An Algorithm for Reducing the Complexity of Interaction Networks

Matthew W. Johnson
University of North Carolina at Charlotte
Charlotte, NC
mjokimoto@gmail.com

Michael Eagle
University of North Carolina at Charlotte
Charlotte, NC
maikuusa@gmail.com

John Stamper
Carneige Mellon University
Pittsburgh, PA
john@stamper.org

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

ABSTRACT

We present an algorithm for reducing the size and complexity of the Interaction Network, a data structure used for storing solution paths explored by students in open-ended multi-step problem solving environments. Our method reduces the number of edges and nodes of an Interaction Network by an average of 90% while still accounting for 40% of actions performed by students, and preserves the most frequent half of solution paths. We compare our method to two other approaches and demonstrate why it is more effective at reducing the size of large Interaction Networks.

1. INTRODUCTION

One major advantage of computer based tutoring systems, for multi-step problems, is the ability to log data showing 'how' students solved problems, and their mistakes, an aspect not often found in traditional paper based homework. However providing educators a method of understanding the data logged by these systems, efficiently so that it can be acted upon, remains challenging. One approach is to provide a visualization tool to display the solutions to educators, to facilitate gaining insights into how students solved their open-ended multi-step problems. We define open-ended problems, as problems with at least two differing solution paths requiring multiple steps to complete. These types of problems are often seen in Intelligent Tutoring Systems (ITS) and similar computer based instruction, like the Deep Thought Logic Tutor[3].

The interaction network is a network similar to a state-space for tutors, built from data-logs, which leverages student information. One important challenge facing the Interaction Network and InVis, the tool built for exploring those networks, is the size of the network, often resulting in thousands of nodes and edges for roughly a hundred students worth of data. These large networks make it difficult to retrieve a general overview and understanding of student solutions. We present a reduction algorithm that drastically reduces the number of nodes in the network, allowing users to focus on the common approaches used by students to solve problems.

We compare the reduction of nodes and edges between our reduced network and the original, as well as other metrics, like the percent coverage of student solutions. We also compare our approach to two alternative filtering processes to show the benefits of our method. We provide a set of domain experts with one of the problems from the Deep Thought logic tutor and ask they describe the different approaches students use to solve the problem, as well as the common mistakes. We compare the expert provided solutions to the reduced Interaction Network to confirm whether or not our method has appropriately captured the solution paths.

We show that our proposed reduction algorithm when applied to the Interaction Network successfully reduces the number of nodes of the Interaction Network by between 86 and 96 percent, while preserving the solution traces to an average of 52 percent of the goals and 40 percent of the student action frequencies. Furthermore, our method preserves all the solutions suggested by experts. These types of reduced networks could aid in providing a more efficient means of understanding student behaviors, mainly by limiting the network to the most important solutions. When combined with InVis, this could provide an efficient method of understanding how students solved problems in computer based systems, potentially providing a useful role for the educator in their course, by providing a better understanding of student solutions.

2. RELATED WORK

Others have looked at reducing the state space in learning environments for purposes of improving intelligent tutor efficiency and improving interpretation of the data for use by course developers and instructors [9]. Our work differs, as we focus on clustering different student solutions to complex

problems in order to reduce the space of student strategies.

The source for our student data is the Deep Thought logic tutor. This tutor allows students to solve first-order propositional logic problems[3]. Students are provided a set of premises and are challenged with deriving a conclusion. By applying a set of different logic axioms, students can either work from the premises towards the conclusion or alternatively, Deep Thought allows students to work backwards, from the conclusion towards the premises. The Interaction Network can be applied to an individual problem.

Sudol et al. describe a method of generating a similar state space that we use here, but for the domain of programming. In their work, they present the probability distance metric for states in programming problems for introductory students [10]. Menzel and Le have also focused on exploring the state-space of 'ill-defined' domains, using a constraint based system[7]. Mitrovic explores open-ended problems in their web-based SQL tutor which is a constraint based system, but their system has also incorporated a student model to aid users [8].

From our experience with earlier versions of the InVis tool, large networks made it difficult for educators and researchers to efficiently decipher the types of solutions students are using to solve problems from the Deep Thought logic tutor. In our previous work, users explored networks for nearly 20 students at a time. However, our goals for the InVis tool are to make understanding student solutions efficient, which can be achieved by viewing more students at a time. An advantage to looking at more students at once is, it can be easier for users to compare different solutions, as they will not have to maintain those different solutions in their working memory but can quickly make comparisons based on the visualization. Finally, the visualization research community provides us with the Visual Analytics mantra, which argues when there is too much data, visualizations should leverage the machine to analyze the data, identify and present the important features of the data, rather than providing an overview of all the data and relying on the user to filter[2].

