Computers & Education 147 (2020) 103777

Contents lists available at ScienceDirect

Computers & Education

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

Adaptive learning path recommender approach using auxiliary learning objects

Amir Hossein Nabizadeh [a,*], Daniel Gonçalves [a], Sandra Gama [a], Joaquim Jorge [a], Hamed N. Rafsanjani [b]

[a] INESC-ID/Instituto Superior Técnico, University of Lisbon, Portugal
[b] University of Georgia, Athens, USA

ARTICLE INFO

Keywords:
Long Term Goal Recommenders (LTRS)
Learning path
Item Response Theory (IRT)
Matrix Factorization (MF)
Recommendation Systems (RS)
Auxiliary learning objects
e-learning

ABSTRACT

In e-learning, one of the main difficulties is recommending learning materials that users can complete on time. It becomes more challenging when users cannot devote enough time to learn the entire course. In this paper, we describe two approaches to maximize users' scores for a course while satisfying their time constraints. These approaches recommend successful paths based on the available time and knowledge background of users. We first briefly explain a method that has a similar goal to our method, and highlight its drawbacks. We then describe our proposal, which works based on a two-layered course graph (lesson and Learning Object (LO) layers; a lesson includes a few LO). Initially, our method uses the Depth First Search algorithm (DFS) to find all lesson sequences in the graph that start by a lesson (opted by a user). It then assigns LO for lessons of paths and estimates their score and time. Finally, a path that satisfies the user's limited time while maximizing his/her score is recommended lesson by lesson. During a path recommendation, if the user could not get the estimated score from a lesson, our method recommends auxiliary LO for that lesson. To evaluate our method, we first assessed the quality of our estimation methods and then evaluate our recommender in a live environment. Results show that our estimation methods outperformed the ones in the literature. Results also present within the same amount of time, the users of our recommender proceeded more on the course than the users of another e-learning system.

1. Introduction

E-learning systems ease learning tasks and enable users to learn at their own pace and comfort, but they face a few challenges, such as providing learning materials that fit the users' requests and limitations. One of the challenges is to present those learning materials that users can complete (read and learn) on time. Tackling this challenge becomes harder when the users' available time is limited and they cannot dedicate the required time to learn the materials.

Users might not have enough time due to different reasons, like multi-tasking and mismanaging time while they attend a course having different professions, ages, backgrounds, etc. So, it is expected that they cannot devote equal time for a course. Despite having the time restriction, they all aim to enhance their knowledge in their available time. Knowledge enhancement can be measured by a proper metric, such as the obtained score from a course (Gawthrop, 2014).

* Corresponding author.
E-mail addresses: amir.nabizadeh@tecnico.ulisboa.pt (A.H. Nabizadeh), Daniel.goncalves@inesc-id.pt (D. Gonçalves), sandra.gama@ist.utl.pt (S. Gama), joaquim.jorge@inesc-id.pt (J. Jorge), hr@uga.edu (H.N. Rafsanjani).

https://doi.org/10.1016/j.compedu.2019.103777
Received 5 June 2019; Received in revised form 22 November 2019; Accepted 25 November 2019
Available online 9 December 2019
0360-1315/

To this end, in this paper, we explain two approaches, that are examples of Long Term Goal Recommender Systems — LTRS (Nabizadeh, Jorge, & Leal, 2015; Nabizadeh, Mário Jorge, & Paulo Leal, 2017), to recommend paths (sequences) through the learning materials under the users' available time while maximizing their score. These approaches recommend paths regarding the knowledge background and available time of the users. The first approach uses a one-layer course graph while the second one (our proposal) works based on a two-layered course graph. Since the first method is detailed in Nabizadeh, Jorge, and Leal (2018), we explain it briefly for having a better comparison between (Nabizadeh et al., 2018) and our method.

In the first approach (Nabizadeh et al., 2018), a user initially specifies his/her knowledge background by opting a Learning Object (LO) in the course graph. This method then finds all paths (LO sequences) using a Depth-First Search (DFS) while estimating their time and score. Finally, it recommends a sequence of LO that satisfies the user's limited time while maximizing his/her score.

In our approach (second approach), after specifying a user's knowledge background (opting a lesson), a DFS is used to find all lesson sequences that start by that lesson. It then assigns LO to lessons of the sequences and estimates score and time for them. Finally, a path that satisfies the user's time constraint while maximizing the expected score is recommended lesson by lesson. It enables the recommender to collect the user's transactions data for adapting the path for him/her. In the case that the user could not complete a lesson with the estimated score, this approach recommends auxiliary LO for that lesson. These LO are not in the initial path, and they will be recommended when the user could not complete a lesson with the estimated (expected) score.

For evaluation, we first evaluated the quality of our estimation methods, and then assessed the quality of our recommender in a live environment. For that, we conducted an A/B test. The experimental group used our recommender while the control group used an e-learning system that delivered LO with a predefined order (defined by an expert). Finally, we compared the groups performances.

The main highlights of this study are:

1. Presenting 3 time/score estimation methods and assessing their performances.
2. Describing the drawbacks of Nabizadeh et al. (2018) that has similar goal to our method.
3. Presenting an adaptive method that generates paths using a two-layered course graph.
4. Recommending auxiliary LO when a user couldn't complete a lesson with expected score.
5. Performing various online and offline tests to assess the quality of the recommender.

In this paper, Section 2 illustrates related work while our problem is detailed in Section 3. Section 4 describes a method presented in Nabizadeh et al. (2018) and highlights its drawbacks. It then describes our method and explain how it estimates score and time for paths. In Section 5, the data description as well as the evaluation methodology and results are discussed. Section 6 presents a discussion about our methods and results while Section 7 suggests several research directions for the researchers in the path recommendations area. Finally, Section 8 presents our conclusion.

2. Related work

Personalizing paths regarding the users' preferences and objectives is the main goal of e-learning recommenders. These recommenders automatically retrieve the learning materials from a repository and assemble them to generate paths. Since the early 70 s, different e-learning recommender approaches have been introduced using various sets of goals, algorithms, and techniques. According to our survey, these approaches can be classified into:

- Course Generation (CG).
  - Sequential Pattern Recognition (SPR).
- Course Sequence (CS).

2.1. Course Generation (CG)

In the CG approaches, by identifying users' needs and preferences, a path will be generated in a single recommendation (Belacel, Durand, & Laplante, 2014; Bhaskar, Das, Chithralekha, & Sivasatya, 2010; Carchiolo, Longheu, & Malgeri, 2010), and users' knowledge will be assessed after completing a path. So far, CG methods are introduced using different algorithms and techniques, like using a Hierarchical Task Network (HTN) (Garrido et al., 2013), a decision tree classifier (Lin, Yeh, Hung, & Chang, 2013), a Markov decision process (Durand, Laplante, & Kop, 2011), a Case-Based Reasoning/Planning (Dharani & Geetha, 2013; Garrido, Morales, & Serina, 2012), a Bayesian network (Suazo, Rodríguez, & Rivas, 2012), greedy algorithms (Durand, Belacel, & LaPlante, 2013), genetic algorithms (Tam, Lam, & Fung, 2012), etc.

Among the CG methods, there are ones to help a group of users rather than a user, like (Kardan, Ebrahim, & Imani, 2014; Xie et al., 2017). As an example, the ACO-Map method, which was presented in Kardan et al. (2014). First, it used a K-means algorithm (Jain, 2010) to classified users based on a pre-test's results. Second, the ant colony optimization algorithm (Dorigo & Stützle, 2010) was utilized to recommend a path to each group. As another example, we mention the groupized learning path discovering (GLPD) (Feng, Xie, Peng, Chen, & Sun, 2010). In GLPD, initially a topic graph was built and the users' pre-knowledge and preferences were obtained. Then, the method estimated the time boundaries (min & max) that a group needed to learn a path. Finally, a strategy was opted to discover a path based on the estimated boundaries and the required time to learn a path.

Table 1
Proportion of personalization methods per year. 56 were reviewed.

Year	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
CG	1	10	1	2	6	5	2	3	2	9
SPR	0	1	1	0	1	0	0	0	0	1
CS	0	1	0	3	3	1	0	1	0	2

There are other CG methods that consider a user instead of a group, like (Dwivedi, Kant, & Bharadwaj, 2018; Krauss, 2018; Krauss, Salzmann, & Agathe, 2018; Ye et al., 2018; Zhou, Huang, Hu, Zhu, & Tang, 2018; Zhu et al., 2018). For example, in Belacel et al. (2014) a graph based method is presented that initially dropped the irrelevant LO to obtain the goal from the graph. It then found the shortest path in the graph by decreasing the number of needed competencies using a branch-and-bound algorithm.

