RUTICO: Recommending Successful Learning Paths under Time Constraints

Amir Hossein Nabizadeh*
LIAAD, INESC TEC, Porto
Faculdade de Ciências
Universidade do Porto
Porto, Portugal
amirhossein@dcc.fc.up.pt

Alípio Mário Jorge
LIAAD, INESC TEC, Porto
Faculdade de Ciências
Universidade do Porto
Porto, Portugal
amjorge@fc.up.pt

José Paulo Leal
CRACS, INESC TEC, Porto
Faculdade de Ciências
Universidade do Porto
Porto, Portugal
zp@dcc.fc.up.pt

ABSTRACT

Nowadays using E-learning platforms such as Intelligent Tutoring Systems (ITS) that support users to learn subjects are quite common. Despite the availability and the advantages of these systems, they ignore the learners' time limitation for learning a subject. In this paper we propose RUTICO, that recommends successful learning paths with respect to a learner's knowledge background and under a time constraint. RUTICO, which is an example of Long Term goal Recommender Systems (LTRS), after locating a learner in the course graph, it utilizes a Depth-first search (DFS) algorithm to find all possible paths for a learner given a time restriction. RUTICO also estimates learning time and score for the paths and finally, it recommends a path with the maximum score that satisfies the learner time restriction. In order to evaluate the ability of RUTICO in estimating time and score for paths, we used the Mean Absolute Error and Error. Our results show that we are able to generate a learning path that maximizes a learner's score under a time restriction.

CCS CONCEPTS

•Information systems → Recommender systems; •Applied computing → E-learning; •Mathematics of computing → Graph algorithms;

KEYWORDS

Long Term Goal Oriented Recommenders (LTRS), E-learning, Recommendation System (RS), Depth-first search (DFS).

1 INTRODUCTION

E-learning systems aim to deliver educational resources to users. These systems have been able to progressively remove the barriers of space and time by delivering educational resources anytime and anywhere. E-learning systems have several advantages compared to traditional learning methods, such as reducing cost, availability, flexibility (give users the freedom to learn at their own convenience), etc [7]. In these systems, if a user plans to learn a course, he/she requires to spend a specified amount of time, which is often fixed and given by the system. The problem is if a learner's available time is less than the required time by the system, the learner might not be able to learn the materials properly. Therefore the learner fails to complete a learning process (i.e. dropout problem [6, 18]) and subsequently his/her obtained learning score from the course could not be good.

Long Term goal Recommender Systems (LTRS) [19] can be used as a possible solution to address the learners' time constraint. The LTRS, beside satisfying current needs of learners (what they need to learn immediately) and engaging them more with the system by personalizing the learning materials , guide them toward a predefined goal by generating a set of relevant strategic recommendations. In our study, the predefined goal is obtaining the maximum possible learning score in a limited time.

The LTRS method that we present (RUTICO) uses a graph traversal algorithm to generate personalized learning paths from a course graph. In a course graph, nodes represent the learning objects (LO) and directed edges indicate the prerequisite relations among the LO. To traverse the course graph, Depth First Search (DFS) algorithm [22] is used in order to generate all paths from an initial point, which is defined by a learner. RUTICO also estimates the learning time and score for each generated path using the transaction data and finally, it recommends a path that satisfies the learner's time restriction while maximizing his/her score.

Our proposal has three main contributions:

(1) RUTICO provides personalized support for learners. This assists both the learners and teachers to be more productive.
(2) RUTICO uses the learners' transaction data to estimate learning score and time for the target learner. Researchers often estimate a learning time and score for a path which is a fixed value for all learners, while our estimation methods estimate personalized learning time and score for each learner.
(3) Our method considers the learners' time limitation and personalizes the learning process under the learner's time restriction.

The remainder of this paper is structured as follows: section 2 introduces related work and section 3 clarifies our problem in detail. Our method is described in section 4 which utilizes the DFS algorithm as the basis. The data description and evaluation methodology are mentioned in section 5. Finally, in section 6, we present our conclusion.

