betweenness			In-degree			Out-degree			In-closeness			Out-closeness		
	β	SE	p	β	SE	p	β	SE	p	β	SE	p	β	SE	p
Affective	**0.27**	**0.12**	**0.024**	**0.18**	**0.054**	**0.001**	**0.23**	**0.059**	**<0.001**						
Interactive	**0.38**	**0.12**	**0.002**	**0.65**	**0.064**	**<0.001**	**0.65**	**0.07**	**<0.001**	**0.27**	**0.11**	**0.015**	0.37	0.15	0.017
Cohesive				**0.2**	**0.061**	**0.001**	**0.14**	**0.066**	**0.041**				-0.23	0.15	0.137
$F(3, 77)$	**19.6**		**<0.001**	**159**		**<0.001**	**130**		**<0.001**	**6.24**		**0.015**	3.03		0.054
Adjusted R^2	**0.32**			**0.86**			**0.83**			**0.061**			0.048		

had the largest standardized β coefficient, while the affective and cohesive categories had roughly the same standardized coefficients.

Regarding the two closeness centrality measures, the regression model for in-closeness was statistically significant, explaining 6.1% of the variability in the students' in-closeness centrality scores, while the model for out-closeness failed to reach the significance by a very small margin. The model for in-closeness consisted of only the interactive category, which was found to be a statistically significant predictor of in-closeness centrality. Similarly, the regression model for out-closeness consisted of the interactive and cohesive social presence categories, and explained 4.8% of the variation in the students' out-closeness centrality scores. In the model for out-closeness centrality, the only statistically significant predictor was the interactive category of social presence, while interestingly, the cohesive category of social presence was negatively associated with the change in the out-closeness centrality values, although statistically insignificantly.

5. DISCUSSION

One finding immediately stands out of the regression analyses results: *Interactive social presence is the most strongly associated with all of the network centrality measures, indicating a significant relation with the development of the students' social capital*. A possible explanation of this lies to some degree in the nature of students' social networks. Given that the primary goal of social networks in online courses is to serve as a communication medium for fostering of collaborative learning [27], it is reasonable to expect that interactivity in communication can explain a significant proportion of the differences in network positions, and ultimately the differences in the development of students' social capital. The reason why the interactive category is had the strongest association might be that only after the students have gott familiar with each other through focused, on-task interactions, and after they have started developing trust within a learning community, the expression of emotions and the sense of group belonging begins to emerge. This is aligned with the findings of Garrison [27] who suggested that interactive social presence is dominant at the beginning of a course, but decreases over time, while affective and cohesive social presence increase over time [27]. However, as Garrison [27] points out, too much of the interpersonal and affective interactions undermine the productivity of the collaborative learning activities. There is a certain amount of social interactions that is beneficial for learning [27], and the focus of the instructional interventions should be on: i) stimulating the right amount of the different social interactions that support productive and purposeful collaborative learning activities, and ii) the development of trust and the sense of community among the group of learners [17].

One practical implication of these results is that they suggest the effective way for fostering the productive social climate – and that is *focusing on the student interaction and open communication*. In order to guide the development of the social relationships in a learning community, it seems that the instructional emphasis should be on the interventions that require engaging in an open exchange of ideas and opinions, that would in turn lead to more affective expression, and eventually to the development of the sense of community belonging. Still, this hypothesis warrants further investigation, and in the future we plan to analyze the evolution of the students' social presence and the corresponding social network structures over time, which would shed new light on this important question.

The results of individual network centrality measures revealed that both in-degree and out-degree centrality measures were significantly predicted by all the three categories of students' social presence. By looking at the description (Section 2.2.2) and the indicators (Table 1) of the interactive category of social presence, we can see that interactive social presence is mainly about stimulating open and direct communication between the students. Thus, the students who exhibit a high level of interactive social presence have higher chances of "provoking" a response from the other students. Activities such as asking questions, explicitly referring to other students by name, quoting their messages, complementing them or agreeing with their messages, are all activities associated with an interactive and open communication, and can be used to elicit a response from the other students. It would be interesting to further investigate the relationship between different indicators of social presence and social capital, as certain indicators – such as `I4` "Asking questions" – seem to have more impact than the other indicators. Besides the interactive category, the regression model revealed that the affective and cohesive categories of social presence were also significant predictors of in-degree and out-degree centralities. These findings are even more interesting, as affective and cohesive exchanges are not directly stimulating discussions in the same manner as the interactive category. Further investigation is needed to examine particular time periods over the duration of a course in which those different dimensions of social presence contribute to the degree centrality measures of students.

With respect to betweenness centrality that is most closely related to the notion of social capital [13, 12], the regression model was statistically significant and explained 32% of the variability in the betweenness centrality scores. This corresponds to Cohen's $f^2 = 0.47$ effect size, which is considered to be a large effect size [16]. Both the interactive and affective categories of social presence were statistically significant predictors of the betweenness centrality, with the interactive category having a bit greater standardized β coefficient. This might be due to the nature of student communication networks and their focus on collaborative learning, which resulted in the emphasis on information exchange. Still, these are very intriguing findings, given that betweenness centrality is not directly related to the number of interactions the student has, but more to the overall diversity of the interactions within a group of learners. In a follow-up study, it would be very interesting to investigate whether there are any particular ways in which the students with the high betweenness centrality differ from the other students (e.g., asking many questions or exhibiting higher self-disclosure).

Regarding the closeness centrality measures, the regression model for in-closeness was also statistically significant. The model explained 6.1% of the variability, and the stepwise model selection

using the AIC criteria resulted in a simple regression model with only the interactive category of social presence. In contrast to degree centrality, which considers only direct relationships, closeness centrality also considers the indirect relationships. Such indirect relationships could be the reason why only interactive category was rendered as important. The affective and cohesive exchanges between students A and B, although very important, provide very little, or no influence on the indirect relations of student B and the rest of the students. The similar findings we could see in the model for out-closeness, which was marginally significant with the p-value of 0.054. However, it could be expected that the significance of this model would be conformed in a larger replication study.

The major limitations of this study is the sample size and the use of the single course from a single institution. Even though there were six offerings of the course taught by the two instructors, there might still be significant effects of the adopted pedagogical approach, which could have shaped a specific social dynamics, and thus, potentially distort the findings of our study. Likewise, we considered all interactions among the students as contributing to their social capital, it is very likely that the certain interactions (e.g., adversarial interactions) might have a negative effect on the student social capital. In the future work, we plan on replicating our findings on a bigger sample and with more diverse courses from different subject matter domains. Finally, we plan to investigate the temporal aspects of the relationship between social capital and the social presence, which might give us a deeper insight into the complexity of the social interactions in learning communities.

6. CONCLUSIONS

The study presented in this paper investigated some of the social processes that can contribute to the development of students' social capital. We have looked at the relationship between students' *social presence*, operationalized through the Community of Inquiry model, and students' *social capital*, operationalized through the three network centrality measures. The implications of our findings are twofold: *First*, our results indicate that a significant part of the variability in network centrality scores can be explained using the three dimensions of the social presence, and this in turn indicates the existence of the relationship between the development of social presence and social capital. All three categories of social presence were significant predictors of in-degree and out-degree centrality measures while interactive and affective categories were significant predictors of the betweenness centrality. Also, interactive category of social presence was significantly predictive of the in-closeness and out-closeness centrality measures, although the overall regression model for out-closeness was marginally significant. A possible explanation is that given the task-oriented nature of discussions in online courses, students' social presence develops mostly through interactions focused on learning, and then over time, with the development of trust among a group of learners, the other dimensions of social presence start to emerge. *Second*, the study shows the significant relationship between the interactive category of social presence and betweenness, in-degree, out-degree, and in-closeness network centrality measures. This provides an empirical basis for fostering the productive social climate in discussions through interventions that increase interactivity and open communication among the students. By engaging students to participate in discussions with the clearly defined expectations, students develop social relationships which can in turn have positive impact on the attainment of the learning objectives and their overall academic experience.

References

[1] Special issue on the community of inquiry framework: Ten years later. *The Internet and Higher Education*, 13(1–2), 2010.

[2] T. Anderson and J. Dron. Three generations of distance education pedagogy. *The International Review of Research in Open and Distance Learning*, 12(3):80–97, 2010.

[3] T. Anderson, L. Rourke, D. R. Garrison, and W. Archer. Assessing teaching presence in a computer conferencing context. *Journal of Asynchronous Learning Networks*, 5:1–17, 2001.

[4] J. Arbaugh, M. Cleveland-Innes, S. R. Diaz, D. R. Garrison, P. Ice, J. C. Richardson, and K. P. Swan. Developing a community of inquiry instrument: Testing a measure of the community of inquiry framework using a multi-institutional sample. *The Internet and Higher Education*, 11(3–4):133–136, 2008.

[5] A. W. Astin. *What Matters in College: Four Critical Years Revisited*. Jossey-Bass, 1 edition edition, 1997.

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

[7] T. T. Baldwin, M. D. Bedell, and J. L. Johnson. The social fabric of a team-based M.B.A. program: Network effects on student satisfaction and performance. *The Academy of Management Journal*, 40(6):1369–1397, 1997.

[8] J. P. Bean. Conceptual models of student attrition: How theory can help the institutional researcher. *New Directions for Institutional Research*, 1982(36):17–33, 1982.

[9] R. A. Bjork, J. Dunlosky, and N. Kornell. Self-regulated learning: beliefs, techniques, and illusions. *Annual review of psychology*, 64:417–444, 2013.

[10] B. L. Bowerman and R. T. O'Connell. *Linear Statistical Models: An Applied Approach*. Duxbury Press, 1990.

[11] S. Buckingham Shum and R. Ferguson. Social learning analytics. *Journal of Educational Technology & Society*, 15(3):3–26, 2012.

[12] R. S. Burt. Structural holes versus network closure as social capital. In N. Lin, K. Cook, and R. S. Burt, editors, *Social Capital: Theory and Research*. Aldine Transaction, 2001.

[13] R. S. Burt. The social capital of structural holes. In M. F. Guillen, R. Collins, P. England, and M. Meyer, editors, *The New Economic Sociology: Developments In An Emerging Field*. Russell Sage Foundation, 2005.

[14] B. V. Carolan. *Social Network Analysis and Education: Theory, Methods and Applications*. SAGE Publications, Inc., 2014.

[15] H. Cho, G. Gay, B. Davidson, and A. Ingraffea. Social networks, communication styles, and learning performance in a CSCL community. *Computers & Education*, 49(2):309–329, 2007.

[16] J. Cohen. The analysis of variance. In *Statistical power analysis for the behavioral sciences*, pages 273—406. L. Erlbaum Associates, Hillsdale, N.J., 1988.

[17] S. Dawson. A study of the relationship between student social networks and sense of community. *Journal of Educational Technology & Society*, 11(3):224–238, 2008.

[18] M. F. De Laat, V. Lally, L. Lipponen, and R.-J. Simons. Investigating patterns of interaction in networked learning and computer-supported collaborative learning: A role for social network analysis. *International Journal of Computer-Supported Collaborative Learning*, 2(1):87–103, 2007.

[19] B. De Wever, T. Schellens, M. Valcke, and H. Van Keer. Content analysis schemes to analyze transcripts of online asynchronous discussion groups: A review. *Computers & Education*, 46(1):6–28, 2006.

[20] J. Dewey. My pedagogical creed. *School Journal*, 54(3):77–80, 1897.

[21] R. Donnelly and J. Gardner. Content analysis of computer conferencing transcripts. *Interactive Learning Environments*, 19(4):303–315, 2011.

[22] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[23] J. E. Eckles and E. G. Stradley. A social network analysis of student retention using archival data. *Social Psychology of Education*, 15(2):165–180, 2011.

[24] S. L. Feld. The focused organization of social ties. *American Journal of Sociology*, 86(5):1015–1035, 1981.

[25] R. Ferguson and S. B. Shum. Social learning analytics: five approaches. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, page 23–33, New York, NY, USA, 2012. ACM.

[26] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978.

[27] D. R. Garrison. *E-Learning in the 21st Century: A Framework for Research and Practice*. Routledge, New York, 2 edition edition, 2011.

[28] D. R. Garrison, T. Anderson, and W. Archer. Critical inquiry in a text-based environment: Computer conferencing in higher education. *The Internet and Higher Education*, 2(2–3):87–105, 1999.

[29] D. R. Garrison, T. Anderson, and W. Archer. The first decade of the community of inquiry framework: A retrospective. *The Internet and Higher Education*, 13(1–2):5–9, 2010.

[30] D. R. Garrison and J. Arbaugh. Researching the community of inquiry framework: Review, issues, and future directions. *The Internet and Higher Education*, 10(3):157–172, 2007.

[31] R. Garrison, M. Cleveland-Innes, and T. S. Fung. Exploring causal relationships among teaching, cognitive and social presence: Student perceptions of the community of inquiry framework. *The Internet and Higher Education*, 13(1–2):31–36, 2010.

[32] D. Gasevic, A. Olusola, S. Joksimovic, and V. Kovanovic. Externally-facilitated regulation scaffolding and role assignment to develop cognitive presence in asynchronous online discussions. *The Internet and Higher Education*, (submitted), 2014.

[33] D. Gasevic, A. Zouaq, and R. Janzen. "Choose your classmates, your GPA is at stake!": The association of cross-class social ties and academic performance. *American Behavioral Scientist*, 2013.

[34] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[35] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, New York, NY, 2013.

[36] C. Haythornthwaite. Social network analysis: An approach and technique for the study of information exchange. *Library & Information Science Research*, 18(4):323–342, 1996.

[37] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[38] K. H. Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, 2003.

[39] N. Lin. Social resources and instrumental action. In P. V. Marsden and N. Lin, editors, *Social structure and network analysis*, pages 131—145. Sage Publications, 1982.

[40] P. Long and G. Siemens. Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5):31–40, 2011.

[41] R. Luppicini. Review of computer mediated communication research for education. *Instructional Science*, 35(2):141–185, 2007.

[42] A. Mehrabian. Some referents and measures of nonverbal behavior. *Behavior Research Methods & Instrumentation*, 1(6):203–207, 1968.

[43] 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.

[44] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *Trans. Sys. Man Cyber Part C*, 40(6):601–618, 2010.

[45] L. Rourke, T. Anderson, D. R. Garrison, and W. Archer. Assessing social presence in asynchronous text-based computer conferencing. *The Journal of Distance Education*, 14(2):50–71, 1999.

[46] J. Scott and P. J. Carrington. *The SAGE Handbook of Social Network Analysis*. SAGE Publications, 2011.

[47] P. Shea, S. Hayes, S. U. Smith, J. Vickers, T. Bidjerano, M. Gozza-Cohen, S.-B. Jian, A. Pickett, J. Wilde, and C.-H. Tseng. Online learner self-regulation: Learning presence viewed through quantitative content- and social network analysis. *The International Review of Research in Open and Distance Learning*, 14(3):427–461, 2013.

[48] P. Shea, S. Hayes, J. Vickers, M. Gozza-Cohen, S. Uzuner, R. Mehta, A. Valchova, and P. Rangan. A re-examination of the community of inquiry framework: Social network and content analysis. *The Internet and Higher Education*, 13(1–2):10–21, 2010.

[49] R. A. Smith and B. L. Peterson. "Psst ... what do you think?" the relationship between advice prestige, type of advice, and academic performance. *Communication Education*, 56(3):278–291, 2007.

[50] S. L. Thomas. Ties that bind: A social network approach to understanding student integration and persistence. *The Journal of Higher Education*, 71(5):591–615, 2000.

[51] V. Tinto. *Leaving College: Rethinking the Causes and Cures of Student Attrition*. University of Chicago Press, 1993.

[52] S. Wasserman. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[53] Y. Woo and T. C. Reeves. Meaningful interaction in web-based learning: A social constructivist interpretation. *The Internet and Higher Education*, 10(1):15–25, 2007.

Cross-Domain Performance of Automatic Tutor Modeling Algorithms

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

ABSTRACT

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

Keywords

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

1. INTRODUCTION

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

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

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

2. RELATED WORK

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

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

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

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

3. GENERATING BEHAVIOR GRAPHS

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

3.1 Behavior Graphs

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

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

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

3.1.1 Effective

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

3.1.2 Readable

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

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

3.1.3 Complete

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

3.1.4 Accurate

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

3.1.5 Robust

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

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

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

3.2 ABGG Algorithms

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

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

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

First order transition matrix computed from solution traces can be used to represent a directed graph. *Algorithm 4* considers ABGG as the process of finding multiple paths in a directed graph. Specifically, the longest (non-repeating) path in this directed graph represents the most likely path through the solution steps. Since, the problem of finding longest paths in general graphs is known to be NP-hard, we employ a combination of bounded

longest path finding and an algorithm for finding multiple shortest paths [18] in a transformed transition matrix to obtain a number of different paths through the directed graph. These paths are merged to construct a behavior graph similar to the process of constructing an interaction network.

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

3.3 Discussion

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

Table 1. Comparison of Algorithm Capabilities

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

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

4. DATASETS

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

Table 2. Problems & Traces for the three learning domains

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

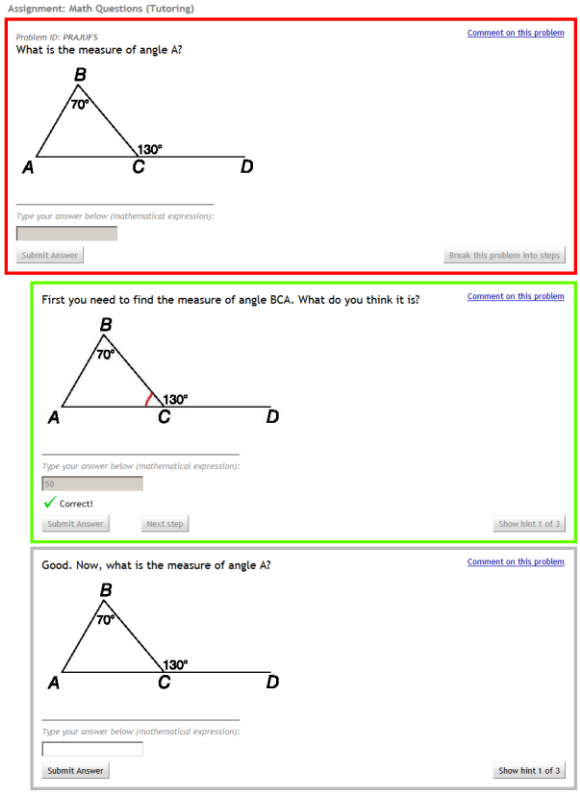


Figure 1. Example Math Problem from *Assistments*
Source: www.assistments.org, April 2014

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

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

of each other to account for change in UI designs of the problems common to the three datasets.

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

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

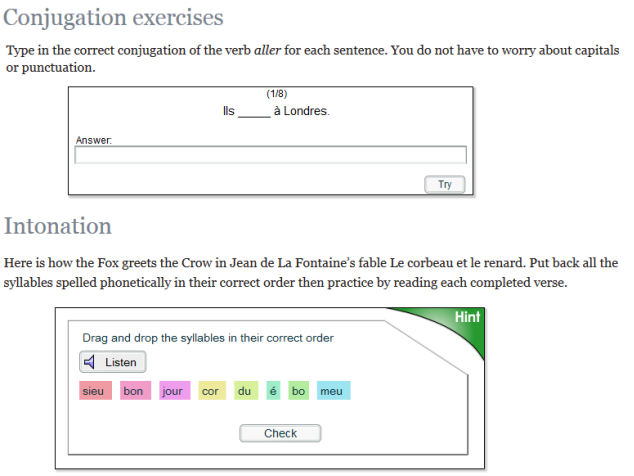


Figure 2. Example Steps from Problem from the French Online Course *Source: oli.cmu.edu, April 2014*

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

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

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

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

5. EXPERIMENTS

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

5.1 Results

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

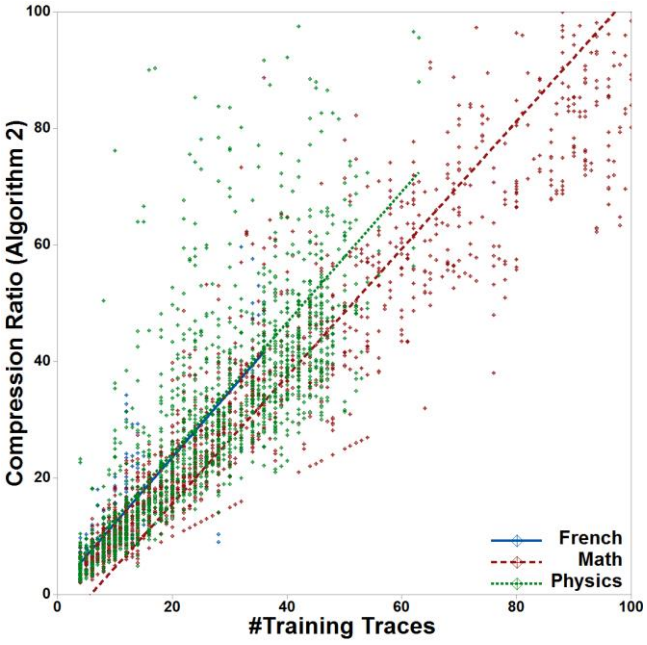


Figure 3. Compression Ratio of *Algorithm 2*

5.1.1 Mathematics

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

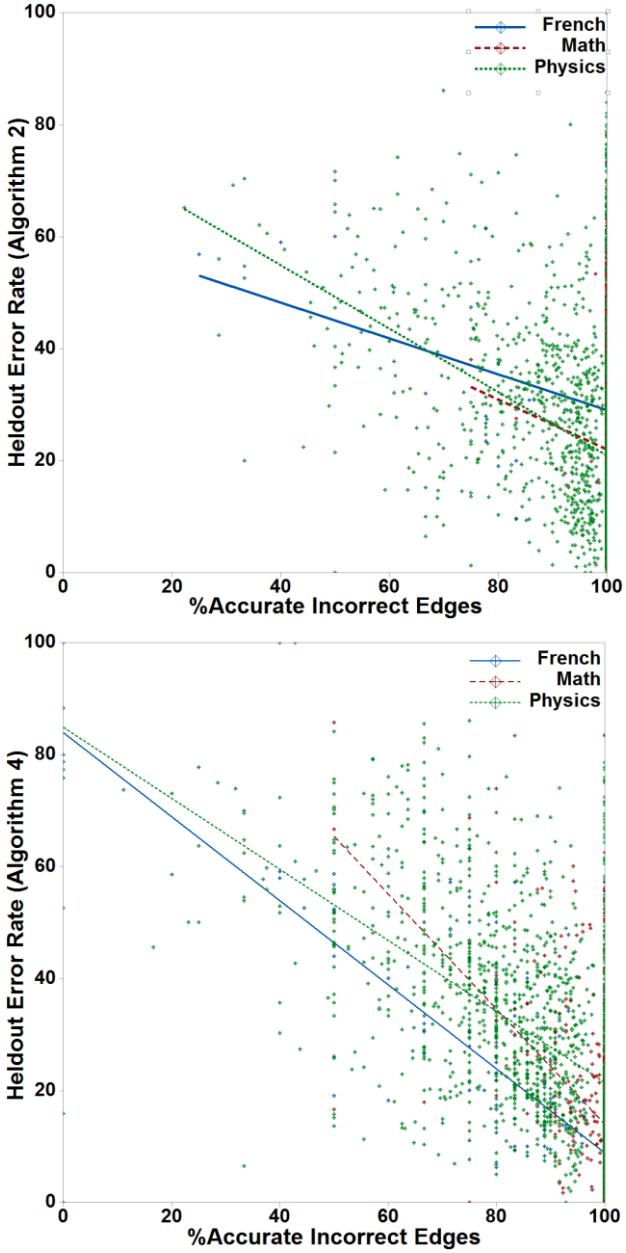


Figure 4. Heldout Error Rate of *Algorithms 2 and 4*

5.1.2 Physics

On the primary readability metric (Compression Ratio), *Algorithm 2* outperforms the others on the Physics dataset as was the case with Mathematics. This is consistent with prior conclusion [4] on the use of *Algorithm 2* for readability. We note that the Physics

dataset has significantly lower compression ratio than the previous dataset. Figure 3 shows a scatter plot and domain-specific regression fits for the compression ratio of *Algorithm 2* for different problems with different number of training traces and UI elements. We see that for equivalent number of training traces, the compression ratio for Physics is actually slightly better than Mathematics. However, as we know from Table 2, fewer training traces are available for the Physics problems on average.

