A long term goal recommender approach for learning environments

Amir Hossein Nabizadeh Rafsanjani

Programa Doutoral em Informática das
Universidades do Minho, Aveiro, Porto
Departamento de Ciência de Computadores
2018

Orientador
Alípio Mário Guedes Jorge
Professor Associado
Faculdade de Ciências da Universidade do Porto

Coorientador
José Paulo Leal
Professor Auxiliar
Faculdade de Ciências da Universidade do Porto

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: [illegible] DATE: 25. June. 2018

Dedication and acknowledgements

First, I would like to thank my supervisor, Prof. Alípio Mário Jorge and my co-supervisor, Prof. José Paulo Leal. Their commitment to my success in the development of this thesis was notorious, both by the enriching discussions we had and by their constant encouragement to pursue my goals.

I also would like to thank my loving parents for inspiring and supporting me unconditionally and for being proud of me. My greatest thank is for my two brothers, Dr. Hamed and Dr. Hesamaddin and my beloved sister, Dr. Azadeh. This work would not have taken place, and this thesis would not exist without them.

سپاس گزارم

In addition, I would like to thank José Carlos Paiva for his wonderful collaboration, in particular during the online evaluation phase of this thesis. His supported us greatly and was always willing to help. Finally, I want to thank my friends, especially Joao Vinagre and Diogo Marcelo Nogueira, who serve as inspirations while doing this thesis.

Abstract

Current recommenders concentrate on the immediate needs of users. This is insufficient for achieving long term goals. Therefore, we propose Long Term Recommender Systems (LTRS) that besides satisfying immediate needs of users, drive them toward a predefined long term goal by generating a set of relevant recommendations step by step. LTRS can be applied in different domains such as music, tourism, and E-learning.

One of the main challenges in E-learning is recommending learning materials that students can timely complete. Fulfilling this requirement becomes more challenging when students are not able to devote enough time. Thus, to study, a student faces two main questions: (1) What am I able to learn from a course in my limited time? (2) Do the learning outcomes (e.g. score) justify the time that I spend? Therefore, in this thesis we introduce two approaches, which are instances of LTRS, to maximize the students' grades from a course while satisfying their requirements and constraints. These approaches recommend potentially successful paths based on the available time and knowledge background of a student. The first approach uses a one-layer directed graph (course graph) to generate paths while the second one is based on a two-layered course graph to cover the problems of the first approach. In the course graphs, vertices show the learning objects (LO) or lessons while the edges indicate the precedence relations among them.

These approaches start by generating paths considering the knowledge background of a student. Paths are generated from the course graphs. These approaches then estimate time and score for the paths using the same estimation methods. Furthermore, they use identical methods to estimate the probability of underachieving the estimated score for a path, and also of not completing a path under a time constraint of the student. Finally, they recommend a path that satisfies the limited time of the student while maximizing the score.

The evaluation of the proposed approaches was twofold. Firstly, we assessed the quality of time and score estimation methods using offline approaches. Secondly, we evaluated the quality of the approach based on a two-layered course graph in a live environment. For that, we implemented and embedded it in an E-learning system. We then performed an experiment to compare the performance of two groups.

Resumo

Os atuais sistemas de recomendação concentram-se nas necessidades imediatas dos utilizadores. Isso é insuficiente para alcançar objectivos de longo prazo. Para esse fim, propomos os Sistemas de Recomendação de Longo Prazo (*Long Term Recommender Systems* - LTRS) que, além de satisfazer as necessidades imediatas dos utilizadores, os direccionam para um objetivo predefinido de longo prazo, gerando um conjunto de recomendações relevantes passo a passo. LTRS podem ser aplicados em diferentes domínios, como música, turismo e E-*learning*.

Um dos principais desafios em E-learning é recomendar conteúdos que os alunos possam concluir em tempo útil. Cumprir esse requisito torna-se mais desafiante quando os alunos não são capazes de lhe dedicar tempo suficiente. Assim, para estudar, um aluno enfrenta duas questões principais: (1) O que sou capaz de aprender com um curso, em tempo limitado? (2) Os resultados da aprendizagem (por exemplo a classificação) justificam o tempo gasto? Nesta tese introduzimos duas abordagens, que são instâncias de LTRS, para maximizar as notas dos alunos de um curso, enquanto se satisfazem os seus requisitos e restrições. Essas abordagens recomendam trajetos potencialmente bem sucedidos com base no tempo disponível e nos conhecimentos do aluno. A primeira abordagem utiliza um grafo direccionado de uma camada (grafo do curso) para gerar trajetos, enquanto a segunda abordagem é baseada num grafo do curso de duas camadas, para cobrir os problemas da primeira abordagem. Nos grafos do curso, os vértices representam os Objectos de Aprendizagem (*Learning Objects* - LO) ou aulas, enquanto as arestas indicam a relação de precedência entre eles.

Essas abordagens começam por gerar trajetos considerando os conhecimentos prévios de um aluno. Os trajetos são gerados a partir dos grafos do curso. Essas abordagens estimam o tempo e a classificação para os trajectos que foram gerados, utilizando os mesmos métodos de estimativa. Além disso, eles utilizam métodos idênticos para estimar a probabilidade de não alcançar a classificação estimada por um trajeto, e também de não completar um trajeto, devido a restrições de tempo do aluno. Finalmente, recomenda um trajecto que satisfaça as restrições de tempo do aluno, enquanto maximiza a classificação.

As abordagens propostas foram avaliadas duplamente. Primeiro, avaliámos a qualidade dos métodos que estimam o tempo e a classificação, utilizando abordagens *off-line*. Depois, avaliámos a qualidade da abordagem com base num grafo do curso de duas camadas, num ambiente real. Para isso, foi implementada e incorporada num sistema de E-learning. Em seguida, realizámos uma experiência para comparar o desempenho de dois grupos.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

With the emergence of complex E-learning environments, characterized by large-scale information, high interactivity and no space-time constraints [ACBT04, MS04], personalization is becoming a significant feature in E-learning systems. Users of these systems have different goals, background, capabilities, and personalities. Personalized learning occurs when E-learning systems are designed regarding educational experiences that match the requirements, goals, and interests of the users. The personalization can be obtained using recommendation techniques.