So far, the mentioned methods mainly generated paths and ignored the users' time restriction. This restriction was considered by a few studies, like Basu, Bhattacharya, and Roy (2013) and Garrido et al. (2013). In Basu et al. (2013), a system was introduced to recommend paths to the users regarding their available time (Basu et al., 2013). It worked based on Learning Path Generating (LPG), and Learning Path Indicator (LPI). Initially, a function was defined regarding the required time to learn a subject, the number of post-requisite and credits for that subject. Then, a fitness function was defined using a user's preferences and limitations. Finally, an LPI was made using the results from the mentioned functions. The LPI was used in the LPG to formulate a path. Similarly, in RUTICO (Nabizadeh et al., 2017), right after selecting a LO by a user as a starting point for the paths, a DFS was executed to generate all paths for the user considering his/her time constraint. In addition, RUTICO computed score and time for the paths, and recommended one of them that maximizes the user's score while fulfilling his/her limited time.

2.1.1. Sequential Pattern Recognition (SPR)

SPR methods mainly use the sequential pattern mining algorithms (Agrawal & Srikant, 1995) to discover paths for users. In comparison with the CG methods, which generate paths even without transactions data, SPR methods need the transactions data to generate paths. In SPR, paths are found for users using transactions of similar users (having similar preferences, objectives, etc.).

A few studies used the SPR approaches, like (Fournier-Viger, Faghihi, Nkambou, & Nguifo, 2010; Klašnja-Milićević, Vesin, Ivanović, & Budimac, 2011). For instance, Protus (Vesin, Milicevic, Ivanovic, & Budimac, 2013) clustered users based on their preferences and features (age, class, etc.). Next, it identified the target user's cluster and considered the paths that were opted by its members (members rated the paths regarding their success in guiding them). Ultimately, this method retrieved the paths (using association rule mining approach) from the target cluster and recommended them regarding their rates.

2.2. Course Sequence (CS)

Contrary to the CG approaches, in CS approaches a path will be recommended LO by LO regarding the progress of a user (Karampiperis & Sampson, 2005; Nabizadeh et al., 2015, 2017). Different techniques and algorithms were used in CS approaches, like Item Response Theory (IRT) (Salahli, Özdemir, & Yasar, 2013; Yarandi, Jahankhani, & Tawil, 2013), a Association Link Network (ALN) (Yang, Li, & Lau, 2012), Bayes theorem (Xu, Wang, Chen, & Huang, 2012), Evolutionary Algorithms (EAs) (Li, Chang, Chu, & Tsai, 2012), etc. For instance, a Parallel Particle Swarm Optimization (an EA) was applied in Govindarajan, Kumar, et al. (2016) to generate a dynamic path for a user. In this method, the users were categorized into four clusters regarding their proficiency level. Then, the clustered information was used to generate a dynamic path for a user. EAs were also used in Li et al. (2012). In this paper, learning concepts were initially assembled for forming a LO sequence. Next, the difficulty level of LO were updated using the user's feedback, and the Maximum Likelihood Estimation (MLE) was applied for analyzing the user's objectives and abilities. Finally, a path was made having the MLE results and using two EAs. After completing a LO, the feedback data was utilized to update the user's objectives, abilities, and the difficulty levels of LO.

The users' abilities was also considered in Salahli et al. (2013). In this system, initially the topics, their difficulties and relations were determined, and the users' profiles were built. Whenever a user used the system, his/her knowledge level and the topic difficulty were retrieved for estimating his/her understanding level using the Item Response Theory (IRT) (An & Yung, 2014). Then, the LO were recommended using the understanding level. By completing a LO, the system analyzed if the user was able to understand the LO. If the user could learn the LO, his/her knowledge level was re-evaluated. If the computed understanding degree was not high, the system recommended those LO that assisted to enhance the user's knowledge on the prior topics.

Another study that used the IRT and considered the users' abilities was Yarandi et al. (2013). In this study, a path was generated given a set of inputs (e.g. a user's preferences, knowledge background, etc.). In this system, the IRT (An & Yung, 2014) was applied for assessing a user's feedback, and later on it updated the user's ability. The modified user's ability was employed to adapt the learning path by recommending those LO that fitted the ability of the user.

In Table 1, we presented the proportion of personalization methods proposed by researchers per year. As shown, most of the methods were the CG ones.

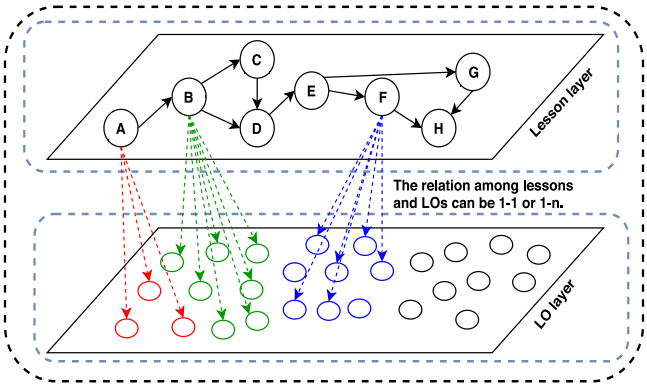


Fig. 1. A two-layered course graph.

3. Problem statement

Recommending a path that users can complete in their available time while maximizing their score is the main objective of our method. In our CS method, paths are extracted from a two-layered course graph (Fig. 1), and their score and time are computed by means of our estimation approaches. Then, our method recommends a path (a sequence of lessons) that satisfies our objective lesson by lesson. It allows the recommender to collect the users' interactions data and update a path for them. In the case of not completing a lesson having the estimated score by a user, our approach recommends auxiliary LO.

In order to attain our main objective, we break the main problem into sub-problems and tackle them one by one. These sub-problems are:

1. **Constructing course graph** : Building a course graph (G) in two layers and determining all relations among the LO and lessons (Fig. 1).
2. **Generating path**: Generating lesson sequences (P) from G for a user (u). Paths need to be generated considering the user's available time (T_u) and knowledge background (sp).
3. **Estimating score**: Score estimation (S_p) for P.
4. **Estimating time**: Time estimation (T_p) for P.
5. **Recommending path** : Designing a mechanism to recommend a path, which takes into account a user's learning score and his/her available time.
6. **Recommending auxiliary LO**: These LO (L_{aux}) are recommended whenever u is not able to obtain the estimated score from a lesson.
7. **Evaluation**: We first compare the performance of our time and score estimation methods with the ones in the literature. Next, we assess the quality of our recommender approach in a live environment using an A/B test.

4. Recommender approaches

As we explained previously, we describe two approaches, which their goals are recommending paths that satisfies the users' restricted learning time while maximizing their scores. First, we briefly explain an approach introduced in Nabizadeh et al. (2018) and highlight its main drawbacks. We then describe our approach.

4.1. First approach

In this CG method, initially a DFS (Tarjan, 1972) is used to find all LO sequences (paths) from a directed course graph (it includes LO and their relations) that begin by a LO (opted by a user). Meanwhile, this approach computes the score and time for the paths by various approaches. In the end, it recommends a path, which is under the user's restricted time while maximizing his/her score.

In this paper, two methods are also proposed, one for estimating the possibility of not finishing a path in the restricted time, and another one to compute the possibility of finishing a path having a score lower than the computed one. Although, these methods are well explained but they are not evaluated.

4.1.1. Drawbacks of the first approach

In spite of having some advantages, like computing score, time, and risk of not finishing a path within the restricted time or with the expected score, this method has a few drawbacks which are:

- **Proceeding on a concept (lesson) rather than the whole course.** In this method, all the LO in a path could cover the same concept (lesson). Thus, the recommended path results in maximizing scores on a single lesson rather than a course. For example, in a programming course, all the recommended LO could cover the "loop" concept (e.g. For, Do-While loops) while other lessons might be ignored, like "conditions" and "arrays".
- **Time consuming and computationally expensive**. In this method, adding LO to the course graph can increase the number of paths exponentially, which makes the path generation time consuming and computationally expensive. For instance, the time complexity of DFS to search a graph is b^d (b : out-going degree, d : max depth), and growing the d exponentially raise the time complexity. This problem can become bigger when we deal with large graphs. In addition, since the time and score for all paths need to be estimated, it makes the problem even harder.

4.2. Our approach

As mentioned above, the first method has a few deficiencies. To this end, we propose a lesson-based method that works based on a two-layered course graph (Fig. 1). In our method, since paths are made of lessons, thus, learning paths ensure that users are learning different concepts (not sticking to a single lesson). Furthermore, in our proposal the lesson layer is searched to find paths, which has a more restricted space than the LO graph. Therefore, it results in reducing the cost of path generation both in terms of time and computation. As a real example, for this paper we built a two-layered course graph using 5 vertices (lessons) and 8 precedence relations, while the same course had 83 links and 59 nodes (LO) using the setup of the first method. It shows how the number of paths and subsequently time and computation costs are reduced in our method.

