Information Sciences xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Graph theory based model for learning path recommendation

Guillaume Durand *, Nabil Belacel, François LaPlante

National Research Council Canada, Institute for Information Technology, Moncton, NB, Canada

ARTICLE INFO

Article history:
Received 1 December 2011
Received in revised form 8 February 2013
Accepted 12 April 2013
Available online xxxx

Keywords:
Learning object recommendation system
Learning path
Graph theory
Soft computing

ABSTRACT

Learning design, the activity of designing a learning path, can be a complex task, especially for learners. A learning design recommendation system would help self-learners find appropriate learning objects and build efficient learning paths during their learning journey. Educational Data Mining (EDM) has provided an impressive amount of novelties related to learning object recommendation systems. However, most of the solutions proposed thus far do not take into account eventual competency dependencies among learning objects and/or are not designed for large repositories of interdependent learning objects. We propose a model to build a learning design recommendation system based on graph theory. From this model, we propose, implement and test an approach using the concept of cliques to recommend learning paths.

1. Introduction

For many years, learning design has been the subject of a significant amount of research papers and implementations. Learning design [8,12] consists in the process of designing learning activities. Usually, this task is thought to be assumed by a teacher preparing a formal learning session in his/her favorite Virtual Learning Environment (VLE). In the emerging Personal Learning Environments (PLEs [26]), the teacher's role has evolved since learners are expected to be more pro-active in the learning process. It is up to learners to choose learning material and organize their custom-fit learning process; the learner is the learning designer.

Many approaches have emerged in the last few years to facilitate learners' task of finding learning materials. Among those figures the contribution of data mining and recommendation technologies to provide learners with adequate learning materials. By recommending educational papers [35] or internet links [16], authors use collaborative and/or content-based filtering approaches for their recommendation systems. There are two ways of building such a recommendation system. Usually, some cluster profiles are discovered among a learner or content population using unsupervised learning (clustering) and from these new clusters, learners or content are predicted or classified by supervised learning methods (classifiers). Hence, depending on the classification, learners will be recommended with the resources completed by other learners sharing the same cluster and/or with content similar to those they themselves have already completed.

These approaches tend to satisfy the immediate interest of learners rather than considering the recommendation of content fitting in an organized sequential learning process. This approach might be problematic since learning design puts a lot of emphasis on the sequence of the learning material. More precisely, the sequence of the learning material has to match a particular learning process strategy in which each learning object (LO) is consistent with the former and the next one as well as the evolving learner knowledge [11]. To some extent, learning design tends to think of learning with more guidance, considering it as a journey among activities where the proposed itinerary makes as much sense as the learning materials offered. A learning design recommendation system should propose a sequence of learning objects in a well-defined order

* Corresponding author. Tel.: +1 506 861 0961; fax: +1 506 851 3630.
E-mail addresses: Guillaume.Durand@nrc.gc.ca (G. Durand), Nabil.Belacel@nrc.gc.ca (N. Belacel), Francois.Laplante@nrc.gc.ca (F. LaPlante).

0020-0255/$ - see front matter Crown Copyright © 2013 Published by Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.04.017

with a starting and ending point, rather than submit a sequence of unordered content elements simply recommended because they are used by "similar" learners [9]. Furthermore, the sequence of learning objects, or learning path, followed by a learner can be seen as an itinerary connecting several learning objects. The constraints on learning objects defined within the itinerary are not distance or time taken to reach one from another, but rather prerequisite competencies and competency gain. Considering sequential pattern [48] and process mining [36] could prove interesting in recommending learning designs to learners. However, the ways in which these technologies have been used so far are more oriented towards understanding the learner's interaction with content to discover general patterns and trends rather than to recommend adapted learning paths to learners.

Other approaches, especially in the course generation research community, address the need for recommending not only the learning objects themselves, but sequences of learning objects. Sicilia et al. [33] or more recently Ullrich and Melis [39] addressed learning design concepts and requirements through course generation. Though numerous solutions have been proposed, using statistical methods [23], decision rules [40], production rules [18], Markov processes [13] and Hierarchical Task Network Planning [33,37,38], to the best of our knowledge, none have investigated recommendation capabilities for large repositories counting at least hundreds of thousands of objects.

Learning object repositories accessibility and referencing have improved allowing a tremendous growth capacity. Some repositories aggregates, as the Global Learning Objects Brokered Exchange (GLOBE), [15] count already more than one million learning objects [30] and are likely to keep growing in the future.

The goals of the present study were: (1) to elaborate a model for personalized learning paths using graph theory and (2) to apply said model in order to build a learning design recommendation system in the case where learning objects are stored in very large repositories.

2. Modelling of learning path recommendations

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful" [7].

2.1. Definitions and assumptions

In order to facilitate the understanding of the presented model, several key elements and assumptions need to be clearly defined.

A competency can be seen as a knowledge component being part of a "model that decomposes learning into individual knowledge components (KCs)" [31]. In this paper, a competency is "an observable or measurable ability of an actor to perform a necessary action(s) in given context(s) to achieve a specific outcome(s)" [22]. A competency in our situation can be a prerequisite to the efficient completion of a learning object. According to Wiley [46], a learning object is "any digital resource that can be reused to support learning". In the rest of the paper we define the learning object as any digital resource that can be reused to provide a competency gain.

A learner is a dynamic user interacting with learning objects in order to increase his/her competencies from an initial set to a targeted set of competencies. We assume that a learner completing a learning object will gain the competencies targeted to be transmitted by the interaction with the learning object. We also assume that a learner who would not possess the prerequisite set of competencies required by a learning object should not attempt this learning object since this would result in a limited competency gain.

Last but not least, we assume that the number of learning objects available is very large (millions to billions of learning objects) and that each learning object cannot provide the gain of a competency that is a prerequisite to itself.

2.2. Graph theory based model

Graph theory aims at studying mathematical structures composed of elements having relationships or connection between them. The use of directed graphs is not a novelty in e-learning systems [1,4,44,46]. However, we were unable to find a formal model for discussing learning path problems based on graph theory, especially one taking into account the dynamic nature of a learning environment.

A directed graph, or digraph, $G = (V, E)$ consists of:

- A non-empty finite set V of elements called vertices or nodes.
- A finite set E of distinct ordered pairs of vertices called arcs, directed edges or arrows.