Recommendation techniques and algorithms are proposed for managing information overload by autonomously collecting information and proactively tailoring it to individual interests [AT05], e.g., what item to purchase (Amazon), what music to listen to (Last.fm), which place to visit (TripAdvisor). Currently, the main search engines, such as Google, and e-shopping websites like Amazon have applied recommendation techniques in their services for personalizing their results for the users. Unfortunately, the regular recommenders' techniques and algorithms are not directly applicable in E-learning area [KMIN15]. For instance, music recommenders rely on the tastes and the interests of the users, while preferred learning activities might not be educationally sufficient for the users [CLA$^+$03]. Even for users with similar interests, we may require to recommend different learning materials and activities, considering their proficiency levels and learning goals. For example, users with no prior knowledge in a specific area initially should be advised to learn basic learning materials while advanced users need to receive more complex materials.

Researchers have introduced various algorithms and techniques to recommend learning materials or optimum browsing path to users, considering their preferences, knowledge [LOdP11], and the browsing history of other users with similar features [DOS$^+$10]. In an ideal manner, E-learning recommenders should help users in performing relevant learning activities that match the users' profiles. These recommendations need to be made at the right time and in the right context while keeping the users motivated and enable them to complete their learning activities efficiently [TM05].

In order to design an effective E-learning recommender, it is significant to determine users' characteristics [GRVDC09, DHK08], such as learning goal, knowledge background, rated learning activities, and learning restrictions (i.e., learning time). E-learning systems need to be capable of recognizing and exploiting these characteristics that can be used for designing the frameworks and implementing the platform for effective E-learning recommenders [ANRR01, ZL01, SGS07].

1.1 Motivation

Current recommenders focus on the immediate needs of users and do not usually take into account long term interests. These are, however, often important. For instance, when a user needs to learn a concept, current recommenders can recommend learning materials regarding his/her level of knowledge. Assuming this user intends to learn a concept that with his/her current knowledge and background, the user would not be able to learn it. Therefore, the main task of the recommender would be recommending learnable (for the user) learning materials successively while promoting own knowledge in a way that he/she can learn the target concept (long term goal). Most of the current recommenders do not have such a strategy and are not able to satisfy these kinds of requirements and goals. To this end, we propose Long Term Recommender Systems (LTRS) that besides satisfying immediate needs of users, conduct them toward a predefined long term goal by generating a set of relevant recommendations step by step. This goal is domain dependent and can be defined by the owner of the system or by users.

LTRS can be applied in different domains. For instance, in E-learning domain, LTRS aid users (e.g. teachers and learners) to have more productive activities (teaching and learning) meanwhile consuming less time. In this case, a long term goal can be defined by a teacher as doing a relevant assignment or passing an exam after getting a long sequence of recommendations.

Another example is in music domain. For example, a music company may be interested in promoting a certain artist or genre. In this situation, the company may use LTRS to guide the users from a preferred music genre to a target genre for enhancing its profit on selected products. In this case, LTRS can gradually influence users' interests through time.

The main task in LTRS is: how to generate recommendation sequences that successfully conduct the user to a target area in the item space or lead to the attainment of broad goals, while satisfying immediate user needs?. A goal can be defined as a predetermined area (in case of music, the area can be a specific genre of music) in the item space of interest to both the user and the platform manager. To achieve a long term goal, a recommendation algorithm must act strategically and not simply tactically. Our main objective by introducing the LTRS is to design a recommendation method that is able to attain strategic goals of users and platform managers.

In this thesis, due to the importance of the E-learning domain, our expertise on it, and our access to an E-learning environment, we have decided to focus on implementing LTRS for the E-learning domain.

1.2 Research Objectives

The main research question in LTRS is: "How can we produce recommendation sequences that successfully conduct users toward a long term goal, while satisfying their requirements?". Subsequently, the main research objective in LTRS is to design a sequential recommendation method that is able to attain strategic goals of users and platform managers. As explained in the previous section, in this thesis, we concentrate on implementing a LTRS for the E-learning domain. Hence, we define our main research question as follows: "How can we generate recommendation sequences (learning paths) that maximize a user's score under a given time restriction?". Our main objective is to design a sequential recommendation method that is able to maximize a user's score while satisfying his/her available time.

The detailed objectives of this thesis are:

1. Providing a systematic review on learning path personalization methods. For that, we need to:
   (a) Identify the main concepts in E-learning, such as learning objects (LO) and learning path.
   (b) Identify the main parameters that are used to personalize the learning paths, such as competency level and learning style.
   (c) Present the methods and algorithms that are used to personalize learning paths as well as the methods that are used to evaluate the personalization methods.
   (d) Find the most important challenges of path personalization methods.
2. Designing a method for generating sequences (paths) that maximize a user's score in a limited time. Therefore, to achieve this objective we plan to:
   (a) Identify the required information from a user to generate paths.
   (b) Identify algorithms and techniques that can be useful for designing a method to generate personalized paths. Our aim is to design an adaptive method, which modifies a learning path regarding a user's feedbacks during his/her learning process.

   (c) Design a method to estimate learning time and score for the learning paths.
   (d) Design a method to estimate the probability of error for the estimated learning time and score for the generated paths.

3. Using appropriate evaluation measures and methodologies for assessing the success of the proposed method.

1.3 Contributions

In this thesis, we provide four contributions. These contributions are published. The publications are listed in section 1.4.

