Graph Analysis of Student Model Networks

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

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

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

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

ABSTRACT

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

1. INTRODUCTION

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

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

2. RELATED WORK

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

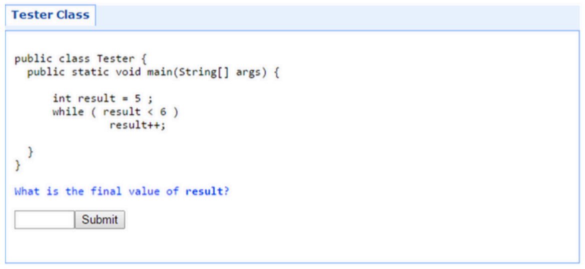
[1]`http://ceur-ws.org/Vol-1183/gedm2014_proceedings.pdf`

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

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

3. THE APPROACH

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



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

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

  }
}
```

What is the final value of result?

Figure 1: Exercise jwhile1

path length, and average node weight can be good indicators of student knowledge compared to the amount of activities done or overall measures of assessment like success rate on exercises. We further explore these graph metrics in relation with motivational factors drawn from a learning motivation theory.

3.1 Domain Model and Content

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

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

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

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

3.2 Graph Metrics

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

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

3.3 Measures of Activity

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

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

3.4 Motivational Factors

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

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

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

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

4. EXPERIMENTS AND RESULTS

4.1 Dataset

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

4.2 Graph Metrics and Learning

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

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

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

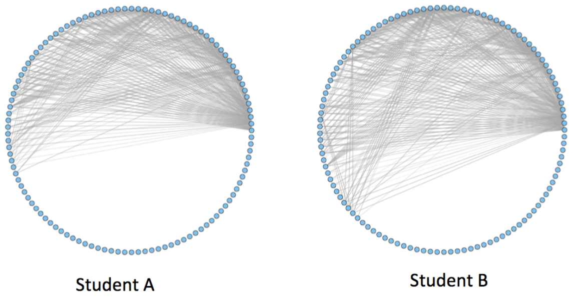


Figure 2: Graph representation of two students.

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

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

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

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

4.3 Metrics and Motivation

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

5. DISCUSSIONS AND CONCLUSIONS

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

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

6. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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