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

5.1.3 French

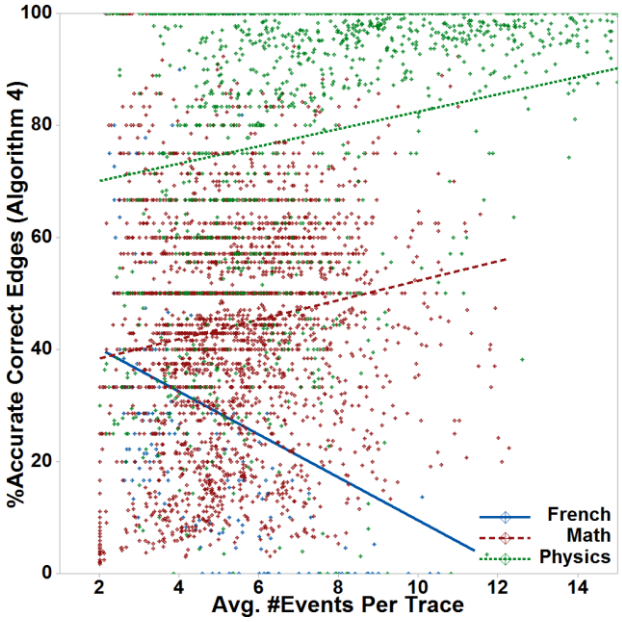


Figure 5. Accuracy of Correct Edges for *Algorithm 4*

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

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

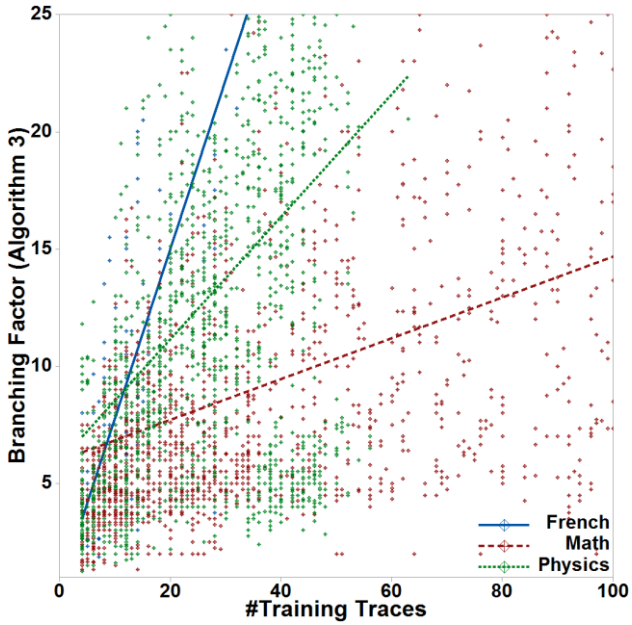


Figure 6. Branching Factor of *Algorithm 3*

5.1.4 Automatically Generated Behavior Graphs

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

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

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

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

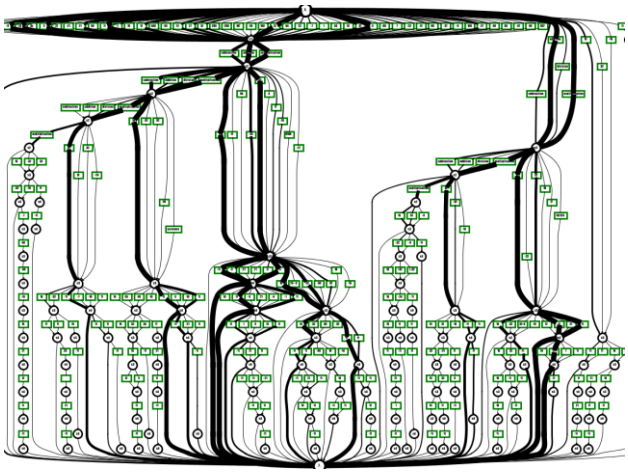


Figure 7a. Behavior Graph: Mathematics, *Algorithm 1*

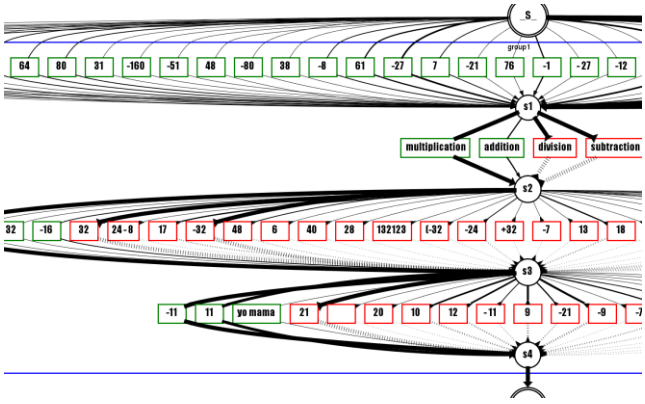


Figure 7b. Behavior Graph: Mathematics, *Algorithm 2*

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

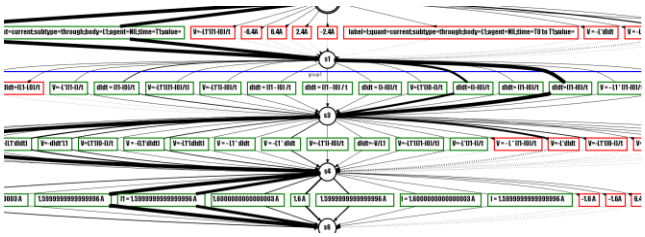


Figure 8. Behavior Graph: Physics, *Algorithm 2*

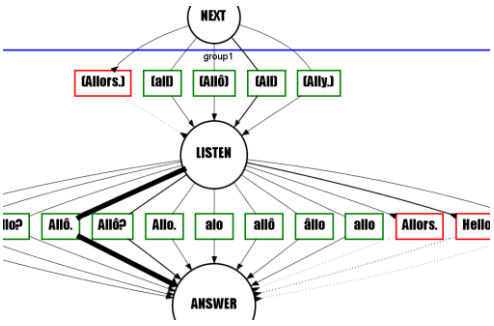


Figure 9a. Behavior Graph: French, *Algorithm 2*

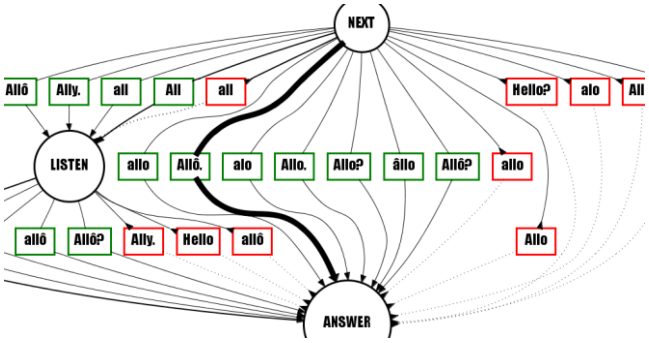


Figure 9b. Behavior Graph: French, *Algorithm 4*

6. CONCLUSIONS

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

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

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

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

ACKNOWLEDGEMENTS

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

7. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

AGG: Augmented Graph Grammars for Complex Heterogeneous Data

Collin F. Lynch
Intelligent Systems Program
and Learning Research & Development Center
Pittsburgh, Pennsylvania, U.S.A.
collinl@cs.pitt.edu

ABSTRACT

The central goal of educational datamining is to derive crucial pedagogical insights from student, course, and tutorial data. Real-world educational datasets are complex and heterogeneous comprising relational structures, social connections, demographic information, and long-term assignments. In this paper I describe *Augmented Graph Grammars* a robust formalism for graph rules that provides a natural structure for evaluating complex heterogeneous graph data. I also describe *AGG* an Augmented Graph Grammar engine written in Python and briefly describe its use.

Keywords

Augmented Graph Grammars, Graph Analysis, Argument Diagrams, Complex Data, Heterogeneous Data

1. INTRODUCTION

The central goal of educational datamining is to draw pedagogical insights from real-world student data, insights which can inform instructors, students, and other researchers. While robust analytical formalisms have been defined for categorical, numerical, and relational data most real-world educational data is complex and heterogeneous combining textual, numerical, and relational features. In large course settings such as a lecture course or MOOC, for example, students may form dynamic working groups and collaborate on complex assignments. They may also be given a flexible set of reading, writing, or problem-solving tasks that they can choose to complete in any order. This process data can be encoded as a graph with nodes representing individual assignments and reading materials and arcs representing group relationships or traversal order. In order to capture important features of this rich graph data and to identify key relationships between teamwork, written text, and performance, it is necessary to apply a rule structure that can capture them naturally.

Individual student assignments can also contain heterogeneous data. Argument diagrams, for example, have been used to teach writing, argumentation, and scientific reasoning [10, 2, 19]. These structures reify real-world arguments as graphs using complex node and arc types to represent argumentative components such as hypothesis statements, citations, and claims. These complex elements can include types, text fields for short notes or free-text assertions, links to external resources, and other data.

A sample student-produced argument diagram drawn from my thesis work at the University of Pittsburgh is shown in Figure 1. This work focused on the use of argument diagrams to support students in developing written scientific reports and in identifying *pedagogically-relevant* diagram structures that can be used to predict students' subsequent performance (see [8]). The diagram contains a central *claim* node representing a research claim. This node has a single text field in which the claim is stated. This is, in turn, connected to a set of *citation* nodes representing related work via a set of *supporting*, *opposing*, and *undefined* arcs colored green, red, and grey, respectively. The citation nodes each contain two text fields, one for the citation information and the other for a summary of the cited work, while the arcs contain a single text field for the *warrant* or explanation of why the relationship holds. At the top of the diagram there is a single disjoint *hypothesis* node which contains two text fields: a *conditional* or IF field, and a *conditional* or THEN field.

This diagram contains a number of pedagogically-relevant issues. Some of them are purely structural such as the disjoint hypothesis node, and the fact that the supporting and opposing arcs are drawn from the claim to the citations and not vice-versa. It also contains more complex semantic issues such as the fact that the text fields on the arcs contain summary information for the cites not explanations of the relationship, and the fact that the opposing citations, citations that disagree about the central claim node have not been distinguished from one-another via a comparison arc. Problems such as these can be detected via complex rules, and I have previously shown that the presence of such problems are predictive of students' subsequent performance [8, 10, 9]. This detection and remediation, however requires the development of rules that can incorporate complex structural and textual information.

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

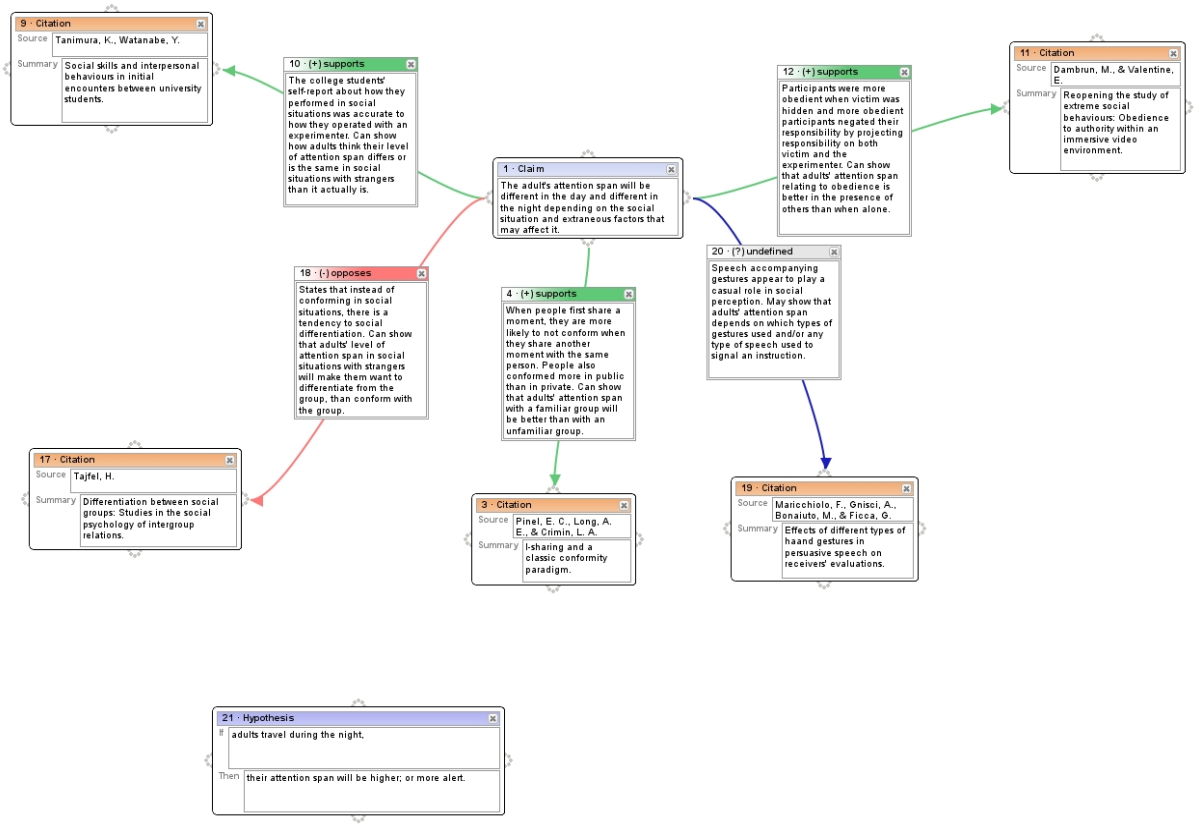


Figure 1: A segment of a student-produced LASAD diagram representing an introductory argument. It contains a central *claim* node surrounded by *citation* nodes. The isolated node is a *hypothesis* that has not been integrated into the argument.

Automatic graph analysis is central to a number of research domains including strategy transfer in games [4], automatic recommendations [1], cheminformatics [12], and social network detection [11]. Graph analysis algorithms have been used to define educational communities [15, 16, 5]) and to automatically grade existing datasets [8, 10, 9]. Graphical structures have also been used in tutoring contexts to represent student work via argument diagrams of the type shown above (see [14, 7] or to provide connection representations [19] for student guidance.

My focus in the present work is on the development of graph *rules* that is logical graph patterns that match arbitrary graph structures based upon content and structure information. While arbitrary graph matching is NP-Hard (see [18]) it is of practical importance, particularly in relational domains such as argument diagrams or student groups where our goal is to identify complex structures that may be evidence of deeper pedagogical issues. To that end, I will introduce *Augmented Graph Grammars* a robust rule formalism for complex graph rules and will describe *AGG* and augmented graph grammar engine for educational datamining. Both were developed as part of my thesis work at the University of Pittsburgh.

2. AUGMENTED GRAPH GRAMMARS

Graph Grammars, as described by Rekers and Schürr, are formal grammars whose atomic components are graphs or graph elements, and whose productions transpose one graph to another [17]. More formally, they define graph-grammars and productions as:

Definition 3.6 A graph grammar GG is a tuple $(A; P)$, with A a nonempty initial graph (the axiom), and P a set of graph grammar productions. To simplify forthcoming definitions, the initial graph A will be treated as a special case of a production with an empty left-hand side. The set of all potential production instances of GG is abbreviated with $PI(GG)$.

Definition 3.2 A (graph grammar) production p := (L; R) is a tuple of graphs over the same alphabets of vertex and edge labels LV and LE. Its left-hand side lhs(p) := L and its right-hand side rhs(p) := R may have a common (context) subgraph K if the following restrictions are fulfilled:

- $\forall e \in E(K) \Rightarrow s(e) \in V(K) \wedge t(e) \in E(K)$ with $E(K) := E(L) \cap E(R)$ and $V(K) := V(L) \cap V(R)$ i.e. sources and targets of common edges are common verticies of L and R, too.

- $\forall x \in L \cap R \Rightarrow l_L(x) = l_R(x)$ i.e. common elements of L and R do not differ with respect to their labels in L and R.

Thus graph grammars are systems of production rules analogous to context-sensitive string grammars (see [18]). For reasons of efficiency Rekers and Schürr restrict their focus to *layered graph-grammars* where all productions must be *expansive* with the left-hand-side being a subgraph of the right. Classical graph grammars, like string grammars, assume a fixed alphabet of simple statically-typed node and arcs and can be used both to generate matching graphs programmatically or to parse matching graphs via mapping and decomposition. My focus in the present work is on graph matching which occurs via iterative mapping.

Let $G_i =< \{n_o, \ldots\}, \{e(n_p, n_q), \ldots\} >$ and $G_j =< \{m_o, \ldots\}, \{e(m_k, m_l), \ldots\} >$ be graphs and let $M = \{< n_a, M - b > \ldots\}$ me a *mapping* from G_i to G_j that links nodes of the two. In the context of a mapping, G_i and G_k are called the *source* and *target* graphs respectively. A mapping M_{G_i,G_j} from G_i to G_j is *valid* if and only if the following holds:

$$\forall n_x \in G_i : \exists < n_x, m_y > \in M_{G_i,G_j}$$

$$\neg\exists\{< n_x, m_y >, < n_r, m_k >\} \subseteq M_{G_i,G_j} : (x = r) \vee (y = k)$$

$$\forall e(n_x, n_y) \in G_i : \{< n_x, m_y >, < n_r, m_k >\} \subseteq M_{G_i,G_j} : \exists e(m_y, m_k) \in G_j$$

For the remainder of this paper all elements in a source graph will be labeled alphabetically (e.g. a, Q) while elements in the target graphs will be referenced numerically (e.g. $1, 2, e(2,3), \overrightarrow{e(4,5)}$).

Augmented Graph Grammars are a richer formalism for graph rules that treat nodes and arcs as complex components with optional sub-fields including flexible text elements or other types. Augmented graph grammars have been previously described by Pinkwart et al. in [13]. There the authors focused on the use of augmented graph grammars for tutoring. An Augmented Graph Grammar is defined by: a graph ontology that specifies the complex graph elements and functions available; a set of graph classes that define matching graphs; and optional graph productions and expressions that provide for recursive class mapping and logical scoping. I will describe each of these components briefly below. For a more detailed description see [8].

2.1 Graph Ontology

In a simple graph grammar of the type used by Rekers and Schürr the set of possible node and arc types ($\sum$) is fixed with the elements being atomic, static, and unique. In order to process complex structures such as the argument diagram shown in Figure 1, a more complex structure is required. Thus augmented graph grammar ontologies are defined by a set of element types $O = \{N_0, \ldots N_m, E_0, \ldots, E_p\}$ such that each element has a unique list of fields and field types as well as applicable functions over those fields. The ontology must also specify appropriate relationships between the fields and operations that can be used on them.

While showing a complete ontology is beyond the scope of this paper an illustrative example can be found in Figure 2. This illustrates the field definitions for the citation and hypothesis nodes shown above. Both node types contain two sub-fields of type String. For each of these fields an additional function is defined '*.Words' which returns a set of all the words found in the field.

```
{
 Nodes:{
   Citation:{
     Cite(String)
     Cite.Words(StringSet)
     Summary(String)
     Summary.Words(StringSet)
   }

   Hypothesis: {
     If(String)
     If.Words(StringSet)
     Then(String)
     Then.Words(StringSet)
   }
 }
Arcs:{
  Comparison: {
    ...

Types: { String, StringSet }
...
```

Figure 2: An illustrative subset of a sample graph ontology for scientific argument diagrams.

2.2 Graph Classes

The core component of an augmented graph grammar is the graph *class*. A class C_i is defined by a 2-tuple $< S_i, O_i >$ where S_i is a graph *schema* and O_i is a set of *constraints*. A class defines a space of possible graphs which satisfy both the schema and the constraints. Classes are not required to be unique nor are the set of matching graphs for a given pair of classes required to be disjoint. A sample named class $R07a$ is shown in 3. This class is designed to detect instances of *Related Uncompared Opposition* in scientific argument diagrams. That is subgraphs where there exists a pair of citation nodes a, and b that disagree about a shared target node t, are not connected via a comparison arc c, and which share some relevant textual content. As I noted above, this type of structure can be found in Figure 1.

