Applying Graph Sampling Methods on Student Model Initialization in Intelligent Tutoring Systems

Marija Vištica, Ani Grubišić, Branko Žitko
University of Split, Faculty of Science, Teslina 12, Split, Croatia,
marija.vistica@gmail.com, ani.grubisic@pmfst.hr, branko.zitko@pmfst.hr
Tel: ++385 21 385 133, Fax: ++385 21 384 086

Abstract

In order to initialize a student model in intelligent tutoring systems, some form of initial knowledge test should be given to a student. Since we cannot include all domain knowledge in that initial test, a domain knowledge subset should be selected. In order to generate a knowledge sample that represents truly a certain domain knowledge, we can use sampling algorithms. In this paper, we present five sampling algorithms (Random Walk, Metropolis-Hastings Random Walk, Forest Fire, Snowball and Represent algorithm) and investigate which structural properties of the domain knowledge sample are preserved after sampling process is conducted. The samples that we got using these algorithms are compared and we have compared their cumulative node degree distributions, clustering coefficients and the length of the shortest paths in a sampled graph in order to conclude about the best one. This approach is original as we could not find any similar work that uses graph sampling methods for student modeling.

Keywords

Intelligent tutoring systems, graph sampling, domain knowledge, domain knowledge graph

Introduction

Throughout history, much time was dedicated to the idea of using the computer as a personal teacher. The complexity and the possibility of executing that idea grew after the appearance of intelligent teaching, with reference to artificial intelligence. Therefore, we can say that intelligent tutoring systems began to develop at the same time with artificial intelligence, in the 1950s, and they are just as interesting now as they were before.

Intelligent tutoring systems (ITS) are computer systems based on the artificial intelligence techniques that simulate human tutors who know what they teach, who they teach to and how to teach, so it is said they belong to the category of the knowledge-based systems ((Carbonell, 1970), (Sleeman & Brown, 1982), (Wenger, 1987), (Ohlsson, 1987), (Shute & Psotka, 1996)). More than forty years of development divides today's intelligent tutoring systems from the first attempts. Many systems have been developed, implemented and tested in the teaching process in schools and universities, and a unanimous agreement on the architecture of ITS has been reached. The traditional structure of ITS consists of four components (Burns & Capps, 1988): domain knowledge, the tutor module, student module (Self, 1974) and a communication module (Woolf, 1992).

The domain knowledge carries the information necessary 1) for designing the structure and the course content presentation plan, 2) for student diagnosis and 3) for communication with the student is called domain knowledge. In many ITSs, domain knowledge is presented in a form of a conceptual graph or network or ontology (Tangjin & Xiahong, 2010). These knowledge structures have advantages that enable solving an optimization problem related to defining some sort of domain knowledge representative subset. Knowing this, we can transform domain knowledge into graph. The domain knowledge in the ITS can be presented in a form of domain knowledge graph where the domain knowledge is a set of triplets (K_1, r, K_2) with concepts K_1 and K_2 connected with relation r (Grubišić, Stankov, Peraić, 2013). We say that K_1 is the super-concept (or a parent) of K_2, and K_2 sub-concept (or a child) of K_1. With regard to the hierarchy, such a set can be easily presented with the directed acyclic graphs. The direction of the edge in the directed acyclic graph goes from a super-concept to a sub-concept (from a parent to a child).

The problem that exists in the ITSs is how to start learning and teaching process. Specifically, the problem is how to initialize a student model. Usually this is done with an initial test or questionnaire that should include questions about a representative sample of knowledge that the student needs to learn. As domain knowledge can be very extensive, it is necessary to find sub-graph of a domain knowledge graph that will include the concepts and relationships that are relevant to some domain knowledge. A domain knowledge sample can be generated using some of the most known graph sampling algorithms. In this paper, we compare several sampling algorithms: Random Walk (Deo and Gupta, 2001), Metropolis-Hastings Random Walk (Metropolis, Rosenbluth, Rosenbluth, Teller, Teller, 1953), (Hastings, 1970)), Forest Fire (Leskovec, Kleinberg and Faloutsos, 2005), Snowball ((Coleman, 1958), (Goodman, 1961)) and Represent algorithm (Grubišić, 2012).

Comparing samples gained after conducting different sampling methods over the same graph, enables finding “a measure for a good sample”. In this work, we are comparing parameters necessary for graph samples comparisons: the average degree of the nodes, the average clustering coefficient, the average shortest path length between two nodes, the density and the number of related and highly related components. In addition, we are also concentrating on the degree distribution, clustering coefficients and shortest paths and those distributions will be shown with the graphs. For presentation simplicity, we will show the cumulative distributions.

In the next chapter we will describe how the problem of student model initialization was resolved in other similar approaches. In the third chapter, we will describe complex networks and describe the chosen sampling methods. Furthermore, in the fourth chapter, we will present the results of the algorithms implementation.

Related work

Initial tests or some questionnaires are usually used for student model initialization. They are usually created manually by teachers for particular domain knowledge. When the domain knowledge changes, then all the questions for initial test have to be created and selected once again.

A CLARISSE (Aïmeur et al., 2002) uses an intelligent pre-test where the questions are focused on the “important” concepts. The selection of the “important” concepts, that is, the selection of questions related to the “important” concepts, is done by cluster analysis. The main drawback is the fact that the starting set of questions is manually created by teacher. That larger set of questions has to be tested by students, and then cluster analysis reduces the pre-test. This process is rather time consuming.

