Interaction Networks: Generating High Level Hints Based on Network Community Clustering

Michael Eagle, Matthew Johnson, and Tiffany Barnes
The University of North Carolina at Charlotte
College of Computing and Informatics
9201 University City Blvd, Charlotte, NC 28223
{mjeagle, matjohns, tiffany.barnes}@uncc.edu

ABSTRACT

We introduce a novel data structure, the Interaction Network, for representing interaction-data from open problem solving environment tutors. We show how using network community detecting techniques are used to identify subgoals in problems in a logic tutor. We then use those community structures to generate high level hints between subgoals. The preliminary results show that using network analysis techniques are promising for exploring and understanding user data from open problem solving environments.

1. INTRODUCTION

This paper introduces a data structure for the analysis of interaction-data collected from open problem solving environments. This structure embeds meaningful information into a complex network, which is subject to analysis through network science techniques. We apply a method of network community clustering to derive subgoals in problems. These clusters allow us to derive high-level hints, which direct students to subgoals in the open problems.

Other methods for modeling students, such as the Bayesian knowledge tracing model [5] are difficult to apply to ill defined domains, such as open problem solving tutors. First, Bayesian knowledge tracing requires each interaction to be labeled as correct or incorrect. Second, each interaction needs to be assigned a single knowledge component. For open procedural problems, both of these assumptions are challenging. As each interaction represents a step towards a goal, it is difficult to address the correctness of an individual step. While errors in the application of actions can be easily marked, errors in obtaining the correct solution require special attention. The open nature of the environment makes it possible for each interaction to provide opportunities to apply several skills. Furthermore, the skills needed for an interaction include action-application, action-opportunity recognition, and problem-solving skill.

For these reasons, many of the traditional methods for data mining on problem solving environment tutor data is either non-applicable or difficult to structure in appropriate ways. We propose using an Interaction Network as a method of structuring tutor data from problem solving environments and show one example of how this has been useful.

Next, we use the interaction network in a method for automatically generating high level hints. This is an extension of previous work where automated feedback was generated from student data [2]. We extend that work by using network community clustering on a interaction network derived from student data; this allows us to generate higher level hints based on derived subgoals. Automatically generated hints have shown positive educational results [10], and have been applied across domains [8].

2. PROBLEM SOLVING ENVIRONMENTS

The effect that the tutor has on how students solve problems is important. While the pedagogical benefits of scaffolded problems are well known, open problem solving environment based tutors may encourage learning in higher 'levels' of cognitive domains [4]. For this work, we define problem solving environments as non-scaffolded tutors, where students are free to apply one of many different actions and are required to complete many steps to solve a single problem, as in the Deep Thought tutor [6].

One advantage to these types of environments, compared to scaffolded problems, is that less time is required for the authoring of problems, as scaffolding is less necessary. Also there are several existing simulations and educational environments that were developed without intelligent feedback. Our goal is to provide data-driven techniques to automatically generate intelligent feedback based on previously recorded data from such environments.

3. INTERACTION NETWORK

In sequential problem solving environments a solution path describes a sequence of state changes from a starting position towards a desired end position. For this work we will only consider discrete time environments with deterministic state transitions. An interaction network is a data structure designed to concisely describe the information contained in a large number of such sequences. This structure is modeled as a complex weighted network, in which information relevant to educational data mining is encoded into the edges and vertices. Interaction networks provide a structure on which to perform data mining, and are also useful for visually displaying information via state diagram visualizations. The interaction network, in terms of ACT-R, is primarily concerned with the results of the Manual Control module [1]. We mention this to make a distinction between the Imaginal module, which contains steps the user makes in internal cognition. That is, the interactions are empirical observations between the subject and the tutor and do not represent internal cognitive states which may occur between

recorded actions.
An interaction network is based on individual student-tutor interactions, as recorded in the log file of the tutoring environment. We define an interaction, I, is a 5-tuple $I = (S_t, A_., S_{t+1}, U, I)$, where

- S_t is the state at step t
- A_s is the action performed on S_t
- S_{t+1} is the resulting state after A has been performed
- U is the unique case ID responsible for this interaction
- I is a set of additional information about the interaction. For example, I_{time} would return a value for how long this interaction took. Included here are I_{error}, which stores the error value, and I_{goal}, which is true if this action resulted in a goal state.