Let $G = (V, E)$ be a directed graph for a personalized learning path. In G each vertex or node corresponds to a learning object. Two vertices are connected if there exists a dependency relation, such that one vertex satisfies the prerequisites of the other. So, each edge between two vertices $ARC\{u, v\}$ means that the learning object v is accessible from u. The accessibility property required to define edges between vertices relies on post and prerequisite competencies associated to each learning object. Considering $Arc\{u, v\}$, this edge means that after having completed the learning object u, the learner should have the required competencies to undertake resource v. By extension, each vertex v is represented by a pair (C_{pre}, C_{post}) where:

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

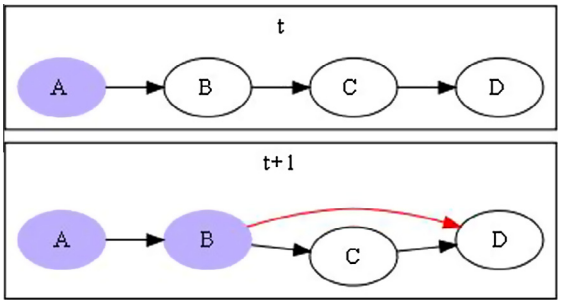


Fig. 1. Edge dynamicity.

- C_{pre}, is a set of the competencies required by vertex v.
- C_{post} is a set of competencies offered by vertex v.

The relationship between learning objects and competencies is multidimensional [9]: a learning object can require several competencies and transmit more than one competency to the learner as well. The existence of an edge between two learning objects u and v can be formalized by the following formula:

$$C_{pre}(v) \subseteq C_{post}(u) \Rightarrow Arc\{u, v\} \quad (Condition\ 1)$$

where $C_{pre}(v) \subseteq C_{post}(u)$ means that the competencies required by v are provided by learning object u. *Condition 1* is sufficient but not necessary. For example, before having completed u, the learner might already have some or the totality of the competencies required by v. This means that we may have an arc between u and v even though none the competencies required by v are provided by u. In other words, edge set E also depends on the learner's competency set at time t: $E = E(C_{learner}(t))$ and $C_{learner}(t) = \{c_1, \ldots, c_n\}$ where $c_1 \cdots c_n$ are competencies which the learner possesses. As a result, graph G is a dynamic directed graph and *Condition 1* can be strengthened by the necessary and sufficient *Condition 2*:

$$Arc\{u, v\} \iff C_{pre}(v) \subseteq C_{post}(u) \cup C_{learner}(t) \quad (Condition\ 2)$$

2.3. Explanation of the model dynamicity

The dynamicity of our model is due to the fact that a learning object can bring competencies that could be among the prerequisites of future learning objects.

For example, as shown in Fig. 1, a learning object D could be accessible to a learner if he has acquired the competencies c_1 and c_2. Assuming that competency c_1 is provided by learning objects A and C and competency c_2 is provided by learning objects B and C; D is reachable if learning objects A and B are completed or if learning object C is completed. If a learner completes learning object A at time t and learning object B at time $t + 1$, the learner will have the competencies required to reach D and according to the *Condition 2*, a new edge between B and D will be created (red[1] edge in Fig. 1).

3. Proposed approach

The long term vision of this work is to be able to provide each learner with the most efficient learning path. As a first step towards this goal, our approach focuses on ways to search for such potential learning paths.

3.1. Induced subgraph

Eliminating irrelevant learning objects is generally the first step of a course generation tool [1,27]. In our case, as the learning object repository is supposed to be very large, the learning objects cannot all be checked individually. The approach we chose consists in reducing the considered solution space by obtaining an induced subgraph[2] H which consists of all the vertices and edges between the vertices in G that could be used in the learning path.

Only the potential learning objects selected by our algorithm as being relevant to the learning path are considered when building this subgraph (see Fig. 2). The algorithm can be seen as a loop generating subgraphs, or cliques, until one such clique is generated whose prerequisites are a subset of the learner's competencies. Cliques are generated in a top-down fashion where we begin with the target clique, which is composed of a single learning object (we create a fictitious learning object,

[1] For interpretation of color in Fig. 1, the reader is referred to the web version of this article.
[2] An induced subgraph H of a graph G is a graph whose vertex set is a subset of G's vertex set, and whose edges between vertices are kept from G.

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

```
targetClique = new clique with only the target learning object
clique = targetClique
while clique's prerequisites are not a subset of the learner's competencies
  preClique = a new clique with all learning objects leading to any of clique's prerequisites
  if preClique's prerequisites contain all of clique's prerequisites AND are not a subset of the learner's competencies
    break, as an infinite loop would ensue
  clique = preClique
```

Fig. 2. Clique generation algorithm.

	β_6	
v_1	$A^6{}_5 \quad E^6{}_{3,5}$	↑ **6**
v_2	$T^{3,2,4}{}_7 \; U^5{}_0$	↑ **3,5**
v_3	$L^{0,7}{}_{-1,-7} \; I^7{}_{-7} \; K^0{}_{-1}$	↑ **0, 7**
	$\alpha^{-1,-7}$	↑ **-1, -7**

α: Fictitious LO with initial learner competency state

β: Fictitious LO with targeted learner competency state

$LO^{\,list\ of\ gained\ competencies}\ LO_{\,list\ of\ prerequisite\ competencies}$

Fig. 3. Induced subgraph generation.

β, whose prerequisite competencies correspond to the list of the learner's target competencies). Cliques are then generated by finding every vertex where at least one output competency is found in the prerequisite competencies of the clique (the union of all prerequisite competencies of every learning object within the clique) to which it is prerequisite. As such, cliques contain the largest possible subset of vertices which satisfies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible". We simplify the stopping condition by adding a second fictitious object, α, into the dataset with no prerequisite competencies and with the learner's current competencies as its output competencies. If a clique contains this object, the stopping condition is true. The union of all the generated cliques results in the subset H.

Considering the target competencies β as shown in Fig. 3, all the vertices leading to those competencies (competency 6 in Fig. 3) are selected in a set v_1, then the learning objects leading to the prerequisites of set v_1 (competencies 3 and 5) are selected from graph G to create the set v_2. This mechanism continues until the prerequisite competencies of the set v_n are all competencies which the learner has already acquired.