In a Web-EasyMath (Tsiriga & Virvou, 2004), the initial test contains representative questions that cover the whole domain being taught. Shortcomings of this approach are manually created representative questions used in the initial test, as the creation and selection of those questions has to be done for every new domain knowledge. The same approach is used in (González, Burguillo, Llamas, & Laza, 2013).

In a SIETTE (Guzmán & Conejo, 2004), a student model is initialized using a pre-test of the whole domain knowledge based on the hierarchically structured curriculum and the “complete assessment mode”. The mechanisms used to carry out the selection of the most suitable question items are based on a psychometric theory called Item Response Theory (IRT). The main disadvantage is the manual creation of question items that have to cover the whole domain knowledge.

A LS-Plan (Limongelli, Sciarrone, & Vaste, 2008) uses the Knowledge Space Theory and the Felder-Silverman’s Learning Styles Model. The student model is initialized using initial questionnaire prepared by teacher. Again, already mentioned disadvantage of manually created and selected questions is present. The similar approach is used in Wayang Outpost that uses 28 problems in a pre-test (Ferguson, Arroyo, Mahadevan, Woolf, & Barto, 2006).

In a DEPTHS (Jeremic, Jovanovic, & Gasevic, 2009), student model is initialized based on the student’s self-assessment. Shortcoming of this approach is subjective self-assessment that cannot give valid predictions about student’s knowledge.

In a LearnSquare (Esichaikul, Lamnoi, & Bechter, 2011), the student model initialization is based on pre-test results analyzed by Dempster-Shafer (DS) theory. The results from the pre-test answers are input to the DS formula for determining the level of the student’s knowledge in all domain knowledge concepts. The creation of questions and their selection is not described.

Graph sampling

Since domain knowledge can be extensive, it is necessary to define such a subset of the domain knowledge that will worthily represent the whole domain knowledge. Thus, the representative subset of domain knowledge includes all the concepts and relationships that are relevant for some domain knowledge. In this paper, the selection of concepts in domain knowledge representative subset is done uniformly for every domain knowledge, using graph sampling methods. In this way, the problems related to semantical analysis done by field experts are avoided. The diversity of semantic representation showed the necessity of finding a mechanism for non-semantic mathematical determination of the domain knowledge representation independent of domain knowledge itself.

Our idea for determining domain knowledge representation has found confirmation in the field of social networks, citation networks and communication networks because of their size (so-called real world or complex networks). In these areas, graph sampling or graph reduction is used. Graph sampling or reduction is the process of selecting the nodes from the network so that the sample is a smaller network, which retains the properties and similar structure that had a larger network. More precisely, for the graph $G = (V, E)$, sample is a graph $G_1 \subseteq G$ such that $G_1 = (V_1, E_1)$, $V_1 \subseteq V$ and $E_1 \subseteq E$.

Complex networks

A network is a set of nodes with links between them. Mathematically, it is represented by the graph in which the nodes are vertices and the links are edges. It can be said that the network comes from the real world, and that the graph is a network abstraction.

The networks from the real world are not static, nor small, and often do not fit the graph theory's models such as regular graphs and random graphs. In this century, the focus of research is on topology, growth and dynamic networks' applications, which are often called the complex networks or the real world networks (Bilgin, Yener, 2008). Perhaps the most prominent examples of such networks are social networks. However, metabolic networks, transportation networks, etc. also behave similarly (Newman, 2003).

The description of the network structure involves determining its structural properties such as: node degrees, node distribution, clustering coefficient and clustering coefficient distribution, the average length of the shortest path and the shortest path length distribution, density, diameter, betweenness, etc. The most significant topological properties are the small-world effect, degree scale-free distributions, correlations and the presence of clustering ((Boccaletti, Latora, Moreno, Chavez, Hwang, 2006), (Newman, 2003)). Thus, in this work we analyze the structural properties of the network, and based on these, make conclusions about its topology.

Complex networks are sometimes not available throughout, so studying and analyzing the network's properties may lead to understanding its structure without studying the whole network. Below we explain main network properties used in this paper as explained in (Leskovec, Faloustos, 2006), (Borge-Holthoefer and Arenas, 2010) and (Boccaletti, Latora, Moreno, Chavez, Hwang, 2006).

The *small world effect* is complex network's property by which the average shortest distance between two nodes is small. The shortest path between two nodes *u* and *v* is property that tells us in how many steps we can get from the node *u* to the node *v*. It comes from a well-known experiment of American social psychologist Stanley Milgram from the 1960s, in which he asked randomly selected people in Nebraska to send a letter to randomly selected people in Boston who were known with full name, occupation and location. The idea was that people send letters to those people whom they assume might know the target people. The number of average steps where the letter came from the sender to the recipient was six (Travers and Milgram, 1969).

Small world networks were described by Watts and Strogatz noting that the real networks are somewhere between regular and random networks (Watts and Strogatz, 1998). They showed that starting from a regular graph and connecting the links in the graph randomly will reduce the average length of the shortest path between two nodes.

With regard to node degrees in the network, the complex networks also have *scale-free property* which means that node degrees follow the power-law curve. In other words, a few nodes have a large degree, while a large number of nodes have a small degree. Nodes that have a large degree are called hubs, and their existence can be explained by the growth of the network and the preferential connectivity property. Specifically, as the network grows, more likely a new node will be connected to the node that has more connections.

A *clustering coefficient* in the graph represents the probability that two nodes, which are connected with the same node in the network, are also connected. In the words of social networks, that is the probability that one's two friends are also friends. More precisely, it is a ration of the number of cycles of length 3 that are passing through the vertex *v* and number of all possible cycles of length 3, which could pass through that node.