A case represents an individual user of the tutoring system, specifically a case is a ordered pair $c = (U, I)$, where

- U is a unique identifier
- I is a set of additional information about the individual. For example, $I_{pretest}$ would return a value for this case's pretest score.

Finally, we define the interaction network for a problem P is as,
$IN_P = (C, S, A, t, s, S_0, G, I_A, M)$, where

- C is a set of *cases*.
- S is the set of observed tutor program states
- A is the set of observed actions, which connect two states
- $s : A \to S$ and $t : A \to S$ are two maps indicating the *source* and *target* states of an action
- S_0 is the starting state of the problem
- G is the set of goal states
- $I_A : A \to I$ is a map to the source set of Interactions
- M is the set of maps, which allow the lookup of relevant state, action, and case information. For example: $M_{Freq} : S \to Frequency$ will map from the state x to the frequency value for that state.

We model a solution attempt as a ordered set of interactions. We use *case* to refer to individual students, as well as student specific information. We create the interaction network for a problem by conjoining the set of all the path graphs. We use *state* to describe the state of the software environment, representing enough information so the program's state could be regenerated in the interface. We use *actions* to describe user interactions and their relevant parameters. We also store the set of all cases who visited any particular state-vertex or action-edge, allowing us to count frequencies and connect case specific information to the interaction network representation.
This representation results in a sparse, weighted, directed, labeled pseudograph, which can contain loops and cycles; with states as vertices, directed action edges to connect the states, and cases that provide additional information about states and edges. This representation allows us to build a interaction network model from any system that logs interactions in state, action, resulting-state tuples. This results in a network graph which represents the interactions of a large number of users in a relatively concise space.

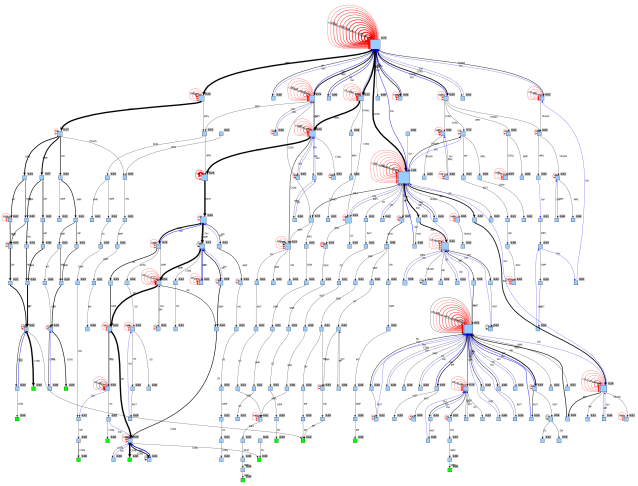

Figure 1: An Interaction Network from the Deep Thought data set. Error actions are shown as red edges, edge width depicts frequency, green squares are goal states.

4. THE DEEP THOUGHT TUTOR

We apply the interaction network to data from Deep Thought, a propositional logic tutor in which students are tasked with performing first-order logic proofs [6]. Students are given a set of premises and a desired conclusion; the student must then use the basic logic axioms to prove the conclusion. As the student works through the proof, the tutor records each interaction. We model the application of axioms as the actions. We model the state of the logic tutor as the conjoined set of each premise and derived proposition.
For example a student starts at state $A \vee D, A \to (B \wedge C), \neg D \wedge E$, where each premise is separated by a comma. The student performs the interaction $SIMP(\neg D \wedge E)$, applying the simplification rule of logic to the premise $\neg D \wedge E$ and derives $\neg D$. This leads to the resulting-state of $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$. Errors are actions performed by students that are illegal operations of logic and the tutor. For example: The student is in state $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$. The student performs the interaction $SIMP(A \vee D)$ in an attempt to derive A. The resulting-state would remain $A \vee D, A \to (B \wedge C), \neg D \wedge E, \neg D$, the log-file would mark this edge as an error.