As shown in Fig. 4, G', consisting of the vertices E of sets $v_1 \cdots v_n$, is an induced subgraph of G. If the learner has completed all the vertices of v_i, he/she will have access to all the vertices of v_{i+1}, thus all subsets of vertices of v_i can be considered to be a clique.

3.2. Greedy algorithms

Once the induced subgraph is obtained, we use a greedy algorithm that would search for a local optimum within each clique. The definition of such a local optimum, depending on the dataset and the results pursued, has to follow a specific heuristic or strategy. The shortest path strategy seems to be widely accepted in the literature [1,47]. We are aware that this strategy is not necessarily the best to adopt in any situation since the proposed learning path might require a steep learning curve. However, the adequacy of heuristic not being the purpose of this paper, it will not be addressed at this time but will be considered for future developments (see Fig. 5). Q3

The greedy algorithm considered attempts to find the shortest path by considering each clique one after the other and reducing it to a minimal subset of itself which still verifies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible".

The first clique considered will be the one leading to the targeted competencies (the clique satisfying the prerequisites of β). In the case of the three cliques v_1 to v_3 as illustrated by Fig. 6, v_1 will be considered first followed by v_2 then by v_3.

For each clique, the local optimum will be considered obtained when the minimum subset of vertices with a minimum "degree", being the sum of the number of prerequisite competencies and output competencies of the vertex, are found. For clique v_1, the selected learning object will be A since its number of prerequisites is smaller than that of E while they share the same competency gain. As A has been chosen in v_1, only the objects in v_2 respecting the new v_1's prerequisites will be chosen. As a result, the algorithm chooses U in v_2. In v_3, K and L lead to v_2's prerequisite but K requires fewer prerequisites than L, therefore K is selected and the proposed learning path is KUA.

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

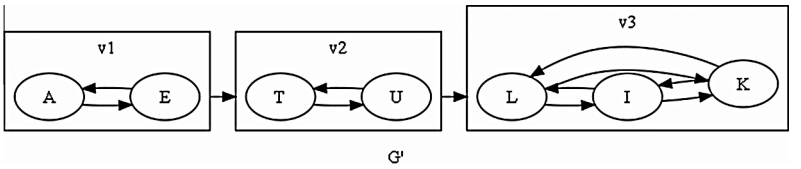
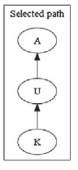


Fig. 4. G' consists of connected cliques.

```
for each prerequisite to satisfy, prerequisite
   selectedObject = a blank object whose degree = ∞
      for each learning object in the clique, object
         if object is already in localOptimum continue to next prerequisite
         else if object produces prerequisite AND object's degree < selectedObject's degree
            selectedObject = object
   localOptimum.add(selectedObject)
return localOptimum
```

Fig. 5. Local optimum algorithm.



	β_6	
v_1	$\boxed{A^6_5}$ $E^6_{3,5}$	↑ **6**
v_2	$T^{3,2,4}_{7}$ $\boxed{U^5_0}$	↑ **3,5**
v_3	$L^{0,7}_{-1,-7}$ I^7_{-7} $\boxed{K^0_{-1}}$	↑ **0,7**
	$\alpha^{-1,-7}$	↑ **-1, -7**

α: Fictitious LO with initial learner competency state
β: Fictitious LO with targeted learner competency state
$LO^{\text{list of gained competencies}}$ $LO_{\text{list of prerequisite competencies}}$

Fig. 6. Illustration of our greedy algorithm execution.

```
       LO                Pre              Post
Min.   :      1   Min.   :1.000   Min.   :1.000
1st Qu.: 250001   1st Qu.:2.000   1st Qu.:1.000
Median : 500001   Median :4.000   Median :1.000
Mean   : 500001   Mean   :3.501   Mean   :1.500
3rd Qu.: 750000   3rd Qu.:5.000   3rd Qu.:2.000
Max.   :1000000   Max.   :6.000   Max.   :2.000
```

Fig. 7. Screen shot of *R* data summary for a small dataset of 1 million learning objects and 1500 competencies.

4. Implementation

4.1. Data set

In web accessible learning object repositories, learning objects are searchable through their metadata. There are several standards of metadata, like LOM [20] which is probably the most popular. However, the way these standards were thought out was to allow a resource to be described, searched and used by a human-being. As a result the description elements and filled information tend to lack precision and they are difficult to exploit with machine-based methods [32]. LOM has numerous information fields but no "elements that would describe the suitability of a given learning resource for supporting learning activities that are designed to meet competence-based educational objectives within competence development programmes" [32].

The datasets available from the EDM (Educational Data Mining) community site (PSLC datashop [25]) are designed in accordance with actual interests of the community. Though there are sometimes competencies associated to learning objects in several datasets, their information model is significantly different from our model. For example, a competency would represent the competency gained by digesting the learning object and there is no notion of prerequisite competencies. Therefore, we decided to generate our dataset and modify its topology, measuring the impact of these topology variations on our approach.

The main dataset used in our implementation consisted of learning objects having from 1 to 6 prerequisites (PRE) and providing a gain of one or two competencies (POST) as illustrated by Fig. 7. The ranges of PRE and POST competencies has been chosen according to qualitative studies of existing learning object datasets (KDD cup 2010 dataset [24], The Australian Flexible Learning Framework [6], Globe [15], etc.). These numbers are open to debate seeing as they are drawn from specific contexts. The distribution of competencies among learning object is also open to debate since some competencies might be more popular than others. This dataset is smaller than the targeted dataset but highlights the topology of such a dataset. We therefore assume that the properties of this dataset and the others tested can be extrapolated to larger ones.

4.2. Data set storage

As the number of vertices envisioned in our case will be significant, we decided to store each vertex with its associated pre and post competencies. There are several advantages of doing so, the first of which is to limit the space required to store node adjacencies. As the condition required between edges in our model is weak, this can lead to a graph having a huge number of edges.

$$\begin{aligned} &\textit{If } Arc\{u,v\} \Longleftrightarrow C_{pre}(v) \subseteq C_{post}(u) \cup C_{learner}(t) \\ &\qquad\qquad \textit{Then} \\ &C_{pre}(v) \subseteq C_{learner}(t) \Rightarrow Arc\{u,v\} \end{aligned}$$

Any node v can become accessible from any other node of the graph if the learner's competencies satisfy the prerequisites of v.