The clustering coefficient is closely related to a network's *density*. The density is the ratio of the number of edges in relation to the possible number of edges in the network. The density value is between 0 and 1. A graph without edges has the density 0, and fully connected graph has the density 1.

A *component* of a graph *G* is the maximum possible sub-graph of the graph in which every two nodes are connected. The number of connected components is the number of such sub-graphs in a graph. In a directed graph, component is strongly connected if there is a directed path between any two nodes in the sub-graph.

Sampling methods

There are two goals of graph sampling process. The first is to find a sample and retain the network's properties. Here, we refer to the network properties, such as the average node degree or the average clustering coefficient. The second goal is to find a sample and preserve the network structure, such as shortest path distribution in the network or clustering coefficient distribution.

Sampling algorithms elaborated in this work mainly arose from the need for analyzing social networks. The exception is Represent algorithm that is designed for the domain knowledge sampling in the intelligent tutoring systems. Most of the presented algorithms are designed for undirected graphs, so we have modified them in order to use them on directed acyclic networks.

Except the Represent algorithm, all these algorithms are algorithms in which we start from the random initial node, and then, in each iteration, we examine node neighbors and add some of them to the sample. The term *neighbor* of a node *u* is referring to the node *v* such that there is a directed edge that goes from *u* to *v* (*u* is the parent of *v*). The problem that can occur when generating samples in the acyclic directed graph is the absence of neighbors - we come to the node called a *leaf* that has out-degree equal to 0. In that case, we continue sampling from some other randomly selected node that is not already in the sample.

The effect of certain sampling methods will be shown in the following sections, using the example graph which is composed of 30 nodes (Figure 1).

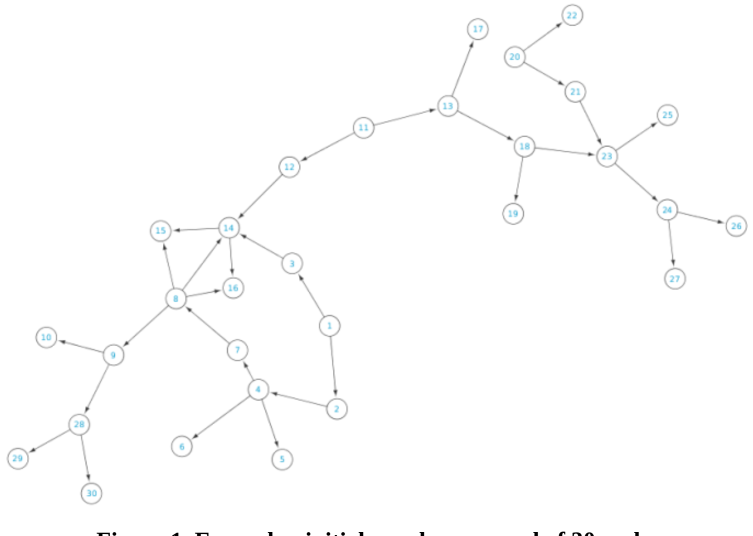

Figure 1. Example - initial graph composed of 30 nodes

Random Walk (RW)

A Random Walk method simulates a random walk on the graph. The initial node is chosen randomly and uniformly. Then, using a 0.85 probability among all its neighbors, if they exist, we choose one, again randomly and uniformly, and if it is not already in the sample, add it to the sample. With probability 0.15 (a fly-back probability) in a random walk, again we visit the initial vertex. It allows finding more neighbors in the sample. The probability does not have to be 0.15, but it is typically used (Frank, Huang, Chyan, 2012).

In the directed unconnected graph, it is possible that the chosen node has no out- degree or that there is no node in the component that could be added to the sample. That is the reason why we define a period *T* and an expected growth size *M* in that period and after *T* iterations check whether the sample growth is large enough and if not, select again initial node. This way we ensure that the sample will reach the assumed size (Wang, Chen, Zhang, Xu, Jin, Hui, Deng, Lin, 2011).

The pseudocode for this method is shown on Figure 2. and the result of applying that method on graph from Figure 1. is shown on Figure 3.

```
Random Walk Algorithm
1. We have a directed acyclic graph, number of iterations $T$, sample
   growth size $M$ in the period $T$ and fly-back probability $p$
2. Select a node randomly and uniformly
3. Until it does not reach the required sample size:
      •   with a probability $1-p$ select a neighbor node and if it is not
          in the sample add him
      •   with a probability $p$ go back to the initial vertex
      •   if the number of added nodes with a number of iteration $T$ is
          less than $M$ return to step 2
```

Figure 2. Random Walk Algorithm

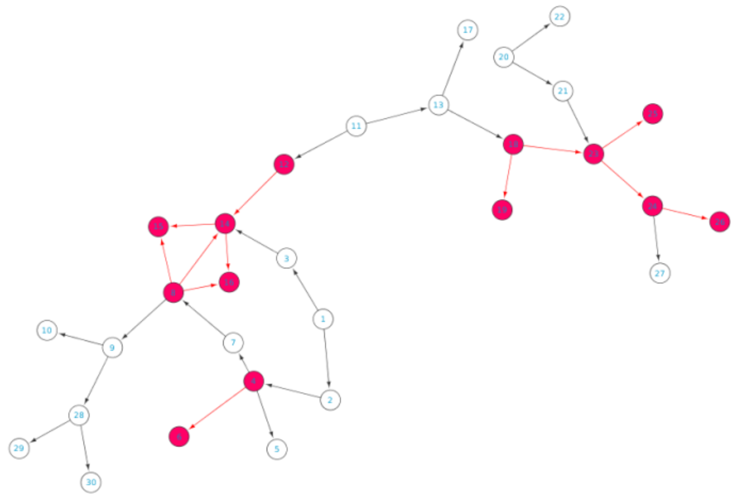

Figure 3. The initial graph after Random Walk sampling

Metropolis-Hastings Random Walk (MHRW)

In a Metropolis-Hastings Random Walk, a sample is built considering the distribution of node degree probability, to avoid the appearance in the sample that the nodes with a high degree are preferred (Nesreen, Neville, Kompella, 2010).