2.2.1 Graph Schema

A *Schema* is a graph structure that defines a space of possible graphs topologically. Schema are defined by a set of *ground nodes* (e.g. t, a, & b in Figure 3) which must match a single node in a target graph, a set of *ground arcs* that must likewise match a single arc in the target graph (e.g. c), and an optional set of *variable arcs* which must match a nonempty subgraph defined by a graph production. By convention, ground elements are denoted via lower-case names while variable elements are denoted by capitalized names.

In addition to the ground and variable distinctions arcs within a schema may be one of four types: *directed* (e.g. O, &

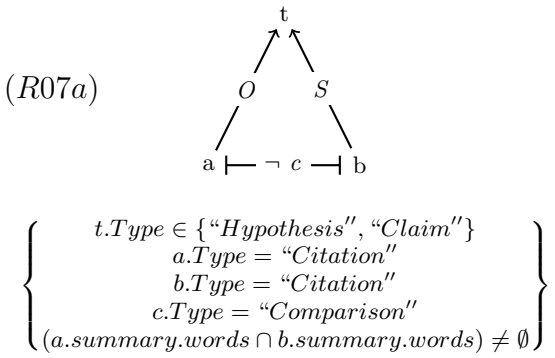


Figure 3: *Related Uncompared Opposition* A simple augmented graph grammar rule that detects related but uncompared counterarguments. The rule shows a two citation nodes (*a*, & *b*) that have opposing relationships with a shared hypothesis or claim node (*t*) and do not have a comparison arc (*c*) drawn between them. The arcs *S* and *O* represent recursive supporting and opposing paths.

S), of *unknown direction*, *undirected* (e.g. c), and *undefined*. Directed arcs will only match directed arcs in the base graph oriented in the same direction. Thus, given a base graph containing an arc $\overrightarrow{e(1,2)}$ and a schema with a directed arc $\overrightarrow{e(n,m)}$ the schema will only match cases where $\{<n,1>,<m,2>\} \subseteq M$. Unknown direction schema arcs may match a directed arc oriented in any order but will *not* match an undirected arc (e.g. $e(2,3)$). Undirected arcs (e.g. $\neg c$) will not match a directed arc. And, undefined arcs may match a directed or undirected arc in any order.

As the example shows arcs may be also be negated (e.g. $\neg c$) in which case the schema matches a graph if and only if no match can be found for the negated arc. Thus the schema shown will only match ground graphs with no arc between the elements assigned to a and b. More complicated cases of negation may be formed using graph expressions which are defined below.

The elements of a Schema must also be *non-repeating* that is, no two elements in a schema may be matched to the same element in the target graph. Thus each element in a schema must match at least one unique node or arc with variable elements possibly accounting for more than one element.

2.2.2 Constraints

Constraints represent individual bounds or limits on the ground elements of a schema. Constraints are specified using a set-theory syntax (e.g. $t.Type \in \{\text{"}Hypothesis\text{"}, \text{"}Claim\text{"}\}$) and may draw on any of the node or arc features, subfields, or functions specified in the ontology. *Unary Constraints* apply to a single element (e.g. $a.Type = \text{"}Citation\text{"}$). *Binary Constraints* (e.g. $(a.summary.words \cap b.summary.words) \neq \emptyset$) specify a relationship between two distinct ground elements.

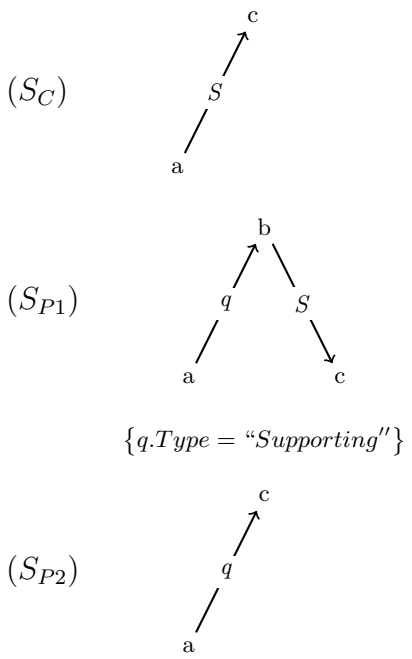


Figure 4: A simple recursive rule production for S that defines a *supporting path*.

2.3 Graph Productions

A graph production $C_l \Rightarrow C_{r1}|C_{r2}...$ is a context-sensitive production rule that maps from a graph class containing a single *production variable* to one or more alternate expansions. Graph productions are used to match layered subgraphs to the variable arcs. A simple recursive production rule for the variable element $\overrightarrow{S(b,t)}$ is shown in Figure 4.

The rule is defined by the *context class* S_C, and the two *production classes* S_{P1} and S_{P2}. The context class is used as a key for the production application. It must contain exactly one variable arc, the *production variable*, and no constraints. The ground nodes a and c are *context nodes* and are used to ground the production for mapping. They must be present in all of the production rules. All production rules must be *expansive* with each of the production classes containing at least one ground element not present in the context class. Recursive productions are thus handled by iteratively grounding the mapping with additional context and, as per the *non-repeating* requirement, these rules must consume additional elements of the graph. Production rules are thus mapped in a layered fashion like the grammars defined by Rekers and Schürr.

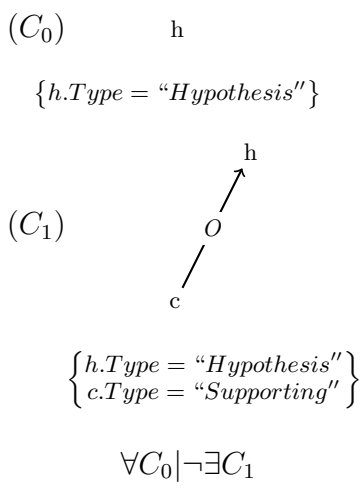


Figure 5: A simple Graph expression that tests for unopposed hypotheses.

2.4 Graph Expressions

Graph expressions are logical rules of the form:

$$S_0C_0 \quad | \quad S_1C_1 \quad | \quad \dots \quad | \quad S_mC_m$$

where each C_i is a graph class and each S_i is a logical quantifier from the set: $\{\forall, \neg\forall, \exists, \neg\exists\}$. The expressions allow for existential and universal scoping and arbitrary negation of graph classes. The expressions represent chained logical structures with each '|' being read as "...such that ...". A sample graph expression is shown in Figure 5. This sample expression asserts that for all hypothesis nodes in the target graph there exist no citation nodes that oppose the target hypothesis. Thus it is a universal claim about a negated existential item. As this example illustrates graph expressions allow for more complex negation structures than are supported by the graph schema.

Graph expressions must be expansive or *right-grounded* such that the following constraints hold:

$$\forall C_{m \leq i > 0} \in E : C_{i-1} \subseteq_g C_i$$

$$S_m \in \{\exists, \neg\exists\}$$

That is, the schema component of class C_i must be a subgraph of all classes class C_{i+n}. This also holds true for the constraints with all constraints present in class C_i being present in classes C_{i+n}. And the rightmost class in the expression must also be an existential ($\exists$) test with optional negation.

3. AGG

AGG is a general-purpose augmented graph grammar engine that implements recursive graph matching. The system was developed in Python to support analysis of the student-produced argument diagrams described above. As such it is flexible, functions across platforms, and supports complex graph ontologies and user-defined functions. The system was designed in a modular fashion and can be linked with third-party libraries such as the NLTK [6].

At present the system uses a straightforward depth-first stack matching algorithm. Given a graph and a set of named *rules*, defined by a single graph class or expression, the system will first match all ground nodes and arcs in the leftmost *target* class. Once each ground element has been matched then the system will recursively match all variable elements in the target. If at any point the system cannot continue to match elements it will pop up the stack and repeat. Rule matching is governed by the aforementioned restrictions of expansiveness and non-repetition. If a rule is defined by a graph expression then each class match will set the context for subsequent rightmost matches. Rules defined by a single class are complete once a single match is found. The system is designed to find matches serially and can be called iteratively to extract all matching items.

In addition to basic graph grammars the AGG toolkit has the capacity to define named rules. These are named graph expressions or individual classes that will be recorded if they match. In my thesis work, I applied the AGG engine to develop a set of 42 such rules the scientific argument diagrams. These ranged in complexity from graph classes defined by a single node to more complex recursive expressions that sought to identify disjoint subgraphs and unsupported hypotheses. The example rules and expressions shown in figures 3 - 5 were adapted from this set. The rules were used for offline processing of the graphs and for prediction of student grades [10, 9].

As part of the analysis process the rules were evaluated on a set of 526 diagrams containing between 0 and 41 nodes each. While exact efficiency data was not retained the performance of the rules varied widely depending upon their construction. General recursive rules such as a test for disjoint subgraphs performed quite inefficiently while smaller chained expressions were able to evaluate in a matter of seconds on a quad-core system.

4. APPLICATIONS & FUTURE WORK

The focus of this paper was on introducing Augmented Graph Grammars and the AGG engine. The formalism provides for a natural and robust representation of complex graph rules for heterogeneous datasets. In prior work at the University of Pittsburgh I applied Augmented Graph Grammars to the detection of pedagogically relevant structures like *Related Uncompared Opposition* (see Figure 3) in argument diagrams of the type shown in Figure 1. The focus of that study was on testing whether student-produced argument diagrams are diagnostic of their ability to produce written argumentative essays. The study was conducted in a course on Psychological Research Methods at the University of Pittsburgh.

The graph features examined in that study included *chained counterarguments* which feature chains of oppositional information, and *ungrounded hypotheses* which are unrelated to cited works, and so on. The study is described in detail in [8], and a discussion of the empirical validity of the individual rules can be found in [9]. The rules were also used as the basis of predictive models for student grades described in [10]. The Augmented Graph Grammars were ideally-suited for this task as they allowed me to define clear and robust rules that incorporated the structural information in the graph, textual information within the nodes and arcs, and the static

element types. It was also possible to clearly present these rules to domain experts for evaluation.

While the AGG system is robust more work remains to be done to make it widely available, and several open problems remain for future development. As noted above, arbitrary graph parsing is NP-Hard. Consequently, many rule classes are extremely inefficient. Despite this limitation, however, real efficiency gains may be made via parallelization and memoization. I am presently researching possible improvements to the system and plan to test them with additional datasets.

Acknowledgments

This work was supported by National Science Foundation Award No. 1122504, "DIP: Teaching Writing and Argumentation with AI-Supported Diagramming and Peer Review," Kevin D. Ashley PI with Chris Schunn and Diane Litman, co-PIs.

5. REFERENCES

[1] Euijin Choo, Ting Yu, Min Chi, and Yan Lindsay Sun. Revealing implicit communities to incorporate into recommender systems. In *Proceedings of the* 15^{th} *ACM Conference on Economics and Computation*, Palo Alto, CA, 2014. Association For Computing Machinery. (in press).

[2] Evi Chryssafidou and Mike Sharples. Computer-supported planning of essay argument structure. In *Proceedings of the 5th International Conference of Argumentation*, June 2002.

[3] Diane J. Cook and Lawrence B. Holder, editors. *Mining Graph Data*. John Wiley & Sons, 2006.

[4] 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.

[5] Rosta Farzan and Peter Brusilovsky. Annotated: A social navigation and annotation service for web-based educational resources. *Journal of the New Review of Hypermedia and Multimedia (NRHM)*, 2008.

[6] Dan Garrette, Peter Ljunglöf, Joel Nothman, Mikhail Korobov, Morten Minde Neergaard, and Steven Bird. The natural language toolkit for python (NLTK), 2014. [Online; accessed 04-29-2014].

[7] Frank Loll and Niels Pinkwart. Lasad: Flexible representations for computer-based collaborative argumentation. *Int. J. Hum.-Comput. Stud.*, 71(1):91–109, 2013.

[8] Collin F. Lynch. The Diagnosticity of Argument Diagrams, 2014. (defended January 30th 2014).

[9] 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. (In Press).

[10] Collin F. Lynch, Kevin D. Ashley, and Min Chi. Can diagrams predict essays? In Stefan Trausan-Matu, Kristy Elizabeth Boyer, Martha E. Crosby, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, volume 8474 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2014.

[11] Sherry E. Marcus, Melanie Moy, and Thayne Coffman. Social network analysis. In Cook and Holder [3], chapter 17, pages 443–468.

[12] Takashi Okada. Mining from chemical graphs. In Cook and Holder [3], chapter 14, pages 347–379.

[13] Niels Pinkwart, Kevin D. Ashley, Vincent Aleven, and Collin F. Lynch. Graph grammars: An its technology for diagram representations. In David Wilson and H. Chad Lane, editors, *FLAIRS Conference*, pages 433–438. AAAI Press, 2008.

[14] Niels Pinkwart, Kevin D. Ashley, Collin F. Lynch, and Vincent Aleven. Evaluating an intelligent tutoring system for making legal arguments with hypotheticals. *International Journal of Artificial Intelligence in Education*, 19(4):401–424, 2009.

[15] Eve Powell and Tiffany Adviser-Barnes. A framework for the design and analysis of socially pervasive games. 2012.

[16] Eve M Powell, Samantha Finkelstein, Andrew Hicks, Thomas Phifer, Sandhya Charugulla, Christie Thornton, Tiffany Barnes, and Teresa Dahlberg. Snag: social networking games to facilitate interaction. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 4249–4254. ACM, 2010.

[17] J. Rekers and Andy Schürr. Defining and parsing visual languages with layered graph grammars. *J. Vis. Lang. Comput.*, 8(1):27–55, 1997.

[18] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, San Francisco, 1997.

[19] Daniel D. Suthers. Empirical studies of the value of conceptually explicit notations in collaborative learning. In Alexandra Okada, Simon Buckingham Shum, and Tony Sherborne, editors, *Knowledge Cartography*, pages 1–23. Springer Verlag, 2008.

Graph Mining and Outlier Detection Meet Logic Proof Tutoring

Karel Vaculík
Knowledge Discovery Lab
Faculty of Informatics
Masaryk University
Brno, Czech Republic
xvaculi4@fi.muni.cz

Leona Nezvalová
Knowledge Discovery Lab
Faculty of Informatics
Masaryk University
Brno, Czech Republic
324852@mail.muni.cz

Luboš Popelínský
Knowledge Discovery Lab
Faculty of Informatics
Masaryk University
Brno, Czech Republic
popel@fi.muni.cz

ABSTRACT

We introduce a new method for analysis and evaluation of logic proofs constructed by undergraduate students, e.g. resolution or tableaux proofs. This method employs graph mining and outlier detection. The data has been obtained from a web-based system for input of logic proofs built at FI MU. The data contains a tree structure of the proof and also temporal information about all actions that a student performed, e.g. a node insertion into a proof, or its deletion, drawing or deletion of an edge, or text manipulations. We introduce a new method for multi-level generalization of subgraphs that is useful for characterization of logic proofs. We use this method for feature construction and perform class-based outlier detection on logic proofs represented by these new features. We show that this method helps to find unusual students' solutions and to improve semi-automatic evaluation of the solutions.

Keywords

logic proofs, resolution, educational data mining, graph mining, outlier detection

1. INTRODUCTION

Resolution method is, together with tableaux proof method, one of the advanced methods taught in undergraduate courses of logic. To evaluate a student solution properly, a teacher needs not only to check the result of a solution (the set of clauses is or is not contradictory) but also to analyse the sequence of steps that a student performed—with respect to correctness of each step and with respect to correctness of that sequence. We need to take into account all of that when we aim at building a tool for analysis of students' solutions. It has to be said that for an error detection (e.g. resolution on two propositional letters, which is the most serious one) we can use a search method. However, detection of an error does not necessarily mean that the solution was completely incorrect. Moreover, by a search we can hardly discover patterns, or sequence of patterns, that are typical for wrong solutions.

To find typical patterns in wrong solutions, we developed a new method for analysis of students' solutions of resolution proofs [13, 14] and showed its good performance. Solutions were manually rewritten into GraphML and then analysed. First, the frequent patterns were found by Sleuth [16], which was suitable for this type of data—unordered rooted trees. This algorithm finds all frequent subtrees from a set of trees for a given minimum support value. Such frequent subgraphs were generalized and these generalizations used as new attributes.

The main drawback of a frequent subgraph mining algorithm itself is its strong dependence on a particular task, i.e. on the input set of clauses that has to be proved, or unproved, as contradictory. Moreover, a usage of such an algorithm is quite limited, because by setting the minimum support to a very small value, the algorithm may end up generating excessively many frequent subtrees, which consumes both time and space. The problem is that we wish to include the infrequent substructures as well because they often represent mistakes in students' solutions.

In this paper we propose a novel way of subgraph generalization that solves the problems mentioned above and is independent on the input set of clauses. We show that by means of graph mining and class outlier detection, we are able to find outlying students' solutions and use them for the evaluation improvement.

The structure of this paper is following. Section 2 brings related work. In Section 3 we introduce the source data. In Section 4 we introduce the improved method for construction of generalized resolution graphs. In Section 5 we bring the main result—detection of anomalous student solutions. Discussion and conclusion are in Sections 6 and 7, respectively.

2. RELATED WORK

Overview of graph mining methods can be found in [5]. Up to our knowledge, there is no work on analysis of student solutions of logical proofs by means of graph mining. Definitely, solving logic proofs, especially by means of resolution principle, is one of the basic graph-based models of problem solving in logic. In problem-solving processes, graph mining has been used in [15] for mining concept maps, i.e. structures that model knowledge and behaviour patterns of a student, for finding commonly observed subconcept structures. Combination of multivariate pattern analysis and hidden Markov models for discovery of major phases that students go through in solving complex problems in algebra is introduced in [1]. Markov decision processes for generating hints to students in logic proof tutoring from historical data has been solved in [2, 3, 12].

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

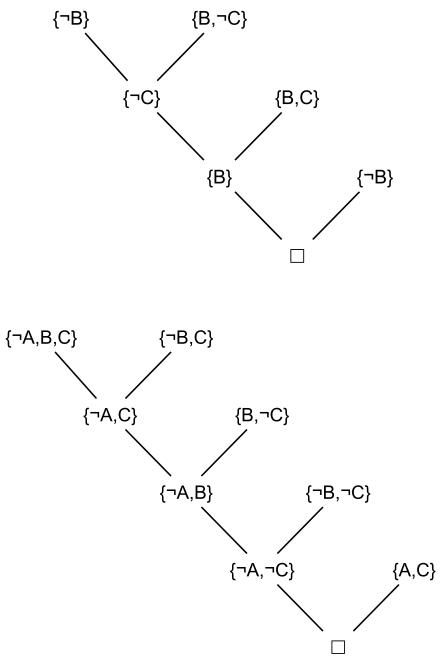


Figure 1: A correct and an incorrect resolution proof.

3. DATA

By means of a web-based tool, each of 351 students solved at least three tasks randomly chosen from 19 exercises. All solutions were stored in a PostgreSQL database. The data set contained 873 different students' solutions of resolution proofs in propositional calculus, 101 of them being incorrect and 772 correct. Two examples of solutions are shown in Fig. 1.

Common errors in resolution proofs are the following: repetition of the same literal in the clause, resolving on two literals at the same time, incorrect resolution—the literal is missing in the resolved clause, resolving on the same literals (not on one positive and one negative), resolving within one clause, resolved literal is not removed, the clause is incorrectly copied, switching the order of literals in the clause, proof is not finished, resolving the clause and the negation of the second one (instead of the positive clause). For each kind of error we defined a query that detects the error. For automatic evaluation we used only four of them, see Table `ERRORS` described in appendix A. As the error of resolving on two literals at the same time is very common and referred later in text, we denote this error as E3.

All actions that a student performed, like adding/deleting a node, drawing/removing an edge, writing/deleting a text into a node, were saved into a database together with time stamps. More details on this database and its tables can be found in appendix A.

In the data there were 303 different clauses occurring in 7869 vertices, see frequency distribution in Fig. 2. Approximately half of the clauses had absolute frequency less than or equal to three.

4. GENERALIZED SUBGRAPHS

In this section we describe feature construction from graph data. Representing a graph by values of its vertices and edges is insufficient as the structure of the graph also plays a significant role. Common practice is to use substructures of graphs as new features [5]. More specifically, boolean features are used and the value of a feature depends on whether the corresponding substructure occurs in the given instance or not.

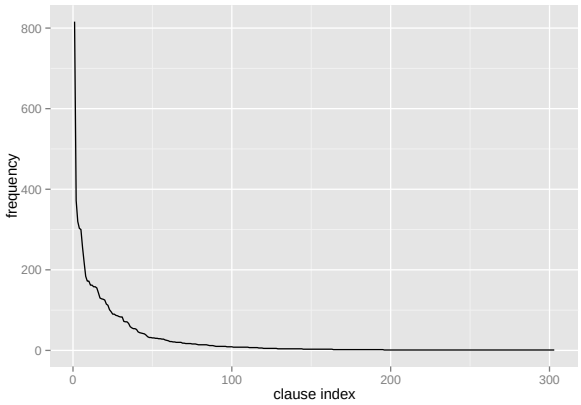


Figure 2: Distribution of clause labels ordered by frequency.

As we showed earlier, a frequent subgraph mining algorihm is inappropriate. To overcome the discussed problems, we created a new method for feature construction from our data. The idea of feature construction is to unify subgraphs which carry similar information but they differ in form. An example of two subgraphs, which differ only in variable letters and ordering of nodes and literals, is shown on the left side of Fig. 3. The goal is to process such similar graphs to get one unique graph, as shown in the same figure on the right. In this way, we can better deal with different sets of clauses with different sets of variable letters. To deal with the minimum-support problem, the algorithm for frequent subgraphs was left out completely and all 3-node subgraphs, which are described later, were looked up.

4.1 Unification on Subgraphs

To unify different tasks that may appear in student tests, we defined a unification operator on subgraphs that allows finding of so called *generalized subgraphs*. Briefly saying, a generalized subgraph describes a set of particular subgraphs, e.g., a subgraph with parents $\{A, \neg B\}$ and $\{A, B\}$ and with the child $\{A\}$ (the result of a correct use of a resolution rule), where A, B, C are propositional letters, is an instance of generalized graph $\{Z, \neg Y\}$, $\{Z, Y\} \rightarrow \{Z\}$, where Y, Z are variables (of type *proposition*). An example of incorrect use of resolution rule $\{A, \neg B\}$, $\{A, B\} \rightarrow \{A, A\}$ matches with the generalized graph $\{Z, \neg Y\}$, $\{Z, Y\} \rightarrow \{Z, Z\}$. In other words, each subgraph is an instance of one generalized subgraph. We can see that the common set unification rules [6] cannot be used here.