By not storing the edges but only the way to find them, we limit the space required to store them. The maximum size of each vector is at most equal to the number of competencies and we assume that the number of competencies is at most in the thousands while the number of vertices counts in the millions. Millions of elements with two vectors of thousands of objects require less space than millions of vectors of millions of elements.

The second advantage is to have a single graph stored. As previously stated, edges are added according to learning objects' prerequisite and output competencies as well as the competencies of the learner at time t. If edges need to be stored, the edge set will be different from one learner to another and would evolve with time. Storing edges would require that a separate graph be stored for each learner and be updated according to the learner's progresses and changes in learning material, leading to a significant waste of storage resources.

In addition to this, a separate reference table was created having competencies as keys and a list of references to all vertices which lead to this competency as values. As such, we can find the vertices which will make up a clique prerequisite to another without the need to examine each learning object individually. This drastically cuts down the computation time required to generate cliques.

4.3. Implementation and performance

Distributed database systems bring a number of advantages over centralized systems, such as economy, modularity and performance. Single site failure has minimal effect on system performance and data loss is avoided by distributing the system. This also allows for the use of many smaller and more economical machines instead of a single large and expensive machine and allows for the load on the system to be balanced among multiple servers.

Apache Cassandra [3] is a highly scalable open source distributed database management system (DDBMS). Although still under development, it has proven itself in real-world implementations (currently in use at Facebook). It was also designed to handle very large amounts of data with no single point of failure and as such seemed to be an appropriate solution for this implementation.

Cassandra's storage model resembles a multi-dimensional hash map. A simplified view of the data model would show 4 or 5 dimensions of hashes. The Keyspace is the first dimension of Cassandra's data model, which could roughly be compared to a schema or database in a relational model. The Keyspace contains Column Families, which could be compared to tables in a relational model. These in turn contain Rows, which are logically related groupings of Columns which will always be stored on a same cluster. Columns are the final dimension of Cassandra's data model and consist of a 3-tuple containing a name, a value and a timestamp. Cassandra's data model can also use Super Column Families instead of Column Families. Super Column Families' Rows contains Super Columns instead of Columns. A Super column is simply a Column whose values are themselves other Columns, adding a fifth dimension to the data model. Cassandra's read and write operations use in-memory Memtables and on-disk SSTables. Memtables contain data from a single Column Family and there is only one Memtable per Column Family. SSTables contain data from a single Column Family but the data from one Column Family can be spread across multiple SSTables.

The application was implemented in Java in order to maintain platform interoperability and because of the availability of high level Cassandra clients. To maximize memory space, learning objects are represented as a single integer reference and two lists of integer references, corresponding to competencies. Correspondence tables can be stored separately and the reference numbers converted to their respective description or object values after the learning path recommendation has been calculated.

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

Due to limited resources, Cassandra was configured with a single cluster on the testing machine. It was configured using the order preserving partitioner (as opposed to the standard random partitioner) so as to facilitate the implementation of paging without the need for additional reference tables. Seeing as our implementation had a single cluster, the potential disadvantage of the order preserving partitioner, being that it may lead to uneven distribution of data between clusters, was moot. Interaction between the application and Cassandra was handled via the Hector [18] high level Java client for Apache Cassandra.

To analyze performance, a series of requests were performed while varying certain parameters to study their impact on performance. Each request was executed 100 times and the results were averaged to obtain each data point. All the tests were conducted on the same machine where the Cassandra server was installed, thus Cassandra and DPLO shared the resources of a single machine (Intel® Core™2 Extreme Q9300 CPU at 2.54 GHz and 8 GB RAM running Microsoft Windows Vista™ Business Service Pack 2).

The goal of the first set of tests was to determine the impact of the dataset size, both in number of learning objects and in number of competencies. For these tests, the datasets had varying numbers of learning objects and competencies both ranging from 10^1 to 10^6. The number of prerequisite competencies per learning object was a random value between 1 and 6 inclusively (average of 3.5) and the number of output competencies was a random value between 1 and 2 (average of 1.5). From these tests the following graphs were created depicting the average calculation time and the average number of cliques generated in order to produce a learning path.

In Fig. 8, we notice a trend where the calculation time increases with the number of competencies with the exception of the lowest values for the number of competencies. This could very well be due to resource limitations of the testing machine. With such a low number of competencies, there will be a very large number of learning objects outputting any given competency and as such, clique sizes will quickly increase to a maximal size. This creates many large requests to the database in rapid succession, filling the Memtables and heap memory and forcing Cassandra to pause for garbage collection (Cassandra will halt for garbage collection and data compaction whenever the current Memtable for a column family reaches a set size or when a set percentage of the available heap memory is used), thus increasing the total time of the process.

In Fig. 9, we notice small spikes in the number of cliques generated where the number of competencies and learning objects are equal, relatively stable and constant values where the number of learning objects is greater than the number of competencies and near-zero values where the number of competencies is greater than the number of learning objects up until the number of learning object reaches 10^3.

Where the number of learning objects is between 10^3 and 10^5 and the number of competencies is between 10^4 and 10^6 we observe a large spike in the number of cliques generated. The near-zero values where the number of learning objects is less than the number of competencies and below 10^3 is due to the fact that in datasets where there are more competencies than learning objects it is possible that there are no possible paths between two given learning objects and given the low number of learning objects in this region, this possibility becomes very likely. As for the stable values where the number of learning objects is greater than the number of competencies, this is due to the fact that in this scenario, a path is almost guaranteed to exist between two given objects because after approximately 3 cliques are generated, the lowest clique can contain the whole dataset. Where the number of competencies is greater than the number of learning objects there are fewer