*The contact author.

UMAP Adjunct'17, July 09-12, 2017, Bratislava, Slovakia

DOI: http://dx.doi.org/10.1145/3099023.3099035

2 RELATED WORK

Over the last decade, E-learning systems have been researched extensively in the various approaches. Several E-learning systems use a Course Generation approach (CG) where the entire learning path is generated and recommended to a learner in a single recommendation. If a learner is not able to follow the path, the recommender generates another one [3, 9, 23]. This method has been frequently used by researchers along with various techniques and algorithms. Vassileva and Deters [24] introduced a method that works based on previous knowledge of a learner and his/her goal, and can be updated dynamically with respect to the learner progress. Their proposal, given a concept (goal), utilizes an AO* algorithm [20] to search for paths in an AND/OR concept graph [8] that connect the concepts known by the learner with the goal concept. Finally, the recommendation of their method is a sequence of teaching materials (each concept has a set of teaching material) for a target learner.

CourseNavigator, that is proposed by Li et al., relies on a graph search algorithm. It generates all paths given a set of learners' inputs. The inputs are constraints (e.g., maximum number of courses to take per semester, courses to avoid), educational goal (e.g., graduation semester, a set of desired courses), student's enrollment status (e.g. starting point), and their preferred ranking for the output paths (e.g., shortest, most reliable, etc.). Their method then is able to generate three types of paths: (a) deadline-driven paths (paths that end by a given semester), (b) goal-driven paths (meet student goal by a given semester) and (c) ranked paths (based on the studentfis ordering preferences) [17]. In their method, the recommended path is a sequence of semesters that in each semester a learner needs to take a definite number of courses regardless of time that each course may take in that semester. Adorni and Koceva presented a method to generate learning path by means of Educational Concept Maps (ECM) [1]. Their personalization process starts with a self-verification of knowledge by a learner which cause pruning the known topics from the ECM map. The output of pruning process will be checked, and finally, by specifying the initial and target point (topic) by the learner, the paths will be generated using ENCODE that performs an algorithm to linearize the map [2].

In CG, there are studies such as [10] and [11] that consider the learners' time restriction. In their methods, although these two studies use different methods to recommend paths, the time of each LO is static (assigned by a course designer) and represented in the metadata of the LO, and time for a path is estimated by accumulating the learning time of LO in that path. For example, a model that is proposed by Farrell et al., receives a learner's inputs (topic keywords, time restriction, and search scope that specifies how much a learner intends to focus on a topic) to generate personalized learning paths under a time constraint. In their method, dynamic assembly of LO is based on the relative match of the LO metadata (LOM) to the learner's needs, preferences, and constraints [11].

Alternatively, the Course Sequence (CS) method, unlike the CG approach, recommends LO one by one as the learner progresses [15]. Different solutions have been proposed, using a Neural Network [13], Evolutionary Algorithms (EAs) [16] and other algorithms and approaches. Karampiperis and Sampson [15] presented an Adaptive Educational Hypermedia System (AEHS) [4]. In their work, goals are divided into three main categories: (1) cognitive capability and skills, (2) practical capabilities and skills, and (3) additional transferable skills. Their method, generates all possible learning paths that obtain the goal, and it then selects a path adaptively based on a decision model [14]. This decision model matches the features of LO with the characteristics and preferences of a target learner. Govindarajan et al. introduced a method using Parallel Particle Swarm Optimization (PPSO) to predict a dynamic learning path for learners. Their method clusters learners into four groups based on their proficiency level. The proficiency includes both measuring the achievement of a target outcome, and the competence and meta-competence changes during the learning process for each defined learning outcome. Second, it predicts the dynamic path based on the clustered information [12].