In this work we focused on generalized subgraphs that consist of three nodes, two parents and their child. Then each generalized subgraph corresponds to one way—correct or incorrect—of resolution rule application.

4.2 Ordering on Nodes

As a resolution proof is, in principal, an unordered tree, there is no order on parents in those three-node graphs. To unify two resolution steps that differ only in order of parents we need to define

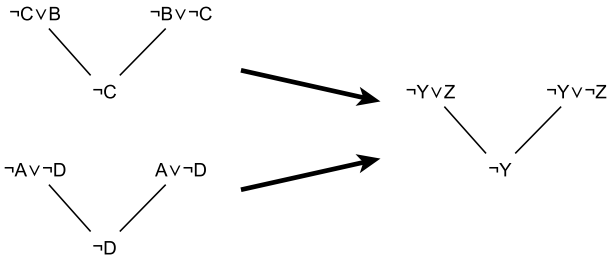


Figure 3: An example of pattern unification.

ordering on parent nodes[1]. We take a node and for each propositional letter we first count the number of negative and the number of positive occurrences of the letter, e.g., for $\{\neg C, \neg B, A, C\}$ we have these counts: (0,1) for A, (1,0) for B, and (1,1) for C. Following the ordering Ω defined as follows: $(X,Y) \leq (U,V)$ iff $(X < U \vee (X = U \wedge Y \leq V))$, we have a result for the node $\{C, \neg B, A, \neg C\}$: $\{A, \neg B, C, \neg C\}$ with description Δ = ((0,1), (1,0), (1,1)). We will compute this transformation for both parent nodes. Then we say that a node is smaller if the description Δ is smaller with respect to the Ω ordering applied lexicographically per components. Continuing with our example above, let the second node be $\{B, C, A, \neg A\}$ with Δ = ((0,1), (0,1), (1,1)). Then this second node is smaller than the first node $\{A, \neg B, C, \neg C\}$, since the first components are equal and (1,0) is greater than (0,1) in case of second components.

4.3 Generalization of Subgraphs

Now we can describe how the generalized graphs are built. After the reordering introduced in the previous paragraph, we assign variables Z,Y,X,W,V,U,... to propositional letters. To accomplish this, we initially merge literals from all nodes into one list and order it using the Ω ordering. After that, we assign variable Z to the letter with the smallest value, variable Y to the letter with the second smallest value, etc. If two values are equal, we compare the corresponding letters only within the first parent, alternatively within the second parent or child. For example, let a student's (incorrect) resolution step be $\{C, \neg B, A, \neg C\}, \{B, C, A, \neg A\} \rightarrow \{A, C\}$. We order the parents getting the result $\{B, C, A, \neg A\}, \{C, \neg B, A, \neg C\} \rightarrow \{A, C\}$. Next we merge all literals into one list, keeping multiple occurrences: $\{B, C, A, \neg A, C, \neg B, A, \neg C, A, C\}$. After reordering, we get $\{B, \neg B, C, C, C, \neg C, A, A, A, \neg A\}$ with Δ = ((1,1), (1,3), (1,3)). This leads to the following renaming of letters: $B \rightarrow Z$, $C \rightarrow Y$, and $A \rightarrow X$. Final generalized subgraph is $\{Z, Y, X, \neg X\}$, $\{Y, \neg Z, X, \neg Y\} \rightarrow \{X, Y\}$. In case that one node contains more propositional letters and the nodes are equal (with respect to the ordering) on the intersection of propositional letters, the longer node is defined as greater. At the end, the literals in each node are lexicographically ordered to prevent from duplicities such as $\{Z, \neg Y\}$ and $\{\neg Y, Z\}$.

4.4 Complexity of Graph Pattern Construction

Complexity of pattern generalization depends on the number of patterns and the number of literals within each pattern. Let r be the maximum number of literals within a 3-node pattern. In the first step, ordering of parents must be done, which takes $O(r)$ for counting the number of negative and positive literals, $O(r \log r)$ for sorting and $O(r)$ for comparison of two sorted lists. Letter substitution in the second step consists of counting literals on merged list in $O(r)$, sorting the counts in $O(r \log r)$ and renaming of letters in $O(r)$. Lexicographical reordering is performed in the last step and takes $O(r \log r)$. As construction of advanced generalized patterns is less complex than the construction of patterns mentioned above, we can conclude that the time complexity for whole generalization process on m patterns with duplicity removal is $O(m^2 + m(4r + 3r \log r))$.

[1]Ordering on nodes, not on clauses, as a student may write a text that does not correspond to any clause, e.g., $\{A, A\}$.

4.5 Higher-level Generalization

To improve performance of used algorithms, e.g. outlier detection algorithms, we created a new generalization method. This method can be viewed as a higher-level generalization as it generalizes the method described in previous paragraphs. This method uses domain knowledge about general resolution principle. It goes through all literals in a resolvent and deletes those which also appear in at least one parent. Each such literal is also deleted from the corresponding parent or parents in case it appears in both of them. In the next step, remaining literals in parents are merged into a new list *dropped* and remaining literals in the resolvent form another list, *added*. These two lists form a pattern of the higher-level generalization and we will write such patterns in the following format:

$$\underset{\text{(added)}}{\{L_{i_1}, L_{i_2}, ..., L_{i_n}\}}; \underset{\text{(dropped)}}{\{L_{j_1}, L_{j_2}, ..., L_{j_m}\}}$$

For example, if we take the generalized subgraph from the right side of Fig. 3, there is only one literal in the resolvent, $\neg Y$. We remove it from the resolvent and both parents and we get $dropped = [Z, \neg Z]$, $added = []$.

As a result, there may be patterns which differ only in used letters and order of literals in lists *dropped* and *added*. For this reason we then apply similar method as in the lower-level generalization. Specifically, we merge lists *dropped* and *added* and compute number of negative and positive literals for each letter in this new list. The letters are then ordered primarily according to number of occurrences of negative literals and secondly according to number of occurrences of positive literals. In case of tie we check ordering of affected letters only in *added* list and if needed, then also in *dropped* list. If tie occurs also in these lists, then the order does not matter. At the end, the old letters are one by one replaced by the new ones according to the ordering and the new lists are sorted lexicographically. For example, let $dropped = [X, \neg X]$, $added = [Y, Z, Z, \neg Z]$. Then $merged = [X, \neg X, Y, Z, Z, \neg Z]$ and number of occurrences can be listed as count(X, merged) = (1, 1), count(Y, merged) = (0, 1), count(Z, merged) = (1, 2). Ordering on letters can be expressed as $Y \leq X \leq Z$. Using letters from the end of alphabet we perform following substitution according to created ordering: $Y \rightarrow Z$, $X \rightarrow Y$, $Z \rightarrow X$. Final pattern will have lists $dropped = [\neg Y, Y]$, $added = [\neg X, X, X, Z]$, provided that $\neg$ sign is lexicographically before alphabetic characters. Examples of patterns with absolute support ≥ 10 are shown in Tab. 1.

4.6 Generalization Example

In this section we illustrate the whole generalization process by an example. Assume that the following 3-node subgraph has to be generalized:

Table 1: Higher-level patterns with support ≥ 10

Pattern (*added*;*dropped*)	Support
$\{\};\{\neg Z,Z\}$	3345
$\{\};\{\neg Y,\neg Z,Y,Z\}$	59
$\{\neg Z\};\{\neg Y,Y\}$	18
$\{\};\{\neg Z\}$	13
$\{\};\{\}$	10

$$P1 = \{\neg C,\neg A,\neg C,D,\neg D\}, P2 = \{\neg D,\neg A,D,C\} \rightarrow \{\neg A,A,\neg C\}$$

First, the parents are checked and possibly reordered. For each letter we compute the number of negative and positive literals in either parent. Specifically, count(A, P1) = (1,0), count(C, P1) = (2,0), count(D, P1) = (1,1), count(A, P2) = (1,0), count(C, P2) = (0,1), count(D, P2) = (1,1). Obtained counts are lexicographically sorted for both parents and both chains are lexicographically compared:

$$((1,0),(1,1),(2,0)) > ((0,1),(1,0),(1,1))$$

In this case, the result was already obtained by comparing the first two pairs, (1,0) and (0,1). Thus, the second parent is smaller and the parents should be switched:

$$P1' = \{\neg D,\neg A,D,C\}, P2' = \{\neg C,\neg A,\neg C,D,\neg D\} \rightarrow \{\neg A,A,\neg C\}$$

Now, all three nodes are merged into one list:

$$S = \{\neg D,\neg A,D,C,\neg C,\neg A,\neg C,D,\neg D,\neg A,A,\neg C\}$$

Once again, the numbers of negative and positive literals are computed: count(A, S) = (3,1), count(C, S) = (3,1), count(D, S) = (2,2). Since count(A, S) = count(C, S), we also check the counts in the first parent, P1'. As count(C, P1') = count(C, P2) < count(A, P2) = count(A, P1'), letter C is inserted before A. Finally, the letters are renamed according to the created order: $D \rightarrow Z, C \rightarrow Y, A \rightarrow X$. After the renaming and lexicographical reordering of literals, we get the following generalized pattern:

$$\{\neg X,\neg Z,Y,Z\},\{\neg X,\neg Y,\neg Y,\neg Z,Z\} \rightarrow \{\neg X,\neg Y,X\}$$

Next, we want to get also the higher-level generalization of that pattern. The procedure goes through all literals in the resolvent and deletes those literals that occur in at least one parent. This step results in a prunned version of the pattern:

$$\{\neg Z,Y,Z\},\{\neg Y,\neg Z,Z\} \rightarrow \{X\}$$

Parents from the pruned pattern are merged into a new list *dropped* and the resolvent is used in a list *added*. Thus, $added = \{X\}$ and $dropped = \{\neg Z,Y,Z,\neg Y,\neg Z,Z\}$. Now it is necessary to rename the letters once again. Lists *added* and *dropped* are merged together and the same subroutine is used as before—now the lists can be seen as two nodes instead of three. In this case, the renaming goes as follows: $X \rightarrow Z, Y \rightarrow Y, Z \rightarrow X$. At the end, literals in both lists are lexicographically sorted and the final higher-level pattern is:

$$\underset{\text{(added)}}{\{Z\}};\underset{\text{(dropped)}}{\{\neg X,\neg X,\neg Y,X,X,Y\}}$$

4.7 Use of Generalized Subgraphs

This section puts all the information from previous sections together and describes how generalized patterns are used as new features. Input data in form of nodes and edges are transformed into attributes of two types. Generalized patterns of the lower level can be considered as the first type and the patterns of higher-level generalization as the second type. One boolean attribute is created for each generalized pattern. Value of such attribute is equal to $TRUE$, if the corresponding pattern occurs in the given resolution proof, and it is equal to *FALSE* otherwise. Thus following this procedure, the resolution proofs can be transformed into an attribute-value representation as shown in Table 2. Such representation allows us to use a lot of existing machine learning algorithms.

Table 2: Attribute-value representation of resolution proofs

Instance	$Pattern_1$	$Pattern_2$	...	$Pattern_m$
1	TRUE	FALSE	...	FALSE
...	...	...	...	...
n	FALSE	FALSE	...	TRUE

5. OUTLIER DETECTION

5.1 Mining Class Outliers

In this section we present the main result, obtained from outlier detection. We observed that student creativity is more advanced than ours, and that results of the queries for error detection must be used carefully. Detection of anomalous solutions—either abnormal, with picturesque error, or incorrectly classified—helps to improve the tool for automatic evaluation, as will be shown later.

Here we focus only on outliers for classes created from error E3, the resolution on two literals at the same time, as it was the most common error. This means that the data can be divided into two groups, depending whether the instances contain error E3 or not. For other types of errors, the analysis would be similar. We also present only results computed on higher-level generalized patterns. The reason is that they generally achieved much higher outlier scores than lower-level patterns.

The data we processed had been labeled. Unlike in common outlier detection, where we look for outliers that differ from the rest of "normal" data, we needed to exploit information about a class. That is why we used weka-peka [9] that looks for class outliers [8, 10] using Random Forests (RF) [4]. The main idea of weka-peka lies in different computation of proximity matrix in RF—it also exploits information about a class label [9]. We used the following settings:

```
NumberOfTrees=1000
NumberOfRandomFetaures=7
FeatureRanking=gini
```

Table 3: Top outliers for data grouped by error E3

instance	error E3	outlier score	significant patterns [(AScore) *added*;*dropped*]	significant missing patterns [(AScore) *added*;*dropped*]
270	no	131.96	(0.96) *looping*	(-0.99) $\{\};\{\neg Z,Z\}$
396	no	131.96	(0.96) *looping*	(-0.99) $\{\};\{\neg Z,Z\}$
236	no	73.17	(0.99) $\{\};\{\neg Y,\neg Z,Y\}$	
187	no	61.03	(0.99) $\{\neg Z\};\{\neg Y,Y\}$ (0.99) $\{\};\{\neg Y,\neg Z,Y\}$	
438	yes	54.43	(1.00) $\{Z\};\{\neg X,\neg Y,X,Y\}$	(-0.94) $\{\};\{\neg Y,\neg Z,Y,Z\}$
389	yes	52.50	(1.00) $\{\};\{\neg Y,\neg Z,Y\}$	(-0.94) $\{\};\{\neg Y,\neg Z,Y,Z\}$ (-0.81) $\{\};\{\neg Z,Z\}$
74	yes	15.91	(0.98) $\{\neg Z\};\{\neg X,\neg Y,X,Y\}$ (0.98) $\{\};\{\neg X,\neg Y,\neg Z,X,Y,Z\}$	(-0.94) $\{\};\{\neg Y,\neg Z,Y,Z\}$
718	yes	15.91	(0.98) $\{\neg Z\};\{\neg X,\neg Y,X,Y\}$ (0.98) $\{\};\{\neg X,\neg Y,\neg Z,X,Y,Z\}$	(-0.94) $\{\};\{\neg Y,\neg Z,Y,Z\}$

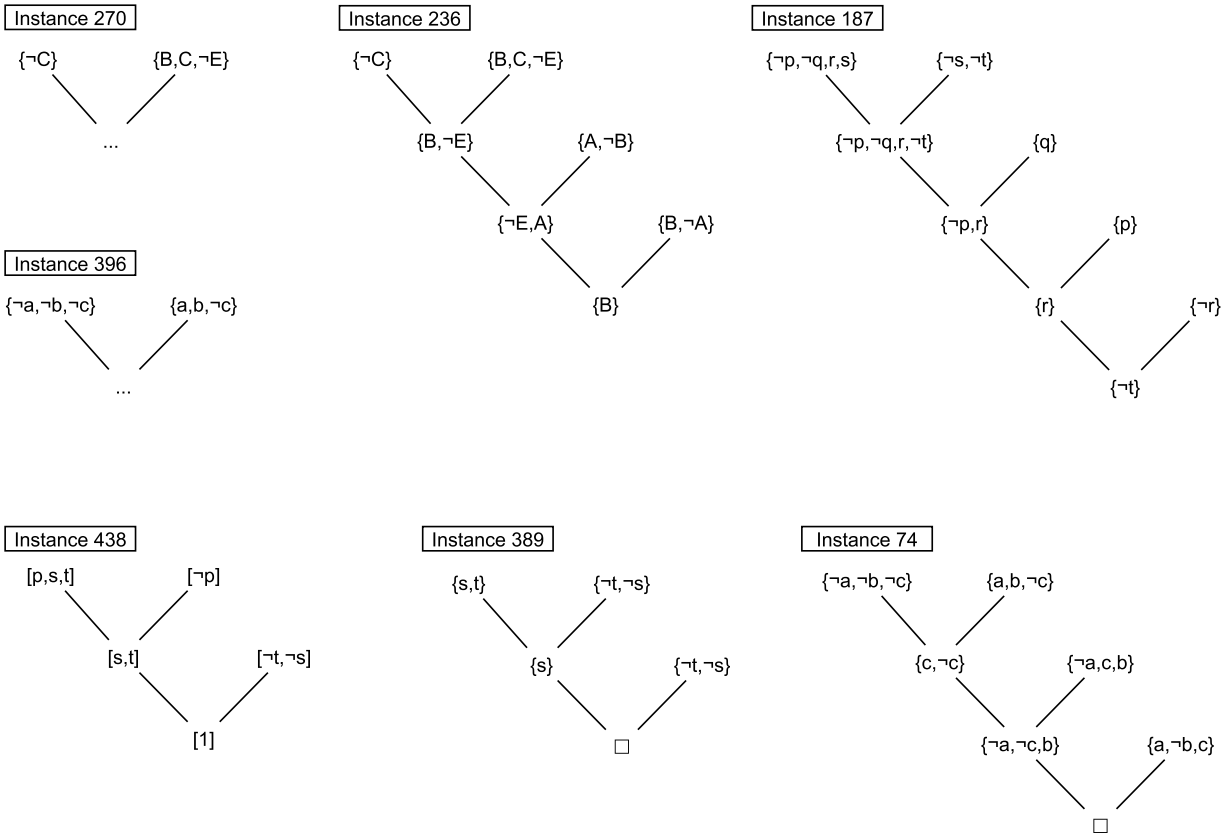


Figure 4: Drawings of the outlying instances from Table 3.

Table 4: Classification results for frequent subgraphs

Used attributes	Algorithm	Accuracy [%]	Precision for incorrect proofs	Recall
low-level generalization	SVM (SMO)	*95.2	0.94	0.61
both levels of generalization	SVM (SMO)	*96.9	0.95	0.74
both levels of generalization	J48	96.1	*0.98	0.68
both levels of generalization E3	J48	*95.4	0.87	0.72

```
MaxDepthTree=unlimited
Bootstrapping=yes
NumberOfOutliersForEachClass=50
```

Main results of outlier detection process are summarized in Table 3. When analyzing the strongest outliers that weka-peka found, we can see that there are three groups according to the outlier score. The two most outlying examples, instances numbered 270 and 396, significantly differ from the others. The second cluster consists of four examples with the outlier score between 50 and 100, and the last group is comprised of instances with the lowest score of 15.91.

As weka-peka is based on Random Forest, we can interpret an outlier by analyzing trees that classify given instance to a different class than it was labeled. Such trees show which attribute or combination of attributes lead to the resulting class. If we search for repeating patterns in those trees, we can find the most important attributes making the given instance an outlier. Using this method to interpret the instance 270, we found out that high outlier score is caused by not-applying one specific pattern (see Table 3). When setting this attribute equal TRUE, outlier score decreases to -0,40. Values of attributes of instances 396 and 270 are equal, it means that also interpretation is the same as in previous case. Similary, we found that outlierness of instance 236 is given by occurence of specific pattern in solution and non-occurence of another pattern. The value of the corresponding attribute is the only difference between instance 236 and 187. Occurence/non-occurence of this pattern is therefore the reason why instance numbered 236 achieves higher outlier score than instance 187. See again Table 3 for information about particular patterns. We further elaborated this approach of outlier explanation in the following section.

5.2 Finding Significant Patterns

As the outlier score is the only output information about the outliers, we created a simple method for finding the attributes with the most unusual values. Let x_{ij} denote the value of the jth attribute of the ith instance, which is either $TRUE$ or $FALSE$ for the pattern attributes, and $cl(i)$ denote the class of the ith instance. Then for instance i we compute the score of attribute j as:

$$AScore(i,j) = \begin{cases} \frac{|\{k|k\neq i \wedge cl(i)=cl(k) \wedge x_{kj}=FALSE\}|}{|\{k|k\neq i \wedge cl(i)=cl(k)\}|} & \text{if } x_{ij} = TRUE \\ -\frac{|\{k|k\neq i \wedge cl(i)=cl(k) \wedge x_{kj}=TRUE\}|}{|\{k|k\neq i \wedge cl(i)=cl(k)\}|} & \text{if } x_{ij} = FALSE \end{cases}$$

AScore expresses the proportion of other instances from the same class which have different value of the given attribute. If outlier's attribute equals *FALSE*, then the only difference is in the sign of the score. For example, consider our data set of 873 resolution proofs, out of which 53 proofs contain error E3. Assume that one of the 53 proofs is an outlier with an attribute equal to $TRUE$ and from the rest of 52 proofs only two proofs have the same value of this attribute as the outlier. Then the outlier's AScore on this attribute is approximately 50/52 = 0.96 and it indicates that the value of this attribute is quite unusual.

In general, the AScore ranges from -1 to 1. If the outlier resolution graph contains a pattern which is unique for the class of the graph, then the AScore of the corresponding attribute is equal to 1. On the other hand, if the outlier misses a pattern and all other graphs contain it, then the AScore is equal to -1. An AScore equal to 0 means that all other instances are equal to the outlier on the specified attribute.

5.3 Interpretation of the Outliers

Using the AScore metrics we found the patterns which are interesting for outliers in Table 3. Patterns, with AScore > 0.8 are listed in the *significant patterns* column and patterns with AScore < -0.8 in the *significant missing patterns* column.

All outliers from Table 3, except for the last one as it is almost identical to the penultimate one, are also displayed in Fig. 4. Analysis of individual outliers let us draw several conclusions. Let us remind that higher-level patterns listed in Table 3 are derived from lower-level patterns consisting of three nodes, two parents and one resolvent, and that the component *added* simply denotes literals which were added erroneously to the resolvent and the component *dropped* denotes literals from parents which participated in the resolution process. Two most outlying instances, numbered 270 and 396, also contain one specific pattern, *looping*. This pattern represents the ellipsis in a resolution tree, which is used for tree termination if the tree cannot lead to a refutation. Both instances contain this pattern, but neither of them contains the pattern of correct usage of the resolution rule, which is listed in the *significant missing patterns* column. The important thing is that these two instances do not contain error E3, but also any other error. In fact, they are created from an assignment which always leads to the *looping* pattern. This shows that it is not sufficient to find all errors and check the termination of proofs, but we should also check whether the student performed at least few steps by using the resolution rule. Otherwise we are not able to evaluate the student's skills. Moreover, there may be situations in which a student only copies the solution.