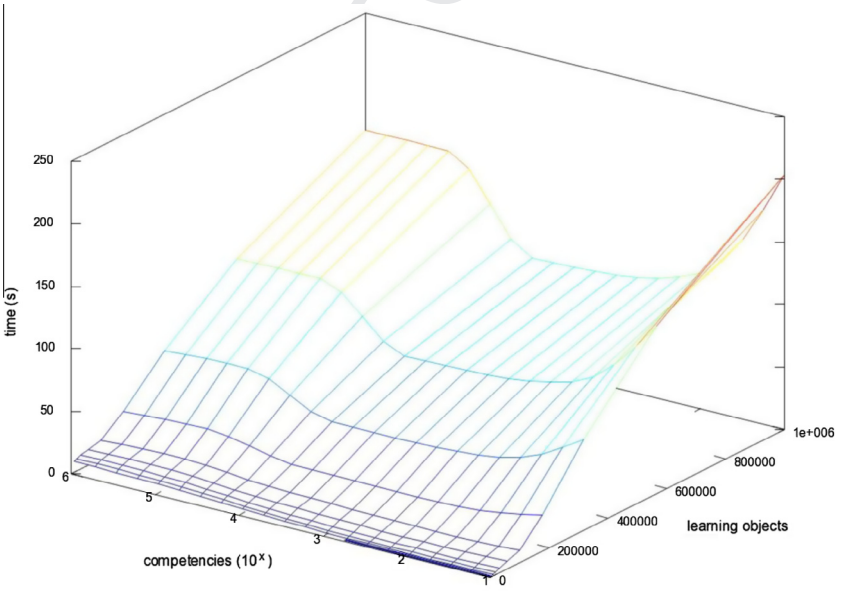


Fig. 8. Average calculation time of learning paths given 1–2 output competencies and 1–6 prerequisite competencies per learning object.

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

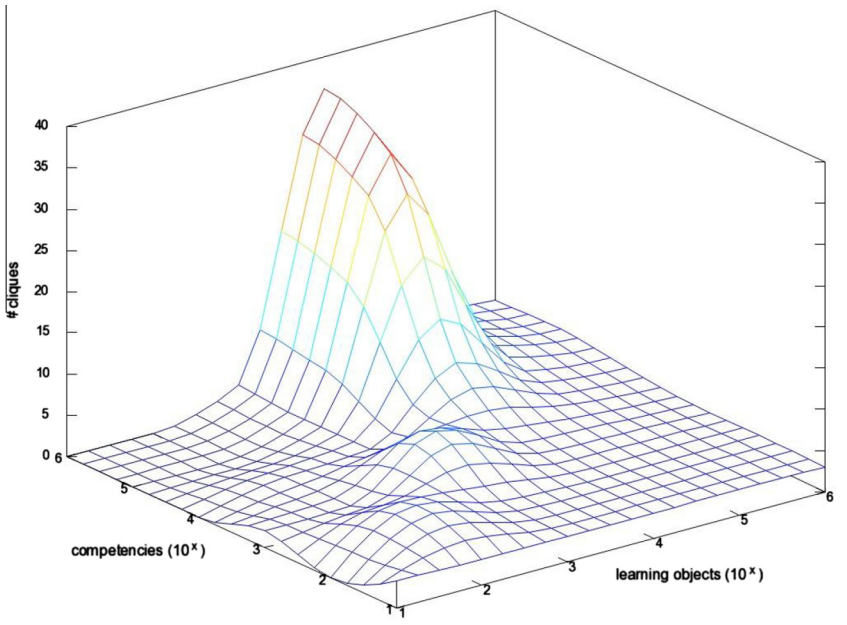


Fig. 9. Average number of cliques in calculated learning path given 1–2 output competencies and 1–6 prerequisites competencies per learning object.

learning object which produce a given clique's prerequisite competencies leading to smaller cliques, thus requiring a larger number of cliques to reach a given learning object. This explains the large spike in number of cliques; however, where the number of learning objects is greater than 10^4 we see a decline in number of cliques generated. This seems to oppose the theory but may prove that there is a threshold after which even if the number of competencies is greater than the number of objects, we still reach any given learning object in a limited number of cliques. It would be interesting to expand the graph further to observe this; however, with the values included in the current graph, we have reached the limitations of our testing machine.

The goal of the second set of tests was to determine the impact on performance of the learning objects, in terms of prerequisite and output competencies. For these tests, the datasets had varying numbers of prerequisite and output competencies per learning object both ranging from 1 to 5 with averages ranging from 1 to 3. The number of learning objects was fixed at 10^5 and the number of competencies was fixed at 10^4. From these tests the following graphs were created depicting the average calculation time and the average number of cliques generated in order to produce a learning path.

In Fig. 10 we notice a general trend of the calculation time increasing with the number of output competencies, although not in a particularly consistent manner. There seems to be little correlation between the calculation time and the number of prerequisite competencies.

In Fig. 11 we see that increasing either the number of output competencies or prerequisites decreases the number of cliques generated. This is due to the fact that increasing the number of prerequisites or output competencies will result in more learning objects being able to produce the required competencies, increasing the average clique size, thus leading to any given learning object in fewer cliques.

As in the first set of tests, there seems to be a point at which the values stabilize to a near-constant value. In this case, for a high enough number of prerequisite and output competencies, the number of generated cliques stabilizes at 5. Similarly to the first set of tests, this is due to the fact that for a sufficiently elevated number of prerequisite or output competencies, after 5 cliques are generated the lowest clique can contain the whole dataset.

Given that the application and Cassandra shared the testing machine's resources and pushed it to its limits, it would be preferable to set up a separate machine for the application. This would probably lead to faster execution and less influence from the machine's limitations, such as excessive memory paging and frequent garbage collection and compaction. It would also be interesting to configure Cassandra on multiple machines, as it is designed as a distributed system, and test with 2, 4, 8 and 16 machines to examine the optimal parameters more closely. Cassandra claims a significant performance increase when concurrent access is used; as the application does not currently implement concurrent access to Cassandra, doing so could potentially cut the average calculation time in a significant way as profiling has revealed that interaction between the application and Cassandra currently takes up between 85% and 95% of the process time.