We are defining a function $Q(v) = d_v$, where is d_v degree of node v. Among the neighbors of node v, we select randomly node w, and then generate number p uniformly between 0 and 1. If $p \leq \frac{Q(v)}{Q(w)}$, node w is added to a sample (Nesreen, Neville, Kompella, 2010).

In each iteration, if the selected neighbor has a lower degree than the degree of its observed parent, it will be certainly added to the sample. On the other side, every neighbor with a higher degree will not be accepted.

In this algorithm, also there is possibility that a node has no neighbors (it is a leaf), and in that case, as in Random Walk, we choose a new randomly selected node to continue.

The pseudocode for this method is shown on Figure 4. and the result of applying that method on graph from Figure 1. is shown on Figure 5.

```
Metropolis-Hastings Random Walk Algorithm
 1. $v$ is initial node
 2. until it is fulfilled the halt criteria
    1.  chose a node $w$ uniformly among the neighbors of $v$
    2.  generate a number $p$ between 0 and 1
    3.  if $p \le min\left(\frac{Q(v)}{Q(w)}\right)$, $w$ is next node, otherwise stay at node $v$
```

Figure 4. Metropolis-Hastings Random Walk Algorithm

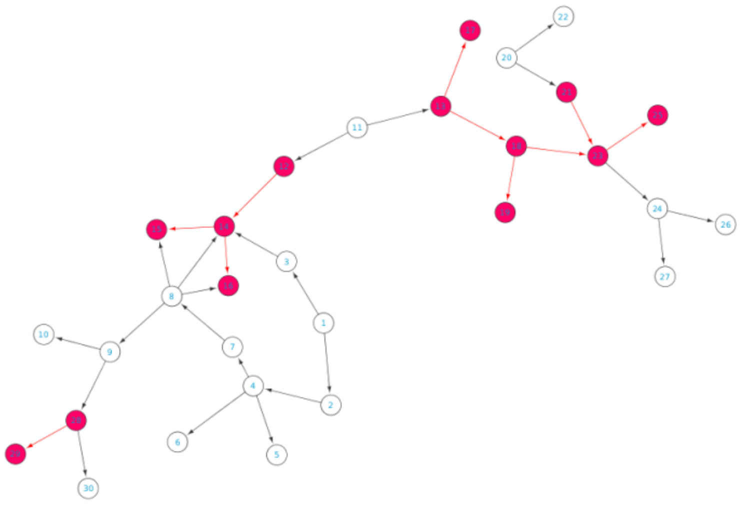

Figure 5. The initial graph after Metropolis-Hastings Random Walk sampling

Snowball

A Snowball is a variant of Breadth First Search where it is possible to limit a search depth *s* and a number of neighbors *k* that are added to the sample (Goodman, 1961). The idea is to begin from some random set of people size *k* and that every man names, for example, *k* of his friends. After that, each of the named people name new *k* people that make the second sampling stage. Naming continues until *s*-th stage (*max_depth*) is reached. In each iteration, in the sample are added only nodes that are not already in the sample and that is the way to avoid repeating. In networks that are not about people (when naming is not possible), the neighbors who enter the sample can be randomly selected.

The pseudocode for this method is shown on Figure 6. and the result of applying that method on graph from Figure 1. is shown on Figure 7.

```
Snowball Algorithm
 1. S is a queue k randomly selected nodes
 2. Select from the queue (dequeue) a node v
    •  if the sample sized is reached, stop
    •  otherwise, take k randomly selected neighbors of a node v and
       add them into the queue S
  3.  If the queue S is not empty, continue from step 2
```

Figure 6. Snowball Algorithm

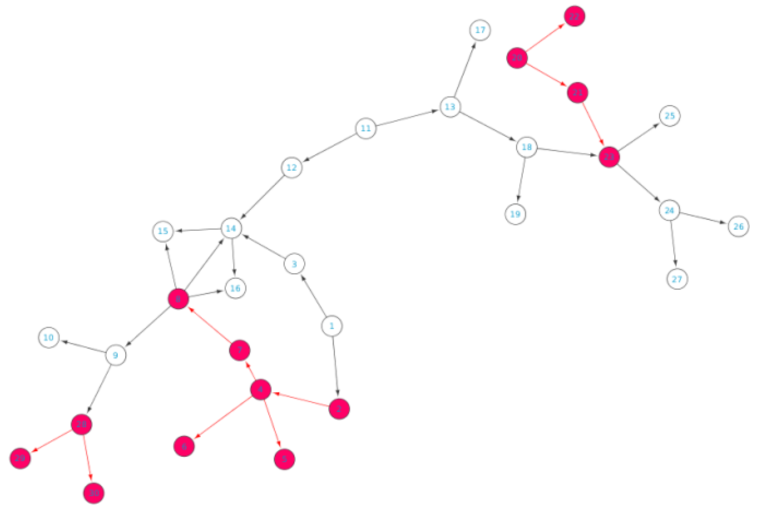

Figure 7. The initial graph after Snowball sampling

Forest Fire

A Forest Fire is a combination of the Snowball and the Random Walk. It starts with randomly and uniformly selection of a node and then, for a random number x, x neighbors of selected node, that have not been visited yet, are selected. The procedure is repeated for the selected neighbors, until the targeted sample size is reached. If the sample stops growing ("the fire went out"), and the target size is not reached, new initial node is selected and the process continues (Frank, Huang, Chyan, 2012).