3 PROBLEM DEFINITION

In this paper, our main goal is to recommend a sequence of LO that maximizes a learner's score under a given time restriction (our proposal is a CG method). These sequences (paths) need to be generated from a course graph. Each LO of this course graph has two main attributes: time and score (Figure 1). These two attributes refer to the learning time and score of learners who already selected the LO. In Figure 1, we show an example of our problem where we intend to find a path that gives the best learning score for a learner who already knows LO 3, while satisfying his/her T_u (the learner's time restriction).

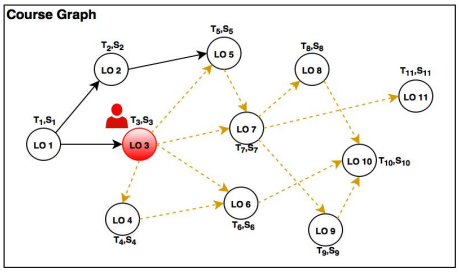


Figure 1: Course graph. Each LO has two attributes: time and score (T,S). The dash links show the potential paths for the learner who is located in LO 3.

In order to tackle the problem, we divide our problem into three sub-problems that are mentioned below. In our problem definition, D is the usage data, G is the course graph, T_u representing time restriction of a learner u, T_p shows the estimated time of path P, sp depicts the current position of u (e.g. LO3 in Figure 1) and finally u that represents the target learner ID.

The sub-problems that we need to overcome are:

(1) **Path Generation**: Generating all possible paths P for a learner u regarding his/her knowledge background sp and his/her time restriction T_u.
(2) **Time estimation**: Estimating learning time for a generated path P (T_p).
(3) **Score estimation**: Estimating learning score for a path P (S_p).

Regarding the sub-problems, we can formalize our main problem in the form of equation 1.

$$Max\ Score\ (P)\ \ where\ \ T_p \leqslant T_u \tag{1}$$

4 RUTICO METHOD

As mentioned in section 3, RUTICO approach is divided into three main phases which are path generation, time estimation and score estimation. In this section, we describe our method (Algorithm 1) in detail, explain how we generate paths from the course graph, and what are our methods in order to estimate the learning score and time for the personalized paths.

4.1 Path generation

In RUTICO, by identifying the starting point *sp* by a target learner u (line 2 in Algorithm 1), the depth-first-search (DFS) which is a well-known algorithm is utilized to generate all possible paths from the course graph G that start by *sp*. The DFS is selected since as we exhaustively search the graph and enumerate all possible paths, DFS statistically consumes less memory for its stack in comparison to the queue of Breadth First Search (BFS) algorithm [5]. Therefore, for scalability reasons (the BFS may need too much memory and impractical to use) the DFS is chosen. The generated paths must satisfy the learner's time restriction (time estimation in section 4.2).

4.2 Time estimation

To estimate the time for each path, RUTICO estimates the learning time of each LO (a path consists of several LO) for the target learner. For time estimation, we consider four different methods (results are in section 5). In **Mean** and **Median** methods, we calculate the time's mean and median of other learners that already selected a target LO. Besides the **Mean** and **Median** methods, we propose a time estimation method which is called **User Adjusted.Mean** (UA.Mean) by using equations 2 and 3 (**UA.Median** method follows the same instruction as **UA.Mean** while it uses the median instead of mean). The purpose of introducing the **UA.Mean** and **UA.Median** for time estimation is to estimate how fast is a target learner in comparison to the rest of the learners in learning the LO.

In equation 2, the numerator (t_{uLOi}) refers to the learning time of learning object i that is already visited by the target learner u, and the denominator ($mean(t_{.LOi})$) indicates the average learning time of other learners on learning object i. n shows the total number of LO that are visited by u.

$$R_u = \frac{1}{n}\sum_{i=1}^{n}\frac{t_{uLOi}}{mean(t_{.LOi})} \tag{2}$$

Then, the learning time of LO_x (LO_x is not visited by the target learner) can be calculated by a multiplication between the R_u and the average learning time of other learners that already visited the LO_x (equation 3). Eventually, learning time of a path will be estimated by summing the learning time of LO in that path.

$$t_x = R_u \times mean(t_{.LOx}) \tag{3}$$

RUTICO keeps generating a path (i.e. adding LO to the end of a path) as long as the estimated time for that path follows the time condition (i.e. $T_P \leqslant T_u$). If the estimated time of a path reach the time restriction of the learner, RUTICO stops adding LO to that path and starts generating another path.

