Interaction Network Estimation: Predicting Problem-Solving Diversity in Interactive Environments.

Michael Eagle, Drew Hicks, and Tiffany Barnes
North Carolina State University, Department of Computer Science
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206
{mjeagle, aghicks3, tmbarnes}@ncsu.edu

ABSTRACT

Intelligent tutoring systems and computer aided learning environments aimed at developing problem solving produce large amounts of transactional data which make it a challenge for both researchers and educators to understand how students work within the environment. Researchers have modeled student-tutor interactions using complex networks in order to automatically derive next step hints. However, there are no clear thresholds for the amount of student data required before the hints can be produced. We introduce a novel method of estimating the size of the unobserved interaction network from a sample by leveraging Good-Turing frequency estimation. We use this estimation to predict size, growth, and overlap of interaction networks using a small sample of student data. Our estimate is accurate in as few as 10–30 students and is a good predictor for the growth of the observed state space for the full network, as well as the subset of the network which is usable for automatic hint generation. These methods provide researchers with metrics to evaluate different state representations, student populations, and general applicability of interaction networks on new datasets.

1. INTRODUCTION

Data-driven methods to provide automatic hints have the potential to substantially reduce the cost associated with developing tutors with personalized feedback. Modeling the student-tutor interactions as a complex network provides a platform for researchers to automatically generate next step hints. An *Interaction Network* is a complex network representation of all observed student and tutor interactions for a given problem in a game or tutoring system. In addition to their usefulness for automatically generating hints, interaction networks can provide an overview of student problem-solving approaches for a given problem.

Data-driven approaches cannot reliably produce feedback until sufficient data has been collected, a problem often referred to as the Cold Start problem. The precise amount of data needed varies by problem and environment. However, some properties of Interaction Networks allow us to estimate how much data is needed. Eagle et al. explored the structure of these student interaction networks and argued that networks could be interpreted as an empirical sample of student problem solving [5]. Students employing similar problem-solving approaches will explore overlapping areas of the Interaction Network. The more similar a group of students is, the smaller the overall explored area of the interaction network will ultimately be. Since we expect different populations of students to have different interaction networks, and different domains to require varying amounts of student data before feedback can be given, good metrics for the current and predicted quality of Interaction Networks are important.

In this work, we adapt Good-Turing frequency estimation to interaction level data to predict the size, growth, and "hintability" of interaction networks. Good-Turing frequency estimation estimates the probability of encountering an object of a hitherto unseen type, given the current number and frequency of observed objects [8]. It was originally developed by Alan Turing and his assistant I. J. Good for use in cryptography efforts during World War II. In our context, network states (vertices) are the object types, and the student interactions (edges) leading to those states are observations.

We present several metrics, derived from Good-Turing frequency estimation. Our hypotheses are that these metrics: **H1:** Predict the probability that a student interaction will result in a state which was not previously observed **H2:** Describe the proportion of the network that has been observed for a population **H3:** Predict the expected size and growth of an interaction network when additional student data is added **H4:** Provide a quantitative comparison of different state representations for their ability to represent greater proportions of the network **H5:** Are useful for comparing different populations of users in how they explore the problem space

Additionally, we use the metrics to explore the subset of the interaction network that is useful for providing automatically generated hints. This provides us with estimates of the size, growth, and coverage of automatically generated hints. We find that our metrics quickly become accurate after collecting a sample of about 10 students. This has value as a metric to compare the quality of the interaction networks,

and will aid future researchers in determining an adequate state representation. We also show how two experimental groups, despite having the same amount of network coverage, have substantially different numbers of unique states. This supports previous work, suggesting that different populations of students produce different interaction networks [5], which has broad implications for generating hints as well as using the networks to evaluate student behavior.

1.1 Previous Work

Creation of adaptive educational programs is costly. This is, in part, because developing content for intelligent tutors requires multiple areas of expertise. Content experts and pedagogical experts must work with tutor developers to identify the skills students are applying and the associated feedback to deliver [13]. In order to address the difficulty in authoring intelligent tutoring content, Barnes and Stamper built an approach called the Hint Factory to use student data to build a Markov Decision Process (MDP) of student problem-solving approaches to serve as a domain model for automatic hint generation [18]. Hint Factory has been applied in tutoring systems and educational games across several domains [7, 14, 6], and been shown to increase student retention in tutors [19].