5. Related work

Ullrich and Melis [38,39] proposed PAIGOS a HTN-Planning based (HTN: Hierarchical Task Network) courseware generator to assemble a "sequence of educational resources that support a student in achieving his learning goals" [39]. PAIGOS

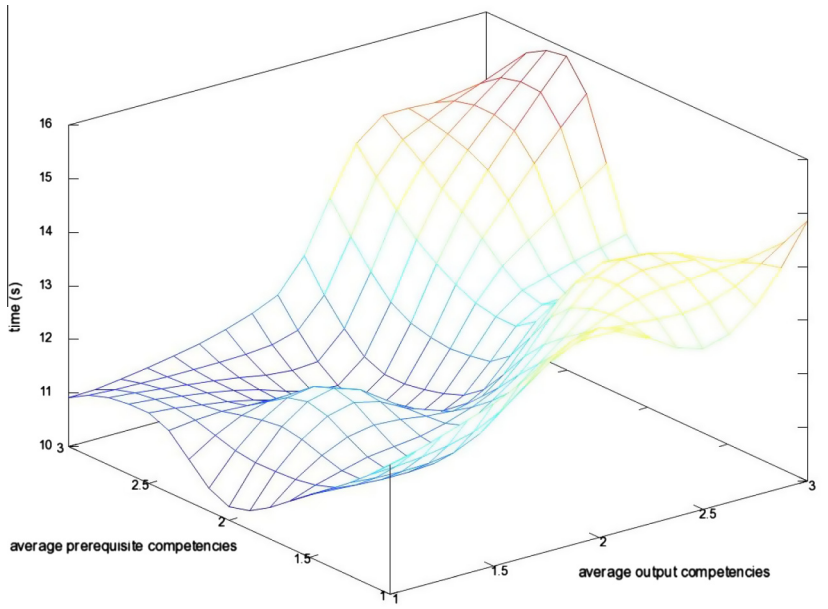


Fig. 10. Average calculation time of learning paths given 10^5 learning objects and 10^4 competencies.

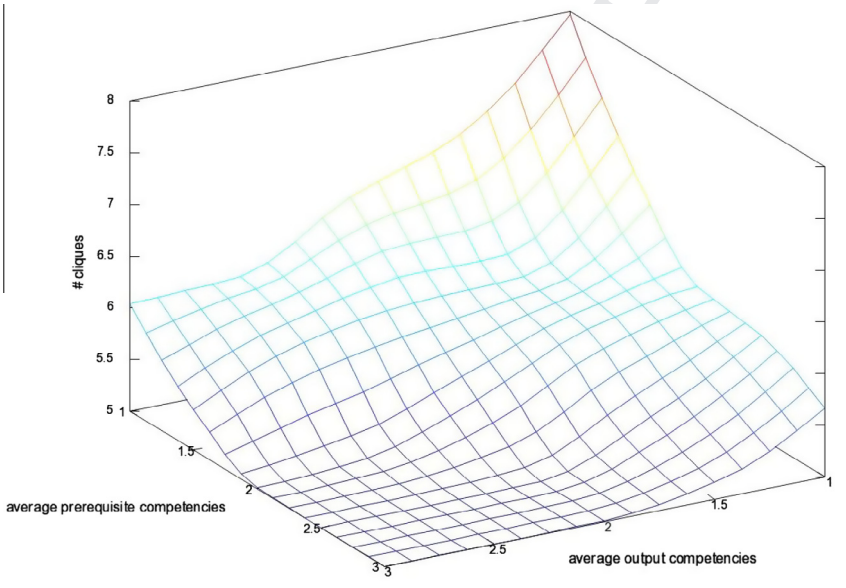


Fig. 11. Average number of cliques in calculated learning path given 10^5 learning objects and 10^4 competencies.

builds a learning scenario based on learner skills and instructional knowledge. The instructional knowledge is deemed to model the pedagogical experience a teacher would use to choose and sequence learning materials. Contrary to hereafter listed approaches PAIGOS allows to produce complex courses based on predefined patterns including sections and subsections. PAIGOS is often presented as a component of the mathematical exerciser ActiveMath [28,29]. ActiveMath contains between ten and thirty thousand content items (learning objects) depending on the version considered. Scalability has been an issue that Ulrich managed to work out by using a caching mechanism [14] to limit the number of requests made to the learning object repositories. However, the scalability of PAIGOS planner algorithm has not been questioned probably because traditionally "HTN planners neither address nor take into account the issue of scalability" [14].

Sicilia et al. [33] proposed a theoretical use of HTN-planning to create IMS-LD [21] learning scenarios. Considering learning design as a planning problem, they introduced the notion of "pedagogical design algorithm" by specifying a specific planner for each pedagogical target. For example a "Socio-Cultural" planner would propose collaborative activities while a

"content oriented" planner would select contents based on concepts hierarchies. The result of each planner would then be submitted to an instructional designer in charge of choosing and adapting the learning design to the learners. The structural and hierarchical approach of learning design as in IMS-LD naturally echoes HTN's use of schemata to solve a planning problem. A schemata can be seen as "standard ways, about how to implement abstract plans" [2]. More accurately, "given a planning domain, the description of a planning problem will contain an initial state like that of classical planning—but instead of a goal formula, the problem specification will contain a partially ordered set of tasks to accomplish" [5]. A solution of the planning problem is generated by recursively applying methods in order to replace abstract actions with actions which are directly executable [2]. Unfortunately the definition of this abstract or generic structure of the final solution of the problem is not always known in advance. In our case, we have no indications of the learning object sequence that should be chosen to solve our problem.

Karampiperis and Sampson [23] proposed a statistical approach to mimic the way an instructional designer would proceed to create a learning path. The learning material is previously rated by an instructional designer according to the specificities of the learner. In a first step, the algorithm lists all the possible learning paths and in a second step selects the learning paths that are the most aligned with learner's "cognitive characteristics and preferences" [23]. Because of the combinatorial nature of listing all the possible learning paths, this method can lead to a particularly inefficient algorithm especially in large repositories.

Idris et al. [19] pursued with the idea of imitating an instructional designer or instructor's role by considering sequencing as a classification problem and proposing the use of Artificial Neural Networks (ANN) to recommend the most adapted learning objects to a specific learner profile. For this purpose, learning objects' required competencies are rated by a human expert and then divided into learning object clusters using a clustering approach. The ANN associates a learner's profile to a matching learning object cluster. Recommendations made depend strongly on the number of clusters considered. Moreover, learning object clusters produced should not be considered as sequences but rather as sets.

Huang et al. [18] proposed a Mobile E-learning Authoring Tool (MEAT) to help teachers create courses for mobile applications. MEAT possesses a learning material arrangement agent using Dynamic Fuzzy Petri Nets (DFPN). The arrangement agent adapts the learning paths designed by the lecturer to fit the needs of the learner. The adaptation consists in the suggestion of auxiliary material in order to increase the capacities of the learner to succeed at tests. With MEAT, the learning path is ruled as a petri network where the learner has to reach the test threshold in order to get access to the next learning material. This subsequent adaptation of a learning path could prove to be an interesting add-on when initially recommended learning paths are pedagogically weak.