1. We survey the state-of-the-art on learning path personalization methods as well as detailing their advantages and disadvantages. In addition, we highlight the most significant challenges of these methods, which need to be tackled in order to enhance the quality of the personalization (5'th paper in section 1.4.1).
2. We propose an adaptive learning path generation method that takes into consideration a user's feedbacks (e.g. score), restriction and background. It uses an algorithm to recommend learning paths to users and if they could not learn the recommendations, it recommends auxiliary LO (3'rd and 4'th papers in section 1.4.1).
3. We develop various methods to estimate time and score for paths for each user using Item Response Theories [AY14], Clustering techniques [JK13] , Matrix Factorization [KBV09] (3'rd and 4'th papers in section 1.4.1). Researchers often use static learning time values that are specified by a course expert and mentioned in the metadata of LO for estimating the time for paths.
4. We have implemented our recommender using *R* programming language (6'th item in section 1.4.1). For that, we used the SQLite, which is a free library that implements a self-contained SQL database engine. We also have recoded our recommender method in Java and embedded it in Enki. Enki is a web-based learning environment for programming languages. We expect this system will be used at the department of computer science of the University of Porto to help students for learning programming languages.

1.4 Publications and Applications

This section provides a list of the author's publications and applications that are produced during his Ph.D. The listed published papers are divided in two parts, the works that are used directly in this Ph.D. thesis and the ones that are not used.

1.4.1 Works included in this thesis

1. [NJL15a] Nabizadeh, A. H., Jorge, A. M., & Leal, J. P. (2015). Long Term Goal Oriented Recommender Systems. In WEBIST (pp. 552-557).

2. [NJL15b] Nabizadeh, A. H., Jorge, A. M., & Leal, J. P. (2015). Long Term Goal Oriented Recommender Systems. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), Ph.D. consortium.

3. [NMJPL17] Nabizadeh, A. H., Mário Jorge, A., & Paulo Leal, J. (2017, July). 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.

4. Nabizadeh AH, Jorge AM, Leal JP. Estimating time and score uncertainty in generating successful learning paths under time constraints. Expert Systems. 2018;e12351. https://doi.org/10.1111/exsy.12351.

5. Nabizadeh, A. H., Mário Jorge, A., & Paulo Leal, J. (2018). Learning path personalization and recommendation methods: a survey of the state-of-the-art. Submitted to the IEEE Transactions on Learning Technologies.

6. E-learning recommender system. Implementing our learning path recommender approach using *R* programming language. We also implemented and embedded it in Enki [PLQ16], which is a web-based learning environment for programming languages. This system is a part of Mooshak 2.0, a web environment for automated assessment in computer science (https://mooshak2.dcc.fc.up.pt/rutico).

In [NJL15a] and [NJL15b], we present the main idea of Long Term Goal Recommender Systems (LTRS), which directly is related to this thesis. We also introduce the research areas that can be used for developing LTRS, and the methods that can be applied to evaluate the quality of this system.

In [NMJPL17], we present RUTICO, which is an example of Long Term goal Recommender Systems (LTRS). RUTICO utilizes a Depth-first search (DFS) algorithm to find all possible paths for a learner given a time restriction. It 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 "Estimating Time and Score Uncertainty in Generating Successful Learning Paths under Time Constraints", we extend the idea of RUTICO. In this paper, in addition to introducing several methods for estimating time and score for the generated learning paths, we present probability of order for the estimated time and score for the paths. The probability of error for score is the probability of underachieving the estimated score for a learning path by the user. The probability of error for time indicates the probability of not completing a learning path in the user's limited time.

In our survey, which is on the learning path personalization methods, we present an overview of the methods that are applied to personalize learning paths as well as their advantages and disadvantages. We also describe the main parameters for personalizing learning paths. In addition, we present approaches that are used to evaluate path personalization methods. Finally, we highlight the most significant challenges of these methods.

The last item is an E-learning recommender system that we have developed based on paper number 5. We plan to use our recommender in order to help students in their studies.

1.4.2 Works not included in this thesis

1. [NJTY16] Nabizadeh, A. H., Jorge, A. M., Tang, S., & Yu, Y. (2016, July). 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.

In this paper, we propose a method for predicting preferred music feature's value (e.g. Genre as a feature has different values like *Pop*, *Rock*, etc.) of users by modeling not only usage information, but also music description features (music attribute information and usage data are typically dealt with separately). Our method is based on Matrix Factorization (MF) and considers music feature's values as virtual users and retrieves the preferred feature's value for real target users.

1.5 Thesis Outline

The thesis structure is as follows.

- Chapter 2 provides the related work on learning path personalization methods and explains the parameters that these methods used to personalize paths. In addition, it describes the approaches that are applied to evaluate the learning path personalization methods. Finally, it highlights the most significant challenges of these methods.
- Chapter 3 details the main objective of this thesis. It also explains the main problem that needs to be solved during this Ph.D. thesis. In this chapter, our main problem is also divided into sub-problems and each of them is explained in detail.
- Chapter 4 presents two methods for generating learning paths as well as the proposed approaches for estimating time, score and probability of errors for the paths. This

chapter first explains a method that uses a one-layer course graph to generate paths, and describes its drawbacks. It then details the second method, which is based on a two-layered course graph, to cover the problems of the first method.

- Chapter 5 presents the quality assessment of time and score estimation methods that are presented in chapter 4. In this chapter, we also describe the datasets that we have used for our evaluation.
- Chapter 6 details an experiment that is performed to compare the performance of our recommender with another E-learning system that delivers LO to users without any recommendation. This chapter also explains the development of our recommender as well as tools that are used to implement it.
- Chapter 7 concludes the work, describes the main limitations and suggests future research directions.

Chapter 2

State of the art

Technological and pedagogical innovations are redefining education. At the nexus of this convergence is E-learning. Nowadays using E-learning systems such as Intelligent Tutoring Systems (ITS) [PR13] has become a routine. These systems aim to deliver educational resources to users [NMJPL17]. They have several advantages over the traditional learning methods where a teacher was playing the main role and controlling a classroom. The main advantages are availability, reduced cost, improved collaboration, enhanced flexibility (students learn at their own convenience). [TJ15, DSM12].