Early work with the Hint Factory method used a Markov Decision Process constructed from students' problem-solving attempts. Eagle and Barnes further developed this structure into a complex network representation of student interactions with the system, called an *Interaction Network* [5]. Complex networks are graphs or networks which contain non-trivial topological features unlikely to appear in simple or random networks. The Interaction Network representation can be used as a visualization of student work within tutors. The effectiveness of Interaction Networks as visualizations was shown by Johnson et al. who created a visualization tool *InVis* to aid instructors in analyzing student-tutor data [11].

Other approaches to automated generation of feedback have attempted to condense similar solutions in order to address sparse data sets. One such approach converts solutions into a canonical form by strictly ordering the dependencies of statements in a program [15]. Another approach compares *linkage graphs* modelling how a program creates and modifies variables, with nested states created when a loop or branch appears in the code [10]. In the Andes physics tutor, students may ask for hints about how to proceed. Similarly to Hint Factory-based approaches, a solution graph representing possible correct solutions to the problem was used. However their solution space was explored procedurally rather than being derived from student data, and they used plan recognition to decide which of the problem derivations the student is working towards [20].

Interaction networks are scale-free networks. This is a property of complex networks whose degree distribution is heavy-tailed, often a power law distribution. In practice, this means that a few vertices have degree that is much larger than the average, while many vertices have degree somewhat lower than average [5]. Eagle et al. argued that students with similar problem solving ability and preferences would travel into similar parts of the network, resulting in some states being more important to the problem than others [5]. Using these "hub" states, sub-regions of the network corresponding to high-level approaches to the problem were derived. These sub-regions captured problem-solving differences between two experimental groups [4].

2. METHODS AND MATERIALS

For the purposes of this work, we are using datasets from three different environments to build our interaction networks. Summaries of these datasets are found in Table 1. The first dataset is from the Deep Thought tutor, used in previous work by Stamper et al. [19]. This dataset was collected for a between groups experiment investigating the use of data-driven hints, so we split the dataset into two groups, DT1-C, the control group from that experiment, and DT1-H, the group that received hints. We selected this dataset to explore and evaluate H5.

The second dataset comes from the game BOTS. Here, we have the same students and interactions represented in two different ways: First. using *codestates* (the programs users wrote) and second using *worldstates* (the output of those programs). The advantages and disadvantages of these state representations were explored in previous work by Peddycord and Hicks [14]. We split this dataset into two groups as well (BOTS-C and BOTS-W) one for each state representation used. We selected this dataset for evaluation of H4.

Our third and largest dataset comes from an updated version of the Deep Thought tutor, called Deep Thought 3. Unlike with the other datasets, Deep Thought 3 features an AI problem selection component [12]. This means that not all students will have had access to all problems. In addition, there is a larger number of problems in this dataset. We selected this dataset, as the larger number of problems effectively splits student data across multiple networks. H1–H3 are relevant towards measuring the quality of networks produced for new problems.

Table 1: Dataset summary: the total number of students in the dataset, the number of distinct problems, and the average number of students represented in each network.

Dataset	Total N	Num Problems	Mean Net N
DT1-H	203	11	83.73
DT1-C	203	11	63.82
DT3	341	59	78.41
BOTS-C	125	12	99.75
BOTS-W	125	12	99.75

2.1 Constructing an Interaction Network

An *Interaction Network* is a complex network representation of all observed student and tutor interactions for a given problem in a game or tutoring system. To construct an Interaction Network for a problem, we collect the set of all solution attempts for that problem. Each solution attempt is defined by a unique user identifier, as well as an ordered sequence of interactions, where an interaction is defined as {initial state, action, resulting state}, from the start of the

problem until the user solves the problem or exits the system. The information contained in a *state* is sufficient to precisely recreate the tutor's interface at each step. Similarly, an *action* is any user interaction which changes the state, and is defined as {action name, pre-conditions, post-conditions}. In Deep Thought, for example, an action would be the logical axiom applied, the statements it was applied to, and the resulting derived statement. Figure 1 displays two Deep Thought interactions. The first interaction works forward from STEP0 to STEP1 with action $SIMP$ (simplification) applied to $(Z \wedge \neg W)$ to derive $\neg W$. The second interaction works backward from STEP1 to STEP2 with action $B - ADD$ (backwards addition) applied to $(X \vee S)$ to derive the new, unjustified statement S.