Instances with the outlier score less than 100 are less different from other instances. In particular, instances number 236 and 187 are more similar to correct resolution proofs than the instances discussed above. Yet, they both contain anomalous patterns such as $\{\}; \{\neg Y, \neg Z, Y\}$. This particular error pattern does not indicate error E3, as can be seen in Table 3. It is actually not marked as any type of error, which tells us that it is necessary to extend our list of potential errors in the automatic evaluator.

Continuing with outlier instances we get to those which contain error E3. Two of them exceed the boundary of outlier score 50, which suggests that they are still relatively anomalous. The first outlier, instance number 438, differ from other instances in an extra literal which was added into a resolvent. Specifically, the number 1, which is not even a variable, can be seen at the bottom of the resolution proof in Fig. 4. More interesting is the second instance with number 389. Error E3 was detected already in the first step of resolution, specifically when resolved on parents $\{s,t\}$ and $\{\neg t, \neg s\}$. This would not be a strange thing, if the resolvent was not s. Such a resolvent raises a question whether it is an error of type E3 or just a typing error. The latter is a less serious error.

Last two outliers in the table are almost the same so only the instance number 74 is depicted in Fig. 4. These two instances have quite low outlier score and they do not expose any shortcomings of our evaluation tool. Yet, they exhibit some outlying features such as resolving on three literals at the same time.

6. DISCUSSION

As we observed it is not sufficient to detect only the errors but we need to analyze a context in which an error appeared. Moreover, there are solutions that are erroneous because they do not contain a particular pattern or patterns. Outlier detection helped to find wrong students' solutions that could not be detected by the system

of queries even though the set of queries has been carefully built and tested on the test data. We also found a situation when a query did not detected an error although it appeared in the solution. We are convinced that with increasing number of solutions we will be able to further increase performance of wrong solution detection.

As we stressed in the introduction, this method has not been developed for recognition of correct or incorrect solutions. However, to verify that the feature construction is appropriate, we also learned various classifiers of that kind. In previous work we used only generalized patterns as attributes for classification with allerrors class attribute. However, these patterns were not sufficient for our current data. Repeating the same experiments we got the best result for SMO Support Vector Machines from Weka [7], which had 95.2% accuracy, see Table 4. Precision and recall for the class "incorrect" were 0.94 and 0.61, respectively. Minimum support for pattern selection was 0% in this case. To improve performance of classification we used the new level of generalization. Using the same settings, but now with both levels of generalized patterns, we achieved 96.9% accuracy, 0.95 precision and 0.74 recall for the class "incorrect". Similar results were obtained when only the new level of generalization was used, again with SMO. When ordered according to precision, value 0.98 was achieved by J48, but the accuracy and recall were only 96.1 and 0.68, respectively.

As one of the most common errors in resolution proofs is usage of resolution rule on two pairs of literals at the same time, we repeated the experiment, but now discarding all patterns capturing this specific kind of error. In this scenario the performance slightly dropped but remained still high—J48 achieved 95.4% accuracy, 0.87 precision and 0.72 recall. For the sake of completeness, F1 score for the class "correct" varied between 0.97 and 0.99 in all the results above.

We also checked whether inductive logic programming (ILP) can help to improve the performance under the same conditions. To ensure it, we did not use any domain knowledge predicates that would bring extra knowledge. For that reason, the domain knowledge contained only predicates common for the domain of graphs, like node/3, edge/3, resolutionStep/3 and path/2. We used Aleph system [11]. The results were comparable with the method described above.

7. CONCLUSION AND FUTURE WORK

In this paper we introduced a new level of generalization method for subgraphs of resolution proof trees built by students. Generalized subgraphs created by this special graph mining method are useful for representation of logic proofs in an attribute-value fashion. We showed how a class-based outlier detection method can be used on these logic proofs by utilization of the generalized subgraphs. We also discussed how the outlying proofs may be used for performance improvement of our automatic proof evaluator. This method may also be used for other types of data such as tableaux proofs.

As a future work we are going to analyse the temporal information, which was saved together with the structural information of logic proofs.

ACKNOWLEDGEMENTS

This work has been supported by Faculty of Informatics, Masaryk University and the grant CZ.1.07/2.2.00/28.0209 Computer-aided-teaching for computational and constructional exercises.

8. REFERENCES

[1] J. R. Anderson. Discovering the structure of mathematical problem solving. In *Proceedings of EDM*, 2013.

[2] T. Barnes and J. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pages 373–382, 2008.

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

[4] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[5] D. J. Cook and L. B. Holder. *Mining Graph Data*. John Wiley & Sons, 2006.

[6] A. Dovier, E. Pontelli, and G. Rossi. Set unification. *CoRR*, cs.LO/0110023, 2001.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[8] N. Hewahi and M. Saad. Class outliers mining: Distance-based approach. *International Journal of Intelligent Technology*, 2.

[9] Z. Pekarcikova. Supervised outlier detection, 2013. `http://is.muni.cz/th/207719/fi_m/diplomova_praca_pekarcikova.pdf`.

[10] P. Spiros and F. Christos. Cross-outlier detection. In *Proceedings of SSTD*, pages 199–213, 2003.

[11] A. Srinivasan. The Aleph Manual, 2001. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/` [Accessed: 2014-01-09].

[12] 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.

[13] K. Vaculik and L. Popelinsky. Graph mining for automatic classification of logical proofs. In *Proceedings of the 6th International Conference on Computer Supported Education CSEDU 2014*, 2014.

[14] K. Vaculik, L. Popelinsky, E. Mrakova, and J. Jurco. Tutoring and automatic evaluation of logic proofs. In *Proceedings of the 12th European Conference on e-Learning ECEL 2013*, pages 495–502, 2013.

[15] J. S. Yoo and M. H. Cho. Mining concept maps to understand university students' learning. In *Proceedings of EDM*, 2012.

[16] M. J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundam. Inf.*, 66(1-2):33–52, Jan. 2005.

APPENDIX
A. DESCRIPTION OF DATA

CLAUSE – list of nodes from all graphs
. idclause – ID of the node
. coordinatex – x position in drawing
. coordinatey – y position in drawing
. timeofcreation – when the node was created
. timeofdeletion – when the node was deleted (if not deleted, value is "NA")
. idgraph – in which graph the node appears
. text – text label

EDGE – list of (directed) edges from all graphs
. idedge – ID of the edge
. starting – ID of the node from which this edge goes
. ending – ID of the node to which this edge goes
. timeofcreation
. timeofdeletion
. idgraph

ERRORS – errors found in resolution graphs (found by means of SQL queries)
. idgraph – ID of the graph
. error3 – resolving on two literals at the same time (1 = error occurred, 0 = not occurred)
. error4 – repetition of the same literal in a set
. error5 – resolving on identical literals
. error8 – no resolution performed, only union of two sets
. allerrors – any of the previously listed errors occurred / not occured

GRAPH – list of graphs
. idgraph – ID of the graph
. logintime – start of graph creation
. clausetype – either set or ordered list
. resolutiontype – type of resolution, encoded by numbers (see table RESOLUTIONTYPES)
. assignment – textual assignment of task
. endtime – end of graph creation

MOVEMENT – list of coordinate changes of nodes
. idmovement – ID of the change
. idclause – ID of the node whose coordinates were changed
. coordinatex – new x coordinate
. coordinatey – new y coordinate
. time – time of the change

RESOLUTIONTYPES – encoding of resolution types
. typeid – ID (numeric encoding)
. typetext – textual value

TEXT – list of text (label) changes of nodes.
. idtext – ID of the change
. idclause – ID of the node whose text label was changed
. time – time of the change
. text – new text (label) value

TYPES – list of resolution type and clause type changes
. idtypes – ID of the change
. resolutiontype – new value of resolution type for specific graph
. clasetype – new value of clause type for specific graph
. timeofchange – time of the change
. idgraph – ID of the graph whose values were changed

Snag'em: Graph Data Mining for a Social Networking Game

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

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

Collin Lynch
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
cflynch@ncsu.edu

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

ABSTRACT

New conference attendees often lack existing social networks and thus face difficulties in identifying relevant collaborators or in making appropriate connections. As a consequence they often feel disconnected from the research community and do not derive the desired benefits from the conferences that they attend. In this paper we discuss Snag'em, a social network game designed to support new conference attendees in forming social connections and in developing an appropriate research network. Snag'em has been used at seven professional conferences and in four student settings and is the subject of active research and development. The developers have sought to make the system engaging and competitive while preventing players from 'gaming' it and thus accruing points while neglecting to form real-world connections. We briefly describe the system itself, discuss its impact on users, and describe our ongoing work on the identification of critical *hub* players and important social networks.

Keywords

Social Networks, Gamification, Conferences, Underrepresented Populations

1. INTRODUCTION

Social networking is an essential task at any academic conference or professional venue. One of the primary goals of attendees is to seek out relevant work, identify potential collaborators, and to maintain existing connections. Many of these contacts are made by building upon existing relationships and by expanding the attendees existing social network. New conference goers however, particularly students and historically underrepresented groups, lack these foundational networks and thus face difficulties making connections. Based on Tinto's Theory of University Departure, increased interaction with other students, faculty, staff and community supporters can increase the retention rate of minority populations and sense of community within secondary and post-secondary academic communities [7].

In academia, sense of community has a strong positive correlation with retention [7]. Research indicates that students who do not feel as if they are part of a larger academic community are less likely to participate in extracurricular activities and organizations. This leads to lower retention rates, especially amongst minority students who suffer without a strong student support group [7]. A feeling of community can be nurtured with small group activities that augment the individual's role within a setting and helps students to foster connections [8].

Snag'em was designed as a pervasive game to encourage valuable professional networking and promote sense of community. The system's pervasive features are designed to help players translate their in-game networks directly into real world peer groups. The system was originally created for the 2009 Students and Technology Academia Research & Service (STARS) conference. This conference is unusual in that it is an academic conference designed specifically to engage with minority and female undergraduates majoring in computing fields. Students who attend the conference participate in competitions and attend training sessions to support engagement and research. Studies conducted at prior conferences has shown that while students were engaged in the training sessions and vigorously involved in learning they did not develop the lasting social connections that can arise out of conferences. Snag'em was designed to engage students in social networking through gamification of the process. Prior research has shown that social games can help people to engage in otherwise challenging or uncomfortable situations [6, 4, 2, 3].

Snag'em functions as a large human scavenger hunt. Players are assigned a set of relevant tags (e.g. "I'm a games researcher", or "I'm interested in data-mining"). They are

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

Figure 1: The browser interface for mission assignments. Snag Snapshots highlight missions recently completed.

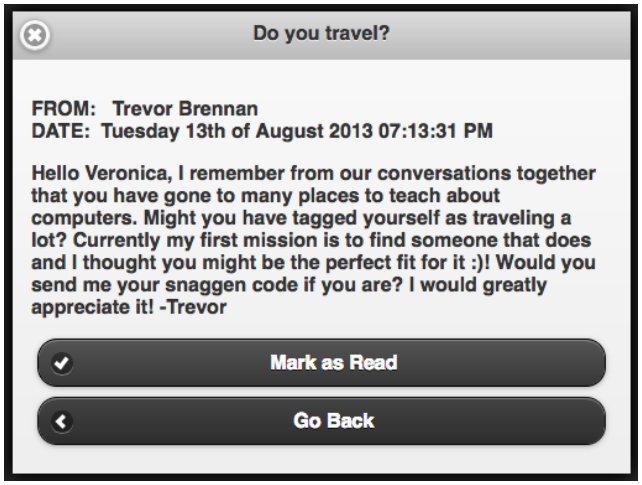


Figure 2: Here is an example of a message sent in game after a conversation between players.

then assigned a set of missions (e.g. "Find someone who specializes in HCI") which they must complete by identifying and engaging with an appropriate individual. The system was developed in PHP with a MySQL backed and provides a web-based front end for players to edit their profile and to record interactions. We have also developed a mobile version of Snag'em which allows players to access the game via tablets and smartphones. The game itself is designed for easy deployment to new conferences and we are presently adding features that will allow us to automatically populate the database with initial tags.

Figure 1 shows a snapshot of the mission browser screen from the web version of Snag'em. Contact is registered when the players enter a 4-digit ID from the other person. In addition to missions the systems also allows players to record notes about one-another for future reference (e.g. "I should e-mail my proposal to him after the conference") and to send one-another messages. A sample message from the mobile interface is shown in Figure 2. Snag'em can also be configured to suggest specific individuals that students should make contact with based upon their mutual interests or social connections.

The system logs all player interactions including tag updates, missions completed, notes made, messages, sent, connections added, and so on. This provides a rich dataset of information that we can use to analyze social patterns at conferences and to improve the impact of the intervention. In addition to the raw logs the game contains a number of features to support easy analysis. The developers have created a set of badges that allowed administrators to easily track the number of people playing via the mobile or web interfaces as well as the number of missions completed. The badge system also provides a simple visual record of the types of features (i.e. notes, tags, avatars) each player is using. The badge systems also allows administrators to note the frequency of use, time of day that players are online and so on.

To date, Snag'em has been used at seven academic conferences. It has also been deployed to help incoming freshman and transfer students connect at four academic institutions. In 2009, for example, Snag'em was used by new students in the College of Computing and Informatics at the University of North Carolina at Charlotte. Students were able to play the game during the freshman orientation week with kiosks available for students to sign up located in the College of Computing and Informatics. SNAG'EM was used alongside other social activities to get students acquainted with each other, the faculty, and the CCI campus.

2. PRIOR ANALYSIS

We have studied the impact of Snag'em on users and found that playing the game improved conference attendees' sense of community [6, 1]. We have also analyzed the existing dataset both to test the implementation of the Snag'em features, and to identify *hubs* or critical players whose activity predicts the behavior of others.

In analyzing the game mechanisms we have focused primarily on the STARS 2009 dataset. As mentioned above STARS is primarily targeted at undergraduate students specifically females and underrepresented minorities. We deployed the system via the conference infrastructure and set up a table near the registration booth. The game was active during the first two full days of the conference. The conference had 280 attendees 60.0% of whom were female (N=168) and 70% of which (N=196) were students. Roughly 28% of the conference-goers played the game (N=80) of whom 50% were female. In previous analysis 35.0% of the players were classified as active. It is important to note that this data was collected on an earlier version of SNAG'EM where players could snag each other only once, and only a single mission was available at a time. Because completing missions was significantly more difficult in this version of the game, players were classified as active if they completed at least two missions. An additional 50% of the players were classified as Interested, meaning they did more than just register for the game or that they completed one mission.

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

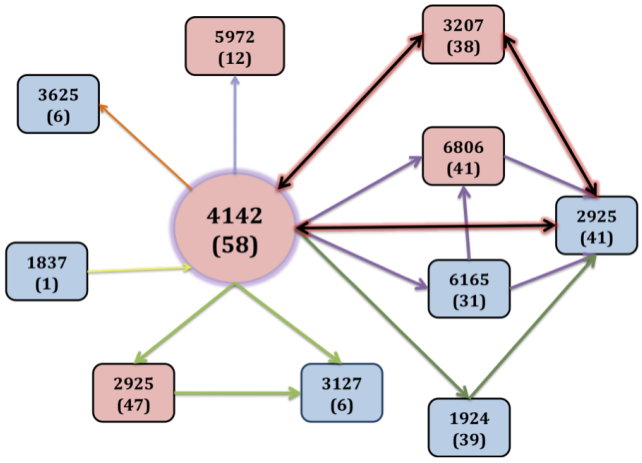


Figure 3: Visualization of community center 4142, with one of that user's maximal cliques highlighted.

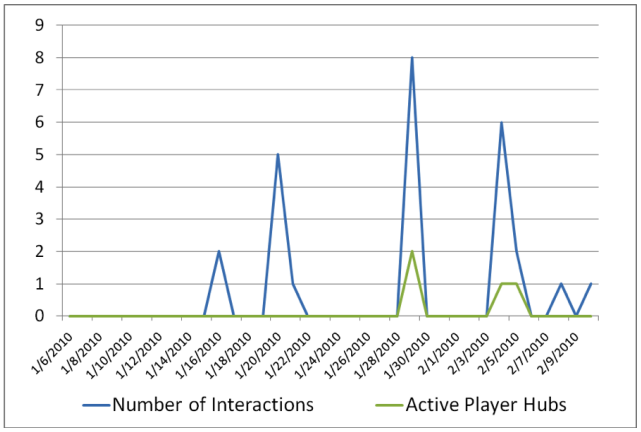


Figure 4: Correlation between active player hubs and number of interactions.

Our analysis of this data was focused primarily on the mission and scoring systems. In 2009 the mission system was relatively simple and focused solely on guiding students to locate a single individual with a desired tag. Players were then guided to record the match via the ID system discussed above. Both the missions generated and points received were determined by the state of the current network. When generating missions we attempted to ensure that they were of varying difficulty, and were relevant to the current user. In this iteration of the system the missions could only be satisfied by identifying someone whom the user had not previously snagged. The target tags were selected from the full set listed in the system. Easy missions were assigned high frequency tags (more than $\frac{1}{2}$ of the non-adjacent users), while medium missions were assigned tags that are present in $\frac{1}{4}$ of non-adjacent users and hard missions required tags present in less than $\frac{1}{4}$ of the non-adjacent community.

The difficulty of the mission determined the base score which was then modified by a *connectedness factor*. This factor was greater than 1 if adding this connection expanded your "Friends of friends," that is, the number of vertices less than 2 edges distant from the user. The connectedness factor was less than 1 if you completed the mission using the ID of a person you were already adjacent to, In this way we hoped to encourage players to branch out.

When developing the system we had hoped that players would develop social networks that exhibited breadth (i.e. meeting lots of people), depth (i.e. getting to know some individuals well), and mutuality (i.e. snags in both directions). We therefore hoped that users' immediate neighborhoods would be large and relatively dense with multiple snags between some people and bidirectional connections. When analyzing the STARS 2009 dataset, however, we found that this was not the case. Rather the game mechanics encouraged players to make a relatively large number of unrelated connections which, in turn, produced relatively broad and shallow social neighborhoods with very few inbound arcs. In fact some players actually opted to hide their IDs so that no other player could gain points by using them to complete a mission. As a consequence the attendees were actually less likely to engage in the deep and meaningful conversations required or to form lasting connections.

In response to these results we have overhauled the scoring system. This included changing the connectivity bonus to reward players based upon the size of the largest clique that they participate in. Players are now rewarded more for expanding this clique, thus deepening their social networks, than they are for adding an unrelated individual to their friends of friends. We have also allowed players to re-snag the same individual for multiple missions with a low penalty for re-snags, and have begun to reward players with points for allowing themselves to be snagged to help others complete a mission. We have not yet analyzed the effects of these changes on a the dataset.

We have used two measures of importance when identifying critical players. The first is the simple interaction frequency as measured by the number of outgoing arcs from a player in the network. The second is membership in maximal cliques, that is, cliques which are not part of a larger clique. Players that participate in a large number of maximal cliques are *hubs*. We were able to identify three distinct user communities in the STARS 2009 dataset that centered on these hubs. A sample community graph is shown in Figure 3. We also found that the activity of these hub players was highly correlated with the activity of the other players in the community (r=0.827). A graph of these spikes is shown in Figure 4. More specifically, on any day where one or more of the hub players were active, we observed spikes in the number of interactions taking place across users. We were able to observe a similar effect (r = 0.659) on days when the developers had a booth/kiosk available.

We also performed an analysis of hub players using the UNCC Student Orientation dataset described above. In this dataset 91 of the 1290 potential students registered to play Snag'em of which 22% (N=20) were female [5]. This data was collected on a version of Snag'em permitting multiple missions and allowing players to connect with the same user multiple times.We classified players as active if they completed 5 or more missions. In total, 9 users were active users during this study. However, all of these players were

moderators or members of the development team. In this deployment almost all of the game interaction took place at the registration table thus making the administrators responsible for most of the activity. We had hypothesized that the moderators would only need to initiate the game and then it would be self-sustaining. As our analysis shows however, this was not the case. In general the players did not think about the game outside of the advertised area.

3. OPEN QUESTIONS & FUTURE WORK

Our prior research has focused on identifying key players using graph methods. We plan to continue examining these key players in future work and to modify the mission selection criteria to better engage players that have not been active recently. Our chosen method of community detection, based upon maximal cliques, is both computationally expensive on large networks and can change substantially based upon small shifts in the network. Using a simpler, less volatile measure to identify community centers would allow us to adapt the gameplay based upon those communities more efficiently. This would in turn enable us to encourage new players to specifically seek out these active players in an effort to better engage them from the start. Different community detection algorithms might identify different hub players, or provide different ways of scoring missions that help to foster larger communities. Further development in this area might facilitate play in the absence of an instigating 'active player' or outside of areas with an active game station or kiosk.

One open question is how to better identify hub players during the game, and modify mission selection criteria to engage inactive players or players who don't need motivation to network. These 'social elites' are important to attract, as they are precisely who we should be encouraging our players to network with. If we are better able to build and analyze our networks, we may be able to offer features to these social elites that would attract them to Snag'em as a system more than the gamification aspects would. We hope to explore techniques for reliably generating edges and tags for users based on existing data sources like conference proceedings or citations. This would reduce the burden of entry on new players, particularly elites, and make it more likely for those users to participate in networking (if not gameplay) using SNAG'EM.

We also plan to expand our in-game evaluation of Snag'em itself. We are presently adapting the system to poll players for their opinions as the system is used. This will better help us to identify the immediate impact of the system on users' social connections. We will be deploying some of these new features of the system during the 2014 Educational Datamining Conference in London as well as subsequent conferences in 2014 and 2014.

4. ACKNOWLEDGMENTS

This research was supported by the NSF GRFP Fellowships No. 0900860 & No. 1252376 and BPC Grant No. 0739216 and No. 1042468 Thanks to all developers who have worked on the SNAG'EM project. The authors also wish to thank Shaghayegh Sahebi for her expert advice.

5. REFERENCES

[1] S. L. Finkelstein, E. Powell, A. Hicks, K. Doran, S. R. Charugulla, and T. Barnes. Snag: using social networking games to increase student retention in computer science. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 142–146. ACM, 2010.