4.3 Score estimation

To estimate the score, we consider the same instructions (methods) as time estimation while using the score of LO instead of time. Since in our data the learners' scores for LO are represented in the form of 0 (fail) and 1 (pass), therefore our score estimation infers a learner's ability to complete a LO. Hence, in score estimation, **UA.Mean** and **UA.Median** methods indicate how good is a target learner in completing a LO in comparison to the rest of the learners. Finally, by estimating the score for all paths, our method recommends a path that gives the maximum score (see Algorithm 1).

Algorithm 1: RUTICO algorithm.

Input: u, *sp*, T_u, G, D.
Output: A path that gives the best score while satisfies T_u.

```
1  Recom[] ← empty;                                  ▷ Recom is a list.
2  node ← sp;
3  P ← [sp];                                         ▷ P is a list.
4  i ← 1;
5  T_P ← T_sp;        ▷ T:time.Calculated using equations 2&3.
6  S_P ← S_sp;                                       ▷ S:score.
7  Algorithm DFS (node, G, S_P, T_P, T_u, P, i, Recom)
8  begin
9      if (edgelist of node = empty) then
10         Recom[i] ← (P, T_P, S_P);
11         i++ ;
12     else
13         foreach (Newnode in edgelist of node) do
14             if (T_P + T_newnode <= T_u) then
15                 T_P+ = T_Newnode;
16                 S_P+ = S_Newnode;
17                 P ← P + Newnode;
18                 DFS (Newnode, G, P, T_P, S_P);
19             else
20                 Recom[i] ← (P, T_P, S_P);
21                 i++ ;
22 Final Path ← return the Recom. Recom is a set of paths that
   are listed in descending order by score.
```

5 EVALUATION AND DISCUSSION

To evaluate our method we used one dataset that includes two kinds of data, usage data and a course graph. Our data is taken from a web-based learning environment for programming languages named Enki [21]. This data is for the C# programming language course which was conducted by the Polytechnic Institute of Porto in academic year 2015/2016.

5.1 Enki Dataset

The Enki usage data consist of 1186 learners' transactions for 61 learners. This data includes 8 attributes that are listed below.

- **User ID** : 61 unique IDs.
- **Resource ID** : 59 different LO that are in the form of URLs.
- **Title**: Title of each resource id.
- **State**: Indicates the status of each LO for each learner and can have four different values: Available, Unavailable, Seen and Solved. If a LO is accessible to a learner the status value would be Available, and Unavailable otherwise.
- **Grade** : 0 and 1. A grade for a LO is 1 only if a learner could solve (i.e. "Solved" state) the LO and for the rest of the states the grade would be 0.
- **Learning time** : Time to learn each LO by the learners per millisecond. Min ≈ 0, Max = 490 minutes.
- **Type** : Shows the type of each LO. Type Problem indicates the question and Video refers to the expository LO. In our data, we have 36 problems and 23 videos.
- **Learning Object ID** : Learning objects' ID (resource ID).

The Enki course graph, designed by a course expert, consists of 59 LO and 83 links. The links indicate the prerequisite relation among the LO.

5.2 Result and Discussion

In this section, we describe our experiments. Since we applied the DFS which is an off-the-shelf algorithm to traverse a graph, we do not evaluate the DFS. Here, we assess the ability of four different methods (described in section 4) for estimating the learning time and score of each LO for a target learner. To evaluate the estimation methods, we have used the Mean Absolute Error (MAE) and Error.