An Interaction Network is a model of the state space which includes student information on edges and nodes. It is a connected, directed, labeled multi-graph with states as vertices, actions as directed-edges to connect the states. The Interaction Network stores the set of all students who visited any particular state-vertex or action-edge, allowing us to count frequencies and connect other information, like test scores or hint usage values, to the Interaction Network representation. A detailed description of the Interaction Network is provided in previous work[5].

3. REDUCTION ALGORITHM

Data sets containing many student solution attempts can create large state spaces. One of the goals of InVis is to provide an efficient understanding of common student behaviors. However, exploring networks with thousands of nodes can be slow, and is also subject to hardware limitations. In our experience, even professional software tools for viewing graphs start to slow down when the node counts exceed 1500, on typical PC hardware. We developed an algorithm for reducing the network by roughly 90%, while preserving important information.

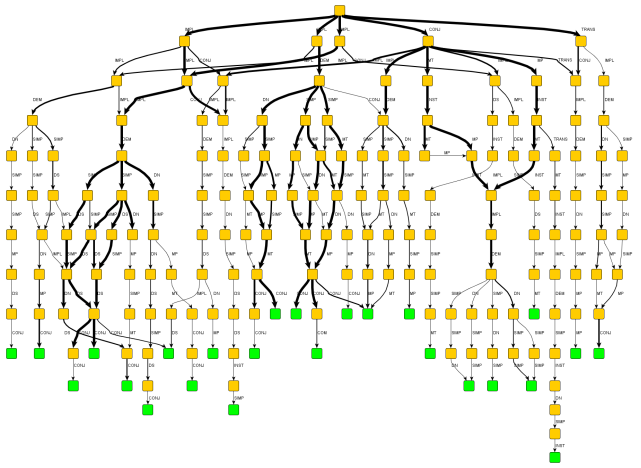

Figure 1: This is the reduced network for problem 3-5. This reduced network contains 186 nodes (85% reduction) and 219 edges (88% reduction).

The purpose of this algorithm is to maximize the amount of information we can gain from the data, while minimizing the number of nodes and edges, to make common approaches more clear. We also want to be as close as possible to a directed simple graph. A simple directed graph is defined as a graph containing no loops or parallel edges. Our assumption is that simple graphs are easier to read when following state-transitions, because they have no parallel edges. Next we want to preserve as many paths from the problem start to the goals as possible, to retain as many student solutions as we can. We would also like to provide continuity and solution variations. Continuity in this case, implies the reduced network maintain complete solution paths, so the graph is understandable, as opposed to a list of the most frequent nodes. By providing variations to similar solutions we should be able to provide better estimations to the numbers of students who performed a particular solution. Without the context of the progression of the states, users would be unable to understand how the problems were solved. We want to provide a means for understanding how many students solved the problem, not just which actions were most frequent.

We will use four metrics for measuring our success.

1. Vertex and Edge Reduction Rates
2. Number of Goals
3. Number of Interactions
4. Average Student Frequency per Edge

The vertex and edge counts will inform us how well we reduced the number of states and actions, we aim for reduction in magnitude. Goal counts will let us know how many of the solution paths we have maintained, from start to finish. For this metric, not only the count is important, but to maintain continuity all goals must have a path from the start of the problem to the respective goal state. The sum of edge frequencies will inform us of the total number of

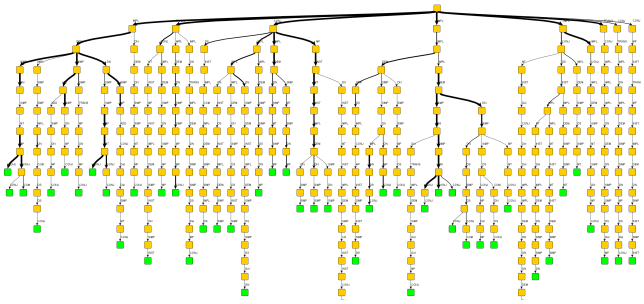

Figure 2: This is the problem 3-5 data set after the shortest paths reduction applied. This network contains 383 nodes(69% reduction), and 382 edges (79% reduction). Note, the two longest solution paths have been cropped in this image.

actions performed by all students, and in turn what percent of student actions are being preserved in our reduction. Arguably, edges with higher frequencies are more informative because more students performed that action, over an action performed by a fewer students. Lastly, the average student frequency per edge will give us an indicator of how important each edge is in the network.