[2] M. Montola. Exploring the edge of the magic circle: Defining pervasive games. In *Proceedings of DAC*, page 103, 2005.

[3] M. Montola. A ludological view on the pervasive mixed-reality game research paradigm. *Personal and Ubiquitous Computing*, 15(1):3–12, 2011.

[4] E. Powell and T. Adviser-Barnes. A framework for the design and analysis of socially pervasive games. 2012.

[5] E. Powell, F. Stukes, T. Barnes, and H. R. Lipford. Snag'em: Creating community connections through games. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 591–594. IEEE, 2011.

[6] E. M. Powell, S. Finkelstein, A. Hicks, T. Phifer, S. Charugulla, C. Thornton, T. Barnes, and T. Dahlberg. Snag: social networking games to facilitate interaction. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 4249–4254. ACM, 2010.

[7] V. Tinto. Taking Student Retention Seriously: Rethinking the First Year of College. *NACADA Journal*, 19(2):5–10, 2000.

[8] S. White. Algorithms for estimating relative importance in networks. *Proceedings of the ninth ACM SIGKDD international*, pages 266–275, 2003.

Social Positioning and Performance in MOOCs

Suhang Jiang
School of Education
University of California, Irvine
Irvine, CA 92697
suhangj@uci.edu

Sean M. Fitzhugh
Department of Sociology
University of California, Irvine
Irvine, CA 92697
sean.fitzhugh@uci.edu

Mark Warschauer
School of Education
University of California, Irvine
Irvine, CA 92697
markw@uci.edu

ABSTRACT
Literature indicates that centrality is correlated with learners' engagement in MOOCs. This paper explores the relationship between centrality and performance in two MOOCs. We found one positive and one null correlation between centrality and grade scores at the end of the MOOCs. In both MOOCs, we found out that learners tend to communicate with learners in different performance groups. This suggests that MOOCs' discussion forum serves to facilitate information flow and help-seeking among learners.

Keywords

MOOCs; Social Positioning; Performance

1. INTRODUCTION
Massive Open Online Courses (MOOCs) have attracted over 7 million users in the past two years. In addition to offering videos and online quizzes that users can watch and take, a key feature of MOOCS is that they contain some platform for discussion among users. Indeed, discussion forums can even be considered a defining feature of a MOOC, because, without such forums, a MOOC is more like a collection of online instructional resources rather than an interactive course.

Our own preliminary data analysis of 15 MOOCs offered at the University of California, Irvine, indicates that the number of posts in MOOC discussion forums significantly predicts the number of people who complete MOOCs. Online discussion forums serve an important role in the collaborative learning process of learners [9]; however, little research explores the relationship between social positioning in the forum and the performance at the end of the course in online learning environments. To better understand learners' interaction patterns in MOOC discussions, we employed social network analysis to study the collaborative learning process in the discussions of two large MOOCs. Social network analysis is a methodology that identifies the underlying patterns of social relations of actors [11]. This paper compares the discussion forum activities of two MOOCs and examines three centrality metrics of online learners—*degree centrality, betweenness centrality, and closeness centrality*—and their relationship with learner performance.

2. RELATED WORK
Threaded discussion forums, an important component of computer assisted collaborative learning, allow learners to connect, exchange ideas, and stimulate thinking [3]. Social network analysis (SNA) is valuable for analyzing the dynamics of these discussions, as it emphasizes the structure and the relationship of actors [2]. SNA is thus a practical means for gaining insight into the relations and collaborative patterns of learners in the forum [8]. Learners' behaviors measured by social network metrics (e.g. *authority* and *hub*) in discussion forums have been identified as positively correlated with learners' engagement in MOOCs [12]. Previous research on online education indicates that network measures of *centrality* (out-degree) and *prestige* (in-degree) is strongly associated with learners' cognitive learning outcomes [10]. Research in online collaborative learning community found out that central actors tend to have higher final grades and suggested that communication and social networks should be central elements in distributed learning environments [4].

The embedded theory states that learners' embeddedness in the social networks that pervades the educational programs predicts their satisfaction and performance [1]. We hypothesize that learners' embeddeness in online learning environment is also positively correlated with their performance. Three centrality metrics, i.e.degree centrality, betweenness centrality and closeness centrality are proposed to reflect embeddness in the online learning networks.

This paper explores whether the correlation between the three centrality metrics and academic performance exists in the MOOC settings. The study mainly focused on learners who took part in the discussion forum.

3. DATASET
The project focuses on two online courses named "Intermediate Algebra" and "Fundamentals of Personal Financial Planning" delivered via the Coursera platform. The Intermediate Algebra MOOC was 10 weeks long and developed by professors from University of California, Irvine. It was open for all to enroll for free. A total 63,100 learners registered in the course, among which 43,342 learners had a record in the gradebook and 23,662 learners accessed course materials. The course consisted of lecture videos, weekly quizzes, and the final exam. The quizzes accounted for 20% of the final course grade while the final exam accounted for 80% of the final grade. Learners who obtained 65% or more of the maximum possible score were awarded with the Statement of Accomplishment, i.e. the Normal certificate. Learners who achieved 85% or more of the maximum possible score were rewarded the Statement of Accomplishment with Distinction, i.e. the Distinction certificate.

The Financial Planning MOOC was 7 weeks long and developed by a certified financial planner practitioner from University of California, Irvine. Over 110,000 learners had enrolled in the course, among which 84,234 leaners have record in the gradbook and about 55, 000 learners accessed course materials. The course evaluation consisted of weekly quizzes (30%), one peer assessment (30%) and the final exam (40%). Learners who

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

received a minimum of 70% on all graded assignment received the Statement of Accomplishment; those who received a minimum of 85% of all graded assignment obtained the Statement of Accomplishment with Distinction.

In the Algebra course, 2,126 learners participated in the forum during the 10 week course duration. Among them, 1,558 were identified as learners with an academic record, who can be found in the gradebook. It is unclear why a certain percentage of users who participated in the forum, but did not have a record in the gradebook. A possible explaination is that some are instructors and teaching assistants. The percentage of MOOC forum participation of the three performance groups is relatively constant, with 68% of forum participants as none-certificate earners. Table 1 shows the composition of forum participants.

Table 1 Composition of Discussion Forum Participants

Performance Group	Algebra		Financial Planning	
Distinction	311	20%	998	24%
Normal	193	12%	337	8%
None	1054	68%	2897	68%
In total	1558	100%	4232	100%

3.1 Network Descriptive

To create each network we used the following procedure. The forum consists of several sub-forums. Users can initiate a thread in a sub-forum, make posts to a thread, and make comments to a post. Each thread and post serves as a site of interaction among learners. Learners engage in a variety of actions: asking questions, seeking help, and providing assistance to fellow learners. We treat individuals as tied if they co-participate in a thread or a post. These ties represent communication among learners. Although one could create directed ties between individuals who address each other directly in the posts/comments, doing so would require extensive reading and coding of the data and tackling issues such as how to define direct communication (e.g., is implied communication sufficient, or must the alter be directly named?). Given the size of our data, such an approach is infeasible for our purposes.

The Algebra course discussion network has 1,389 nodes, as not all 1,558 individuals participated in the discussion forum have a record in the gradebook. The network has 3,540 edges. We illustrate it below in Figure 1. Nodes colored according to their performance groups. The network is dominated by a large, dense component with a periphery of low-degree actors. A few isolates and lone dyads are also present. Nodes of different performance groups appear to be intermixed throughout the main component and the rest of the graph.

Mean degree is 5.10, although mean degree varies slightly by performance group. Those in the "none" category have the lowest mean degree (4.36) while those in the "normal" performance have a mean degree of 8.249 and individuals earning "distinction" have a mean degree of 5.502.

More than twice as large as the algebra course discussion network, the financial planning course discussion network has 3,317 nodes and 5,505 edges. We depict the network in Figure 2. Like the algebra network, the financial planning network is

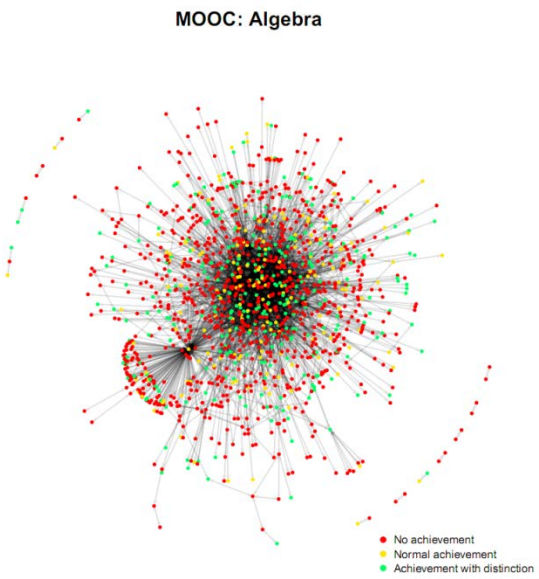


Figure 1: Algebra Network

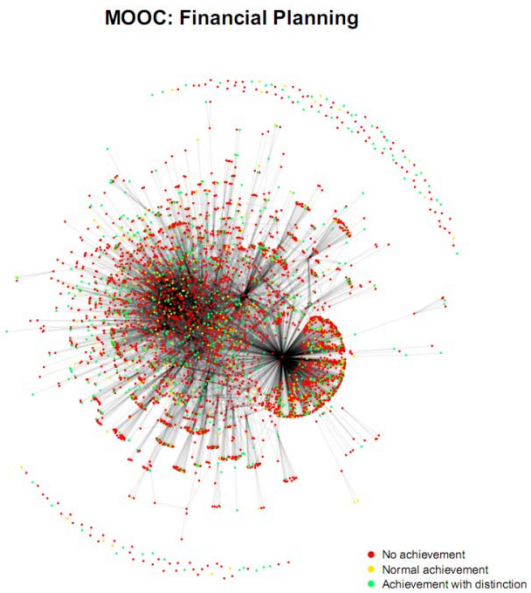


Figure 2: Financial Planning Network

dominated by a large component with a mix of isolates and smaller components. Although the financial planning discussion network is much larger than the algebra network, mean degree is lower. The average degree is 3.32. Like the algebra network, nodes with performance achievements of "normal" or "distinction" have higher degree than those in the "none" category. Those in the "none" category have an average of 2.80 ties, followed by the "normal" category with 4.15 ties, and "distinction" which has an average of 4.48 ties.

4. METHOD

Our analysis consists of analyzing the graph-level centralization and node-level centrality with permutation tests.

4.1 Centrality

Among the most common structural indices employed in the analysis of networks are centrality indices. These measures demonstrate the extent to which a node has a central position in the network [5][11]. Several measures of centrality exist and we utilize three of the most common measures in this paper: *degree*, *betweenness*, and *closeness*. One of the simplest centrality indices, *degree*, measures the total number of alters to which a node is tied. In the context of our MOOC network, this represents the number of other learners to which one is tied through participation in discussion forum threads. Those with high degree have greater levels of participation in a variety of threads that put them in contact with other learners. We also utilize *betweenness*, which measures the extent to which a node bridges other nodes by lying on a large number of shortest paths between them. Nodes with high betweenness have been described as having some degree of control over the communication of others [5] as well as greater opportunities to exert interpersonal influence over others [11]. Nodes with high betweenness in these MOOCs participate in discussions in such a way to learners across multiple forum threads. Finally, we measure *closeness*, which measures the extent to which a node has short paths to other nodes in the network. Nodes with high closeness centrality are described as being in the "middle" of the network structure [2]. Because the standard definition of closeness does not accommodate networks with multiple components, we use the Gil and Schmidt [6]approach of measuring closeness of a node as the sum of the inverse distances to all other nodes.

In addition to measuring node-level centrality, we also measure graph-level centralization. Unlike the node-level centrality indices described above, these graph-level indices produce one measure for the entire graph. These indices measure the difference between the most central node and the centrality scores for all other nodes in the network in order to provide a graph-level measure of the extent to which centrality is concentrated on a small portion of the network's nodes. We compute these centralization scores for the three aforementioned centrality measures: degree, betweenness, and closeness. These measures demonstrate the extent to which centrality is dominated by a small number of learners in the discussion network.

4.2 Permutation Test

Because we cannot guarantee the normality assumptions required by many statistical tests, we use a variety of permutation tests to assess various features of the network. While we use standard, non-parametric correlation tests, we also use non-parametric network methods. These network methods uncover structural biases by using baseline models to determine the likelihood of observing particular structural traits[2]. The results demonstrate the extent to which the network deviates from a reasonable baseline network. These tests allow us to test our hypotheses despite the statistical complexities of the network representation. We use conditional uniform graph (CUG) tests to determine whether features of our observed graph occur at levels exceeding what we would expect by chance. The CUG test conditions on a certain set of network features (typically, size, number of edges, or dyad census) and treats all graphs within that set as equally likely. It then draws at random from this set of graphs and measures whether the statistic of interest is greater, less than, or equal to the measure from our original, observed graph. To the extent that few graphs drawn from the set exceed our observed measure, the measure is higher than we expect by chance. In our analyses, we measure whether the observed levels of centralization in the discussion network are greater than what we could expect from graphs of the same size with the same number of edges.

The second non-parametric network method we employ is the matrix permutation test, often referred to as the quadratic assignment procedure or QAP test [7]. This test evaluates correlations between matrices by permuting rows and columns of the matrices, recalculating the test statistic, and measuring whether it is greater or less than the observed value. This test controls for the structure of the network and allows us to determine whether the labels (i.e., categorical attributes) of the network explain its structure. Where the correlation between the permuted graph rarely exceeds the observed test statistic, we find evidence that the observed statistic is greater than we would expect by chance. We use this technique in our MOOC network to measure whether similarity in grades between any given pair of individuals is associated with the presence of a tie between those individuals.

5. RESULTS

To determine whether observed graph-level centralization exceeds levels we would expect by chance, we use conditional uniform graph (CUG) tests conditioned on the dyad census. We hold constant the number of nodes and number of dyads (either mutual or null, given our undirected graph) when running the test. In our algebra network, degree centralization (.164), betweenness centralization (.269), and closeness centralization (.0001) all exceed chance levels, with p-values less than .01. These results are consistent with the financial planning course, where degree centralization (.354), betweenness centralization (.626), and closeness centralization (.001) were all significantly higher than baseline ($p < .01$). These results indicate that both of our observed networks have much higher levels of centralization than we would expect by chance. These networks are characterized by concentrations of centrality on a handful of nodes. While certain nodes have high levels of centrality, others lack centrality in the network.

We assess node-level centrality by relating our three centrality measures with attainment measures in the course. For each of the nodes in the network, we calculate its degree, betweenness, and closeness and measure the correlation of centrality with the final grade in the course. The correlation between the algebra course grade and degree ($r=.043$, $p=.029$), betweenness ($r=.046$, $p=.018$) are significant while closeness ($r=.028$, $p=.125$) failed to achieve significance in a non-parametric correlation test. Those with high levels of degree and betweenness centrality have higher grades in the algebra course. In the financial planning course we found no evidence of a significant correlation between course grade and degree ($r=.003$, $p=.811$), betweenness ($r=-.002$, $p=.848$), and closeness ($r=-.006$, $p=.582$). Individuals who are more central in the financial planning discussion network did not appear to have notable differences in performance compared to those with lower centrality. Although we find that both these networks have a high level of centralization, we find discrepancies between the correlation between centrality and course grade. While we find no relation between the two in the financial course, we find a weakly positive relation between centrality (except closeness) and grade in the algebra network.

Finally, we look for an association between learners' scores and their propensities to form ties with one another. We use the matrix permutation test, or QAP test, to find an association between tie formation and similar performance in the classes, where performance is measured as the overall grade or end-of-

course distinction status. To measure this association, we correlate the sociomatrix with a similarity matrix *m*, such that the *i*,*j* cell in the matrix represents the similarity in final grade between individual i and individual j. To produce this matrix we found the difference between *i*'s grade and *j*'s grade and subtracted it from 100, the maximum possible difference. The resulting scores represent similarity, where smaller scores indicate similar final grades while larger scores indicate large discrepancies between their final grades. We use the same approach to construct a distance matrix for achievement status, where learners who did not pass the class were scored as 0, while learners who passed received a 1. In the algebra course we found a significant, negative correlation between the observed sociomatrix and grade (r=-.005, p=.01) and achievement (-.007, p < .01). These results suggest that there is an association between tie formation and difference in achievement; that is, algebra learners with high achievement and high grades are *more* likely to be tied to learners with lower performance, and vice versa. In the financial planning course we found similar results: negative correlations between grade similarity (r=-.002, p=.08) and achievement status (r=-.005, p < .01). Although the relation is weak, it suggests that learners are *more* likely to form ties with learners who ended up with different achievement statuses. Learners who failed were *more* likely to communicate with learners who passed, and vice versa.

6. DISCUSSION AND CONCLUSTION

The descriptive statistic shows that the discussion forum is mainly dominated by a small percentage of learners who contributed far more than the rest of learners. This group of opinion leaders or knowledge source helps to build up and maintain the network. It also implies that the MOOCs' network is more an information network than a social network.

According to literature, a likely hypothesis would be that learners who perform well in a MOOC are more central in online discussions. However, our data demonstrated mixed results. In one MOOC (Algebra) we found a significant relationship between centrality in online discussions and student performance, while in the other MOOC (Financial Planning) we found no relationship.

It is worthwhile to consider why there might have been differences in outcomes between the two courses. Though our study was not designed to pinpoint the cause of these differences, they could be related to the differing purposes and audiences of the two MOOCs. The Algebra MOOC is more academically oriented and aims to prepare learners to succeed in higher education, whereas the Financial Planning MOOC is more geared toward assisting people in life skills. Due to the content of the Financial Planning MOOC, learners who were actively involved in the forum discussion may not have been very concerned about obtaining a certificate. Further social network analysis among a larger corpus of MOOC courses could reveal more about the relationship of course content to forum participation; we have recently obtained a corpus of data from 15 Coursera MOOCs at UCI and will conduct follow up research in this area. Additionally, moving beyond permutation tests to model-based approaches such as ERGMs could provide further insight into the properties of these networks and the relations between individual positions and outcomes.

In addition, we find in both networks a weak propensity for individuals to form ties with classmates with very different grades or attainment. This suggests that the discussion forum serves an important role in facilitating help seeking and promoting communication between the knows and the know nots.

The study also has some limitations. For example, it mainly analyzed the behavior of learners who participated in the discussion forum, which only takes up a small proportion of learners in MOOCs. In addition, we did not consider passive forum participation, such as posts or comments viewing. The future research shall include the content analysis to analyze the cognitive engagement of MOOC learners.

7. ACKNOWLEDGMENTS

We are very indebted to the Digital Learning Lab, University of California, Irvine.

8. REFERENCES

1. Baldwin, T.T., Bedell, M.D., and Johnson, J.L. The Social Fabric of a Team-Based M.B.A. Program: Network Effects on Student Satisfaction and Performance. *The Academy of Management Journal 40*, 6 (1997), 1369–1397.
2. Butts, C.T. Social network analysis: A methodological introduction. *Asian Journal of Social Psychology 11*, 1 (2008), 13–41.
3. Calvani, A., Fini, A., Molino, M., and Ranieri, M. Visualizing and monitoring effective interactions in online collaborative groups. *British Journal of Educational Technology 41*, 2 (2010), 213–226.
4. Cho, H., Gay, G., Davidson, B., and Ingraffea, A. Social networks, communication styles, and learning performance in a CSCL community. *Computers & Education 49*, 2 (2007), 309–329.
5. Freeman, L.C. Centrality in social networks conceptual clarification. *Social Networks 1*, 3 (1978), 215–239.
6. Gil, J. and Schmidt, S. The Origin of the Mexican Network of Power. *Proceedings of the International Social Network Conference*, (1996), 22–25.
7. Krackardt, D. QAP partialling as a test of spuriousness. *Social Networks 9*, 2 (1987), 171–186.
8. Nurmela, K., Lehtinen, E., and Palonen, T. Evaluating CSCL Log Files by Social Network Analysis. *Proceedings of the 1999 Conference on Computer Support for Collaborative Learning*, International Society of the Learning Sciences (1999).
9. Rabbany, R., Elatia, S., Takaffoli, M., and Zaïane, O.R. Collaborative Learning of Students in Online Discussion Forums: A Social Network Analysis Perspective. In A. Peña-Ayala, ed., *Educational Data Mining*. Springer International Publishing, 2014, 441–466.
10. Russo, T.C. and Koesten, J. Prestige, Centrality, and Learning: A Social Network Analysis of an Online Class. *Communication Education 54*, 3 (2005), 254–261.
11. Wasserman, S. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.
12. Yang, D., Sinha, T., Adamson, D., and Rose, C.P. Anticipating student dropouts in Massive Open Online Courses. (2013).

Facilitating Graph Interpretation via Interactive Hierarchical Edges

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

ABSTRACT

Graphs visualizations can become difficult to interpret when they fail to highlight patterns. Additionally, the data to be visualized may be hierarchical in nature. Therefore, graphs with hierarchical data need to offer means of telescoping that collapse or expand subgraphs while aggregating their data. In this paper, we demonstrate an interactive hierarchical edge graph on book prerequisite data, which can be generalized to a variety of hierarchical data. We illustrate the importance of ordering nodes (when possible) and coloring by various features. We then demonstrate various ways of performing exploratory data analysis by delivering various pieces of information on mouseovers and utilizing telescoping and filtering.

Keywords

Hierarchical edge bundling, prerequisite relationships

1. INTRODUCTION

When graphs contain many nodes and edges – especially different types of nodes and edges – they can quickly become difficult to visually interpret [7]. The common term is "hairball" as nodes and edges jumble into a tangled morass that occlude any meaningful patterns. Force-directed graphs operate to keep nodes with strong edges closer and nodes with weak or absent edges further apart [2]. This layout can aid in some contexts, but frequently exacerbates the hairball phenomenon. There are two striking visualization designs by Krzywinski and colleagues that aim at revealing interpretable patterns in graphs. At the core of each is at least one meaningful axis on which to align nodes. The first is *Circos*, which arranges sorted nodes along a circle [5]. Nodes are often displayed as arcs along the circle and edges between the arcs are visualized as chords or ribbons that cut through the middle of the circle. Circos has been used in over 500 publications, many related to large-scale genomic data. By arranging nodes along one axis in a circle, Circos easily discriminates nearby and distant edges. The widths of the nodes (length of the arc) can carry meaning and so can the width of the chord between connected arcs. Nodes and edges can also be colored to highlight features such as the node type, the source, and the target. It is also common to display many node features such as histograms of different measures within an arc, for example, Figure 3 in [6].