Like the first method, we use the DFS to find the paths, while we introduce our methods to compute score and time for them. In the following sections, we explain how we generate lesson sequences (paths), select LO for them, compute their score and time, and update them whenever users cannot follow them.

4.2.1. Path generation

To generate paths, first, a user opts a lesson as a starting point (sp) for the paths (line 2 in algorithm 1). Selecting the sp implicitly defines the user's knowledge background since there are prerequisite relations among the lessons, and when a user specifies a lesson as the sp, we can conclude that he/she already knew the prior lessons. Then, to find paths, we did the same to compound on the first method (i.e. using DFS). It is detailed in algorithm 2. These paths need to fulfill the user's limited time (time estimation is detailed in Section 4.2.3).

Algorithm 1: Algorithm for path generation.

Input: u, T_u, sp, D, G.
Output: A path having the maximum score.

```
1 node ← sp;                                              ▷ sp:starting point.
2 P ← [sp];                                               ▷ P is a list.
3 Select initial LO for P;                                ▷ LO selection : Section 4.2.2.
4 i ← 1;
5 T_P ← Estimating T for sp;                              ▷ T:time.
6 S_P ← Estimating S for sp;                              ▷ S:score.
7 AllPaths ← DFS_tuwo(node, T_u, G, D, S_P, T_P, P, i);   ▷ Algorithm 2.
8 P_max ← Select the path with the max score from the AllPaths;
9 Return P_max;
```

4.2.2. LO selection for each lesson

By generating lesson sequences, there is a set of LO that can be opted for each lesson. Considering our goal, the opted LO should maximize the score while their accumulated time needs to be under a user's limited time. So, those LO will be opted that a user most likely can complete successfully in his/her available time (T_u). In this paper, we initially start by assigning two LO (which are related) for each lesson, one expository (e.g. PDF, video) and one evaluative (e.g. question). Two LO are opted since the concepts that various LO of a lesson are delivering are related, like explaining different types of "Loop", while knowing one type is sufficient to answer all practical questions about the loop (also we need to consider that the user's time is limited). In the case of not obtaining the expected score from a lesson, more LO (auxiliary LO in Section 4.2.6.1) are recommended. To this end, we formalize our LO selection for a lesson as follows:

$$Maximize\left(S(LO_{ex\in\mathbb{L}}) * W(LO_{ex\in\mathbb{L}}) + S(LO_{ev\in\mathbb{L}}) * W(LO_{ev\in\mathbb{L}})\right) where \underbrace{T_P}_{\text{Path}} + \underbrace{(T_{LO_{ex}} + T_{LO_{ev}})}_{T_{newnode} \text{ in alg 2}} \leqslant T_u \quad (1)$$

In Eq. (1), $\mathbb{L}$ is a lesson and S is the estimated score for a LO. LO_{ex} and LO_{ev} are the expository and evaluative LO, and W is their weights. Here, the weights of all LO_{ex} and LO_{ev} are equal, and also LO_{ex} have no score (zero score). Since in our method lessons are added to a path P one by one, hence T_P is the accumulated time of the lessons that are already added to P, and $\mathbb{L}$ is a lesson that might be added to P if it satisfies the user's limited time.

To select LO, since their scores might tie (equal), their time is used to break the ties (minimizing time). If their time also tied, their completion rate and later on (if needed) their visited rate are used to break the tie (maximizing these two rates). The completion rate shows how many times other users could complete a LO while visited rate indicates how many times a LO is visited.

Algorithm 2: DFS algorithm for two-layered course graph (DFS_{two}).

Input: $node$, T_u, G, D, S_P, T_P, P, i.
Output: Generate all paths under T_u.
if *(edgelist of* ***node*** $=$ *empty)* **then**
  Recom[i] $\leftarrow (P, T_P, S_P)$; ▷ Recom is a list to contain the paths.
  i++ ;
else
  foreach *(Newnode in edgelist of node)* **do**
    $LO_{all} \leftarrow$ Estimate time and score for LO of $Newnode$; ▷ $Newnode$ is a lesson.
    $LO_{selected} \leftarrow$ Select LO_{ex} and LO_{ev} from LO_{all}; ▷ Detailed in Section 4.2.2.
    $S_{newnode} \leftarrow$ Accumulating the score of $LO_{selected}$;
    $T_{newnode} \leftarrow$ Accumulating the time of $LO_{selected}$;
    if $(T_P + T_{newnode} <= T_u)$ **then**
      Assign $LO_{selected}$ to $Newnode$;
      $T_P += T_{newnode}$;
      $S_P += S_{newnode}$;
      $P \leftarrow P + Newnode$;
      DFS_{two} $(Newnode, T_u, G, D, P, T_P, S_P, i)$;
    else
      Recom[i] $\leftarrow (P, T_P, S_P)$;
      i++ ;
Return **Recom**;

4.2.3. Time estimation approaches

In this paper, the time for paths is estimated using three methods. It is computed by estimating and summing the learning time of its LO, taking into account the collection of previous users–LO interactions.

4.2.3.1. Clust.Mean and Clust.Median approaches. Our first method to compute time is **Clust.Mean** that uses a k-means clustering algorithm (Hartigan & Wong, 1979). It first finds all LO that are seen by a user. Then, it finds users that have seen those LO and divides them into three clusters (using k-means) considering their time on LO (Distance function:Euclidean). The idea for having three groups is to segment the users based on their learning speed (slow, normal, and quick users). In a situation where there is not enough data to make three clusters (like estimating score for a user having users' binary scores for only one LO), we generate two (slow and quick users). Finally, to estimate time for a target LO, we compute the average time of the users (who are in the same cluster as the target user u) on the target LO (algorithm 3).

Algorithm 3: Clust.Mean algorithm.

Input: User u, Transaction data D, target LO_{tgt} (unvisited by U)
Output: Estimated time/score for LO_{tgt}.
$Seen_u \leftarrow$ Determining all LO seen by u.
$ALL_{seen} \leftarrow$ Determining users that have visited $Seen_u$.
$Clust \leftarrow$ Clustering (ALL_{seen}+$Seen_u$) in 3 or 2 groups using their time/score.
$Clust_{tgt} \leftarrow$ Find the target cluster (includes u) from $Clust$ and drop u from that cluster.
$T_{tgt} \leftarrow$ Average time/score of users in $Clust_{tgt}$ on LO_{tgt}.
Return T_{tgt}

Although other clustering algorithms could have been used, we used k-means, due to its simple implementation and efficiency (Rokach & Maimon, 2005).

Clust.Median is another approach for estimating time. This approach is similar to **Clust.Mean** but it employs the median operation rather than mean.

4.2.3.2. MF.Predict approach. We also used a Matrix Factorization (MF) approach (Koren, Bell, Volinsky, et al., 2009; Nabizadeh, Jorge, Tang, & Yu, 2016; Rafsanjani, Salim, Aghdam, & Fard, 2013) to estimate time. MF is opted since it performs well having $Scalability$ and $Sparsity$ issues (Gillis, 2012). $Sparsity$ occurs when a small portion of LO is opted while $Scalability$ (Burke, 2002) happens when we deal with a large data (users–LO).

MF discovers latent relations between LO and users (Koren, 2008). Assume that T is a matrix that contains n users and m LO, while entries are users' time for LO. T will be decomposed into two matrices by applying a MF technique:

$$T \approx \hat{T} = A \cdot B^T \tag{2}$$

In Eq. (2), A is a user matrix with n rows (as users) and f columns, while B is a LO matrix that has m LO (as rows) and f columns. f is the total number of latent features that are learned from previous users' feedback. Finally, we use a dot product Eq. (3) to predict the time of a LO i for a user u.

$$\hat{T}_{ui} = A_u \cdot B_i^T \tag{3}$$

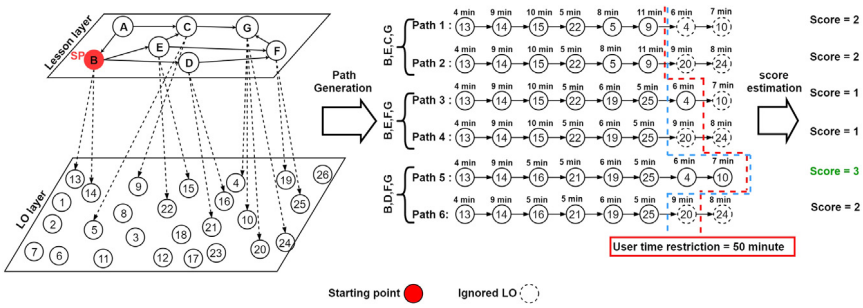


Fig. 2. Example of our method; Red dash-line: ignored LO, blue dash-line: ignored lessons. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2.4. Score estimation approaches

To estimate score, we used similar methods as time estimation while using the users' score instead of time. In datasets that we have used in this paper, score is presented as a binary variable (1: completing a LO, 0: not completing a LO). Thus, a score indicates the ability of a user to correctly complete a LO.