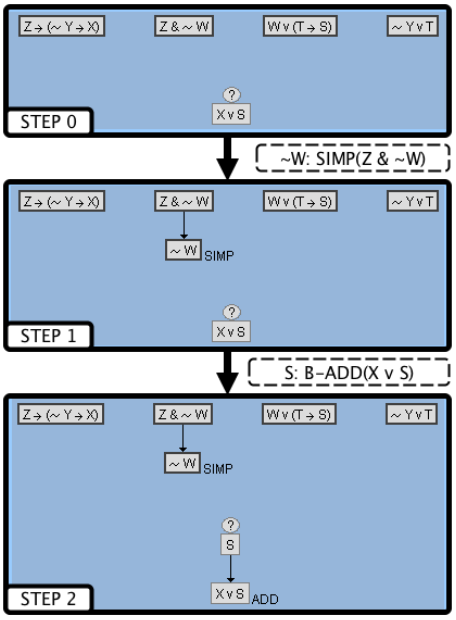

Figure 1: Example of state to state transitions within the Deep Thought (DT1) propositional logic tutoring system.

Once the data is collected. we use a *state matching function* to combine similar states. In Deep Thought, we combine states that consist of all the same logic statements, regardless of the order in which those statements were derived. This way, the resulting state for a step STEP0, STEP1, or STEP2 in Figure 1 is the set of justified and unjustified statements in each screenshot, regardless of the order that each statement was derived. In BOTS, two state matching functions were used: one which combined states based on the code in students' programs, and another which instead used the output of those programs. Similarly, we use an action matching function to combine actions which result in similar states, while preserving the frequency of each observed interaction.

2.2 Providing Hints

Stamper and Barnes' Hint Factory approach generates a next step Hint Policy by modeling student-tutor interactions as a Markov Decision Process [18]. This has been adapted to work with interaction networks by using a Value Iteration algorithm on the states [5]. We generate a graph of all student interactions, combining identical states using a state matching function. Then, we calculate a fitness value for each state. We assign a positive value (100) to each goal state, that is a state configuration representing a solution to the problem. We assign an error cost (-5) for error states. We also assign a small cost to performing any action, which biases hint-selection towards shorter solutions. We then calculate fitness values $V(s)$ for each state s, where $R(s)$ is the initial fitness value for the state, γ is a discount factor, and $P(s, s')$ is the observed frequency with which users in state s take an action resulting in state s'. After this, we use value iteration [2] to repeatedly assign each state a value based on its neighbors and action costs, weighted by frequency.

After applying this algorithm, we can provide a hint to guide the user toward the goal by selecting the child state with the best value. We can do this for any observed state, provided that a previous user has successfully solved the problem after visiting that state. In the original work with Hint Factory on the Deep Thought tutor, the algorithm was permitted to backtrack to an earlier state if it failed to find a hint from the current state. However, not all environments allow the user to backtrack and there are risks of the backtracking hints to provide irrelevant information. Because of this inconsistency across domains, we did not permit backtracking for the purposes of the comparisons in this paper.

We define a state, S to be *Hintable* if S lies on a path which ends at a goal state. We define the *Hintable* network to be the subset of the interaction network containing only *Hintable* states and edges between hintable states; That is, the induced subgraph on the set of *Hintable* states.

2.3 Cold Start Problem

Barnes and Stamper [1] approached the question of how much data is needed to get a certain amount of overlap in student solution attempts by incrementally adding student attempts and measuring the step overlap over a large series of trials. This was done with the goal of producing automatically generated hints, and solution attempts that did not reach the goal were excluded. Peddycord et al. [14] used a similar technique to evaluate differences in overlap between two different interaction network state representations.

The "Cold Start problem" is an issue that arises in all data-driven systems. For early users of the system, predictions made are inaccurate or incomplete [17, 16]. If there are insufficient data to compare to (not enough user ratings, or not enough student attempts) then the quality of the recommendations suffers and in some cases no recommendation can be provided. The term is commonly used in the field of collaborative filtering and recommender systems, but it can be used to describe three related issues, the "new user," the "new item," and the "new community" [3] Cold Start problems. The "new user" problem refers to the difficulty of making recommendations to a user who has performed no actions. The "new item" problem refers to the difficulty of suggesting users visit a newly added, unobserved state. The new community Cold Start problem refers to situations where not enough observations exist to make recommendations for new users. The "new community" definition corresponds most closely to the difficulty of generating hints for

an entirely new problem in an intelligent tutoring system or educational game.

To measure our ability to address this problem, we add all interactions from a single student, one at a time, to the interaction network. This is in order to simulate the growth of the network. We repeat this process for each student, measuring the performance of our model each time. We measured the proportion of currently observed states to total observed states for the entire data set, as well as for the subset of states from which a goal is reachable. To control for ordering effects, we repeated this trial 1000 times using a different random ordering of students each time, and aggregated the results.