4.1 Working Backwards

Deep Thought allows students to work both forward and backwards in the logic domain to solve problems [7]. Working backwards allows a student to select the goal premise and use actions to change the conclusion by adding unjustified propositions. Since students can solve a problem completely by only a single direction or a hybrid approach, the size of our state space is much larger than if only a single direction were possible, however this provides opportunities for

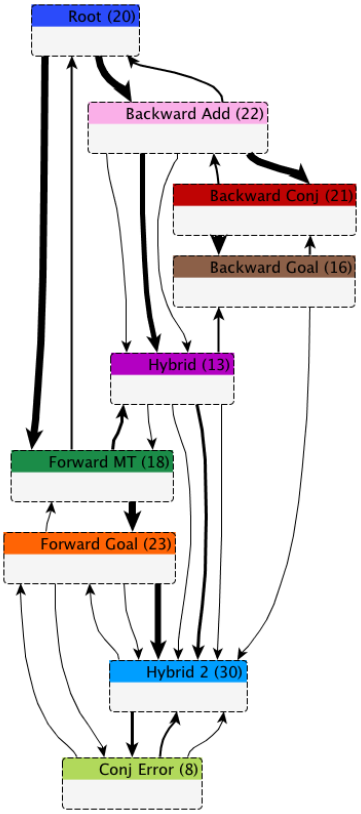

Figure 2: The Interaciton Network from figure 3 represented as a cluster graph. This network is used for the high level hinting process.

diverse problem solving techniques.

4.2 Data

We have six sections of student data from the year 2009, between three professors. Students were required to solve 13 problems from in the Deep Thought tutor as part of the coursework in a introduction to logic class. The problems were generally solved in order, but students could access any problem at any time. In total we have data for 303 students who submitted 1454 attempts across 13 problems totaling 64677 interactions.

5. GENERATING HIGH LEVEL HINTS USING NETWORK CLUSTERING

By formulating our data into an interaction network and using network invariants and metrics, we theorized we could identify sub-goals and in turn student strategies. We also theorized that problems would have underlying structures of sub-goals. We used the Girvan-Newman algorithm to cluster our interaction networks [9]. This algorithm is used to detect communities in networks, where a community has dense node-node connectedness and the edges to nodes in other communities are sparse. Since our data represents interactions, states which are similar are clustered into the same community. Performing similar actions will result in similar states. Performing similar actions in varying orders results in states which are more connected. This causes

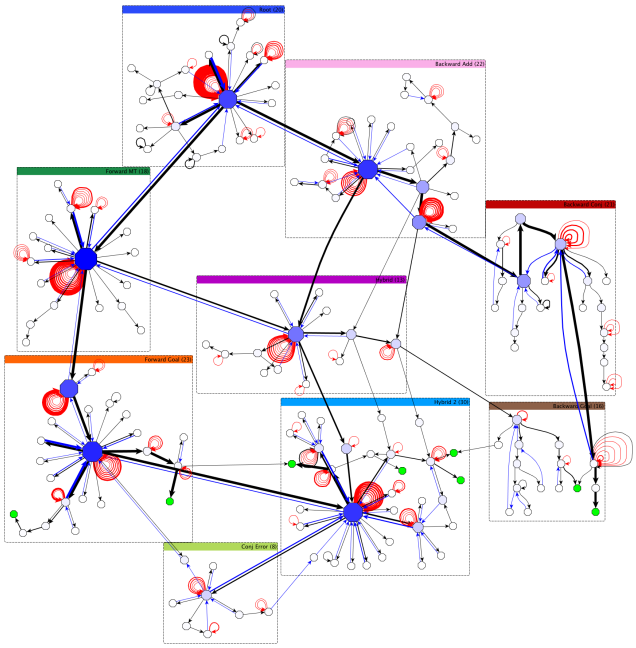

Figure 3: An Interaction Network from Deep Thought with clustering applied. This interaction network has nine clusters. Green nodes are goal states, errors are denoted by red looping edges. Blue edges are deletions where students returned to their previous state. Node size and color is based on node betweenness, and used for visual clarity.

similar interaction orders to be highly connected and result in the same community. For each sub goal, sequences of actions to reach those subgoals will be similar. Different strategies will separate different subgoals into distinct subpopulations, ie. communities. Since actions are defined by edges and Girvan-Newman separates communities by edge-betweenness, in our interaction network we are separating communities based on dominating actions.