4.2.5. Illustrative example of path generation

Fig. 2 depicts an instance of our approach. As presented in this figure, our approach finds six paths (start by sp) using the DFS algorithm and computes their score and time (time of LO are written on their top). To compute the time for a path, we sum the time of its LO. For instance, the time for **Path 1** is 60 min, which it exceeds the user's time (50 min). Hence, the last lesson is dropped (LO 4 & 10). In Fig. 2, blue dashed-line depicts the ignored lessons while the red one indicates the dropped LO (due to exceeding the user's time). Similar to the time, a path's score is computed by summing the score of its LO.

4.2.6. Path recommendation

After generating a path, it will be recommended lesson by lesson (algorithm 4). This algorithm allows the recommender to monitor and collect a user's interactions data, like his/her learning score and time. It has two main benefits:

1. Monitoring users constantly results in detecting their failure in early stages and avoid wasting their time by adjusting the recommended path.
2. Collected data is used to make paths that fit users' competency level. It raises users' satisfaction and keeping users engaged with the recommender.

In our method, by completing a lesson, a user's score (U_s) is compared with the estimated (expected) score (E_s) for that lesson. If $U_s \geqslant E_s$, the user will receive LO for the next lesson, otherwise, the U_s needs to be compared with a score threshold (δ_s). The δ_s is determined by a course expert considering the usual educational principles to pass a lesson or a course, and it is equal to the minimum score to pass a lesson (i.e. 50% of the max possible score). If $U_s < \delta_s$, the user could not learn the lesson, and the recommender reshows the visited LO by the user for that lesson. In the case that $\delta_s \leqslant U_s < E_s$, auxiliary LO (Section 4.2.6.1) will be recommended to help the user on that lesson.

4.2.6.1. Auxiliary LO. Auxiliary LO are not in the initial generated path for a user, and they will be recommended when he/she could not obtain the estimated/expected score from a lesson. Whenever auxiliary LO are needed, our recommender generates two ranking lists for LO of a lesson using the method in Section 4.2.2. Two lists are made since there are two types of LO, expository and evaluative. To recommend auxiliary LO, the method goes through each list and recommends LO with the highest rank, which are not seen by the user. In the case that recommending auxiliary LO result in exceeding the user's limited time (T_u), our recommender ignores the lessons from the end of the path that exceed the T_u (algorithm 5).

5. Evaluation and discussion

In our evaluation, we first used offline methods to compare the performance of our estimation approaches with the ones introduced in Nabizadeh et al. (2018), Rafsanjani (2018) and Johns, Mahadevan, and Woolf (2006). We then assessed the quality of our recommender approach in a live environment. For that, the performance of two groups of users were compared, one group used our recommender while another group used an e-learning system that delivered LO with a predefined order (A/B test).

Algorithm 4: Path Recommendation algorithm.

Input: Path P, User available time T_u, transaction data D.

1 **for** *(i = 1 to number of lesson in P)* **do**
2 Recommend LO of L_i to u; ▷ L indicates a lesson.
3 $U_s \leftarrow$ Obtained score of u on L_i;
4 $\delta_s \leftarrow$ Minimum score to pass a lesson;
5 **if** *($U_s \geqslant E_s$)* **then**
6 $T_u \leftarrow T_u - T_{L_i}$; ▷ E_s Computed score for a lesson.T_{L_i}:Time of U on L_i.
7 **if** *($T_u < T_{L_{i+1}}$)* **then**
8 Terminate; ▷ $T_{L_{i+1}}$:Estimated (expected) time for the next lesson.
9 **else if** *($\delta_s \leqslant U_s < E_s$)* **then**
10 $L_{aux} \leftarrow$ Estimating auxiliary LO using algorithm 5;
11 $T_{aux} \leftarrow$ Estimating time for L_{aux};
12 **if** *($T_{aux} \leqslant T_u$)* **then**
13 Recommend L_{aux};
14 $T_u \leftarrow T_u - T_{aux}$;
15 $U_{s_{aux}} \leftarrow$ Obtained score of u on L_{aux};
16 **if** *($U_{s_{aux}} \geqslant E_s$)* **then**
17 **Go to** Line 1;
18 **else if** *($\delta_s \leqslant U_{s_{aux}} < E_s$)* **then**
19 **Go to** Line 9;
20 **else**
21 **Go to** Line 24;
22 **else**
23 Terminate;
24 **else**
25 $T_{reshow} \leftarrow$ Time to re-read the same LO;
26 **if** *($T_{reshow} \leqslant T_u$)* **then**
27 Reshow the same LO;
28 $T_u \leftarrow T_u - T_{reshow}$;
29 $U_{s_{re}} \leftarrow$ Obtained score of u on same LO;
30 **if** *($U_{s_{re}} \geqslant E_s$)* **then**
31 **Go to** Line 1;
32 **else if** *($\delta_s \leqslant U_{s_{re}} < E_s$)* **then**
33 **Go to** Line 9;
34 **else**
35 **Go to** Line 24;
36 **else**
37 Terminate;

Algorithm 5: Generating auxiliary LO for a lesson.

Input: User u,Lesson L,Threshold δ_s,Data D,Estimated score (E_s),Obtained score (u_s).
Output: Auxiliary LO for a lesson.

1 $LO_{Ex} \leftarrow$ Select all Ex LO of L ; ▷ Ex: Expository LO.
2 $LO_{Ev} \leftarrow$ Select all Ev LO of L ; ▷ Ev: Evaluative LO.
3 **if** *($\delta_s \leqslant u_s < E_s$)* **then**
4 **for** *(i = 1 to n (number of Ex LO))* **do**
5 $T_i \leftarrow$Estimating time for Ex_i using D ; ▷ No Score/Completion Rate for LO_{Ex}.
6 $V_i \leftarrow$Estimating visited rate for Ex_i using D;
7 $Aux_{Ex} \leftarrow$ Selecting an unvisited LO using T and V ; ▷ Explained in 4.2.2.
8 **for** *(j = 1 to m (number of Ev LO))* **do**
9 $S_j \leftarrow$ Estimating score for Ex_j using D;
10 $Tj \leftarrow$ Estimating time for Ex_j using D;
11 $C_j \leftarrow$ Estimating completion rate for Ex_j using D;
12 $V_j \leftarrow$ Estimating visited rate for Ex_j using D;
13 $Aux_{Ev} \leftarrow$ Selecting an unvisited LO using S, T, C and V ; ▷ Explained in 4.2.2.
14 **Return** (Aux_{Ex}, Aux_{Ev});

5.1. Offline evaluation

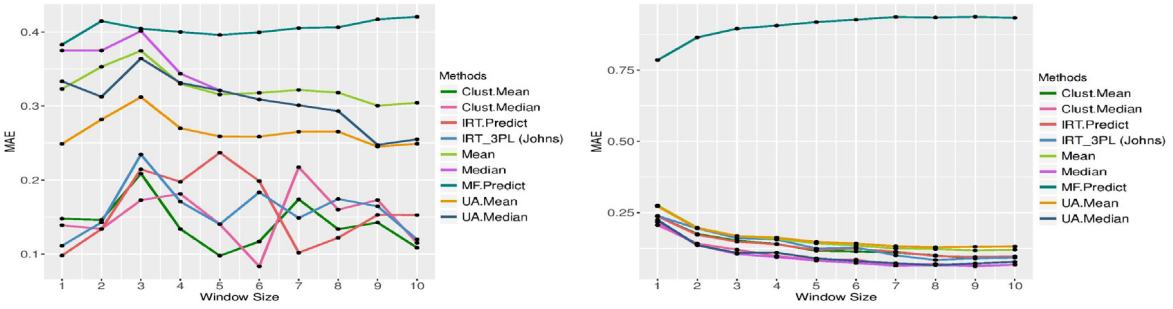
This section is aimed to assess the quality of our estimation approaches. Since we compared our approaches with the ones introduced in Nabizadeh et al. (2018), to have a more accurate comparison, we used the datasets (Table 2) and the evaluation approach used in that paper. To this end, we initially specified a path (a series of LO) that a user selected previously. Next, we hid (ignored) the score and time for a few LO (from the end of the path) and re-estimated them using our methods (these LO are called unobserved LO). Here, the training data was the set of observed LO while test data was the set of unobserved LO.

Table 2
Brief explanation of the datasets.

Datasets	Users	LO	Transactions	Course graph
Mooshak	144	31	2646	×
Enki	61	59	917	✓

Table 3
Optimum MF parameters. Determined by the cross-validation (Nabizadeh et al., 2016).

Datasets	λ	η	f	$iter$
Mooshak	0.09	0.01	10	60
Enki	0.01	0.09	5	30



(a) Enki. (b) Mooshak.