2.4 Good-Turing Network Estimation

We present a new method for estimating the size of the unobserved portion of a partially constructed Interaction Network. Our estimator makes use of Good-Turing frequency estimation [8]. Good-Turing frequency estimation estimates the probability of encountering an object of a hitherto unseen type, given the current number and frequency of observed objects. It was originally developed by Alan Turing and his assistant I. J. Good for use in cryptography efforts during World War II. Gale and Sampson revisited and simplified the implementation [8]. In its original context, given a sample text from a vocabulary, the Good-Turing Estimator will predict the probability that a new word selected from that vocabulary will be one not previously observed.

The Good-Turing method of estimation uses the frequency distribution, the "frequency of frequencies," from the sample text in order to estimate the probability that a new word will be of a given frequency. Based on this distribution, the probability of observing a new word in an additional sample is estimated with the observed proportion of words with frequency one. This estimate of unobserved words is used to adjust the probabilities of encountering words of frequencies greater than one.

We adapt the Good-Turing Estimator to interaction networks by using the states with an observed frequency of one to estimate the proportion of "frequency zero" states. Interaction networks represent the observed interactions and therefore we also use this value to estimate the probability that a new interaction will transition into a new state. We use P_0 as the expected probability of the next observation being an unseen state. P_0 is estimated by:

$$P_0 = \frac{N_1}{N} \tag{1}$$

Where N_1 is the total number of frequency 1 states, and N is the total number of interaction observations. Since N_1 is the largest group of states, the observed value of N_1 is a reasonable estimate of P_1. P_0 can then be used to smooth the estimation proportions of the other states. The proportion of states with observed frequency r is found by:

$$P_r = \frac{(r+1)S(N_{r+1})}{N} \tag{2}$$

where $S()$ is a smoothing function that adjusts the value for large values of r [8].

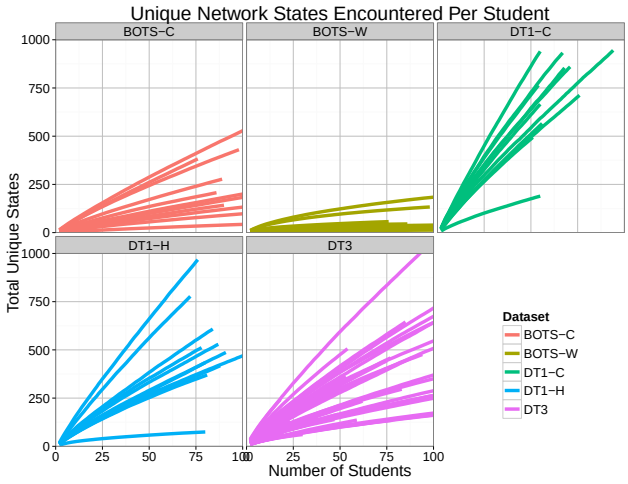

Figure 2: The growth of new states as new students are added for each problem, for each dataset.

Our version of P_0 is the probability of encountering a new state (a state that currently has a frequency of zero,) on a new interaction. We also interpret this as the proportion of the network missing from the sample. We will refer to an interaction with a unobserved state as having *fallen off* of the interaction network. We will use the complement of P_0 as the estimate of *network coverage*, I_C, the probability that a new interaction will remain on the network: $I_C = 1 - P_0$.

The *state space* of the environment is the set of all possible state configurations. For both the BOTS game and the Deep Thought tutor the potential state space is infinite. For example, in the Deep Thought tutor a student can always use the addition rule to add new propositions to the state. However, as argued in Eagle et. al. [5], the actions that reasonable humans perform is only a small subset of the theoretical state space; the actions can also be different for different populations of humans. We will refer to this subset as the *Reasonable State Space*, with *unreasonable* being loosely defined as actions that we would not expect a human to take. An interaction network is an empirical sample of the problem solving behavior from a particular population, and is a subset of the state space of all possible *reasonable* behaviors. Therefore, our metrics P_0 and I_C are estimates of how well the observed interaction network represents the reasonable state space.

3. RESULTS

In order to evaluate the performance of the unobserved network estimator, P_0, and the network coverage estimator, I_C, for each problem in each of our 5 datasets we randomly added students from the sample, one at a time until all student data had been included. At each step, T, we recorded the values of our estimators using only the data that had been encountered up until then. This simulates a real world use-case, where additional students are added over time. We repeated this process 1000 times and averaged the results. Figure 2 shows the growth of unique states as students are added for the interaction networks generated by each problem (line) in each of the five datasets.