To evaluate the performance of our estimation methods, we take a sequence of LO that a learner already visited. We then ignore some of the LO from the sequence and consider them as unobserved LO (Figure 2), and attempt to estimate time and score of unobserved LO. For ignoring the LO we use a window. A window is composed of unobserved LO, and its size ranges from 1 to 10. Every time we ignore a number of LO (it depends on the size of the window) for a learner and estimates their time and score. We then calculate the MAE between the estimated time and score of unobserved LO and their actual values (Figures 3 and 4). The purpose of using different window size is to check which method consistently performs better than the rest of approaches.

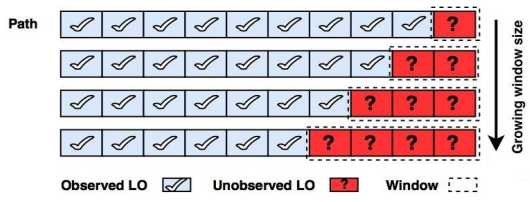
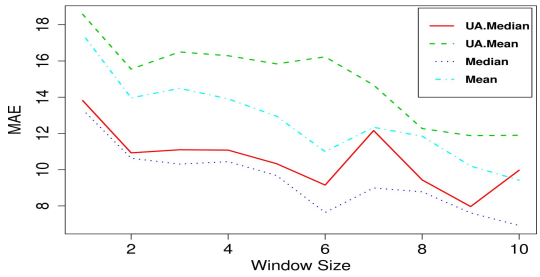
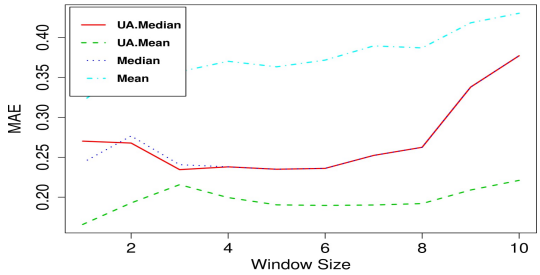


Figure 2: Evaluation method. For a path, the time and score of observed LOs are used to estimate the unobserved ones.



(a) Learning Time estimation. Timescale is in minutes.



(b) Learning Score estimation.

Figure 3: Mean Absolute Error for time and score estimation.

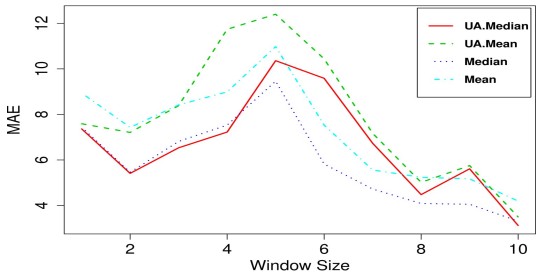
5.2.1 Evaluating estimation methods

In this section, we intend to evaluate the ability of the methods to correctly estimate the learning time and score of LO for a target learner. In Figure 3, we consider both types of LO together (Problems and Videos as described in section 5.1). According to the results in Figure 3, **UA.Mean** performs better than the other methods in score estimation, while **Median** approach outperforms other methods for time estimation. In addition, Figure 3 shows that the error increases by decreasing the size of the window. It happens since learners were devoted more to the course at the beginning. Therefore, the data is less noisy which results in less error in time estimation. During the time that the learners lose their learning motivation, the data gets more noise and the error increases. In the case of score, unlike the time, error increases by increasing the size of the window since the score is independent from learners' devotion to the course.