The pseudocode for this method is shown on Figure 8. and the result of applying that method on graph from Figure 1. is shown on Figure 7.

```
Forest Fire Algorithm
Until it reaches the target sample size
    1.  $v$ is a randomly selected node
    2.  Generate $x$ and select $x$ neighbors of node who had not been
        visited yet and add them to a sample
    3.  Repeat the previous procedure to the selected neighbors
    4.  If the sample has not increased, go back to step 1
```

Figure 8. Forest Fire Algorithm

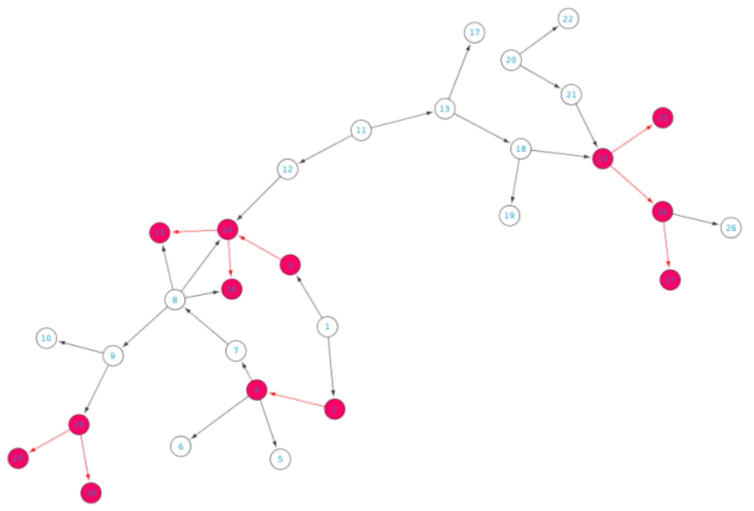

Figure 9. The initial graph after Forest Fire sampling

Represent Algorithm

A Represent Algorithm is based on the search of the shortest paths in a graph. For every *root* in a graph (node with in-degree 0), among all the shortest paths between that root and all reachable leaves, the one with the maximum length is chosen and added to a sample.

The pseudocode for this method is shown on Figure 10. and the result of applying that method on graph from Figure 1. is shown on Figure 11.

```
Represent Algorithm
    1.  Roots is the list of all roots, and leafs is the list of all
        leaves in the directed acyclic graph
    2.  For every root:
        •   Find the shortest paths from the root to the all leaves and
            among them select the one with the maximum length and add it
            to the sample
```

Figure 10. Represent Algorithm

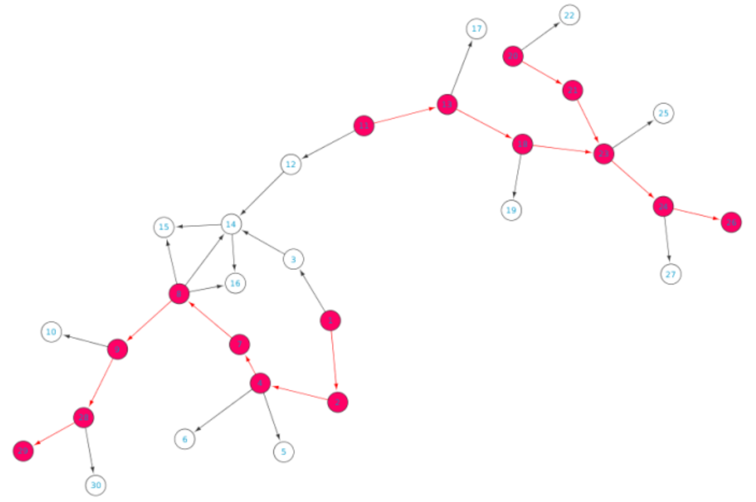

Figure 11. The initial graph after Represent sampling

Evaluation

To test previously described algorithms, we have used citation networks that are directed (one newer work (child) cites another older (parent)) and acyclic (if work A cites work B, and work B cites work C, that means that work C is the oldest one between them, so it is impossible that work C cites work B or that work B cites work A). Two networks, cit-HepPh and cit-HepTh, are downloaded from the page *http://snap.stanford.edu/* and are used for testing the properties of samples gained by applying mentioned sampling methods.

The algorithms were implemented in the Python programming language and its library Networkx. The gained samples were compared using a tool Gephi (*http://gephi.github.io/*).

The results shown in tables Table 1 and Table 2**Error! Reference source not found.** correspond to results we attained with the Gephi, and they show a basic data about the networks and generated samples: the number of nodes, connections, the number of connected components and strongly connected components.

Considering that Represent method has predefined sample size because the number of nodes in the sample depends on the number of nodes in the observed shortest paths, this method has determined the targeted sample size for the other methods. All other methods have as a parameter a size of the sample. In our test examples, we have set the target size to 36% of the original graph size (this percent is determined by the Represent Algorithm that generated sample with 12861 nodes – 12861/34546=0.36 – 36%). The number of connections between nodes, or network density, varies from method to method.