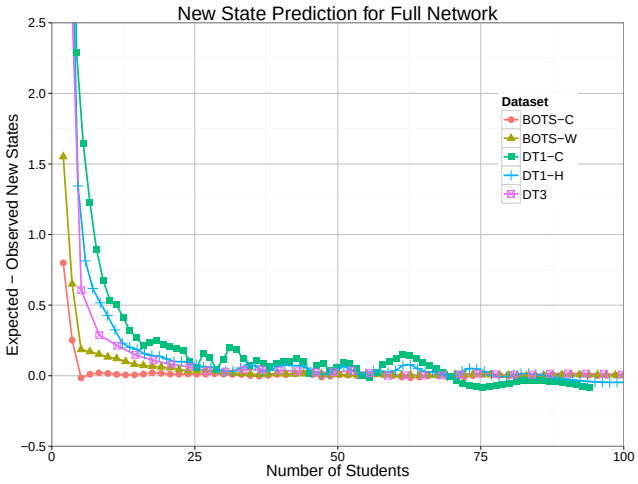

Figure 3: The average absolute error between the estimated number of new states and the observed new states over the number of students for all problems in each of the four datasets. P_0 accurately predicts the observed values after roughly 10 students, rarely being off by more than one after that.

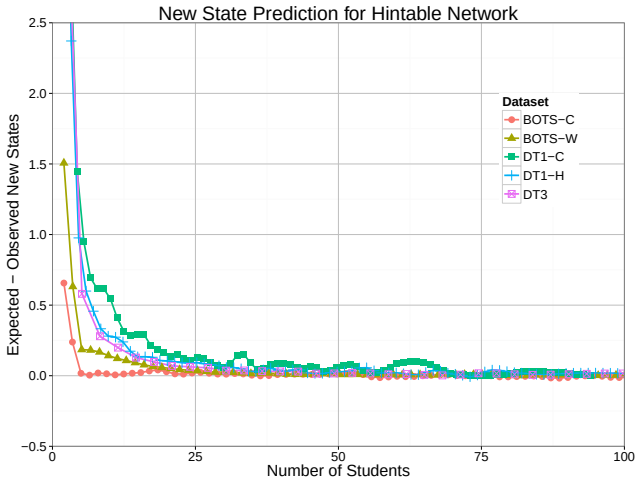

Figure 4: For the hintable states, the average difference between the estimated number of new states and the observed new states over the number of students for all problems in each of the four datasets. P_0 accurately predicts the observed values after roughly 10 students, rarely being off by more than one after that.

3.1 H1: Prediction of New States

In order to evaluate P_0 for the prediction of new states (states that are frequency $= 0$ on time T_i, but will be frequency $= 1$ on T_{i+1}. At each T we add an additional student and compare the expected number of frequency 1 states, E_{S1}, vs. the observed number, O_{S1}. Across all five datasets, Figure 3 shows the differences between the expected and observed number of new states. The $P_O \times Interactions$ prediction for new states follows closely with the observed number, the estimates increase in accuracy rapidly over the first ten students and are rarely off by more than a fraction of a state afterwards. Figure 4 shows the results of running this process on only the hintable portion of the interaction network for each data set.

3.2 H2: Network Coverage

We have defined network coverage I_C as the proportion of interactions which lie within the previously observed network. Another interpretation is that I_C is the probability of an interaction resulting in a state that has been previously observed. This value is the complement of P_0. Figure 5 and 7 display the results of network coverage and its growth as additional students are added.

3.3 H3: Predicting Future Network Size

In order to further evaluate the use of P_0 and I_C we calculated a prediction for the final size of the network, given the number of students in each dataset, at each time stamp. The equation for this prediction is:

$$|V(IN)| = (NewSample * P_0) + U_T. \quad (3)$$

Where $|V(IN)|$ is the number of unique vertices (states) in the final network, *NewSample* is the number of new interactions added, P_0 is the estimation of new states added, and U_T is the number of unique states observed at time T. The results are averaged across all problems for each dataset and are presented in figures 8 and 9. This prediction rapidly improves and after roughly 20% of the sample is added, can accurately predict the final number of unique states for the network. This combined with the accuracy of P_0 reveals the short term and long term accuracy for the estimator.

3.4 H4: Comparing State Matching Functions

The network coverage metric, I_C, allows an easy method of estimating the differences in state matching functions and student network overlap. We can use I_C with two potential matching functions, and get an estimate of the remaining network, to quickly compare different potential state representations as well as to find a state generalization that will allow for a desired amount of network coverage.