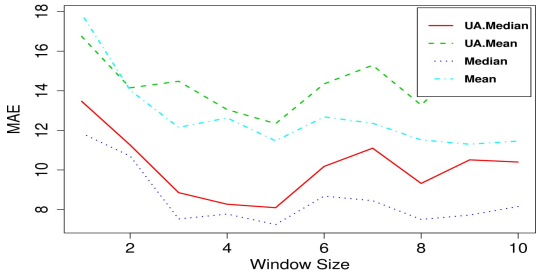
In order to check how different types of LO influence the results, we estimate the time and score for expository (Video) and evaluative (Problem) LO separately. Since the expository LO has no grade (only Problems have grades), we only estimate the learning time for different types of LO (results are in Figure 4).

According to the results in Figure 4, the **Median** could have a better performance than the other approaches in estimating time for the LO. By comparing the results in Figure 3a with the results in Figure 4, we conclude that considering different types of LO separately will reduce the error, especially for expository LO (Videos).

Regarding the results that are represented in Figures 3 and 4, the **UA.Mean** outperforms the other three approaches in score estimation, while in time estimation the results are competitive and the best result is obtained by using the **Median** method.

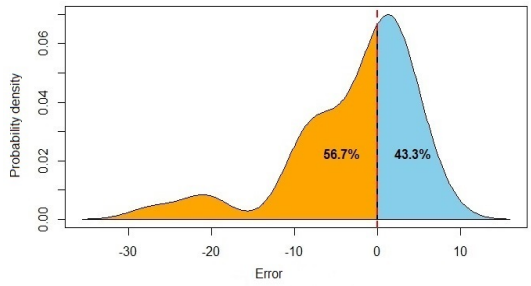


(a) Expository type (Video). Timescale is in minutes.

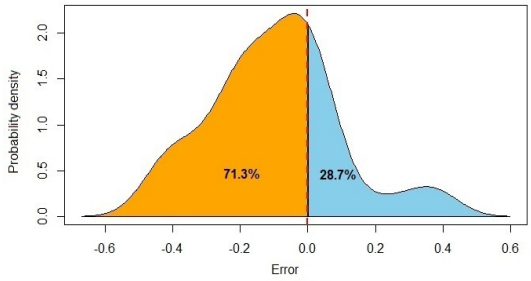


(b) Evaluative type (Problem). Timescale is in minutes.

Figure 4: Mean Absolute Error for different LO.



(a) Time estimation using Median approach. Timescale is in minutes.



(b) Score estimation using UA.Mean approach.

Figure 5: Probability density of Error for time and score estimation.

5.2.2 Evaluating overestimation and underestimation of the selected approaches

By identifying the appropriate methods for estimating time and score, we evaluated the selected approaches by using estimation error. The error results are shown in the form of probability density (Figures 5 and 6) to show the percentage of overestimation (estimated values > real values) and underestimation (estimated values < real values) of the selected methods. In learning time, underestimation indicates higher risk than overestimation since a learner might not complete a generated path in the estimated time. Unlike time, score overestimation signifies higher risk because a learner may not obtain the estimated score from a path.

In Figures 5 and 6, the orange highlighted part (negative error) indicates the underestimation and the blue part (positive error) shows the overestimation. By comparing the results in Figure 5a with the results in Figure 6, we see that considering different types of LO separately reduces the error while increases the type of underestimation error. Since the learning time of LO varies from ≈zero to ≈500 minutes, ≈10 (for videos) and ≈20 minutes (for problems) can be considered as minor errors.

5.3 RUTICO example

Figure 7 shows an example of our proposal for generating a path for a target learner. It generates learning paths that start by the starting point of the learner (LO 13) under a given time restriction. Regarding the evaluation results, **Median** and **UA.Mean** are the selected methods to estimate the learning time (we estimate time for different type of LO separately) and score of LO respectively. In our example, the RUTICO uses the time and score of LO 1, 3, 4, 5 and 9 (i.e. observed LO) of the learner along with the time and score of other learners that they already visited the LO that are in the generated paths to estimate the time and score of the paths for the target learner. Finally, RUTICO recommends a path with maximum score to the learner.