Fig. 3. Score computation-Average MAE.

5.1.1. *Generating mf.predict model*

Before evaluating our estimation approaches, we needed to build an optimal model for the **MF.Predict** approach by optimizing its parameters (Regularization: λ, Learning rate: η, Iteration's number: $iter$, Factors' number: f). The λ is to avoid overfitting to training data while the η is useful, especially when the training data is large. The η is significant for having the balance between the convergence rate and the accuracy of the algorithm (Luo, Xia, & Zhu, 2013) (Optimized parameters are in Table 3).

5.1.2. *Evaluating estimation approaches*

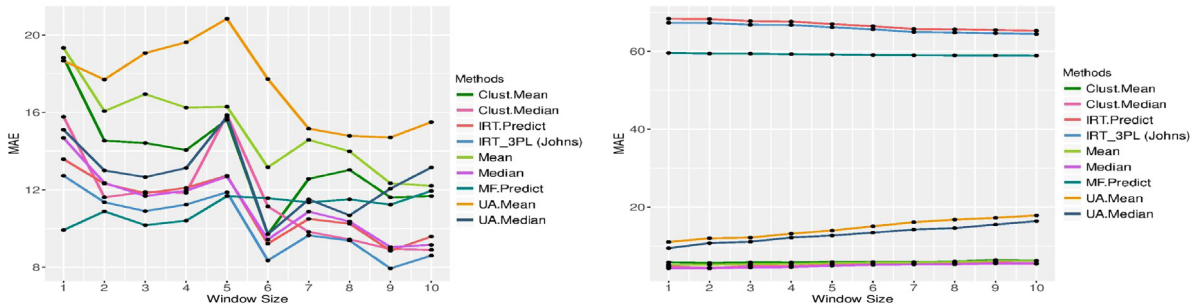
In this section, the quality of our introduced estimation approaches were compared with six other methods in the literature, which are:

1. **Median, Mean**:
   The median and mean of time (or score) of users that already seen a LO was computed and assigned as time (or score) of that LO for a target user. These methods were used in Nabizadeh et al. (2018) as baselines.
2. **UA.Median, UA.Mean**: First, authors computed how good (for score) or quick (for time) was a user in comparison to the others in completing LO (ratio). Next, a multiplication of the average (or median) of time (or score) of other users on the target LO and the ratio gave the time (or score) of the LO for the user (Nabizadeh et al., 2018).
3. **IRT.Predict**: It was introduced in Nabizadeh et al. (2018) and it was relied on a 2 Parameters Item Response Theory (2PL-IRT) (An & Yung, 2014).
4. **IRT-3PL**: It was proposed in Johns et al. (2006), which predicted time and score using a 3 parameters Item Response Theory (3PL IRT).

Fig. 3 shows the average Mean Absolute Error (MAE) results for the score estimation. Regarding these results, **MF.Predict** performed worse than the others. It could have two reasons: MF-based methods required enough data to train; or MF-based methods often performed well for sparse data. Fig. 3 also presents a few approaches outperformed other methods for each dataset. Among these approaches, the **Clust.Median** performed well for both datasets.

In time estimation, like estimating score, the performance of a few approaches were competitive. Regarding the results in Fig. 4, **Median**, **Mean**, **Clust.Median** and **Clust.Mean** outperformed the rest of approaches in Mooshak data, while in Enki it was not clear. Among the mentioned methods, **Clust.Mean** and **Clust.Median** were the ones that performed well in estimating score for both datasets (**IRT-3PL** performed well in estimating score but it generated more outliers or bad estimations than the **Clust.Mean** and **Clust.Median**). So, we wanted to select one of the two approaches, **Clust.Mean** or **Clust.Median** to estimate time and score. For that, we conducted another evaluation (Section 5.1.3).

In the evaluation approach, raising the number of unobserved LO (window size in Figs. 3 and 4) leads to reducing the observed LO cases, which it reduces the size of the training dataset. In general, a larger training set leads to have more precise results but in our case it was not clear due to not having enough data for monitoring how the window size influenced the results of evaluation.



(a) Enki. (b) Mooshak.

Fig. 4. Time computation-Average MAE.

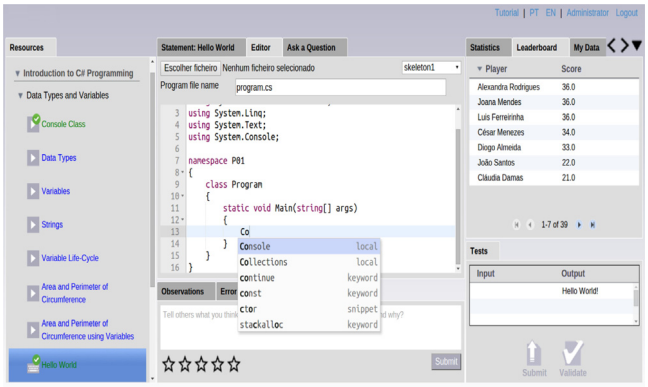


Fig. 5. Interface of our recommender. Lessons are in the **Resources** tab, like **Strings**.

Table 4
Under and Over estimating time and score in Clust.median and Clust.Mean.

	Methods	Enki		Mooshak		Enki		Mooshak		
		Over	Under	Over	Under	Over	Under	Over	Under	
Score	Clust.Median	88%	12%	90%	10%	98.80%	1.20%	99.70%	0.30%	Time
	Clust.Mean	79.80%	20.20%	90.90%	9.10%	98.80%	1.20%	99.60%	0.40%	

5.1.3. Underestimation and overestimation assessment

As mentioned previously, both **Clust.Median** and **Clust.Mean** performed well in score and time computation. For opting one of them, we evaluated them in the case of underestimating and overestimating the score and time. This evaluation was performed because time underestimation is riskier than overestimation since a user might not finish a path within the computed time, while overestimating score is riskier since user might not get the computed score.

Regarding the results in Table 4, although both methods performed well in time estimation, **Clust.Mean** performed better in score estimation. Thus, we selected **Clust.Mean** to estimate time and score for users.

5.2. Online evaluation

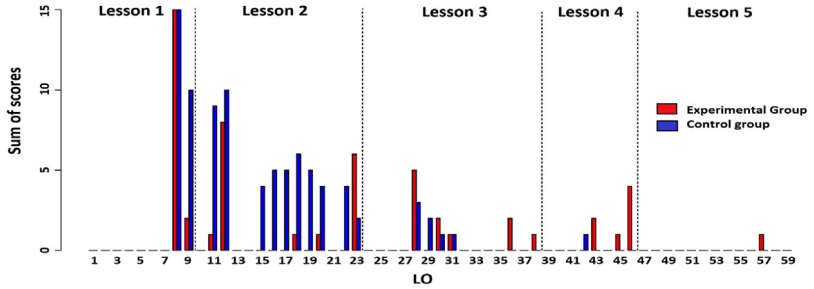
This section is aimed at assessing the quality of our recommender approach. Thus, we briefly describe the implementation of our recommender. We then explain our experiment to assess the quality of the recommender.

5.2.1. Recommender implementation

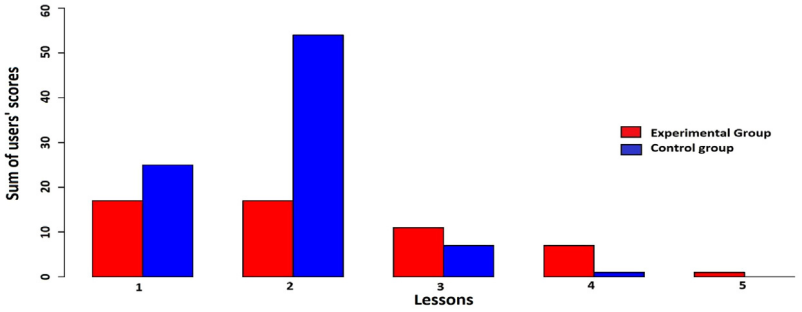
We developed our approach using Google Web Toolkit (GWT) (Hanson & Tacy, 2007) and integrated it with an e-learning system, called Enki (Paiva, Leal, & Queirós, 2016). Enki was embedded in Mooshak system (Leal & Silva, 2003). Mooshak manages coding contest (automating evaluation of codes), and can communicate with other e-learning systems, like learning management system (LMS). Fig. 5 depicts the GUI of our recommender.

Table 5
Testing Groups' homogeneity. *P*-value was estimated using the Fisher exact test (McDonald, 2009) that did not reject the null hypothesis (two groups were similar).

	Gender		Portuguese level		Coding level		
	M	F	Know	None	None	Generic	Know C#
Control	13	3	14	2	5	11	0
Experimental	13	3	10	6	7	9	0
	P-value = 1		*P*-value = 0.22		*P*-value = 0.716		



(a) Sum of participants' scores on each LO.