In the traditional form of E-learning systems, these often caused learning disorientation and cognitive overload by providing users with a bag of disorganized learning materials [BBR13, YLH10]. These problems could become nontrivial when the users had a restricted learning experience, particularly when they were not familiar with a course, or when they had limited time to learn a course [Nai16].

Hence, during the early 1960s, E-learning systems started using a directional sequence of learning materials [YLH10, YD16], and became relying on curriculum sequencing mechanisms. These mechanisms provided users a learning path through learning materials [Chi10, MKKP10]. By generating the learning paths, the E-learning systems offered a "one size fits all" approach, since they provided the same educational resources in the same way to users with different profiles.

"One size fits all" causes several problems. One of the problems is frequent users' failure, since by using this approach the E-learning systems simply ignore the users' knowledge background and their ability to learn. Thus, users are at risk of wasting time with the materials that they are not able to learn. Inability to persuade the users and engage them with the system is another problem, since the users' preferences (e.g. learning style) are disregarded by using the "one size fits all". Another problem is ignoring the users' progresses and changes during the learning process, which negatively influences the efficiency of the E-learning systems [KS05]. All mentioned problems cause users' abstention from using the system before completing a learning process. This incident is called dropout [MLO$^+$16], and its high rate indicates users' dissatisfaction or mismatch with the learning process and method [KS05].

Personalized learning is proposed as an alternative to the "one size fits all", in order to cover the mentioned problems [VA13]. It refers to approaches that generate learning paths considering the individual differences in learning preferences, goals, abilities, knowledge background, etc [DHC17]. Since the late 1960s, researchers have attempted to address the personalization, using different parameters and approaches, but they faced different problems and challenges. In this section, a systematic and comprehensive research on learning path personalization methods is presented. We expect that after reading this chapter the reader will have a fairly broad background on the recent personalization studies (main focus is on the studies after 2010), and is able to understand:

- The key concepts in E-learning, such as learning object, learning path, etc.
- The main parameters to personalize the learning paths, such as users' learning style, competency level.
- The methods and algorithms that are proposed to personalize the learning paths.
- The approaches and techniques to evaluate these methods.
- The most significant challenges of these methods.

In addition to this chapter, Appendix A provides more information about the main studies that are explained in this chapter. The remainder of this chapter is structured as follows. In section 2.1, we present the terminology that is commonly used in the E-learning area. Section 2.2 highlights the parameters that are applied for personalizing the learning paths. This section is followed by section 2.3, which covers the main personalization methods (Course Generation (CG), Sequential Pattern Recognition (SPR), and Course Sequence (CS)), as well as their advantages and disadvantages. Section 2.4 describes the methods that are used to evaluate the learning path personalization methods. In section 2.5, we present the main challenges that the personalization methods are facing.

2.1 Terminology

Currently, a large variety of terms is used in the literature on E-learning systems. We start by providing a set of operational definitions on some of the main terms. For this purpose, we use a modular content hierarchy, which is defined by the standard Autodesk structure [Hod06, DH03]. In this hierarchy, the contents are divided into five abstraction levels, but we only describe three of them: learning object (LO), Lesson, and Course. The two other levels (i.e., Raw content, Information object) are disregarded since they have never been mentioned in any path personalization study. Furthermore, reviewing the literature enabled us to add one more level, called Topic, to the modular content hierarchy (figure 2.1).

1. **Course**: In the course level, which is the topmost level, the collections are gathered from the topic level (regarding a large objective) in order to build a thematic course. A course might be mentioned as a "subject" in some studies. The courses are often represented as oriented graphs. In these graphs, vertices indicate the LO or topics (depending on the abstraction level), and directed edges represent the prerequisite relations among the vertices [NMJPL17].

2. **Topic**: A course is composed of a few learning units called topics. Each topic covers a unique concept. For example, in the *C* programming language course the topics are arrays, data types, pointers, functions, loops, etc. The topics might be referred as "chapters" or "learning units".

3. **Lesson**: Each course needs several lessons in order to be learnt, and each lesson can cover one or more topics of a course. For instance, in the *C* programming language course, the loop topic needs a lesson, while several topics, such as data types and arrays, require one lesson.

4. **Learning object**: LO are the small units of learning content that are reusable and constructed regarding a certain learning objective [BDL14, DG13]. A LO might appear in different form, such as a text file, a power point, an audio, a video, etc. In some studies, LO are referred as "learning materials" or "knowledge units".

Any sequence of the mentioned contents (LO, topics, etc.) that satisfies their (LO, topics, etc.) prerequisites, while guiding the users in order to accomplish the learning goals, is called a **learning path** [MZB$^+$16, AK16]. Generating a path that satisfies the preferences and requirements of a user is the main goal of path personalization methods. For this purpose, the **personalization parameters** are applied to determine the users' characteristics and needs. These parameters explain the users' requirements and their divergent features, such as learning styles, knowledge background, etc., and are applied to deliver personalized learning scenarios [EAJ$^+$10]. **Learning scenarios** help not only focus on generating a path to keep the users motivated and engaged with the learning process, but also providing them with the best possible educational materials that effectively improve their knowledge. The combination of the personalization parameters to personalize the learning scenarios is called **personalization strategy** [EAJ$^+$10].