We asked two professors with over a decade of experience teaching logic, and two graduate students who have either taught the course or performed a teaching assistant role to provide us with the set of solutions they expected to see from students. These four experts provided us with eight solutions total, four of which differed in direction or actions used to solve the problem. Problem 3-5 was chosen because it has one of the larger ranges of possible solutions in our problem set. We will use these provided solutions in comparison to the reduced network and compare how many of those solutions are preserved in the reduced network.

3.1 Algorithm

The idea for this algorithm is inspired by compression algorithms. We want to identify the edges with the highest frequencies and preserve them, then find goal states which are close to those paths. The Interaction Network for the problem 3-5 data set has 1252 nodes and 1835 edges. The proposed algorithm works by focusing on high frequency edges, of which there are few, and filtering out the low, and often frequency one edges, for which there are hundreds. This algorithm works by accepting three parameters, the Interaction Network on which to act upon, the percent of desired reduction, and a growth parameter. Prior to reduction, we first calculate a set of values in a pre-reduction step. In tutors which do not contain 'undo' actions, this step will not be necessary. To adjust for the behavior of moving forward, followed by an undo, we calculate a table of negative weights. For each state, an incoming action followed by an 'undo', will increment a negative weight counter for the incoming action. This will be used to devalue the frequency of these actions. Next we remove the 'undo' edges from the network, this reduces the number of cycles and parallel edges presumably making the flow of state-transitions in the Interaction Network easier to follow.

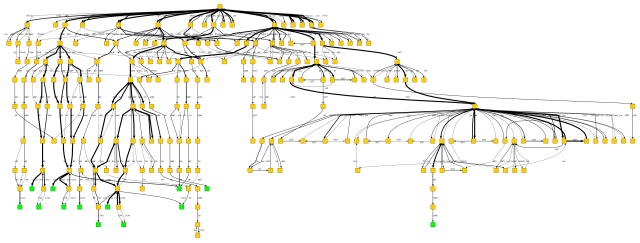

Figure 3: This is the problem 3-5 data set after the frequency one filter reduction is applied. This filtered network contains 235 nodes(81% reduction), and 400 edges(78% reduction).

Next we calculate the adjusted edge frequencies, which are equal to an edge's original frequency minus the weight calculated in the previous step. Now, the network is reduced using the percent reduction parameter. We aimed for an order of magnitude reduction and so this parameter was set to 10%. For the reduction step, we generate a new network using the edges with the top 10% of student frequencies, and their source and target nodes. Depending on the network a set of disjoint graphs will be created, we find the roots of each disjoint graph, which are the nodes with zero in-degree. We then calculate the shortest paths from the problem start to each disjoint-root, and inject the necessary edges and nodes to reconstruct a connected graph. Following this step we check the list of all goal nodes and attempt to connect any node in the reduced graph to any of the goal nodes, again using the shortest path in the original network. We use the growth parameter to limit the distance of the shortest path, for this work we used a value of ten. That is, if a goal node can be reached within ten edges, the path is added, otherwise it is ignored. As a final step, we attempt to connect all the nodes within the reduced graph to any other node in the reduced graph, again using shortest path. The reduced network for problem 3-5 is provided in figure 1.

4. RESULTS

For each problem we generate the Interaction Network, the reduced Interaction Network using our algorithm described here, as well as two other reductions described below. Next we average the values across all 11 problems and compare. We discuss the results presented in table 1 and the comparisons with the other methods of reduction. Experts provided eight total solutions independently, four of which differed in either the actions or direction in which the problem was solved. Our reduced graph contained three out of the four solutions provided by experts. The fourth solution, a forward disjunctive syllogism and Modus Tollens approach was not present in our reduced network or the original full-data network. A working backward version of this solution is in the reduced graph which was solved by a single student.

4.1 Comparison

We compare against two alternative methods of filtering or reduction to help confirm the quality of our chosen approach. Those methods were a shortest path approach and a frequency 1 removal approach, which are somewhat naive but provide for good comparison. The shortest path method of reduction, takes in the start state of the problem and a

Table 1: The average metric scores across 11 problems and 2239 problem sessions. Original refers to the full network values. Each column is a method and its score, with percentile comparison to the Original network in parenthesis. For vertices and edges the percent is the amount of reduction, for goals and interactions it is the amount of coverage or inclusion.