Table 1. Basic data about samples for cit-HepPh network

method	num nodes	num edges	num conn comp.	num strongly conn. comp.
original graph	34546	421578	61	21608
randomwalk	10364	95163	158	8281
metropolis	10364	77921	208	10187
snowball	7482	58442	1	5240
forestfire	10364	94088	1	5918
represent	12861	75215	96	10228

Table 2. Basic data about samples for cit-HepTh network

method	num nodes	num edges	num conn comp.	num strongly conn. comp.
original graph	27770	352807	143	20086
randomwalk	8331	86701	283	7194
metropolis	8331	90995	267	6909
snowball	8332	102715	1	5468
forestfire	8331	109496	1	5519
represent	10229	54040	218	8946

In the following tables (Table 3 and Table 4) we see that the *densest* sample is the one generated by the Forest Fire method, while the least dense and the closest to the original network is the one generated by the Represent Algorithm.

Table 3. The average values of cit-HepPh network samples

method	density	avg.degree	avg. cluster-ing coef.	avg. shortest path
original graph	0.0003532604	24.4067619985	0.1423980660	11.6929683986
randomwalk	0.0008860439	18.3641451177	0.1423495075	10.9662936521
metropolis	0.0007255070	15.0368583558	0.1383382135	6.1395873218
snowball	0.0010441135	15.6220261962	0.1414274989	11.7135521588
forestfire	0.0008760348	18.1566962563	0.1435007294	11.3741443194
represent	0.0004547668	11.6966021305	0.1213122805	11.4584582118

Table 4. The average values of the cit-HepTh network samples

method	density	avg.degree	avg. clustering coef.	avg. shortest path
original graph	0.0004575105	25.4092185812	0.1560097479	8.4601372939
randomwalk	0.0012493438	20.814067939	0.1519643048	5.4215171934
metropolis	0.0013112195	21.8449165766	0.1505411279	5.5178244424
snowball	0.0014797470	24.6555448872	0.1605898646	7.2584830648
forestfire	0.0015778151	26.2864001921	0.1596729024	6.4806495411
represent	0.0005165251	10.5660377358	0.1206504112	8.7349940612

Regarding the *number of connected components*, the Snowball and the Forest Fire as a result have the sample with one component, while the other samples have more connected components than there are in the original graph. The reason for this is because in these two methods add a larger number of neighbors of the same node in one iteration, while Random Walk and Metropolis-Hastings Random Walk methods add the most one neighbor in each iteration, depending on the random function. On the other side, the Represent Algorithm uses a completely different method of building the sample, and connecting the roots with the leaves and looking for the longest shortest paths between them, as expected, locates the longest paths in different components.

The *average node degree* is the closest to the average node degree in the original network in the sample generated with the Forest Fire method. The reason for this is that the number of neighbors of some node added in the sample varies depending on the random function, and in each iteration, greater number of neighbors is added. In other samples, the number of neighbors is limited (the Snowball method) or one selected neighbor is added. In the Represent Algorithm, only neighbors that are on the shortest path between a root and leaf are added, and there can be a maximum one of them for one shortest path.

The *average clustering coefficient* in all the samples reflects the original network (from 0.3% variation in the Snowball method to 15% difference in the Represent Algorithm compared to the original network), while the *average length of the shortest path* in the sample gained with the Snowball, the Forest Fire and the Represent Algorithm, is almost equal to the average length of the shortest path in the original network.

The figures Figure 12 and Figure 13 show the cumulative distribution of degrees, Figure 14 and figure 15 show cumulative distribution of the shortest paths in the networks and Figure 16 and figure 17 show cumulative distribution of clustering coefficients. Observing mentioned figures we can conclude that all samples preserve well the *node degree*, among them the closest to the original degree distribution is the sample that is generated with the Fire Forest. The graph of the cumulative distribution of clustering coefficient shows us also that all algorithms preserve well the *clusters* from the original network. The samples that are given with the Represent and Snowball methods have the largest deviation. That is because the algorithm stops if there are no more neighbors or it comes to the maximum depth search, and the search does not continue away from some other node. Finally, the frequency of the *shortest path length* in the graph is the most correct in the sample that is given with the Represent Algorithm, which is a consequence of the way the sample is created as a union of the shortest paths.