Still but maybe less related work on learning path recommender systems could be considered in the Educational Data Mining community [41,42].

Su et al. [34] used sequential pattern mining to extract frequent learning patterns, clustering to group learners having similar behavior and using a decision tree on a learner's profile to recommend some learning sequences to new learners. More than prescribing complete learning sequences in order to achieve a final goal, the system can provide some guidance during learner navigation among learning objects.

Wang et al. [45], assigned different learning paths to a pool of learners having a homogenous level of English language. Using a decision tree on learners' profile elements (gender, personality, cognitive style, learning style, grade), and considering results of pre and post-tests, the authors managed to match profiles and learning paths allowing to recommend predefined learning paths to new learners.

Durand et al. [13] proposed a system based on Markov Decision Processes assisting teachers during the learning design phase. According to a predefined policy, comparable to HTN planner rules, and experience learned from the teaching practice (machine learning) several learning paths are proposed to the teacher. The system is designed and sized to a classical learning management system (a few thousand learning objects).

Vialardi et al. [43] used decision trees and association rules to build the Author Assistant A^2. A^2 models student's behavior by mining learning activity logs. A^2 supports an "a posteriori" analysis to help the teacher detect patterns and improve the course. Though A^2 does not recommend learning paths, it helps to iteratively improve existing paths.

Besides learning path recommendations, EDM is showing more interest to apply its techniques to big data. Cechinal et al. [10] consider in a recent paper collaborative filtering in a "large" repository counting 8905 learning objects. This remains considerably smaller than the targeted environment of our approach.

6. Conclusions and future work

Learning design recommendation systems should draw a lot of attention in the coming years since the potential need for such systems should be significant in today's digital economy. The model proposed in this paper gives a formal framework to the field and should help researchers from several communities to clearly understand the problem being addressed. The second contribution of this paper is the method we propose to address the problem of learning design recommendation in large repositories: first reducing the problem space then using a greedy algorithm. This approach is not optimal and we doubt that an optimal solution could be easily found in an acceptable computation time; however, this method could be computationally improved by local and global optimization. The third contribution of this work is its implementation results that provide interesting information related to the dataset topology and its impact on the performance of the method used. Implemen-

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

tation results open the way for competitive contributions allowing researchers from several communities to compare their solutions with our results. For this purpose the generated datasets used and the generation scripts are available upon request by contacting one of the authors by email. These datasets should evolve in the future so as to investigate the impact of competency distribution among learning objects on our approach. In the current dataset, competencies have been randomly distributed (Fig. 7) and this may not reflect reality since some competencies might be more often associated to certain learning objects rather than others in real data. Therefore, it would make sense to study the performance of our implementation using several competency distribution models as yet to be determined.

7. Uncited reference

Q4 [17].

References

[1] M. Alian, R. Jabri, A shortest adaptive learning path in e-learning systems: mathematical view, Journal of American Science 5 (6) (2009) 32–42.
[2] L. Anselma, S. Montani, Planning: supporting and optimizing clinical guidelines execution, in: Computer-based Medical Guidelines and Protocols: a Primer and Current Trends, IOS Press, 2008, pp. 101–120.
[3] Apache Cassandra Project. <http://wiki.apache.org/cassandra/> (retrieved 06.02.13).
[4] Y. Atif, R. Benlarmi, J. Berri, Learning objects based framework for self-adaptive learning, Education and Information Technologies, IFIP Journal 8 (4) (2003) 345–368.
[5] T.-C. Au, O. Ilghami, U. Kuter, J.W. Murdock, D.S. Nau, D. Wu, F. Yaman, SHOP2: an HTN planning system, Journal of Artificial Intelligence Research 20 (2003) 379–404.
[6] Australian Flexible Learning Framework, National Toolbox Learning Object Repository. <http://toolboxes.flexiblelearning.net.au/repository/> (retrieved 06.02.13).
[7] G.E.P. Box, N.R. Draper, Empirical Model-Building and Response Surfaces, Wiley, 1987. p. 424.
[8] S. Britain, A Review of Learning Design: Concept, Specifications and Tools, A Report for the JISC E-learning Pedagogy Programme, May 2004. <www.jisc.ac.uk/uploaded_documents/ACF83C.doc> (retrieved 06.02.13).
[9] V. Carchiolo, A. Longheu, M. Malgeri, Reliable peers and useful resources: searching for the best personalised learning path in a trust- and recommendation-aware environment, Information Sciences 180 (10) (2010) 1893–1907.
[10] C. Cechinel, M.-A. Sicilia, S. Sanchez-Alonso, E. García-Barriocanal, Evaluating collaborative filtering recommendations inside large learning object repositories, Information Processing and Management 49 (1) (2013) 34–50.
[11] H. Drachsler, H.G.K. Hummel, R. Koper, Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model, International Journal of Learning Technology 3 (4) (2008) 404–423.
[12] G. Durand, S. Downes, Toward Simple Learning Design 2.0, in: Proc. 4th Int. Conf. on Computer Science and Education 2009, Nanning, China, 2009, pp. 894–897.
[13] G. Durand, F. Laplante, R. Kop, A learning design recommendation system based on markov decision processes, in: Proc. 17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD) Workshop on Knowledge Discovery in Educational Data, San Diego, CA, 2011.
[14] I. Georgievski, M. Aiello, An Overview of Hierarchical Task Network Planning, University of Groningen, JBI 2012-12-5, 2012.
[15] Global Learning Objects Brokered Exchange. <http://www.globe-info.org/index.php/> (retrieved 06.02.13).
[16] D. Godoy, A. Amandi, Link recommendation in e-learning systems based on content-based student profiles, in: C. Romero, S. Ventura, M. Pechenizkiy, R. Baker (Eds.), Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, 2010, pp. 273–286.
[17] Hector – A High Level Java Cassandra Client for Apache Cassandra. <http://hector-client.github.com/hector/build/html/index.html> (retrieved 06.02.13).
[18] Y.M. Huang, J.N. Chen, T.C. Huang, Y.L. Jeng, Y.H. Kuo, Standardized course generation process using dynamic fuzzy Petri nets, Expert Systems with Applications 34 (2008) 72–86.
[19] N. Idris, N. Yusof, P. Saad, Adaptive course sequencing for personalization of learning path using neural network, International Journal of Advance in Software Computing 1 (1) (2009) 49–61.
[20] IEEE Draft Standard for Learning Object Metadata. <http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf> (retrieved 06.02.13).
[21] IMS Learning Design, Information Model, Best Practice and Implementation Guide, Version 1.0 Final Specification IMS Global Learning Consortium Inc. <http://www.imsglobal.org/learningdesign/> (retrieved 04.02.13).
[22] ISO 24763/Final Version: Conceptual Reference Model for Competencies and Related Objects, 2011.
[23] P. Karampiperis, D. Sampson, Adaptive learning resources sequencing in educational hypermedia systems, Educational Technology and Society 8 (4) (2005) 128–147.
[24] KDD Cup 2010, Educational Data Mining Challenge Hosted by Pittsburgh Science of Learning Center (PSLC) Datashop. <https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp> (retrieved 06.02.13).
[25] K.R. Koedinger, R.S. Baker, K. Cunningham, A. Skogsholm, B. Leber, J. Stamper, A data repository for the EDM community: the PSLC DataShop, in: C. Romero, S. Ventura, M. Pechenizkiy, R. Baker (Eds.), Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, 2010, pp. 43–55.
[26] R. Kop, The design and development of a personal learning environment: researching the learning experience, in> European Distance and E-learning Network annual Conference, Valencia, Spain, 2010.
[27] J. Liu, J. Greer, Individualized selection of learning object, in: Workshop on Applications of Semantic Web Technologies for e-Learning, Maceió, Brazil, 2004.
[28] E. Melis, G. Goguadze, M. Homik, P. Libbrecht, C. Ullrich, S. Winterstein, Semantic-aware components and services of ActiveMath, Briti.sh Journal of Educational Technology 37 (3) (2006).
[29] E. Melis, J. Siekmann, Activemath: an intelligent tutoring system for mathematics, in: Proc. Seventh International Conference 'Artificial Intelligence and Soft Computing' (ICAISC), LNAI, vol. 3070, Springer, 2004.
[30] X. Ochoa, J. Klerkx, B. Vandeputte, E. Duval, On the use of learning object metadata: the GLOBE experience, in: Proc. 6th European Conference on Technology Enhanced Learning, Palermo, Italy, 2011, pp. 271–284.
[31] P.I. Pavlik Jr., N. Presson, K.R. Koedinger, Optimizing knowledge component learning using a dynamic structural model of practice, in: Proc. 8th International Conference on Cognitive Modeling, Ann Arbor, MI, 2007.
[32] D.G. Sampson, Competence-related metadata for educational resources that support lifelong competence development programs, Educational Technology and Society 12 (4) (2009) 149–159.
[33] M.-A. Sicilia, S. Sánchez-Alonso, E. García-Barriocanal, On supporting the process of learning design through planners, in: Proc. Virtual Campus Post-Selected and Extended Papers, 2006, pp. 81–89.