	Original	Reduced	Shortest Paths	Greater than Frequency One
Vertices	1172	**114 (90.26%)**	238 (77.85%)	203 (81.73%)
Edges	1690	**132 (92.23%)**	237 (84.75%)	348 (78.33%)
Goals	38	20 (52.54%)	**38 (100.00%)**	12 (33.04%)
Interactions	3332	1283 (39.90 %)	1162 (36.89)	**1990 (60.77%)**
Avg. Edge Freq.	2.10	**12.34**	6.60	6.03

set of goal nodes for the problem. Next, Dijkstra's shortest path algorithm[4] is run and the result is the union of the shortest paths to each goal. The Frequency one filter approach, simply removes all edges from the network with student frequency one.

Referring to table 1 we can see some advantages and disadvantages of each approach. First, as expected, the shortest path approach naturally has 100% goal coverage, that is we can see *a* path to every goal from the original Interaction Network. The disadvantages of this approach is that the paths chosen do not optimize the frequencies of edges, because the shortest path can contain many frequency one edges. Next the overall reduction rates are half as effective as our method, leaving on average twice as many nodes and edges. This method preserves fewer actions performed by students while having lower rates of reduction. The resulting shortest paths network for problem 3-5 is shown in figure 2. Note, if the growth parameter is set to infinity, the path to all goals will be preserved - same as the shortest path method, though naturally reduction rates will be affected. Thus, our method can facilitate 100% goal coverage.

Alternatively, the frequency one filter, maintains a higher rate of interactions, as we would expect since fewer edges are removed. However, frequency one filtering suffers from low rates of reduction, having double the number of nodes and triple the number of edges on average, while also having lower rates of goal coverage, 33% compared to our method which achieved 52%. This method has lower reduction rates and lower goal coverage. Figure 3 shows the resulting network for problem 3-5 using the frequency one filtering process. By comparing the average edge frequencies in table 1 we can see our method has double the value than either of the other two approaches. This score is meaningful because it is the average number of actions performed by students, per edge within the network.

5. CONCLUSIONS

We provide an algorithm for reducing the complete Interaction Network to a summary of the most common problem solving approaches used by students. We showed that this algorithm was capable of reducing the number of vertices and edges of the Interaction Network by an average of around 90%, while still depicting more than half the of the solution paths and accounting for 40% of interactions performed by students.

6. REFERENCES

[1] G. Cobo, D. García-Solórzano, J. A. Morán, E. Santamaría, C. Monzo, and J. Melenchón. Using agglomerative hierarchical clustering to model learner participation profiles in online discussion forums. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, pages 248–251, New York, NY, USA, 2012. ACM.

[2] B. Craft and P. Cairns. Beyond guidelines: What can we learn from the visual information seeking mantra? In *Proceedings of the Ninth International Conference on Information Visualisation*, pages 110–118, Washington, DC, USA, 2005. IEEE Computer Society.

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

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] M. Eagle, M. Johnson, and T. Barnes. Interaction networks: Generating high level hints based on network community clusterings. In *EDM*, pages 164–167, 2012.

[6] D. Feng, J. Kim, E. Shaw, and E. Hovy. Towards modeling threaded discussions using induced ontology knowledge. *In Proceedings of National Conference on Artificial Intelligence (AAAI-2006)*, 2006.

[7] N.-T. Le and W. Menzel. Using constraint-based modelling to describe the solution space of ill-defined problems in logic programming. In *Proceedings of the 6th international conference on Advances in web based learning*, pages 367–379, Berlin, Heidelberg, 2008.

[8] A. Mitrovic. An intelligent sql tutor on the web. *Int. J. Artif. Intell. Ed.*, 13(2-4):173–197, Apr. 2003.

[9] S. Ritter, T. K. Harris, T. Nixon, D. Dickison, R. C. Murray, and B. Towle. Reducing the knowledge tracing space. In *EDM*, pages 151–160, 2009.

[10] L. A. Sudol, K. Rivers, and T. K. Harris. Calculating probabilistic distance to solution in a complex problem solving domain. In K. Yacef, O. R. ZaÃŕane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *EDM*, pages 144–147. www.educationaldatamining.org, 2012.

[11] D. D. Suthers, H. U. Hoppe, M. de Laat, and S. B. Shum. Connecting levels and methods of analysis in networked learning communities. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, pages 11–13, New York, NY, USA, 2012. ACM.