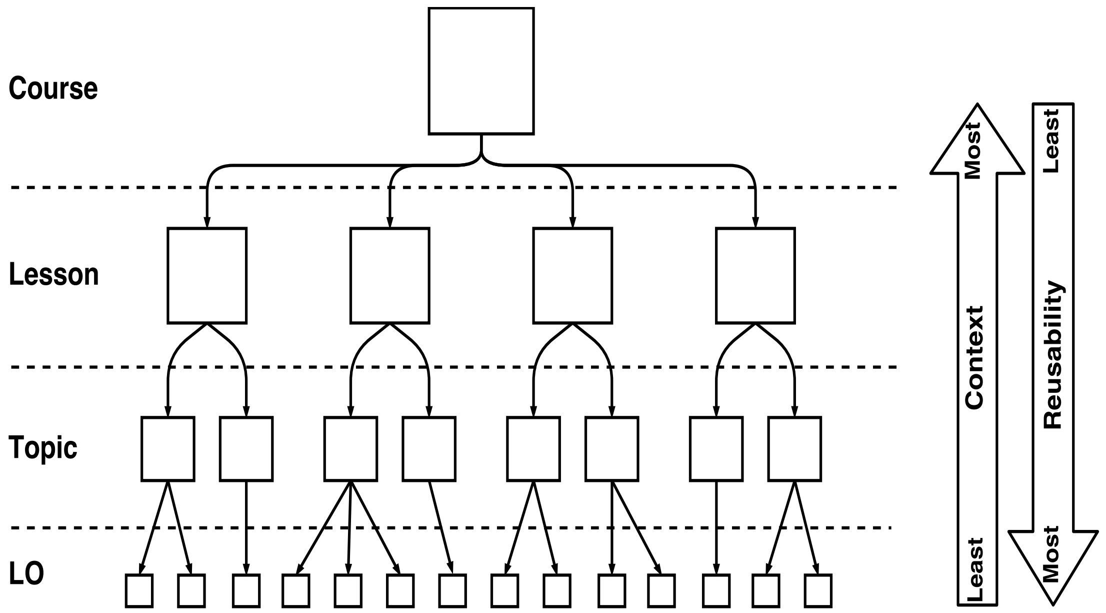


Figure 2.1: Content hierarchy.

2.2 Personalization parameters

As mentioned before, personalization parameters are critical for providing essential information to personalize the learning paths. These parameters describe various characteristics and requirements of the users, such as users' knowledge background, their goals and learning styles, etc. [EAJ+10, ZTPS07]. In this section, we detail the personalization parameters that are mainly used in path personalization methods.

Several researchers have considered different personalization parameters to personalize the learning paths, but all can be classified into three main classes (Figure 2.2). Any personalization parameter responds to one of these questions (1) why to learn?, (2) what to learn?, or (3) how to learn?. The parameters that respond to "why to learn" question, consider the learning goal and the motivation of the users, while those that concern "what to learn", enable personalizing by taking into account the users' knowledge background and their competency level. The ones that answer to "how to learn?", consider the users' preferences (e.g. learning style) [EAJ+15, Jin11]. Although researchers, such as [EAJ+10, AJ09], have considered different personalization parameters, according to our survey the most significant ones are as follows:

1. **Users' time limitations**: This refers to a user's available time [LPK16, NMJPL17]. In existing path personalization methods, if users intend to learn a path, they are required to spend a specified amount of time, which is often fixed and given by the method. Due to various reasons, a user might not be able to allocate enough time

to follow an entire path. Some of the main reasons are: user's lack of time because of multitasking, mismanaging time, etc. Hence, the information that this parameter provides is used to generate a path that a user is able to learn properly in his/her available time.

2. **Users' mastery learning**: Mastery learning, which is a stringent form of competency-based education, indicates how much the users have mastered the knowledge and skills (competencies) required for a particular course or task [MIBW14]. This is a dynamic parameter that might change during the learning process.

3. **Users' learning style**: This is an important parameter that indicates how a user learns and likes to learn [DG13]. There are several well-known learning style theories and indexes that are used by researchers, such as La Garanderie, Honey-Mumford, Kolb, and Felder and Silverman, which are described in [EAJ+10, KMVIB11]. According to our survey, the Felder and Silverman is the most frequent used index in the path personalization area. This index assesses variations in individual learning style preferences across four dimensions: Active/Reflective users, Sensing/Intuitive users, Visual/Verbal users, and Sequential /Global users.

4. **Users' knowledge background**: It considers the knowledge that the users obtained before receiving the recommendations. This knowledge has several benefits such as easing the learning process, improving reading comprehension, etc. [AK16, GMS12]. It can be divided in two different types:

   (a) Objective pre-knowledge level : Objective data such as user's grades on a past course, or pre-test scores on a course.

   (b) Subjective pre-knowledge level : Users specify their pre-knowledge levels explicitly with respect to their own understanding [XZW+17, FXP+10].

5. **Users' goal**: Learning goals are applied to design and plan the learning process, and to arrange the LO in the form of paths that satisfy the users' goals. Depending on the users, the learning goals might be different. Goals can be deadline-driven, when a user intends to complete a learning process by a given time [LPK16]. They can be score-driven, when a user aims to maximize his score [NMJPL17]. Learning rewards [DLK11], users' competency [BDL14, Chi10], and length of the paths (i.e., shortest path) [BDL14] are other types of goals that have already been considered by the researchers.

It should be clear that some of the personalization parameters are dynamic (e.g. learning style, mastery learning) and their values might change during the learning process. Furthermore, some of the parameters (e.g. knowledge level) might not be identified accurately in advance, but only during the users' interactions with the system. Therefore, a user profile,

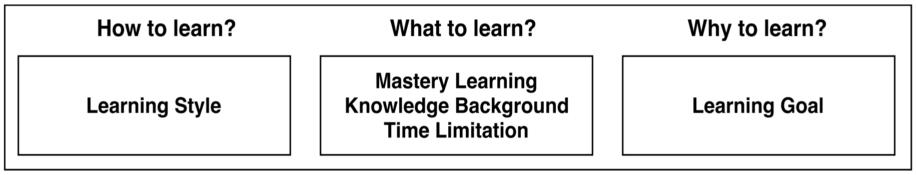


Figure 2.2: Personalization parameters classification.

which is modeled based on the personalization parameters, needs to be contemplated and updated regularly [DG13, IALS12].