(b) Sum of participants' scores on lessons.

Fig. 6. Comparing the effectiveness of both groups on different LO and lessons.

5.2.2. Groups formation

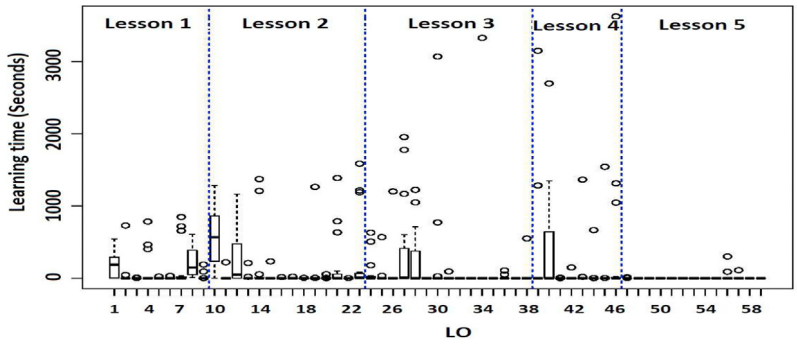
32 participants were divided in control and experimental groups to attend a short course on C# programming (5 lessons,59 LO). Both groups used two different models of the Enki. The experimental group used the one that guided participants using recommendations while control group used the model that delivered LO having a predefined order. Finally, we compared their performances.

Due to having a small group of participants that were so diverse, to ensure that both groups had negligible differences, we assigned participants to groups manually using three criteria (Table 5). Knowing Portuguese language was considered since some of the participants did not know Portuguese and it could affect their performance. These participants could use the Google translator.

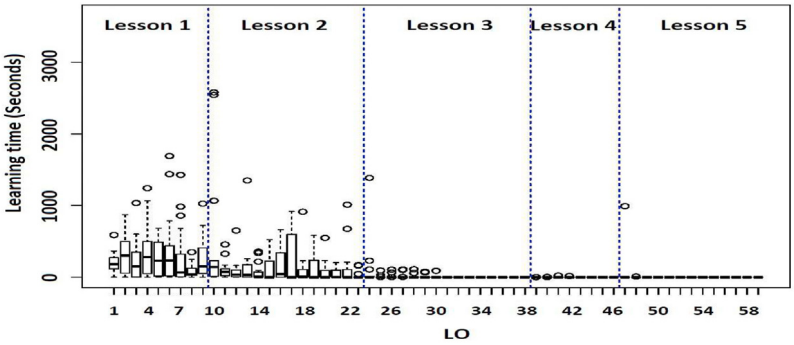
5.2.3. Experimental methodology

Users use a system when it makes the learning process more efficient, effective and attractive. These are the common measures in educational studies (Vesin et al., 2013). Effectiveness is the number of correctly completed LO/lessons by users. Efficiency is the time that users spend to get their goals, while the attractiveness shows their satisfaction with the system. We used these measures to assess the groups' performances and to validate the following hypotheses:

1. Our recommender promoted a higher lesson coverage than the baseline (assessing the effectiveness and efficiency on the course).
2. Users of our recommender got better scores on the final exam than the ones without recommendation (assessing the effectiveness on the final exam).
3. Considering the course's time, experimental group was more satisfied with the exam's score than the control group (assessing the attractiveness).



(a) Experimental group.



(b) Control group.

Fig. 7. Comparing the efficiency of both groups on different LO and lessons.

In our experiment, first, both groups attended a course for 2 h. Then, they attended a small exam for 1 h (equal for both groups having 5 practical questions) to assess their knowledge on the learned lessons during the course. The questions should be answered by correct compilable solutions. Finally, groups answered two different short questionnaires to provide their opinions about the systems (control group: 2 questions, experimental group:5 questions).

5.2.4. Course performance

To validate the first hypothesis, we first compared the completion of LO and lessons (effectiveness) by the groups. As shown in Fig. 6, the control group completed more LO from the first two lessons while the experimental group was able to complete more lessons. One possible reason is, the control group received the LO sequentially and answered them as they were received while the experimental group completed only those LO that were recommended. Hence, the experimental group could complete more lessons than the other group.

Secondly, we assessed both groups in terms of efficiency for the course coverage. For that, we monitored the time that groups spent on LO and lessons. As shown in Figs. 7 and 8, the control group mainly focused on the first two lessons while the other group completed four lessons within the same amount of time. Therefore, Figs. 6 to 8 show that the efficiency and the effectiveness of the experimental group were highly enhanced in comparison to the control group.

To evaluate how significant was the difference between the course coverage of two groups (first hypothesis), we estimated the *p*-value (Wasserstein, Lazar, et al., 2016) using the groups' time. For that, we counted the lessons that a user spent more than five 5 min for them, since for each lesson a user needed to complete one expository LO (a video 3–4 min), and one evaluative (at least 1 min). So, two samples having 16 values were made. The estimated *p*-value using the Wilcoxon–Mann–Whitney (Neuhäuser, 2011) and Kruskal–Wallis tests (McKight & Najab, 2010) was **0.001024**, which strongly rejected the null hypothesis (i.e. same course coverage for the groups). In this paper, the significance level is 0.1.

Using score, we also made two samples including 16 values by counting the lessons that had at least one LO graded for each user. The *p*-value **0.07727** (via both tests) rejected the null hypothesis. So, the p-values show that the efficiency and the effectiveness of our recommender were higher than the baseline.

5.2.5. Final exam performance

In the final exam, we compared the effectiveness of both groups. It allowed us to assess the difference between the gained knowledge and also to determine how much our recommender was successful in achieving our main goal. The exam was similar

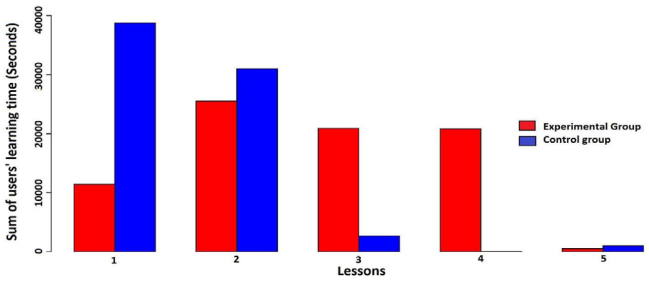


Fig. 8. Sum of time that participants spent on each lesson.

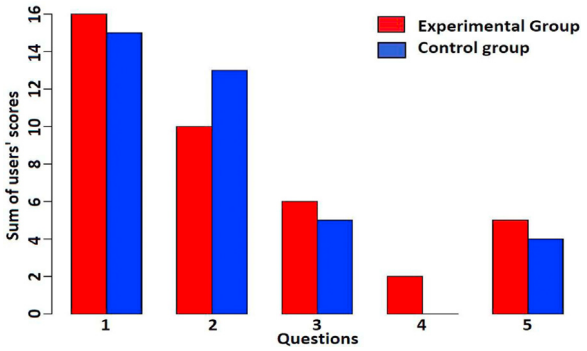


Fig. 9. Sum of the participants' scores on each question.Correct answer = 1, otherwise 0.

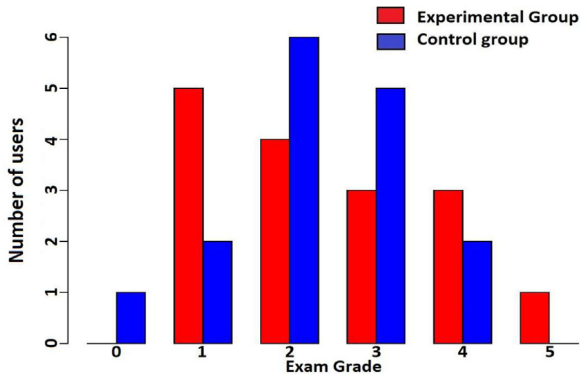


Fig. 10. Frequency of final scores for participants in both groups.

for both groups and had one question per lesson (5 questions). The time of the exam was not considered since in all educational exams participants have a similar amount of time and they can spend it as they want.

Fig. 9 shows that the experimental group performed better the other group in answering all questions except the second one (all experiment group answered 1st question correctly). The reason is, each question was associated with a lesson and the control group could complete the second lesson better (Fig. 6).

The minimum and the maximum scores for the exam (for a participant) were 0 and 5. Fig. 10 shows that the worst result was made by the control group, while the best score was obtained by the experimental group. Although the experimental group got more "4" and "5" than the other group, the number of participants that their scores were equal or less than 2 (9 participants), and more than 2 (7 participants) was equal for both groups. So, if we consider 3 as the passing score for the exam, both groups performed almost similar.

To validate the second hypothesis, we estimated the *p*-value (**0.9571**) using similar methodology applied for the first hypothesis. Fig. 10 and the *p*-value show that although the experimental group got more higher scores than the other group, this difference was not statistically significant. We believe that the relatively small size of the samples may partly explain this lack of significance.