The other design by Krzywinski is *hive plots* [4]. Hive plots are comprised of multiple axes, each radiating from an inner ring. A given node may exist on one or more axes, aligned along the axis in some meaningful way. For example, an axis might sort nodes by different graph features such as a node's *closeness* – the average distance between a node and all others reachable from it. By placing nodes on various axes, a representation of where a node resides along some feature is captured. When edges are added, it may bring out relationships between adjacently-placed features. For example, anti-correlations of two features compared side-by-side will have many criss-crossed edges. In short, ordering nodes in some meaningful way(s) permits Circos and hive plots to better reveal patterns. Circos and hive plots, however, do not capture hierarchical relationships very well.

Hierarchical edge bundling is a visualization technique on hierarchical data that skews edges toward their parent nodes, which may be invisible in the graph [3]. The visual effect is that edges are channeled into larger, striking swaths while avoiding the direct clutter of the parent nodes. Any topology can be employed, but simpler geometric structures are most commonly used.

In this paper, we demonstrate an interactive hybrid of Circos plots with hierarchical edge bundling on mathematical book prerequisites. The books are structured hierarchically as a table of contents with chapters, sections, objectives, and exercises. Prerequisites map between objectives. The goal was to provide a means of highlighting prerequisites at the various levels, to call out important objectives, and also reveal holes. Through coloring, it is simple to discriminate chapters or to highlight nodes by features such as learner interaction frequency. Through telescoping, it is straightforward to determine those prerequisites that map across chapters, within a chapter, and within sections. Through filtering it is possible to display nodes and edges by their degree. Collectively, by aligning a curriculum along a circle, we demonstrate how this template can be used for displaying various relationships and features of hierarchical, educational data.

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

2. METHODS

Two higher education math books were selected that contained a table of contents and prerequisites as mapped by content matter experts. Interactivity data came from students, largely from the U.S., who were enrolled in courses spanning Fall semester 2012 through 2013 that used these books and the accompanying Pearson MathXL® homework system. All data was translated into JSON format for use in a web browser. The graph and its interactivity functionality was programmed using D3.js [1].

3. DEMONSTRATION

Figure 1 shows a screen shot of the graph and user controls. Displayed is a developmental math book with chapters starting at 12 o'clock and progressing clockwise. Nodes are colored by chapter and have ample spacing to easily discriminate them. Most nodes displayed are at the objective level. Within a chapter, slight separations between the nodes delineate the sections. Edges within a section are shown as little arcs. Edges within the chapter have a larger arc, and edges across chapters bend so that they bundle near to where a chapter node would be. We see various features at a glance. For example, the online appendix has no pre- or post-requisites across chapters. This is because it is shared across several books and is independent from this book.

Chapters 2, 11, and 13 are displayed at the chapter level hiding all of their section and objective nodes, whereas chapters 4 and 7 are at the section level. Chapters can be shown or hidden in the column of checkboxes on the right. For example, some appendix items have been removed from this display. The radio buttons correspond to the level of the hierarchy to display.

In Figure 1, the user has centered the mouse over the Chapter 2 node. The color of the node is green, so bold green edges reveal other chapters to which Chapter 2 is a prerequisite. Also shown are bold orange lines from Chapter 1 objectives coming into Chapter 2. The text of these pre- and post-requisites are listed on the left. We see at a glance that while Chapter 2 is prerequisite to Chapter 3, it links to other sections and objectives in a punctate fashion, completely ignoring the middle chapters of the book.

Edges are colored by their outgoing node color. Of course, they can be colored by their incoming node color or other feature. We have also colored nodes by their degree, highlighting critical objectives and important chapters. We have also colored nodes by performance measures such as the frequency of user interactions. Coloring can be selected through the pulldown menu on the top-left.

This utility also has some filtering capabilities to show/hide edges within sections, within chapters, and across chapters. Nodes can also be filtered out their degree or feature by which they are colored. Similarly, edges can be filtered out if they are below a weight threshold.

3.1 Limitations

While this visualization works well with book prerequisites, making graphs interactive as we have demonstrated, limits the quantity of nodes and edges because they have to be large enough to be selectable. Additionally, while edge bundling facilitates an interpretation of convergence, it also makes it difficult to select or hover over any individual edge for information. As presented, more than 3000 edges begins to be problematic. Similarly, when there are over 500 nodes along the circle, it can become difficult to select a node of interest with a mouse.

3.2 Next steps

This plot and circos plots have only one axis. Avenues to explore include reordering nodes along this axis by different features. Alternatively, hive plots could be extended with the ideas presented here, where various axes could utilize hierarchical data by swapping child nodes with aggregated parent nodes.

4. CONCLUSION

It is difficult to interpret graphs without an adequate visualization. In this work, we demonstrated a template that can be used on hierarchical data aligned along the axis of a circle. At a glance, it can reveal a lot of features, but through filtering, telescoping, and interactivity, exploratory data analysis can be performed to reveal features at various scales. As a template, it is quite useful for contrasting several graphs, or alternatively, illuminating various features within a general structure. For example, in our case using prerequisite data, nodes and edges might be colored by difficulty, fraction correct, time-on-task, or other measures of students interacting with these book objectives. Furthermore, this technique can be generically applied to other datasets.

5. REFERENCES

[1] M. Bostock, V. Ogievetsky, and J. Heer. D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[2] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, Nov. 1991.

[3] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, Sept. 2006.

[4] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra. Hive plots—rational approach to visualizing networks. *Briefings in Bioinformatics*, 13(5):627–644, Sept. 2012. PMID: 22155641.

[5] M. Krzywinski, J. Schein, İ. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, Sept. 2009. PMID: 19541911.

[6] E. S. Mace, S. Tai, E. K. Gilding, Y. Li, P. J. Prentis, L. Bian, B. C. Campbell, W. Hu, D. J. Innes, X. Han, A. Cruickshank, C. Dai, C. Frère, H. Zhang, C. H. Hunt, X. Wang, T. Shatte, M. Wang, Z. Su, J. Li, X. Lin, I. D. Godwin, D. R. Jordan, and J. Wang. Whole-genome sequencing reveals untapped genetic potential in africa's indigenous cereal crop sorghum. *Nature Communications*, 4, Aug. 2013.

[7] D. Merico, D. Gfeller, and G. D. Bader. How to visually interpret biological data using networks. *Nature Biotechnology*, 27(10):921–924, Oct. 2009.

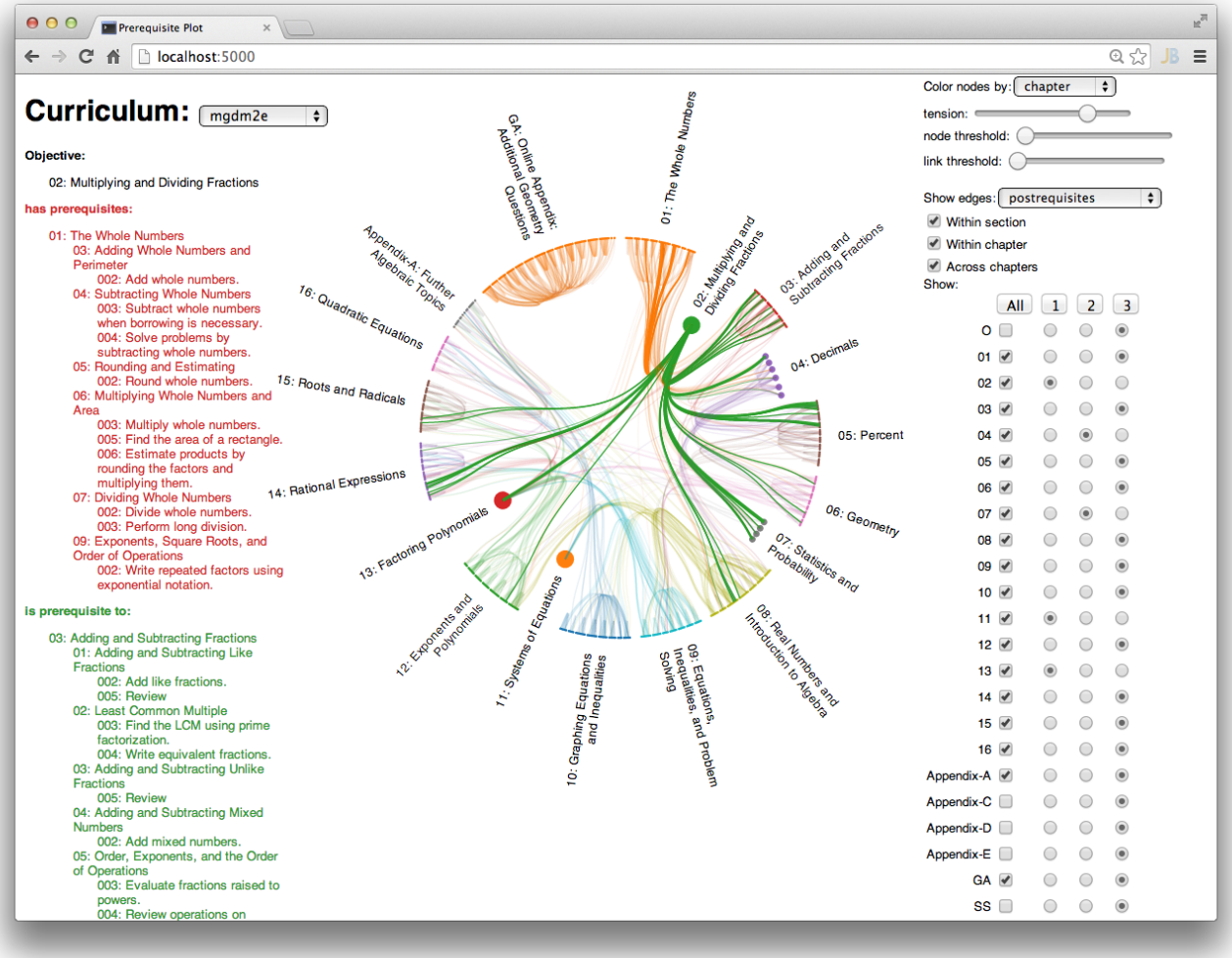


Figure 1: Screen shot of the interactive, hierarchical edge bundling graph.

Evaluation of Logic Proof Problem Difficulty through Student Performance Data

Behrooz Mostafavi
North Carolina State University
Department of Computer Science Raleigh, NC 27695
bzmostaf@ncsu.edu

Tiffany Barnes
North Carolina State University
Department of Computer Science Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT

The interactions of concepts and problem-solving techniques needed to solve open-ended proof problems are varied, making it difficult to select problems that improve individual student performance. We have developed a system of data-driven ordered problem selection for Deep Thought, a logic proof tutor. The problem selection system presents problem sets of expert-determined higher or lower difficulty to students based on their measured proof solving proficiency in the tutor. Initial results indicate the system improves student-tutor scores; however, we wish to evaluate problem set difficulty through analysis of student performance to validate the expert-authored problem sets.

Keywords

Problem Difficulty, Logic Proof, Data-driven Problem Selection

1. INTRODUCTION

Effective intelligent tutoring systems present problems to students in their zone of proximal development through scaffolding of major concepts [3]. In domains such as deductive logic, where the problem space is open-ended and requires multiple steps and knowledge of different rules, it is difficult to choose problems for individual students that are appropriate for their proof-solving ability. We have developed a system that uses the data-driven knowledge tracing (DKT) of domain concepts in existing student-tutor performance data to regularly evaluate current student proficiency of the subject matter and select successive structured problem sets of expert-determined higher or lower difficulty.

We used an existing proof-solving tool called Deep Thought to test the DKT problem selection system. The system was integrated into Deep Thought and tested on a class of undergraduate philosophy students who used the tutor as assigned homework over a 15-week semester. Performance data from this experiment were compared to data from previous use of Deep Thought without the DKT problem selection system. The results of the comparison indicate that the DKT problem selection system is effective in improving student-tutor performance. However, we wish to evaluate the difficulty of presented problems using student performance data to validate the difficulty of expert-determined problem sets, and improve the system for future students.

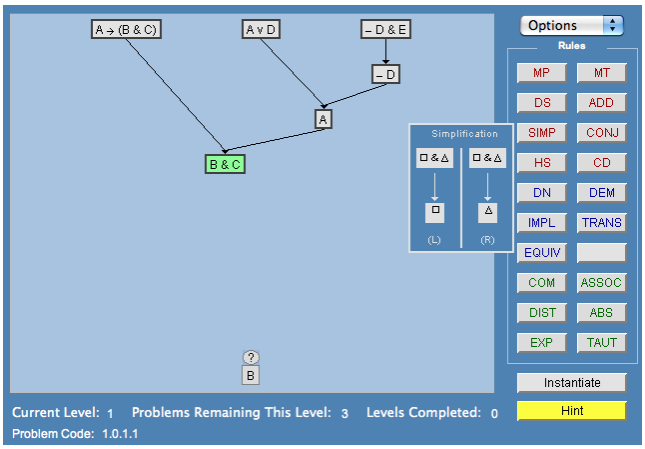


Figure 1: A screen capture of the Deep Thought tutor, showing given premises at the top, conclusion at the bottom, and rules for application on the right.

2. DEEP THOUGHT

Fig. 1 shows the interface for Deep Thought, a web-based proof construction tool created by Croy as a tool for proof construction assignments [1]. Deep Thought displays logical premises, buttons for logical rules, and a logical conclusion to be derived. For example, the proof in Fig. 1 provides premises $A \rightarrow (B \wedge C)$; $A \vee D$; and $\neg D \wedge E$, from which the user is asked to derive conclusion B using the rules on the right side of the display window.

Deep Thought keeps track of student performance for the purpose of proficiency evaluation and post-hoc analysis. As a student works through a problem, each step is logged in a database that records: the current problem; the current state of progress in the proof; any rule applied to selected premises; any premises deleted; errors made (such as illegal rule applications); completion of the problem; time taken

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

per step; elapsed problem time; knowledge tracing scores for each logic rule in the tutor.

2.1 Problem Selection

The problem selection system in Deep Thought presents ordered problem sets to ensure consistent, directed practice using increasingly related and difficult concepts. The system presents set of problems at different degrees of difficulty, determined through evaluation of current student performance in the tutor.

Evaluation of student performance is performed at the beginning of each level of problems. Level 1 of Deep Thought contains three problems common to all students who use the tutor, and provides initial performance data to the problem selection model. Levels 2–6 of Deep Thought are each split into two distinct sets of problems, labeled higher and lower proficiency. The problems in the different proficiency sets are conceptually identical to each other, prioritizing rules important for solving the problems in that level. To prevent students from getting stuck on a specific proof problem, Deep Thought allows students to temporarily skip problems within a level. A unique case occurs if a student skips a problem more than once in a higher proficiency problem set; the student will be dropped to the lower proficiency problem set in the same level, under the assumption that the student was improperly assigned the higher proficiency set (See Fig. 2).

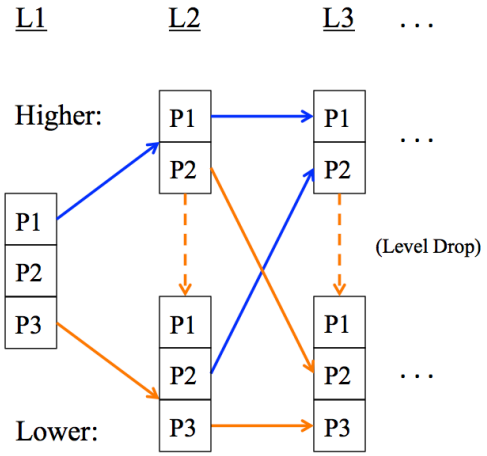


Figure 2: DT2 path progression. At each level, students are evaluated and provided either the higher or lower proficiency problem sets. Students can also be switched from the higher to lower proficiency set within a level.

2.2 Logic Proof Problems

The degree of problem solving difficulty between proficiency sets is different, as determined by domain experts. The problems in the low proficiency set require fewer numbers of steps for completion, lower complexity of logical expressions, and lower degree of rule application than problems in the high proficiency set (See Table 1).

3. DATA GRAPH REPRESENTATION

Table 1: An example of lower and higher proficiency set problems from Deep Thought requiring the same concepts: Level 4 Problem 3 from the lower proficiency set (top); Level 4 Problem 2 from the higher proficiency set (bottom). The prioritized rules required for these problems are Conjunction and Constructive Dilemma.

#	*Premise*	*Derivation*
1	$(A \rightarrow B) \wedge (\neg D \rightarrow F)$	Given
2	$A \vee \neg D$	Given
3	$\neg A \rightarrow (D \vee G)$	Given
4	$\neg A$	Given
5	$B \vee F$	1,2/Constructive Dilemma
6	$\neg D$	2,4/Disjunctive Syllogism
7	$D \vee G$	3,4/*Modus Ponens*
8	G	6,7/Disjunctive Syllogism
9	$(B \vee F) \wedge G$	5,8/Conjunction

#	*Premise*	*Derivation*
1	$Z \rightarrow (\neg Y \rightarrow X)$	Given
2	$Z \wedge \neg W$	Given
3	$W \vee (T \rightarrow S)$	Given
4	$\neg Y \vee T$	Given
5	Z	2/Simplification
6	$\neg W$	2/Simplification
7	$\neg Y \rightarrow X$	1,5/*Modus Ponens*
8	$T \rightarrow S$	3,6/Disjunctive Syllogism
9	$(\neg Y \rightarrow X) \wedge (T \rightarrow S)$	7,8/Conjunction
10	$X \vee S$	4,9/Constructive Dilemma

Deep Thought was used as a mandatory homework assignment by students in a philosophy deductive logic course ($n = 47$). Students were allowed to work through the problem sets at their own pace for the entire 15-week semester. Problem Levels 1–6 were assigned for full completion of the tutor, totaling 13–18 (out of the total tutor-set of 43) problems depending on proficiency path progression.

For the purpose of problem difficulty evaluation, progress through the tutor can be expressed as a directed graph for each individual student, with nodes in the graph each corresponding to a single problem. The node set for the graph represents the problem space for the tutor, and is the same for every student. Each problem node has the following properties:

1. Tutor Level (1–6)
2. Proficiency (High or Low)
3. Problem Number (1–3)
4. Problem Complete (True or False)
5. Expert-Authored
   (a) Required Rules
   (b) Minimal Solution
6. Corresponding Step Logs (See Section 2)

Directed edges between nodes correspond to movement between problems by the individual student, and are assigned a numerical value, ordered by increasing time stamp. The nodes and directed edges together give a map of the student's progression through the tutor. Connected nodes with false Problem Complete status represent a skipped problem, and the node adjacent to the highest numbered edge represents the student terminus point in the tutor. Isolated nodes represent non-visited problems, and are therefore un-useable for problem difficulty evaluation.

Logic proofs can also be represented as directed graphs, with each node containing a proof premise, and each directed edge indicating a node parent-child relationship, along with an applied logic rule. For example, the top proof shown in Table 1 can be represented as a graph with the premise in each line as a node, with the directed edges into that node corresponding to the derivation of that premise from parent nodes. A proof premise can either be a variable (i.e. A), a negated variable or expression (i.e. $\neg A$, or $\neg(A \wedge B)$), or an operational expression in (variable/nested expression)-operand-(variable/nested expression) form (i.e. $A \vee B$, or $(A \wedge B) \vee (A \rightarrow B)$). Nested expressions can be represented in high level form. Therefore, node premises can be categorized by their operand (conjunction, disjunction, negation, implication, equivalence), the complexity of the expression (single variable, simple expression, complex [nested] expression), and the rule used for derivation.

4. PROBLEM DIFFICULTY EVALUATION

The question at hand is how to best use the recorded data to determine proof problem difficulty through student performance. We wish to find both a classification of problem difficulty between proficiency sets in the same level, and difficulty of all problems in the tutor, compared to expert-determined classifications.

Because students follow different problem-solving paths, no student can solve all available problems in the tutor, nor are students likely to solve problems in both proficiency sets within the same level. This makes student performance comparison over multiple problems difficult. We plan to use a combination proof-problem properties weighted by student performance metrics to evaluate problem difficulty; however, we have not determined which combination of methods to use. We are currently looking into weighted cluster-based classification methods to apply to the problems. The hypothesis presumed before applying one of these methods would be that problems of similar difficulty would be placed into the same clusters. Student performance metrics for each problem could be used to determine distance, since it's assumed that students would react most similarly to problems of similar difficulty. Eagle et al. applied network community mining to this student log data in order to form interaction networks [2]; a modified version could be applied here on a student-per-problem level in order to determine prominent similar behaviors that are correlated with problem performance.

This would determine which problems are of similar difficulty, but not necessarily which problems (or groups of problems) are more or less difficult. That determination could be made by analyzing student rule scores across problems, or even the difference in scores at the start and end of a problem. In particular, analyzing the difference in rule scores would both standardize the scores (to account for the scores being calculated at different points in the tutor) and give a measure of forward or backward progress (a student's rule scores should not decrease after solving an easy problem).

Problem properties we feel are valuable to take into consideration when evaluating problem difficulty per student include:

- Classification of problems by operand/expressions
- Deviation of student solutions from expert solutions
  - Number of steps taken
  - Number and frequency of rules used

Student performance metrics that we feel are valuable to take into consideration include:

- Path progression through the tutor, including
  - Order of assigned proficiency sets
  - Number and path location of skipped problems
  - Terminus point in tutor
  - Final tutor grade
- Knowledge tracing scores for each rule, prioritized by problem requirements
- Step and elapsed time
- Type and number of errors committed

We would appreciate any literature recommendations, as well as suggestions for how to use the data from our experiment to measure and compare problem difficulty through student performance.

5. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. 0845997.

6. REFERENCES

[1] M. J. Croy, T. Barnes, and J. Stamper. Towards an Intelligent Tutoring System for Propositional Proof Construction. In *Current Issues in Computing and Philosophy*, pages 145 – 155. 2008.