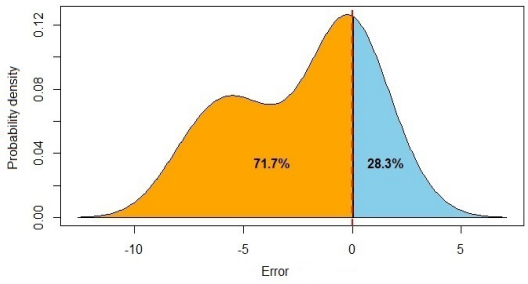
6 CONCLUSION

In this paper, we propose our method RUTICO to recommend a sequence of Learning Objects (LO) that maximizes a learner's score under his/her time restriction. Our approach works by exploiting two kinds of data, i.e., usage data and a course graph. The method uses the Depth First Search (DFS) algorithm to generate all paths for a learner from his/her starting point in the course graph, estimates score and time for all generated paths and finally recommends a path with the maximum score that satisfies the learner time limitation. We evaluate the result of the four different methods that are for estimating time and score in order to select the best one for estimating the time and score of LO.

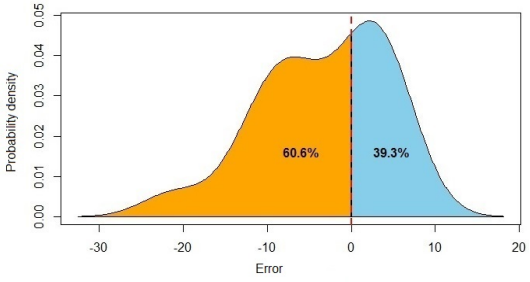
In the future, we aim to tackle the learner cold-start problem, extend our method to an adaptable one (i.e. can be updated regarding the learners progress), automatically identifying the starting point of a learner in a course graph, deal with uncertainty for estimating time and score, and evaluate our method in a live environment to assess the quality of the recommendations.

ACKNOWLEDGMENTS

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme



(a) Expository type (Video).



(b) Evaluative type (Problem).

Figure 6: Probability density of Error for estimating time for different LO using Median method. Timescale is in minutes.

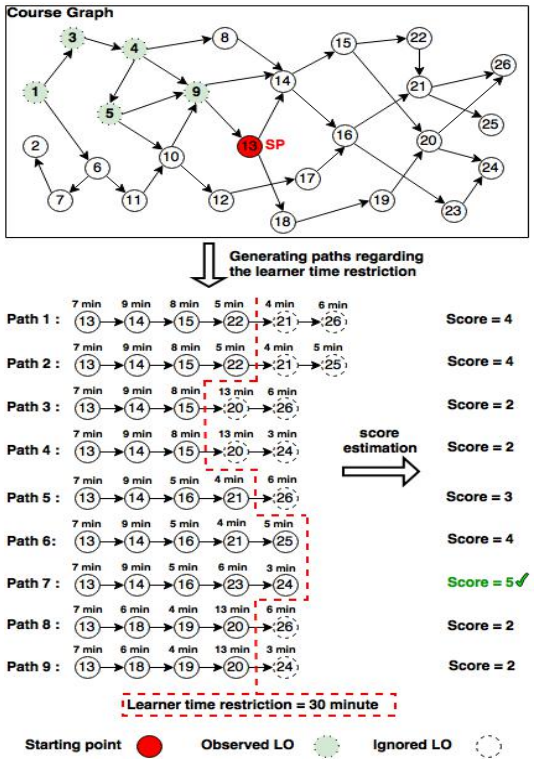


Figure 7: RUTICO example.

within project "POCI-01-0145-FEDER-006961", and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013. The first author would like to thank José Carlos Paiva who was assisting during the data collection.

REFERENCES

[1] G. Adorni and F. Koceva. Designing a knowledge representation tool for subject matter structuring. In *International Workshop on Graph Structures for Knowledge Representation and Reasoning*, pages 1–14. Springer, 2015.