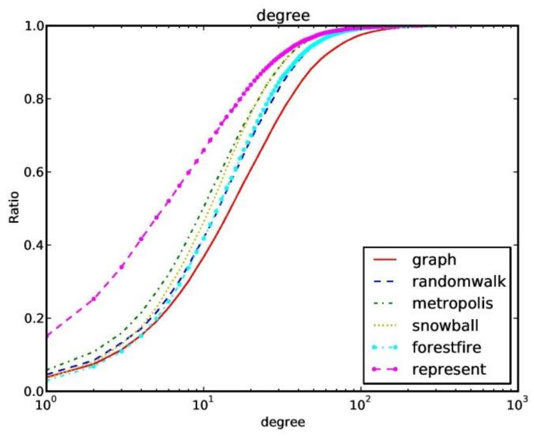

Figure 12. The cumulative distribution of degrees of cit-HepPh network

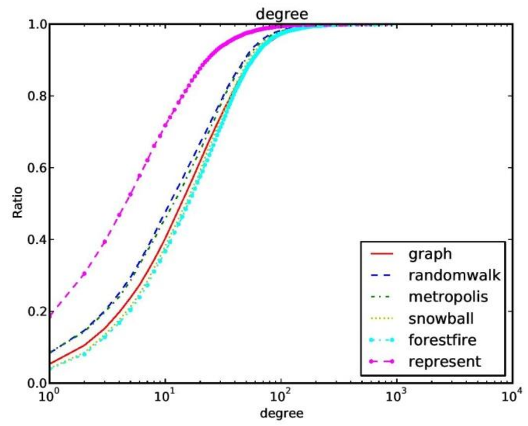

Figure 13. The cumulative distribution of degrees of cit-HepTh network

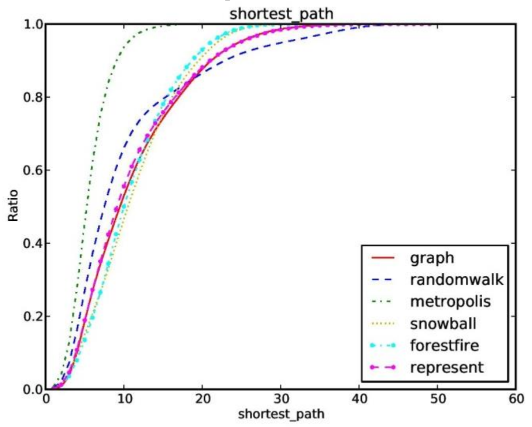

Figure 14. The cumulative distribution of the shortest paths of cit-HePh network

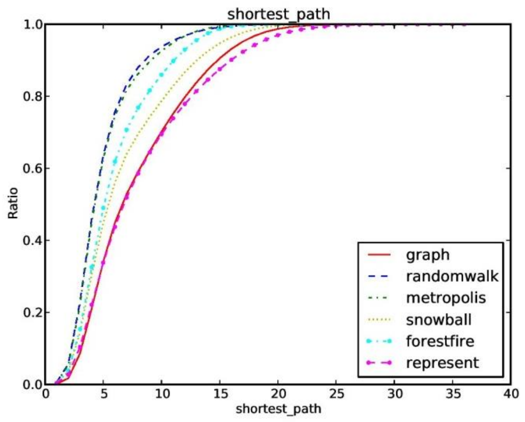

Figure 15. The cumulative distribution of the shortest paths of cit-HeTh network

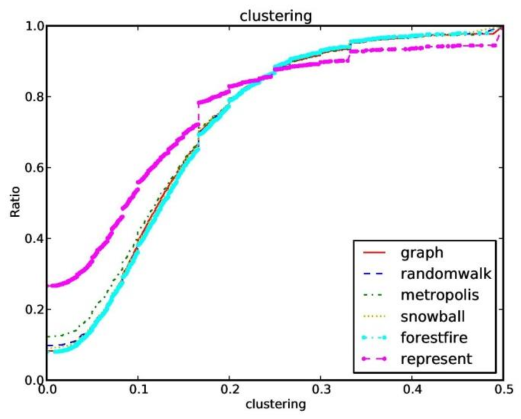

Figure 16. The cumulative distribution of clustering coefficients of cit-HepPh network

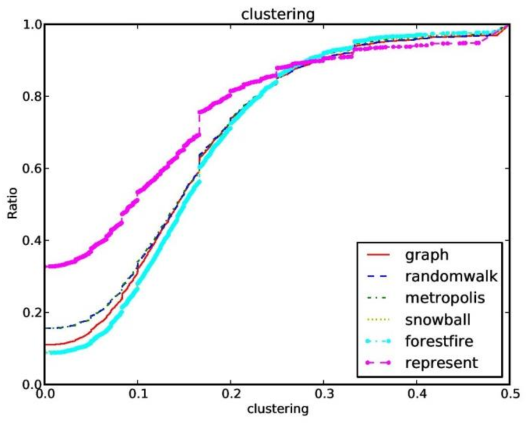

Figure 17. The cumulative distribution of clustering coefficients of cit-HepTh network

Conclusion

The goal of the ITS is to generate the course content offered to the student adapted to student's knowledge. To make this possible it is necessary to initialize the student model in a way that it grasps the state of the student knowledge without interrogating student too much. This can be done only using initial tests based on representative subset of domain knowledge. Since that subset of domain knowledge cannot easily be defined (if we use semantical selection done by experts), a rigid mathematical approach using sampling methods is more than welcome. A subset of domain knowledge that will represent whole domain knowledge can be generated using sampling methods undertaken from the field of complex networks (the RW, the MHRW, the FF, the Snowball) or algorithms created specially for the domain knowledge representation generation (the Represent).

This work describes the results of comparison of the five sampling methods: the Random Walk, the Metropolis-Hastings Random Walk, the Snowball, the Forest Fire and the Represent Algorithm.

The implementation was tested on two citation networks – cit-HepPh (34,546 nodes and 421,578 edges) and cit-HepTh (27,770 nodes and 352,807 edges). The samples generated with the Forest Fire and the Represent Algorithm more precisely represent the original network, only the Represent Algorithm has a longer run time. Therefore, this method is not recommended for large datasets. However, for the domain knowledge representation purposes, that is the best algorithm because it keeps the property of the shortest path length, which in this case, represents a path from super-concepts to sub-concepts.