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017

[34] J.-M. Su, S.-S. Tseng, W. Wang, J.-F. Weng, J.T.D. Yang, W.-N. Tsai, Learning portfolio analysis and mining for SCORM compliant environment, Educational Technology and Society 9 (1) (2006) 262–275.
[35] T.Y. Tang, G.G. Mccalla, Data mining for contextual educational recommendation and evaluation strategies, in: C. Romero, S. Ventura, M. Pechenizkiy, R. Baker (Eds.), Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, 2010, pp. 257–271 (Chapter 18).
[36] N. Trcka, M. Pechenizkiy, W. Van-Deraalst, Process mining from educational data, in: C. Romero, S. Ventura, M. Pechenizkiy, R. Baker (Eds.), Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, 2010, pp. 23–141 (Chapter 9).
[37] C. Ullrich, Course generation based on HTN planning, in: Proc. 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems, Saarbrucken, Germany, 2005, pp. 74–79.
[38] C. Ullrich, E. Melis, Pedagogically founded courseware generation based on HTN-planning, Expert Systems with Applications 36 (5) (2009) 9319–9332.
[39] C. Ullrich, E. Melis, Complex course generation adapted to pedagogical scenarios and its evaluation, Educational Technology and Society 13 (2) (2010) 102–115.
[40] J. Vassileva, R. Deters, Dynamic courseware generation on the www, British Journal of Educational Technology 29 (1) (1998) 5–14.
[41] C. Romero, S. Ventura, Educational data mining: a review of the state-of-the-art, IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 40 (6) (2010) 601–618.
[42] C. Romero, S. Ventura, Educational data mining: a survey from 1995 to 2005, Expert Systems with Applications 33 (1) (2007) 135–146.
[43] C. Vialardi, J. Bravo, A. Ortigosa, Improving AEH courses through log analysis, Journal of Universal Computer Science 14 (17) (2008) 2777–2798.
[44] A. Viet, D.H. Si, ACGs: adaptive course generation system – an efficient approach to build e-learning, in: Proc. 6th IEEE International Conference on Computer and Information Technology, Jeju Island, Korea, 2006, pp. 259–265.
[45] Y. Wang, M.-H. Tseng, H.-C. Liao, Data mining for adaptive learning sequence in English language instruction, Expert Systems with Applications: An International Journal 36 (4) (2009) 7681–7686.
[46] D.A. Wiley, Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy, in: D.A. Wiley (Ed.), The Instructional Use of Learning Objects, 2002, pp. 3–23.
[47] C. Zhao, L. Wan, A shortest learning path selection algorithm in e-learning, in: Proc. 6th IEEE International Conference on Advanced Learning Technologies, Kerkrade, The Netherlands, 2006, pp. 94–95.
[48] M. Zhou, Y. Xu, J.C. Nesbit, P.H. Winne, Sequential pattern analysis of learning logs: methodology and applications, in: C. Romero, S. Ventura, M. Pechenizkiy, R. Baker (Eds.), Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series, Springer, 2010, pp. 107–120 (Chapter 8).

Please cite this article in press as: G. Durand et al., Graph theory based model for learning path recommendation, Inform. Sci. (2013), http://dx.doi.org/10.1016/j.ins.2013.04.017