[2] M. Eagle, M. Johnson, and T. Barnes. Interaction Networks: Generating High Level Hints Based on Network Community Clusterings. In *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)*, pages 164–167, 2012.

[3] T. Murray and I. Arroyo. Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (EDM 2002)*, pages 289 – 294, 2002.

InVis: An EDM Tool For Graphical Rendering And Analysis Of Student Interaction Data

Vinay Sheshadri
North Carolina State University
Raleigh, NC
vshesha@ncsu.edu

Collin Lynch
North Carolina State University
Raleigh, NC
collin@pitt.edu

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

ABSTRACT

InVis is a novel visualization tool that was developed to explore, navigate and catalog student interaction data. InVis processes datasets collected from interactive educational systems such as intelligent tutoring systems and homework helpers and visualizes the student data as graphs. This visual representation of data provides an interactive environment with additional insights into the dataset and thus enhances our understanding of students' learning activities. Here, we demonstrate the issues encountered during the analysis of large EDM data sets, the progressive features offered by the InVis tool in order to address these issues and finally establish the effectiveness of the tool with suitable examples.

Keywords

EDM, visualization, graphs, student interaction data

1. INTRODUCTION

One of the central goals of Educational Datamining (EDM) is to translate raw student data into useful pedagogical insights. That is, educational dataminers seek to analyze student interaction data such as user-system logs with the goal of identifying: common errors, typical solutions and key conceptual challenges among other things. This research is of interest to learners, educators, administrators and researchers [17]. In recent years, the increased adoption of web-based tutoring systems, learning management tools and other interactive systems has resulted in an exponential increase in available data and increased demand for novel analytical tools. The Pittsburgh Science of Learning Center's DataShop, for example, currently stores over 188 datasets, encompassing 42 million student actions and 150,000 student hours [19]. With the increase in available data has come a corresponding increase in the insights EDM can provide and in making analytical tools available to expert instructors.

EDM researchers have generally relied on statistical analyses (see [14, 2, 1], formal rule induction (e.g. [12]), or other modeling methods to extract these insights. While these analytical methods are robust and have led to great progress in model development and evaluation, the increased interest in EDM by non-statisticians and practitioners has accentuated the need for "good visualization facilities to make their results meaningful to educators and e-learning designers" [16].

InVis was initially developed by Johnson, Eagle and Barnes [11]. The present version has been expanded to include changes to the visual editing system, export functions and other features. An example graph is shown in Figure 1. The graphical structure of InVis is designed to facilitate direct exploration of student datasets and easy comparison of individual solution paths. InVis can render individual student solutions or display the work of an entire class thus enabling educators to identify and draw insights from common student strategies and repeated mistakes [11]. InVis was inspired by the work of Barnes and Stamper [3] on the use of graphical representations for logic problems. Similar work has been done by Chiritoiu, Mihaescu and Burdescu who developed the EDM Visualization tool. This tool generates the student clustering models using k-means clustering algorithm [5]. However unlike InVis, the resulting visualization is non-interactive and non-graphical.

EDM researchers generally seek to answer questions such as: What actions can predict student success? Which strategy or solution path is more or less efficient and educationally effective? What decisions indicate student progress? And what are the features of a learning environment that promote learning? (see [15]). In a programming tutor, for example, students might be given the task of implementing an array-sorting algorithm for a large vector of integers. The particular choice of algorithm and the implementation details are left to the students to formulate using a variety of existing tools. This resulting code will proceed in several stages including reading data from disk, sorting the contents in memory, and returning the result. Our goal as researchers is to classify the successful students, identify the most commonly-chosen algorithms and flag individuals who faced difficulties or failed to complete the assignment. In a logic tutor such as Deep Thought [7] or a Physics tutor such as Andes [20] we would like to make similar determinations by focusing on the solutions chosen by the students and the individually-critical steps.

Published in CEUR-WS:
G-EDM workshop (Lynch and Barnes)
In EDM 2014 Extended Proceedings (Gutierrez-Santos and Santos)

The graph representation provided by InVis allows us to answer these questions by constructing and exploring interactive visualizations of the student dataset. By rendering a graph of a class or key subgroup (e.g. low-performing students), we can visually identify garden-path solutions over long isolated chains, identify critical states through which most students traversed and so on. These visualizations can also be used to guide, or evaluate the output of automatic analysis such as MDP models or path-detection algorithms. In the remainder of this paper we will discuss the tool, describe key features of it in detail and illustrate the type of insights it can provide.

2. DATA

We will illustrate the operation of InVis on a typical dataset. For the purposes of the present paper we will use student data collected from the Deep Thought tutor [6, 7]. Deep Thought is a graph-based tutor for first-order logic. Students using the system are presented with a problem defined by a set of given components (e.g. "$A \wedge \neg B \wedge C \Rightarrow B$") and are tasked with proving some goal state (e.g. $\neg C$). Problem solving proceeds through forward or backward-chaining with students applying rules such as Modus Ponens or Modus Tolens to draw new conclusions. For example, given the conclusion B, the student could propose that B was derived using Modus Ponens (MP) on two new, unjustified propositions: $A \rightarrow B, A$. This is like a conditional proof in that, if the student can justify $A \rightarrow B$ and A, then the proof is complete. At any time, the student can work backwards from any unjustified components, or forwards from any derived statements or the premises [8].

The DT data thus has a number of key characteristics that make it amenable to graphical display. The data is grouped into fixed problems covered by many students. Each problem is defined by a static set of given information and a clear goal. And the solutions are constructed via iterative rule applications drawn from a fixed library. As a consequence it is possible to define a fixed, albeit large, space of solution states and to efficiently map the traversal between them. While this seems restrictive this set of criteria applies to data collected from many if not most Intelligent Tutoring Systems. Andes, for example, defines problems by a set of given values (e.g. "$M_{car} = 2kg$") sets fixed variable goals (e.g. "S_{car-t0}": speed of the car at t_0) and groups student actions into a fixed set of rule applications. Similar state representations have also been applied to other datasets such as code-states in the SNAP programming tutor [4].

The figures shown below are drawn from two InVis datasets. We will focus in detail on a small dataset comparing the work of three students on a single problem with a fixed set of givens and two alternate goals. Such a small dataset is designed to allow for efficient illustration but is not an upper limit for analysis. We will also present some qualitative discussion of larger scale analysis with a larger DT dataset as shown in Figure 3.

3. FEATURES OF INVIS

InVis was developed with the *Java Netbeans Framework* and employs the *JUNG* libraries for the rendering of the graphs [13]. It provides an assortment of features that allow the end user to interact with the visualizations and draw observations from the data set. The *Network Display, Network Viewer, Visual Editor and Export Dot Data* are some of the prominent features of InVis which will be illustrated with examples in the upcoming sections. InVis also supports *MDP calculation, between-ness calculation and frequency reduction* which currently are under development and test phases.

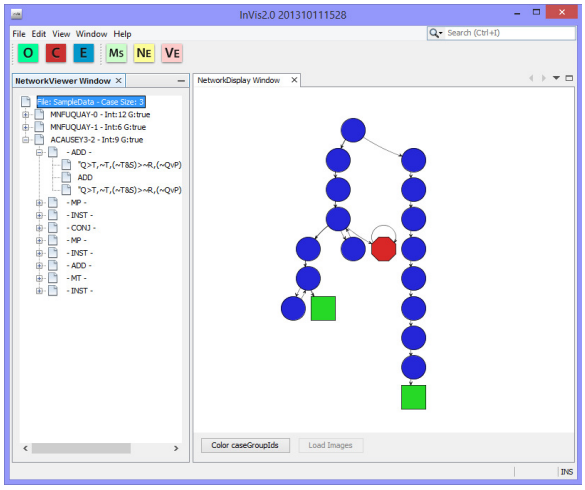


Figure 1: Network Display and Viewer

3.1 Network Display and Viewer

The front-end of InVis is the The *Network Display* component. It displays the interaction network generated by the engine in a graphical format. The user is presented with a cumulative overview of the processed input data. The various logic states of the DT tutor are represented by nodes and the applied propositional logic transformations are represented by edges of the graph. Intermediate states are represented by blue circular nodes while the goal states are represented by green square nodes. Error states in the DT dataset are defined by logical fallacies and are represented by red octagons for easy identification. The sample display shown in Figure 1 contains 16 intermediate nodes arrayed from the top to bottom of the network, one error state located in the center, and two goal states at the bottom.

The *Network Viewer* component represents the InVis input data in the form of a tree structure known as case-set. Each primary node in the case-set represents a student and each sub-node under it represents a transition state executed by the student sequentially. Selecting a student in the Network Viewer window highlights the corresponding path in the Network Display window. Selecting a sub-node highlights the corresponding nodes and edges that were involved in the transformation. Expanding a sub-node will cause the system to display the pre-state and post-state information from the nodes involved in that transition.

The path taken by a student to solve the given problem can be detected by selecting the appropriate student in the Network Viewer window. This will fade the non-path nodes to bring the chosen path to the foreground. An example of this highlighting is shown in Figure 2 where we have selected a single student path within the demo dataset.

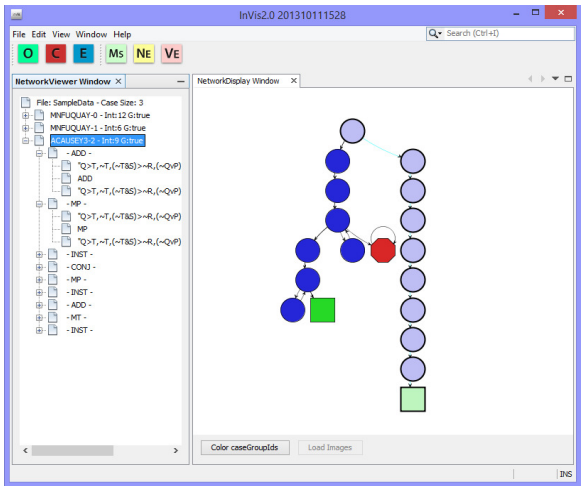


Figure 2: Tracing the path of a student

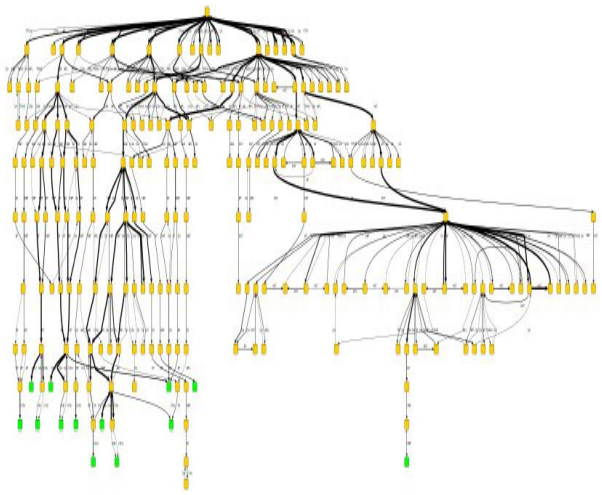


Figure 3: InVis and large data sets

One common use of InVis is to identify frequently-occurring error states. The system can also be used to analyze the different paths taken by students in order to achieve a common goal and isolate the areas where the students face difficulties in solving the given problem or took a garden path. A garden path is an inefficient path from one target state to another with many nonessential intermediate states. From Figure 1, in the current data set, for example, one student performed 11 transitions to achieve the goal, due in part to cycles, whereas a separate student reached the goal with 5 transitions. Each transition is marked by an arc from one state to another in the graph. Thus the Network Display provides an instructor with a cumulative analysis of the input data and aids the instructor in identifying areas of difficulty faced by students during the course of problem solving.

Figure 3 shows the visualization generated by InVis for a sample large dataset. The bold edges indicate the common paths employed by the students in order to solve a given problem. The graph also highlights the garden paths and the succeeding action taken by students towards achieving the goal states. From the rendered visualization it is clear that the cloud space comprises of students who achieved the goal, indicated in green and students who failed to reach the final goal states. InVis can thus be employed to congregate useful observations on large EDM datasets.

3.2 Visual Editor

The *Visual Editor* component of InVis controls the various visual aspects of the graph displayed in the Network Display window. The visual editor provides options for displaying the node and edge data of the graph. InVis renders graphs with the DAG tree layout as the default layout. The visual editor provides options for rendering the graph in different layouts. An ISOM layout of the originally generated graph is shown in Figure 4.

The Visual Editor also provides an option for normalizing the edge widths based on the case frequencies. Case frequencies are defined by the number of students who used the same transition between the given set of states. When the *Normalize Width* option is selected, InVis reloads the graph with width of edges proportional to the case frequency. This feature helps instructors in identifying the logic states and transitions which are most used by the students.

The Visual Editor can be launched by clicking on the Visual Editor icon in the toolbar. Options are provided in the Visual Editor window to control the display of node and edge labels. A notable option provided by the visual editor is the option to normalize edge widths. Normalizing edge widths results in the modification of the edge widths of the graph in proportion to the case frequencies.

Figure 5 displays the zoomed in version of the graph with normalized edges. Edges with case frequency of 2 have thicker connecting lines compared to the edges with case frequency of 1. Thus the thickness of the edge offers a visual cue to the instructor in identifying the most commonly traversed paths by students when achieving the given goal.

3.3 Exporting InVis Data

Graphviz is a heterogeneous collection of graph drawing tools [9]. The software is available under open source license. The input to the Graphviz tool is a description of the required graph in a simple text language such as DOT. The tool processes the input and renders output graphs in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser [10]. Graphviz has many useful features for concrete diagrams, options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

In order to leverage the graph design features offered by Graphviz, InVis now features a new export option which

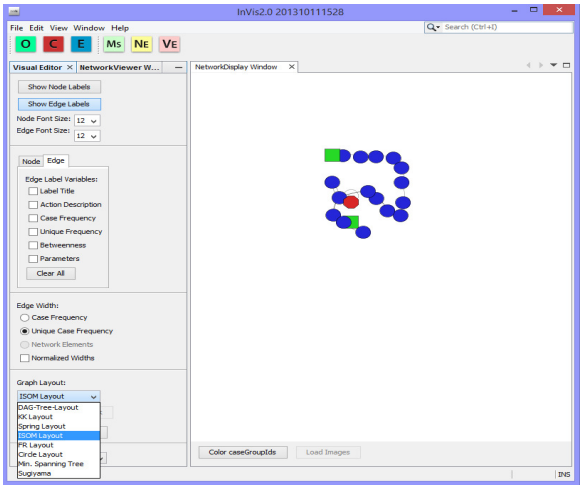


Figure 4: Different graph layouts

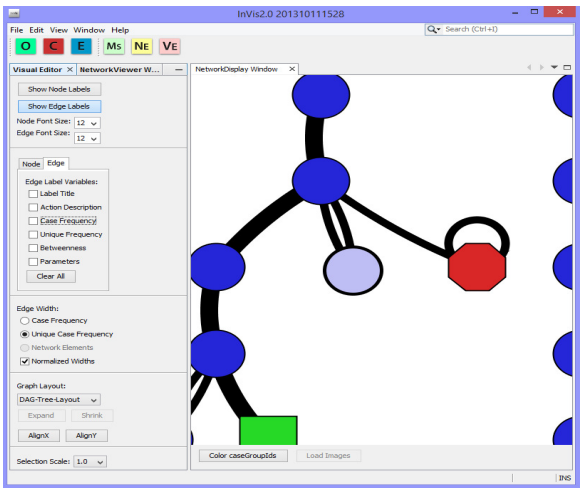


Figure 5: Normalized width - Zoomed in

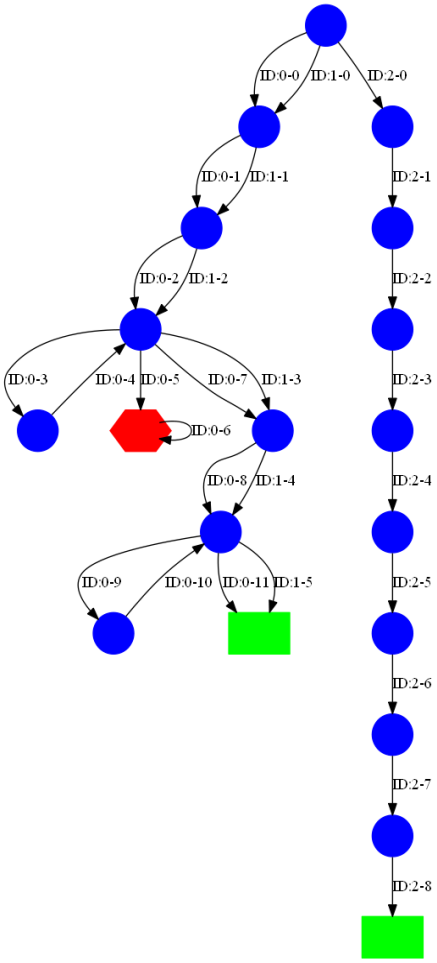


Figure 6: Exported data loaded in Graphviz

renders the input Deep thought data into a DOT format file. The DOT file can be directly imported by Graphviz to generate static images such as PNG, JPEG or interactive formats such as SVG. These visualizations will match those generated by the Network Display tool. Figure 6 shows a graph generated by Graphviz using exported InVis data. Here the arcs are annotated via a static ID number that helps in manually identifying the states and transition information. This data is captured as part of the export process.

4. DISCUSSION

The graphical rendering of EDM data via InVis can yield unique insights into the student interaction data. Romero and Ventura classified EDM objectives depending on the viewpoint of the final user as learner, educator, administrator and researcher [17]. InVis supports learners by providing visual feedback and recommendations to improve performance. Students can compare their approach with that of other students graphically. This can promote real time self-assessment and adoption of better approaches to problem solving.

Educators can use the tool to identify good and poor student solutions and to better understand the students' learning processes which can, in turn, reflect on their own teaching methods. The graphical summary presented by InVis gives an overview, and allows for detailed exploration of, the paths taken by students in achieving a solution to a given problem.

The presence of garden paths, loops and error states illustrate areas where the students have encountered difficulties in deriving a solution to a given problem. This empowers researchers with visual data to model suitable hint generation techniques that can deploy automatic corrective actions [18]. InVis can assist administrators to reorganize institutional resources based on visual evaluation of the effectiveness of a teaching method adopted in a particular course.

In the case of the sorting example introduced in the earlier section, by normalizing the edge width, we can identify the most commonly used sorting algorithm. We can also identify the optimal solution to the given problem comparing the number of transition states between the start and end goal

for each student. Finally the presence of error states, garden paths can be visually identified and corrective actions can be taken to aid students in achieving the goal. Thus the visualizations help in the generation of real time feedback and provides hints for modeling of dynamic hint generation strategies.

InVis is currently limited to the analysis of deep thought tutor data. We are actively working on InVis to extend its capabilities to analyze data sets generated from fields such as: state based games, feedback back based hint generation and others. We are also actively improving the efficiency, user interface, and automatic analysis features of the tool. The InVis project provides the EDM community with a visualization tool for enhanced and accelerated understanding of education based systems. New features will be added to InVis in future to support and sustain this goal. We solicit the EDM community to provide us with additional suggestions for, the InVis tool and help us to enhance the functionality and usability of InVis for EDM applications.

Acknowledgments

This work was supported by NSF-IIS 0845997 "CAREER: Educational Data Mining for Student Support in Interactive Learning Environments" Dr. Tiffany Barnes PI.

5. REFERENCES

[1] R. Baker, A. Corbett, I. Roll, and K. Koedinger. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, 18(3):287–314, 2008.

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

[3] T. Barnes and J. Stamper. Toward the extraction of production rules for solving logic proofs. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education, Educational Data Mining Workshop*, AIED2007, pages 11–20, 2007.

[4] A. H. Barry Peddycord III and T. Barnes, editors. *Generating Hints for Programming Problems Using Intermediate Output*. International Educational Datamining Society IEDMS, 2014. In Press.

[5] M. S. Chirioiu, M. C. Mihaescu, and D. D. Burdescu, editors. *Students Activity Visualization Tool*. International Educational Datamining Society IEDMS, 2013.

[6] M. J. Croy. Graphic interface design and deductive proof construction. *Journal of Computers in Mathematics and Science Teaching*, 18(4):371–385, 1999.

[7] M. J. Croy. Problem solving, working backwards, and graphic proof representation. *Teaching Philosophy*, 2(23):169 – 187, 2000.

[8] M. J. Eagle and T. Barnes. Evaluation of automatically generated hint feedback. *EDM 2013*, 2013.

[9] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz - open source graph drawing tools. In P. Mutzel, M. JÃijnger, and S. Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 483–484. Springer Berlin Heidelberg, 2002.

[10] E. Gansner, E. Koutsofios, and S. North. Drawing graphs with dot. Technical report, Technical report, AT&T Research. URL http://www. graphviz. org/Documentation/dotguide. pdf, 2006.

[11] M. W. Johnson, M. Eagle, and T. Barnes. Invis: An interactive visualization tool for exploring interaction networks. *Proc. EDM 2013*, 65, 2013.

[12] C. F. Lynch, K. D. Ashley, N. Pinkwart, and V. Aleven. Argument graph classification with genetic programming and c4.5. In R. S. J. de Baker, T. Barnes, and J. E. Beck, editors, *EDM*, pages 137–146. www.educationaldatamining.org, 2008.

[13] J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y.-B. Boey. Analysis and visualization of network data using jung. *Journal of Statistical Software*, 10(2):1–35, 2005.

[14] P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance factors analysis - a new alternative to knowledge tracing. In V. Dimitrova, R. Mizoguchi, B. du Boulay, and A. C. Graesser, editors, *AIED*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, pages 531–538. IOS Press, 2009.

[15] A. Pena-Ayala. *Educational Data Mining: Applications and Trends*. Springer, 2014.

[16] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Syst. Appl.*, 33(1):135–146, July 2007.

[17] C. Romero and S. Ventura. Data mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(1):12–27, 2013.

[18] 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.

[19] J. Stamper, K. Koedinger, R. S. J. d. Baker, A. Skogsholm, B. Leber, J. Rankin, and S. Demi. Pslc datashop: A data analysis service for the learning science community. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems - Volume Part II*, ITS'10, pages 455–455, Berlin, Heidelberg, 2010. Springer-Verlag.

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