Finally, the personalization parameters that are used by the researchers (main focus is on the studies after 2010) are summarized in table 2.1. As it is shown in this table, Mastery level is the most frequently used parameter, while time and knowledge background are the ones that are used the least (less than 30 % of the studies used these two parameters). In this table, the learning goal was omitted since each paper has its own learning goals, and only the type of goal is different. In complement to table 2.1, figure 2.3 presents the proportion of personalization parameters that are used by researchers per year.

Table 2.1: Personalization parameters that are used in the studies. In this table, column "Type" indicates the type of personalization methods (CG, SPR, CS) that are used in the studies.

Ref.	Type	Time	Mastery	Style	Know. Back	Ref.	Type	Time	Mastery	Style	Know. Back
[BDL14]	CG		✓			[FXP+10]	CG	✓		✓	✓
[LPK16]	CG	✓				[YLH10]	CG			✓	
[AK16]	CG		✓	✓	✓	[CLM10]	CG	✓	✓		
[DBL13]	CG		✓			[GMS12]	CG			✓	✓
[GFM+13]	CG	✓		✓		[SRR12]	CG		✓		
[YLL14]	CG		✓			[EAJ+10]	CG		✓	✓	✓
[DG13]	CG		✓	✓		[DLK11]	CG		✓	✓	
[YLL10]	CG		✓			[KMVIB11]	SPR		✓	✓	
[XXVDS16]	CG	✓	✓		✓	[VMIB13]	SPR		✓	✓	✓
[XZW+17]	CG	✓			✓	[FVFNN10]	SPR		✓		
[JBH+10]	CG		✓			[YJT13]	CS		✓	✓	
[SP15]	CG		✓			[LCCT12]	CS	✓	✓		
[YHY13]	CG			✓		[GO13]	CS				✓
[Chi10]	CG		✓			[UM10]	CS		✓		
[BDCS10]	CG			✓		[GK+16]	CS		✓		
[BBR13]	CG	✓				[CDSG14]	CS	✓	✓	✓	
[NMJPL17]	CG	✓			✓	[SÖY13]	CS		✓		
[AJ09]	CG	✓	✓	✓	✓	[YLL12]	CS		✓		

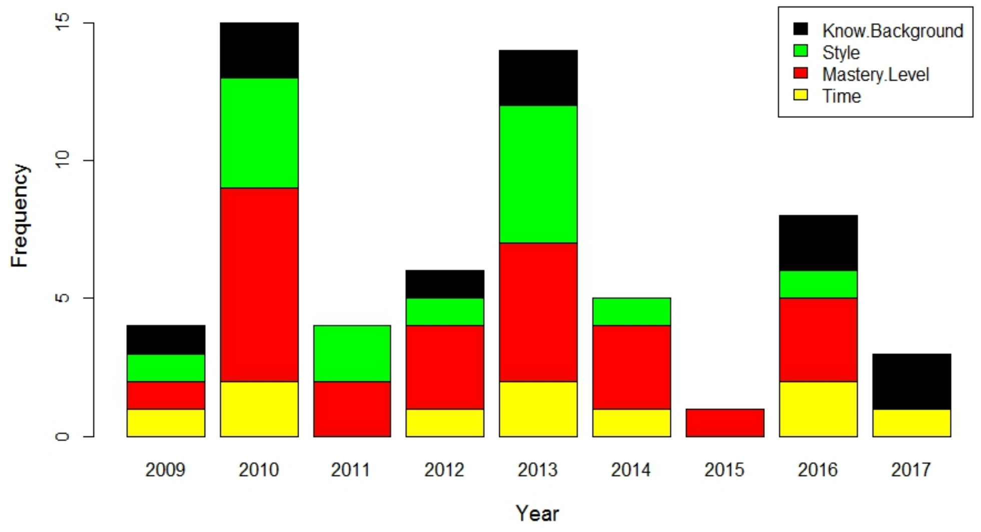


Figure 2.3: Proportion of personalization parameters that are used by publication year.

2.3 Personalization methods

Automatic generation and personalization of a learning path, based on a user's learning goals and preferences, is the main task of path personalization methods. Given a set of LO (lessons or courses) and personalization parameters, researchers have proposed various approaches that automatically generate a personalized learning path for a user. The quality of these methods' outcomes depends highly on the quality of the path personalizer method per se, on the LO, and on the parameters that are used for personalization. These personalization methods automatically retrieve the LO (lessons or courses) from a repository and assemble them in the form of learning paths. Selecting and assembling LO needs to support the users' goal. Such goals require a concise and clear definition and representation.

Since the late 1960s and early 1970s, various learning path personalization methods have been proposed, using different sets of goals, parameters, techniques, and algorithms. According to [NMJPL17, NJL15a], path personalization methods can be categorized into two main classes: (1) course generation, and (2) course sequence. In this section, we describe recent learning path personalization methods in detail, as well as their advantages and disadvantages.

2.3.1 Course Generation (CG)

In the Course Generation methods (CG), after determining a user's characteristics and requirements, the entire learning path is generated and recommended to him/her in a single recommendation [BDL14, BDCS10, CLM10]. Some researchers focused on the course generation to facilitate a group of users rather than a single user [KEI14, XZW^{+}17, FXP^{+}10]. To this end, Kardan et al. presented a method called ACO-Map, which generates paths in two stages [KEI14]. In the first stage, K-means algorithm [Jai10] is applied to divide users into groups based on the results of a pre-test. In the second stage, the ant colony optimization method [DS10] is used to generate a path for each group. Groupized learning path discovering (GLPD), which was introduced by Feng et al. in 2010, is another CG group recommender method. In this method, a topic graph is initially generated, and pre-knowledge and preferences of the users are collected. The GLDP then estimates the temporal boundaries for a group of users (max and min time to learn a path). Finally, regarding the estimated temporal boundaries for a group and the required time to learn a path, a corresponding strategy is selected to discover a path [FXP^{+}10].