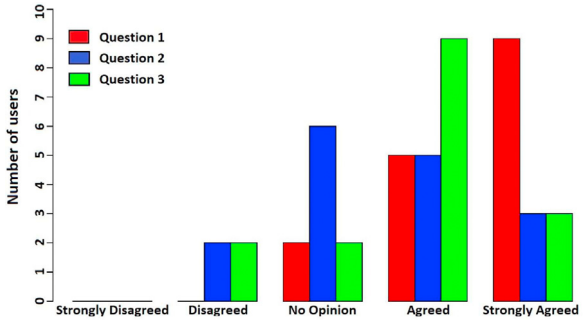


Fig. 11. Participants' opinions about the quality of the recommender.

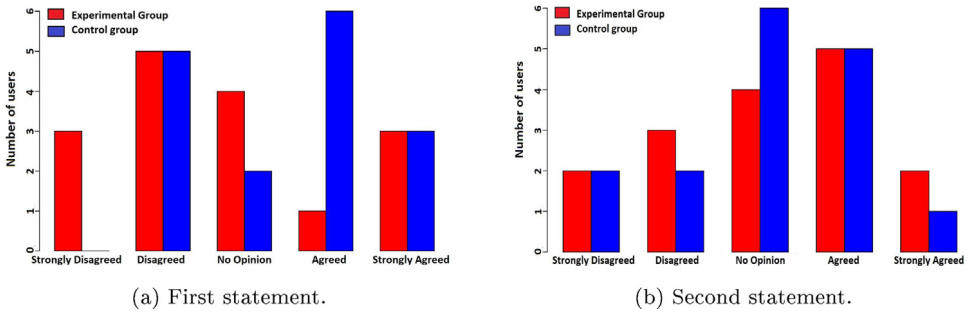


Fig. 12. Participants' opinions about achieving the goal by the systems.

5.2.6. Participants' satisfaction assessment

This evaluation was made in two steps. First, after the exam, a questionnaire (using a five-point Likert scale) was provided to gather the experimental group's opinions about the quality of our recommender. We limited our questionnaire to the following statements since any other statement could be related to the Enki and not our recommender (might mislead us). The statements were:

1. Recommendations were generated quickly.
2. Recommendations were helpful for completing the course.
3. Auxiliary LO were helpful to learn a lesson and answer the questions.

As shown in Fig. 11, 87.5% of participants "agreed" or "strongly agreed" with the first statement, while 75% of them had the same opinions about the third statement. In addition, 50% of them were satisfied with the usefulness of the recommended LO in completing the course (second statement).

In the second step, we collected the groups' opinions about the success of the systems in achieving our main goal. For that, two statements were designed and participants rated them using a five-point Likert scale. The statements were:

1. I could understand most of the course within time.
2. Regarding the time for the course, I am satisfied with my final score.

Fig. 12(a) shows that the control group was more satisfied with the amount of the course that they completed than the other group (first statement). It is against the results in Figs. 6 to 8 (higher course coverage by the experimental group). One reason could be that the participants were not careful enough and rated the statement mechanically. Fig. 12(b) presents almost similar results for both groups, which comply with the results in Fig. 10 (similar groups' results for the exam). A reason for having such results is, confusing the course's score with the exam's score by the participants. Hence, since some of the participants in the control group had a good performance on the course, they high rated this statement. So, due to having this problem, the third hypothesis was not tested.

6. Discussion

Although we could generate paths and maximize users' score under their given time, our recommendation approach has several drawbacks. In this section, we highlight the main drawbacks of our study.

6.1. Dataset availability

One of our main challenges was the lack of publicly available datasets in e-learning area, which contains the users' score and time for Lessons or LO. Although there were several relevant datasets, they were often proprietary and could not be released due to the privacy concerns. To this end, we could not perform extensive experiments (i.e. offline assessment) and evaluate our approaches in the case of scalability issue. So, we employed two relatively small datasets for the offline evaluation.

6.2. Experiment size

In a test involving human subjects, opting an appropriate sample size is one of the main problems. An under-sized test might not challenge the approaches enough to depict their shortages while an over-sized test besides being costly (i.e. money and time), might not be essential. In this study, although we performed offline and online tests, our work could benefit from tests with longer duration and more users.

6.3. Accuracy of estimation methods

As detailed in Section 4.2.3, we introduced various methods to estimate time and score. Although the selected method (**Clust.Mean**) could estimate score and time (results in Section 5.1.2), it generated some outliers (i.e. bad estimations) that need to be studied. Also, this method performed well in overestimating and underestimating time while for score it did not perform as expected.

7. Research recommendations

The proposed methods and obtained results as well as the mentioned limitations suggest additional research directions. The most promising ones are:

7.1. Cold start problem

We selected the **Clust.Mean** approach to estimate time and score (Section 4.2.3), which needed sufficient amount of data (from users and LO) for estimation. So, in the case that there is not enough data, it suffers from Cold-start problem (i.e. not having enough data from LO or users to estimate time or score).

7.2. Users and LO metadata

In this study, we used the minimal users' information (i.e. score and time) to estimate score and time since users often do not share additional information (e.g. feedback on the difficulty of LO). It is of interest to analyze the effect of using various type of information (e.g. difficulty level of LO) in enhancing the results accuracy.

7.3. Scalability and big data

One of the main features of path recommenders is to react rapidly to keep users engaged with the recommender. It can be affected by large scale datasets. So, it is important to design a scalable system, which handles large datasets. Although our opted estimation approach (**Clust.Mean**) theoretically should be able to handle the large datasets, because of inaccessibility of large datasets we could not confirm it.

7.4. Update scheduling

In our method, we updated users' time and score after completing each lesson. It is of interest to find an appropriate time to update a user's profile since users have various progress speed that updating repeatedly might be computationally costly and unnecessary while delaying it might mislead the users. Also, the used information to generate a path can have different priorities for users, so, it can be important to build paths and update users' profiles considering these priorities.

7.5. User-centered course structure

A path recommender often generates paths from a course graph, which is similar for all users (designed by a course expert). It is significant to build paths considering the users' preferences to follow a course because users might not always follow the same structure as a course expert to learn a course. For instance, in our method, expository LO were recommended to users first while users might prefer to initially receive evaluative LO (e.g. Socratic order).

7.6. Evaluation framework

One of the main drawbacks in path recommenders is the lack of a general evaluation framework that enables the researchers to assess and compare their methods. It can include information regarding the available data, key factors that need to be evaluated (e.g. grade), the required information as well as the metrics that need to be used for the evaluation. It can be significantly useful in promoting the research in this area.

8. Conclusion

This paper is aimed at recommending a path that fulfills a user restricted time while maximizing his/her score. For that, we first briefly explain a method that is introduced in Nabizadeh et al. (2018), which has a similar goal to our method, and describe its drawbacks. Next, we present our adaptive method. In this method, initially the DFS algorithm is used to find all lesson sequences from a two-layered course graph, which start by a starting point. Then, our method assigns LO to each lesson of each path, computes score and time for paths by our approaches, and recommends a path that fulfills the restricted time of a user while maximizing his/her score. Recommending a path occurs lesson by lesson, which results in having a control environment for the path recommendation and collecting the user's interaction data to update the path for him/her. During the path recommendation, if the user could not get the expected score from a lesson, our method recommends auxiliary LO. Our recommender is assessed using offline and online approaches while user satisfaction is assessed using questionnaires. In the future, our goal is to extend our approach to identify users' starting points automatically, and also tackle the user and LO cold start.

CRediT authorship contribution statement

Amir Hossein Nabizadeh: Data curation, Formal analysis, Writing - original draft. **Daniel Gonçalves:** Supervision. **Sandra Gama:** Writing - review & editing. **Joaquim Jorge:** Funding acquisition. **Hamed N. Rafsanjani:** Writing - review & editing.

Acknowledgment

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT), Portugal with reference UID/CEC/50021/2019 and project GameCourse, Portugal - PTDC/CCI-CIF/30754/2017.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Data engineering, 1995. Proceedings of the eleventh international conference on* (pp. 3–14). IEEE.

An, X., & Yung, Y.-F. (2014). Item response theory: what it is and how you can use the IRT procedure to apply it. *SAS Institute Inc. SAS364-2014*, *10*(4).

Basu, P., Bhattacharya, S., & Roy, S. (2013). Online recommendation of learning path for an e-learner under virtual university. In *International conference on distributed computing and internet technology* (pp. 126–136). Springer.

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

Bhaskar, M., Das, M. M., Chithralekha, T., & Sivasatya, S. (2010). Genetic algorithm based adaptive learning scheme generation for context aware e-learning. *International Journal on Computer Science and Engineering*, *2*(4), 1271–1279.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370.