The estimate based on the above methods has proven useful for comparing State Matching functions to help determine which produces more relevant hints. Figure 6 shows the BOTS interface, with the user's program (codestate) and the game world (worldstate) both illustrated. In previous work investigating the Cold Start problem on the BOTS data set, we measured "coverage" in terms of how much of the newly added test data was already present in the training set [9, 14]. Compare this analysis to Figure 5 which shows the estimated probability that a student's next action will result in an observed state, I_C. After 100 students, the probability that a student will generate a new *codestate* is still quite high, $P_0 > .25$. In comparison, after the same number of students, the probability of generating a new *worldstate* is extremely low, $P_0 < .02$. This result supports both our intuition and our results from the previous work, that students will continue to generate new *codestates*, but that these different *codestates* will collapse to previously observed *worldstates*.

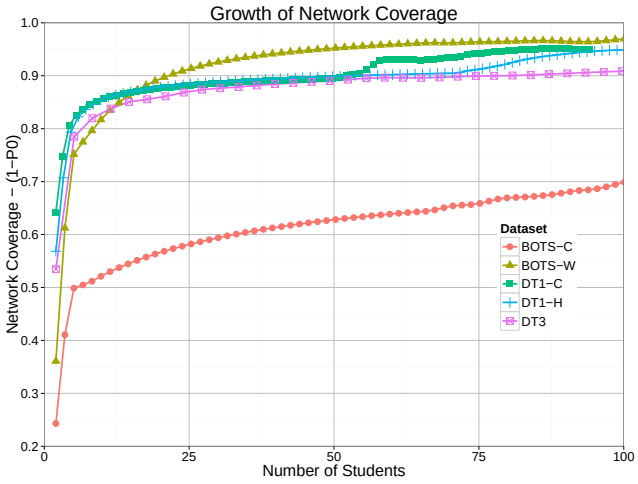

Figure 5: The estimated network coverage I_C for each of the 5 datasets, note the poor coverage for the BOTS-C dataset. The BOTS-W state is more general and has the much higher coverage.

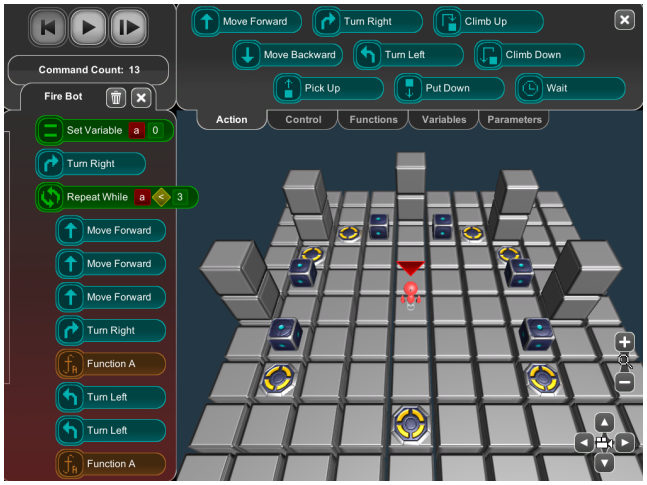

Figure 6: An image of the main gameplay interface for BOTS. The left hand side of the screen shows the user's program, used to derive code states. The right-hand side shows the game world, where the program output determines the world states.

3.5 H5: Comparing Populations

Samples from different populations have different resulting interaction networks. The size of the represented network can tell us about the similarity of student approaches in the sample. If students are more alike in the types of actions they perform, fewer students will be needed to achieve a similar amount of overlap. We can also see that adding students from a dissimilar population will not always increase estimated network coverage (I_C), and can potentially decrease it. This has implications about the importance of building hints for one population and applying it for another. In other work we have already shown that different groups are likely to visit different parts of the networks [4]. Here we expand on that analysis by showing that the two

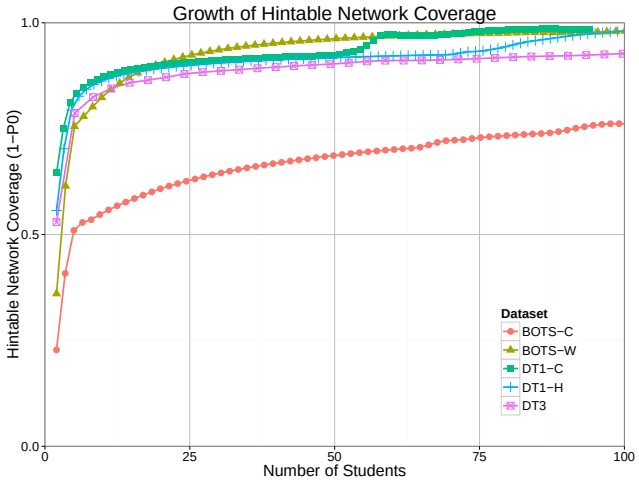

Figure 7: For the hintable network: the estimated network coverage I_C for each of the 5 datasets. Even the lowest performing hint network BOTS-C reaches roughly 70% coverage by 100 students.