Other methods, instead of generating a path for a group of users, concentrate on personalizing a path for a single user. For instance, Belacel et al. proposed a CG method based on graph theory. In their graph, the LO are vertices and the edges present the dependency relations among vertices (prerequisite). Their method starts with reducing the solution space by obtaining an induced sub-graph of the learning graph (eliminating LO that are irrelevant to obtain the goal). It then utilizes the branch-and-bound algorithm in the sub-graph to find the shortest path by minimizing the number of required competencies [BDL14].

Another CG method that is based on graph theory is called CourseNavigator [LPK16]. This method is based on a graph search algorithm. It generates all paths given a set of users' inputs. The users' inputs are constraints (e.g., maximum number of courses to take per semester, courses to avoid), learning goals (e.g., graduation semester, a set of desired courses), users' enrollment status (e.g. starting point), and their preferred ranking for the output paths (e.g., shortest, most reliable, etc.). Given the set of inputs, this method is able to generate three types of learning paths: (a) deadline-driven paths, (b) goal-driven paths and (c) ranked paths (regarding the user's ordering preferences). In the CourseNavigator, a recommended path is a sequence of semesters. In each semester a user needs to take a number of courses. In this method, the researchers do not estimate how much time a single course might take for a user, and therefore, the learning time of a course is the same for all users. Similar to the CourseNavigator method, Xu et al. developed an automated method to generate a sequence of courses for a user [XXVDS16]. The main goal of this method is to minimize the graduation time of the users while maximizing their overall GPA. In this method, a forward-search is first executed, from quarter 1 (each academic year consists of four quarters) to quarter T, to identify all possible course states that can be in a path. Then, a backward-induction is performed, from quarter T to 1, to compute the optimal set of courses that should be considered in each possible course state. Finally, an algorithm, which was developed using multi-armed bandits [GGW11], recommends a course sequence that reduces the graduation time while increasing the overall GPA of a target user. However, in both CourseNavigator and the method that is proposed by Xu et al., the researchers do not take into account how much time a single course takes for a user in a semester (i.e., learning time of a course is a fixed value for all users).

Educational Concept Map (ECM) [AK15] is also employed successfully in CG methods. As an example of an ECM based method, we can refer to Adorni and Koceva's, which is presented in [AK16]. In their method, a user initially determines his/her knowledge background by selecting a set of topics from ECM, which trims the known topics from the map. The output of the trimming process is checked by an expert, and finally, after the user chooses the initial and target topics, the paths are generated using ENCODE [Koc16], which executes an algorithm to linearize the map.

The methods that we have detailed so far, are mainly focused on generating learning paths, regardless of the user's time restriction to learn them. There are some methods that take into account this limitation, such as [GFM+13, BBR13, NMJPL17]. For example, in this thesis we introduced a method called RUTICO, which is an example of Long Term goal Recommender Systems (LTRS) [NJL15a]. The main goal of RUTICO is to generate a path that maximizes a user's score under a time restriction. In this method, after locating a user in the course graph, a Depth First Search (DFS) algorithm is applied to find all possible paths for a user, given a time restriction. RUTICO also estimates learning time and score for the generated paths, and finally, recommends the path with the maximum score that satisfies the user's time restriction. Basu et al. also developed a CG system to recommend a path to a user that satisfies his/her time restriction [BBR13]. Their system consists of two major components: Learning Path Indicator (LPI) generating component and Learning Path Generating (LPG) component. In this system, a function is initially defined based on three system parameters : (1) number of post-requisite of a subject (course), (2) learning time of a subject (course), and (3) number of credit for a subject (course). Then, a fitness function is defined regarding personal restrictions and preferences of a user, such as affordable time. Ultimately, a LPI is generated using the estimated values from the fitness function and the system parameters' function. The generated LPI is passed to the LPG component to formulate a path for a user. In the LPG, to generate a path, each subject (course) is chosen by applying a forward greedy algorithm on the LPI values.

In addition to the mentioned studies, there are other CG methods that have been proposed using different algorithms and techniques : a decision tree classifier [LYHC13], a markov decision process [DLK11], greedy algorithms [DBL13, BBR13], a Hierarchical Task Network (HTN) [GFM+13], a Case-Based Reasoning/Planning [DG13, GMS12], genetic algorithms [BDCS10, TLF12], a Planning Domain Definition Language (PDDL) [GMS12, GFM+13], a Bayesian network [SRR12], etc. In table 2.2, we have summarized the techniques and algorithms that have been used by researchers.

Although CG methods are widely used by researchers to generate learning paths, they have several drawbacks. One of the main disadvantages is ignoring a user performance and the changes that occur during the learning process. Thus, users are at risk of wasting time, by receiving a wrong path or a path that they are not able to follow. Also, these methods often become slow when they receive a large amount of data (e.g. large number of LO and users). Therefore, they might not be able to respond quickly enough to keep the users engaged.

2.3.1.1 Sequential Pattern Recognition (SPR)

Sequential Pattern Recognition methods (SPR), which are a subset of CG methods, are less used than the other learning path personalization methods. In these methods, sequential pattern mining approaches [AS95] are mainly applied to discover a learning path for a user from the transactions of similar users. Users are similar if they have similar initial states, preferences, goals, etc. In comparison with the CG methods that are able to generate paths even without users' transactions data, SPR methods require transactions data for the path generation.

There are a few studies that used SPR methods, such as [KMVIB11, VMIB13, FVFNN10]. As an example, we refer to the Protus method that was introduced by Vesin et al. in 2013 [VMIB13]. In Protus, users are clustered regarding their common attributes (e.g. age, class, etc.) and preferences. Then, the method finds a cluster for a target user and considers the sequence of lessons that each member in that cluster selected (lessons are rated by users and based on sequences which successfully guided the users). Finally, Protus uses association rule mining to find all successful sequences of the target cluster, and recommends a sequence based on users' ratings. As another example, we explain the Fournier-Viger et al. proposal [FVFNN10] that is illustrated in the context of CanadarmTutor [KNB05]. CanadarmTutor is an Intelligent Tutoring Systems (ITS) [PR13] to learn how to control a robotic arm. This system initially (in the observing phase) records the solutions of the users to move the arm from an initial configuration to a goal configuration. In the next phase (learning phase), an algorithm [FVNN08] is applied to find all sequences with a support higher or equal to a minimal support (support is the proportion of transaction in the data in which a sequence X appears). In the final phase (application phase), the system provides assistance to a target user by using the knowledge that was gained in the second phase. The assistance is provided by recognizing a user's plan.