References

1. Aïmeur, E., Brassard, G., Dufort, H., & Gambs, S. (2002). CLARISSE: A Machine Learning Tool to Initialize Student Models. In S. A. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (pp. 718–728). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/3-540-47987-2_72
2. Bilgin, C.C.; Yener, B.: „Dynamic Network Evolution: Models, Clustering, Anomaly Detection“, 2008.
3. Boccaletti, S.; Latora V.; Moreno, V.; Chavez M.; Hwang, D.-U.: „Complex networks: Structure and dynamics“, 2006.
4. Borge-Holthoefer, J.; Arenas, A. Semantic Networks: Structure and Dynamics. *Entropy* **2010**, *12*, 1264-1302.
5. Burns, H. L., & Capps, C. G. (1988). Foundations of intelligent tutoring systems: An introduction. In Poison M. C., & Richardson, J. J. (Eds.) Foundations of intelligent tutoring systems, pp. 1-19. Lawrence Eribaum, London.

6. Carbonell, J. R. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. IEEE Transactions on Man-Machine Systems, 11(4), pp. 190-202.
7. Coleman, J.S.: Relational analysis: The study of social organizations with survey methods. Human Organization, 17:28–36, 1958.
8. Deo, N. Gupta., P. (2001) Sampling the web graph with random walks. Congressus Numerantium, 149:143–154
9. Esichaikul, V., Lamnoi, S., & Bechter, C. (2011). Student Modelling in Adaptive E-Learning Systems. *Knowledge Management & E-Learning: An International Journal (KM&EL)*, *3*(3), 342–355.
10. Ferguson, K., Arroyo, I., Mahadevan, S., Woolf, B., & Barto, A. (2006). Improving Intelligent Tutoring Systems: Using Expectation Maximization to Learn Student Skill Levels. In M. Ikeda, K. D. Ashley, & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 453–462). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/11774303_45
11. Frank, D.; Huang, Z.; Chyan, A.: „Sampling A Large Network: How Small Can My Sample Be?“, 2012.
12. Freedman, R.; Ali, S. S.; McRoy, S.: „Links: what is an intelligent tutoring system?“, Intelligence, pp. 15-16, 2000.
13. González, C., Burguillo, J. C., Llamas, M., & Laza, R. (2013). Designing Intelligent Tutoring Systems: A Personalization Strategy using Case-Based Reasoning and Multi-Agent Systems. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *1*(4), 41–54.
14. Goodman, L. A.: „Snowball Sampling“, The annals of mathematical statistics, pp. 148--170, 1961
15. Grubišić, A.: „Model prilagodljivog stjecanja znanja u sustavima e-učenja“, doktorski rad, 2012.
16. Guzmán, E., & Conejo, R. (2004). A library of templates for exercise construction in an adaptive assessment system. *Technology, Instruction, Cognition and Learning*, *2(1-2)*, 21–43.
17. Hastings, W.K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". Biometrika 57 (1): 97–109. doi:10.1093/biomet/57.1.97. JSTOR 2334940. Zbl 0219.65008.
18. Jeremic, Z., Jovanovic, J., & Gasevic, D. (2009). Evaluating an Intelligent Tutoring System for Design Patterns: The DEPTHS Experience. *Educational Technology & Society*, *12*(2), 111–130.
19. Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD): 177–187.
20. Limongelli, C., Sciarrone, F., & Vaste, G. (2008). LS-Plan: An Effective Combination of Dynamic Courseware Generation and Learning Styles in Web-Based Education. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 133–142). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-70987-9_16
21. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. (1953). "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics 21 (6): 1087–1092. Bibcode:1953JChPh..21.1087M. doi:10.1063/1.1699114.
22. Nesreen, K. A; Neville J.; Kompella R.: „Reconsidering the foundations of network sampling“, Proc. of WIN, 2010.
23. Newman, M. E. J.: „The structure and function of complex networks“, SIAM review, 2003.
24. Self, J. A. (1974). Student models in computer-aided instruction. International Journal of Man-Machine Studies, 6(2), pp. 261-276.
25. Shute, V. J., & Psotka, J. (1996). Intelligent Tutoring Systems: Past, Present and Future. In Jonassen, D. (Ed.) Handbook of Research on Educational Communications and Technology. New York, NY: Macmillan.
26. Sleeman, D. & Brown, J. S. (1982). Introduction: Intelligent Tutoring Systems: An Overview. In Sleeman, D.H., Brown, J.S. (Eds.) Intelligent Tutoring Systems, pp. 1-11. Academic Press, Burlington, MA.
27. Tangjin, J., & Xiahong, W. (2010). Intelligent tutoring system based on computing conceptual graphs. In 2010 International Conference on Artificial Intelligence and Education (ICAIE) (pp. 60–62). doi:10.1109/ICAIE.2010.5641488
28. Travers, J.; Milgram, S. An experimental study of the small world problem. Sociometry 1969, 32, 425–443.
29. Tsiriga, V., & Virvou, M. (2004). A Framework for the Initialization of Student Models in Web-based Intelligent Tutoring Systems. *User Modeling and User-Adapted Interaction*, *14*(4), 289–316. doi:10.1023/B:USER.0000043396.14788.cc

30. Wang, T.; Chen, Y.; Zhang, Z.; Xu, T.; Jin, L.; Hui, P.; Deng, B.; Li, X.: „Understanding graph sampling algorithms for social network analysis“, Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference, pp. 123—128, 2011.
31. Watts, D. J.; Strogatz, S. H.: „Collective dynamics of small world networks“, Nature, pp. 393-440, 1998.
32. Wenger, E. (1987). Artificial Intelligence and Tutoring Systems. Morgan Kaufmann Publishers, Inc., California, USA.
33. Woolf, B. (1992). AI in Education, In: S. Shapiro (ed.) Encyclopedia of Artificial Intelligence, John Wiley & Sons, Inc., New York, pp. 434-444