Table 2: Different populations have different spread in problem exploration.

Group	P_0	States	Interactions	F_1
Hint	0.09	514.61	2709.84	250.09
Control	0.10	720.12	3904.92	340.00

groups, while having the same amount of network coverage, have a different number of unique states. Table 2 shows the results between the Hint group, which received hints on a subset of the problems, and the Control group which never received hints. This corresponds with results from Eagle et al. [4] in which they uncovered significant differences in the student overall approaches. This result adds to that an estimation of how complete each network was, revealing that additional data was not likely to change the result. It also shows some evidence for a *trail blazing effect*. When provided hints, students collectively explore a smaller area of the state space.

3.6 Estimating the effect of filtering

Visualizations must struggle with an "information to ink" ratio. There is a trade-off between displaying full information and overwhelming the viewer, and displaying only the most frequent states and potentially misleading the viewer by eliminating information. *InVis*, a visualization tool for exploring Interaction Networks allowed users to filter by frequency[11]. We can use the Good-Turing Estimation to calculate the amount of information removed by filtering frequency of a certain degree. P_0 is the proportion of the network missing, $I_{C_{>r}} = I_C - P_{1--r} + P_0$, where r is a threshold value for removing low frequency states, and $P_1 - -r$ is the sum of P_1 through P_r. This should be a useful metric for visualizations for measuring the amount of network that is hidden by filtering. It is also useful to show that sometimes a large number of graphical elements can be removed, with only a small amount of interaction information lost.

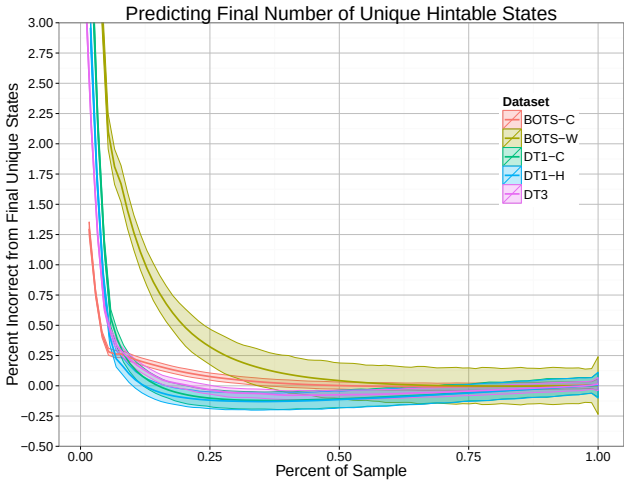

Figure 8: Prediction of total final number of states, as observed number of states increases. Note that for small t, the estimate is very high (up to 300% over prediction), but becomes fairly accurate after roughly 20% of the sample is measured.

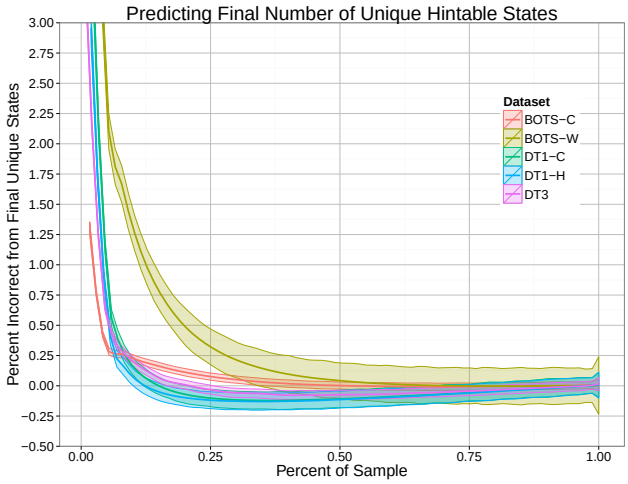

Figure 9: Prediction of total final number of goal states, as observed number of states increases. Note that for small t, the estimate is very high, but becomes an underestimate as t increases. P_0 can predict the number of additional hintable states that can be added for a additional sample of data.

4. DISCUSSION

Good-Turing Estimation works well in the contexts of interaction networks. We were able to provide an easily calculable estimate of the proportion of the network not yet observed P_0. This value alone is a useful high level metric for the percentage of times a student interaction results in a previously unobserved state. The P_0 score for the hintable network is likewise an estimate of the probability that a student will "fall off" of the network from which we can provide feedback. Our network coverage metric I_C allows a quick and easy to calculate method of comparing different state representations, as well as quantifying the difference. We believe that this metric can replace the commonly used cold start method of evaluating the "hintability" of a network. I_C is also valuable to quickly gauge the applicability of a new domain to interaction networks. The majority of the calculations can be performed on the transactional data. The growth trends for our five datasets were often clear after only ten students.