5.1 Cluster Graph

In order to generate high level hints, we first divide the network into community clusters using the Girvan-Newman algorithm. This allows us to create a cluster graph, where the clusters are communities of states, highlighting important nodes we will use as sub-goals, see figure 3. The clusters can be interpreted as problem solving strategies. We manually annotated each cluster based on the strategies we observed the students using, see figure 2. We found the general strategies of working forward and backwards, as well as hybrid approaches.

To determine which hint a student receives we perform the Bellman backup algorithm to assign values to the clusters, similar to previous works by Stamper et al. [3]. Next, when a student requests a hint, we first offer a high level hint directing them to a different cluster, which represents the student's next sub-goal.

We treat each cluster as a node, with edges connecting clusters, with the same weights as in the interaction network. Each goal state is connected to a virtual exit-edge outside of it's original cluster. We assign positive values to the virtual exit-edges. Other edges are assigned a cost, to incentive

shorter solution paths. We then perform Bellman backup to generate cluster values, as is done by Stamper et al. [3] with the key distinction of using the clusters rather than using individual states. The Bellmen backup algorithm will iterate until the cluster values converge, and we use these values to provide our next cluster policy.

5.2 High Level Hints

By generating high level hints we can provide students with hints towards sub-goals in the problem, suggesting different strategies. We add two additional hints to Stamper's hint template, where the first two hints are based on strategies. The first hint directs students to the connected node of the 'next' cluster, if there are multiple out-degree edges from the current cluster, we can offer multiple 'next' clusters, so the student can progress along their desired strategy. Since strategies are at a higher level then individual steps, offering multiple strategies is reasonable, whereas offering multiple next steps is likely less beneficial. Students can request a second hint; this hint is based off of the parameters required to derive the first hint.

After that, if a student is unable to consider their high level strategy, we provide local hints. These hints provide immediate next steps towards the desired exit node of the current cluster. This is done by setting the exit edges of the current cluster with positive weight and performing Bellman backup. These hints are provided using the method described in Stamper [3], however it is performed locally in the current cluster, with the edges leading out of the cluster receiving the highest reward values.

All previously observed states are assigned to some cluster. If a new state is observed we can assign hints based on that student's last known cluster. This means we can still offer hints level 1 and 2 to students in states that we have not previously observed. Should students request lower level hints we can recommend students to go back to the entry point of their previous cluster. In Deep Thought students can do this by deleting prepositions from their solution through a delete action. While the interaction Network, clustering procedure, and the hint policy are domain independent, the hint templates are specific to the tutor and domain.

6. CONCLUSIONS

This work makes three contributions. The first is the Interaction Network, a novel data structure for modeling interaction-data from open problem solving environments. By placing interaction data into a network and encoding meaning in the network relationships, we can derive educational insight from network analysis techniques. The second contribution is the application of community detection, detecting sub-populations of the state space, to derive sub-goals in which the nodes and actions are similar. The final contribution is the method of using the community structure in the interaction network to generate both high and low level hints.

7. FUTURE WORK

The current algorithm used for detecting community structure, via edge betweenness, does not directly take into account problem solving specific characteristics. For example, in the Deep Thought data we have defined start and goal states. By incorporating those characteristics into the clustering algorithm more meaningful clusters may emerge.

8. REFERENCES

[1] J. R. Anderson. *How can the human mind occur in the physical universe?* Oxford University Press, New York, NY, USA, 2007.

[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*, ITS '08, pages 373–382, Berlin, Heidelberg, 2008. Springer-Verlag.

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

[4] B. S. Bloom. *Taxonomy of Educational Objectives: The Classification of Educational Goals.* Taxonomy of educational objectives: the classification of educational goals. Longman Group, New York, 1956.

[5] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 1994.

[6] M. J. Croy. Graphic interface design and deductive proof construction. *J. Comput. Math. Sci. Teach.*, 18:371–385, December 1999.

[7] M. J. Croy. Problem solving, working backwards, and graphic proof representation. *Teaching Philosophy*, 23:169–188, 2000.

[8] D. Fossati, B. Di Eugenio, S. Ohlsson, C. Brown, L. Chen, and D. Cosejo. I learn from you, you learn from me: How to make ilist learn from students. In *Proceedings of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, pages 491–498, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

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

[10] J. C. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. In *Proceedings of the 15th international conference on Artificial intelligence in education*, AIED'11, pages 345–352, Berlin, Heidelberg, 2011. Springer-Verlag.