[2] G. Adorni and F. Koceva. Educational concept maps for personalized learning path generation. In *AI* IA 2016 Advances in Artificial Intelligence*, pages 135–148. Springer, 2016.

[3] N. Belacel, G. Durand, and F. Laplante. A binary integer programming model for global optimization of learning path discovery. In *EDM (Workshops)*, 2014.

[4] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User modeling and user-adapted interaction*, 6(2-3):87–129, 1996.

[5] A. Bundy and L. Wallen. Breadth-first search. In *Catalogue of Artificial Intelligence Tools*, pages 13–13. Springer, 1984.

[6] C. Crawford and C. Persaud. Community colleges online. *Journal of College Teaching & Learning (Online)*, 10(1):75, 2013.

[7] J. Dargham, D. Saeed, and H. Mcheik. E-learning at school level: Challenges and benefits. In *Proceeding of the 13th International Arab conference on Information Technology, ACIT*, volume 13, pages 340–345, 2012.

[8] L. H. De Mello and A. C. Sanderson. And/or graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990.

[9] G. Durand, N. Belacel, and F. LaPlante. Graph theory based model for learning path recommendation. *Information Sciences*, 251:10–21, 2013.

[10] G. Durand, F. Laplante, and R. Kop. A learning design recommendation system based on markov decision processes. In *KDD 2011 Workshop: Knowledge Discovery in Educational Data, ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2011) in San Diego, CA*, 2011.

[11] R. G. Farrell, S. D. Liburd, and J. C. Thomas. Dynamic assembly of learning objects. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 162–169. ACM, 2004.

[12] K. Govindarajan, V. S. Kumar, et al. Dynamic learning path prediction-a learning analytics solution. In *Technology for Education (T4E), 2016 IEEE Eighth International Conference on*, pages 188–193. IEEE, 2016.

[13] N. Idris, N. Yusof, P. Saad, et al. Adaptive course sequencing for personalization of learning path using neural network. *Int. J. Advance. Soft Comput. Appl*, 1(1):49–61, 2009.

[14] P. Karampiperis and D. Sampson. Adaptive learning object selection in intelligent learning systems. *Journal of Interactive Learning Research*, 15(4):389, 2004.

[15] P. Karampiperis and D. Sampson. Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society*, 8(4):128–147, 2005.

[16] J.-W. Li, Y.-C. Chang, C.-P. Chu, and C.-C. Tsai. A self-adjusting e-course generation process for personalized learning. *Expert Systems with Applications*, 39(3):3223–3232, 2012.

[17] Z. Li, O. Papaemmanouil, and G. Koutrika. Coursenavigator: interactive learning path exploration. In *Proceedings of the Third International Workshop on Exploratory Search in Databases and the Web*, pages 6–11. ACM, 2016.

[18] S. Monteiro, J. A. Lencastre, A. J. Osório, B. D. d. Silva, P. De Waal, S. Ç. İlin, Y. K. Türel, and M. Turban. Course design in e-learning and the relationship with attrition and dropout: a systematic review. In *ITTES2016-Fourth International Instructional Technologies & Teacher Education Symposium*. Fırat University, 2016.

[19] A. H. Nabizadeh, A. M. Jorge, and J. P. Leal. Long term goal oriented recommender systems. In *11th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 552–557, 2015.

[20] N. J. Nllsson. Principles of artificial intelligence. *TiogaSpringer Verlag. Palo Alto. Calif*, 1980.

[21] J. C. Paiva, J. P. Leal, and R. A. Queirós. Enki: A pedagogical services aggregator for learning programming languages. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 332–337. ACM, 2016.

[22] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[23] C. Ullrich and E. Melis. Complex course generation adapted to pedagogical scenarios and its evaluation. *Educational Technology & Society*, 13(2):102–115, 2010.

[24] J. Vassileva and R. Deters. Dynamic courseware generation on the www. *British Journal of Educational Technology*, 29(1):5–14, 1998.