Our network estimators also have implications given our previous theories on the network being a sample created from biased (non-random) walks on the problem-space, as the more homogeneous the biased walkers are, the faster the network will represent the population and the fewer additional states will be explored. We revisited our previous results [4], and found that students with access to hints explored less overall unique states. This implies that the students were more similar to each other in terms of the types of actions and states they visited within the problem. Overall, this result supports the idea that different populations of students will have different interaction networks. The implications of this for generating hints are great. Building hints on one population might not work as well in another, and adding interventions or hints can dramatically reduce the number of states visited by the students. Future work should explore the possibility of having multiple network representations and choosing to match the student with the one closely resembling them.

As you can see in figure 8, our estimator starts out drastically overestimating the number of unobserved states in the network. As we collect data, this eventually becomes a slight underestimate, eventually converging on the correct number of states. One explanation for why this might be the case is the method by which undiscovered states are added to the network. By using this model for our estimator, we are making an assumption that states are selected independently of one another. At the beginning, when data is sparse, this assumption is not particularly harmful, since undiscovered states are relatively common. However, as our dataset becomes richer, we underestimate the probability of adding an unobserved state because we do not take into account the effect of "trail-blazing" which increases the probability of adding additional unobserved states after the first. Eagle and Barnes found that interaction networks had properties of scale-free networks. [5]. In particular, their degree distributions follow a power law, with a few vertices having much higher degree than the average for the network. It is likely that taking into account the scale-free and hierarchical nature of the networks will provide methods to improve on our estimators.

5. CONCLUSIONS AND FUTURE WORK

We have adapted Good-Turing frequency estimation for use with networks built from student-tutor interactions. We found that the estimator for the missing proportion of the network P_0 was accurate in predicting the number of new states discovered with new data. We also found that we could accurately measure network coverage with I_C for both the regular network, as well as the network of hintable states. This provides us with a metric to compare different state representations as well as determine the suitability of inter-

action network methods to different tutoring environments. We were also able to use these metrics to provide accurate predictions for the size of networks expected given more data samples, which will be useful for predicting the amount of additional data needed to provide a desired amount of hintable network coverage. Finally, we used the estimate of network coverage to compare different student populations to show that the addition of hints in one environment had an effect on the number of states explored by students.

Future work will include expanding on these *global* measures of the network and exploring *local* measures of coverage. Rather than compute coverage for the entire network we can use methods such as approach map regioning [4] to find meaningful sub-networks and calculate the metrics for those. The region level values of P_0 can estimate the "riskyness" of certain approaches to the problem. The I_C metric can direct attention to parts of the network that are not well explored, perhaps allowing additional hints to be obtained by starting advanced users in those areas.

6. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. #0845997, #1432156, #1015456, #0900860 and #1252376.

7. REFERENCES

[1] 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.

[2] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[3] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26(0):225 – 238, 2012.

[4] M. Eagle and T. Barnes. Exploring differences in problem solving with data-driven approach maps. *Proceedings of the Seventh International Conference on Educational Data Mining*, 2014.

[5] M. Eagle, D. Hicks, P. III, and T. Barnes. Exploring networks of problem-solving interactions. *Proceedings of the Fifth International Conference on Learning Analytics and Knowledge (LAK 15)*, 2015.

[6] M. Eagle, M. Johnson, T. Barnes, and A. K. Boyce. Exploring player behavior with visual analytics. In *FDG*, pages 380–383, 2013.

[7] 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.

[8] W. A. Gale and G. Sampson. Good-turing frequency estimation without tears*. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

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

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

[11] M. W. Johnson, M. Eagle, and T. Barnes. Invis: An interactive visualization tool for exploring interaction networks.

[12] B. Mostafavi, M. Eagle, and T. Barnes. Towards data-driven mastery learning. *Proceedings of the Fifth International Conference on Learning Analytics and Knowledge (LAK 15)*, 2015.

[13] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10:98–129, 1999.

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

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

[16] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[17] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[18] J. Stamper, T. Barnes, L. Lehmann, and M. Croy. A pilot study on logic proof tutoring using hints generated from historical student data. *Proceedings of the 1st International Conference on Educational Data Mining (EDM 2008)*, pages 197–201, 2008.

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

[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.