Carchiolo, V., Longheu, A., & Malgeri, M. (2010). Reliable peers and useful resources: Searching for the best personalised learning path in a trust-and recommendation-aware environment. *Information Sciences*, *180*(10), 1893–1907.

Dharani, B., & Geetha, T. (2013). Adaptive learning path generation using colored Petri nets based on behavioral aspects. In *2013 International conference on recent trends in information technology* (pp. 459–465). IEEE.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics* (pp. 227–263). Springer.

Durand, G., Belacel, N., & LaPlante, F. (2013). Graph theory based model for learning path recommendation. *Information Sciences*, *251*, 10–21.

Durand, G., Laplante, F., & Kop, R. (2011). A learning design recommendation system based on markov decision processes. In *KDD-2011: 17th ACM SIGKDD conference on knowledge discovery and data mining, San Diego*.

Dwivedi, P., Kant, V., & Bharadwaj, K. K. (2018). Learning path recommendation based on modified variable length genetic algorithm. *Education and Information Technologies*, *23*(2), 819–836.

Feng, X., Xie, H., Peng, Y., Chen, W., & Sun, H. (2010). Groupized learning path discovery based on member profile. In *ICWL workshops* (pp. 301–310). Springer.

Fournier-Viger, P., Faghihi, U., Nkambou, R., & Nguifo, E. M. (2010). Exploiting sequential patterns found in users' solutions and virtual tutor behavior to improve assistance in ITS. *Journal of Educational Technology & Society*, *13*(1).

Garrido, A., Fernández, S., Morales, L., Onaindía, E., Borrajo, D., & Castillo, L. (2013). On the automatic compilation of e-learning models to planning. *The Knowledge Engineering Review*, *28*(02), 121–136.

Garrido, A., Morales, L., & Serina, I. (2012). Using AI planning to enhance e-learning processes. In *Proceedings of the twenty-second international conference on automated planning and scheduling* (pp. 47–55).

Gawthrop, J. (2014). Measuring student achievement: A study of standardized testing and its effect on student learning. Retrieved October, 19, 2016.

Gillis, N. (2012). Sparse and unique nonnegative matrix factorization through data preprocessing. *Journal of Machine Learning Research (JMLR)*, *13*(Nov), 3349–3386.

Govindarajan, K., Kumar, V. S., et al. (2016). Dynamic learning path prediction-a learning analytics solution. In *Technology for education (T4E), 2016 IEEE eighth international conference on* (pp. 188–193). IEEE.

Hanson, R., & Tacy, A. (2007). *GWT in action: easy Ajax with the Google Web toolkit*. Dreamtech Press.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Johns, J., Mahadevan, S., & Woolf, B. (2006). Estimating student proficiency using an item response theory model. In *International conference on intelligent tutoring systems* (pp. 473–480). Springer.

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

Kardan, A. A., Ebrahim, M. A., & Imani, M. B. (2014). A new personalized learning path generation method: ACO-map. *Indian Journal of Scientific Research*, *5*(1), 17.

Klašnja-Milićević, A., Vesin, B., Ivanović, M., & Budimac, Z. (2011). E-learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education*, *56*(3), 885–899.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 426–434). ACM.

Koren, Y., Bell, R., Volinsky, C., et al. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Krauss, C. (2018). *Time-dependent recommender systems for the prediction of appropriate learning objects* (Technische universitat Berlin doctoral thesis), http://dx.doi.org/10.14279/depositonce-7119.

Krauss, C., Salzmann, A., & Agathe, M. (2018). Branched learning paths for the recommendation of personalized sequences of course items. In *DeLFI workshops, co-located with 16th e-learning conference of german computer society*.

Leal, J. P., & Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. *Software - Practice and Experience*, *33*(6), 567–581. http://dx.doi.org/10.1002/spe.522.

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

Lin, C. F., Yeh, Y.-c., Hung, Y. H., & Chang, R. I. (2013). Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, *68*, 199–210.

Luo, X., Xia, Y., & Zhu, Q. (2013). Applying the learning rate adaptation to the matrix factorization based collaborative filtering. *Knowledge-Based Systems*, *37*, 154–164.

McDonald, J. H. (2009). *Handbook of biological statistics (Vol. 2)*. Sparky House Publishing Baltimore, MD.

McKight, P. E., & Najab, J. (2010). Kruskal-Wallis test. *The Corsini Encyclopedia of Psychology*, 1. http://dx.doi.org/10.1002/9780470479216.corpsy0491.

Nabizadeh, A., Jorge, A., & Leal, J. P. (2015). Long term goal oriented recommender systems. In *Proceedings of the 11th international conference on web information systems and technologies - Volume 1: WEBIST, 552-557, 2015, Lisbon, Portugal*.

Nabizadeh, A. H., Jorge, A. M., & Leal, J. P. (2018). Estimating time and score uncertainty in generating successful learning paths under time constraints. *Expert Systems*, *36*(2), e12351.

Nabizadeh, A. H., Jorge, A. M., Tang, S., & Yu, Y. (2016). Predicting user preference based on matrix factorization by exploiting music attributes. In *Proceedings of the ninth international c* conference on computer science & software engineering* (pp. 61–66). ACM.

Nabizadeh, A. H., Mário Jorge, A., & Paulo Leal, J. (2017). Rutico: Recommending successful learning paths under time constraints. In *Adjunct publication of the 25th conference on user modeling, adaptation and personalization* (pp. 153–158). ACM.

Neuhäuser, M. (2011). Wilcoxon–Mann–Whitney test. In *International encyclopedia of statistical science* (pp. 1656–1658). Springer, http://dx.doi.org/10.1007/978-3-642-04898-2_615.

Paiva, J. C., Leal, J. P., & Queirós, R. A. (2016). Enki: A pedagogical services aggregator for learning programming languages. In *ACM conference on innovation and technology in computer science education* (pp. 332–337). ACM.

Rafsanjani, A. H. N. (2018). *A long term goal recommender approach for learning environments* (pp. 1–137). Portugal: University of Porto.

Rafsanjani, A. H. N., Salim, N., Aghdam, A. R., & Fard, K. B. (2013). Recommendation systems: a review. *International Journal of Computational Engineering Research (IJCER)*, *3*(5), 47–52.

Rokach, L., & Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook* (pp. 321–352). Springer.

Salahli, M. A., Özdemir, M., & Yasar, C. (2013). Concept based approach for adaptive personalized course learning system. *International Education Studies*, *6*(5), 92–103.

Suazo, I. G., Rodríguez, C. G., & Rivas, M. C. (2012). Generating adaptive learning paths in e-learning environments. In *Informatica (Clei), 2012 Xxxviii conferencia latinoamericana en* (pp. 1–10). IEEE.

Tam, V., Lam, E. Y., & Fung, S. (2012). Toward a complete e-learning system framework for semantic analysis, concept clustering and learning path optimization. In *ICALT* (pp. 592–596). IEEE, http://dx.doi.org/10.1109/ICALT.2012.66.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, *1, 2*(2), 146–160. http://dx.doi.org/10.1137/0201010.

Vesin, B., Milicevic, A. K., Ivanovic, M., & Budimac, Z. (2013). Applying recommender systems and adaptive hypermedia for e-learning personalizatio. *Computing and Informatics*, *32*(3), 629–659.

Wasserstein, R. L., Lazar, N. A., et al. (2016). The ASA's statement on p-values: context, process, and purpose. *The American Statistician*, *70*(2), 129–133.

Xie, H., Zou, D., Wang, F. L., Wong, T.-L., Rao, Y., & Wang, S. H. (2017). Discover learning path for group users: A profile-based approach. *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2016.08.133.

Xu, D., Wang, Z., Chen, K., & Huang, W. (2012). Personalized learning path recommender based on user profile using social tags. In *Fifth international symposium on computational intelligence and design* (pp. 511–514). IEEE.

Yang, F., Li, F., & Lau, R. (2012). Learning path construction based on association link network. In *Advances in web-based learning-ICWL 2012* (pp. 120–131). Springer.

Yarandi, M., Jahankhani, H., & Tawil, A.-R. H. (2013). A personalized adaptive e-learning approach based on semantic web technology. *Webology*, *10*(2), 1.

Ye, J.-m., Song, X., Chen, X., Luo, D.-x., Wang, Z.-f., & Shu, C. (2018). Research on learning path recommendation algorithms in online learning community. In *ECAR international conference* (pp. 326–333).

Zhou, Y., Huang, C., Hu, Q., Zhu, J., & Tang, Y. (2018). Personalized learning full-path recommendation model based on LSTM neural networks. *Information Sciences*, *444*, 135–152.

Zhu, H., Tian, F., Wu, K., Shah, N., Chen, Y., Ni, Y., et al. (2018). A multi-constraint learning path recommendation algorithm based on knowledge map. *Knowledge-Based Systems*, *143*, 102–114.