In SPR methods, researchers apply sequential pattern mining methods and algorithms, such as *Apriori* [AS95], to mine patterns from transactions data, but they often face two main problems. First, current pattern recognition methods such as *Apriori* might require a lot of memory, and second, they find frequent patterns and rare cases are ignored.

2.3.2 Course Sequence (CS)

Unlike CG methods, Course Sequence approaches (CS) recommend a path LO by LO, as a user progresses in the learning path [KS05, NMJPL17, NJL15a]. Different CS approaches have been proposed: using an Association Link Network (ALN) [YLL12], Evolutionary Algorithms (EAs) [LCCT12, GK$^+$16], Item Response Theory (IRT) [YJT13, SÖY13], Bayes theorem [XWCH12], etc. In 2016, Govindarajan et al. applied an evolutionary algorithm (Parallel Particle Swarm Optimization) to predict a dynamic path for users [GK$^+$16]. Their method clusters users into four groups according to their proficiency level. The proficiency comprises both measuring a target outcome achievement, and the competence and meta-competence changes during the learning process for each defined learning outcome. Then, the method predicts a dynamic path based on the clustered information.

Similarly, two evolutionary algorithms were used by Li et al. to develop a CS method. In their method, learning concepts are composed in the form of a sequence, which is the base for presenting a sequence of LO. Next, the collaborative voting method is applied to automatically adjust the difficulty level of LO according to the users' feedback (step 2). In step 3, Maximum Likelihood Estimation (MLE) is used to analyze the users' ability and goals. Finally, a Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are applied to generate a path using the results obtained in step 3. Once a user completes a LO, the feedback information will be used to adjust the difficulty level of LO in step 2, and update the user's ability and goals in step 3 [LCCT12]. The users' ability is also noted by [YJT13] to personalize learning paths. In this study, an adaptive e-learning system is proposed using an ontology-based knowledge modeling. This system receives the user's ability, knowledge background, learning style and preferences as inputs and recommends a path. It then analysis the user's responses using the Item Response Theory (IRT) [AY14] and updates the user's ability. The updated data is used to modify the path by recommending the LO that matches the user's ability.

Another system that uses the Item Response Theory was proposed by Salahli et al. [SÖY13]. Their system takes a few steps for path personalization. Initially, the topics are identified, their relations and difficulties are determined, and the users' profiles are also generated. Item Response Theory (IRT) [AY14] is then applied to estimate the understanding degrees of the topics for each knowledge level. In the next step, when a target user starts using the system, his/her knowledge level and the difficulty of a selected topic are retrieved to estimate his/her understanding degree. Then, the LO are recommended to the user according to his/her understanding degree. After completing a LO, the system checks if the user understood the LO. If the user was able to understand the LO, the user's knowledge on the topic is tested, and his/her knowledge level is re-estimated. Accordingly, the understanding level of the user is re-estimated with the Law of Total Probability (LTP) [Khr10]. If the understanding degree is low, the system recommends the LO to improve the user's knowledge on the prior topics.

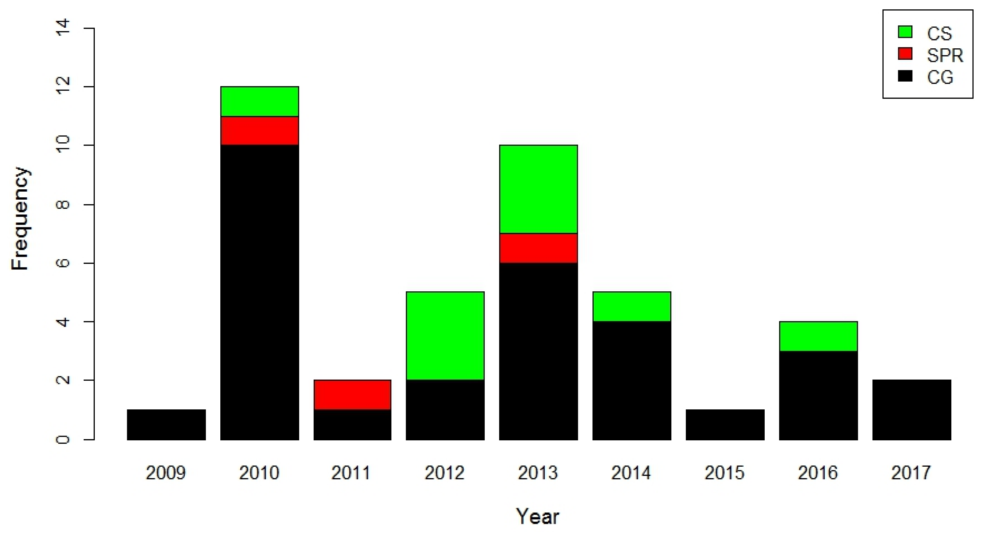


Figure 2.4: Proportion of personalization methods per year.

Although CS methods take into account the users' progresses and changes during the learning process, which is one of the main problems with CG methods, they still have several problems that need to be considered. The first one is estimating a personalized time period to evaluate the user's knowledge and updating his/her profile. The current studies consider a fixed amount of time for all users to evaluate and update their profiles. Evaluating a user and updating his/her profile is time consuming and might be unnecessary, while postponing the tasks might result in recommending improper LO (mismatching the user's ability), which causes misleading the user and wasting his/her time. In addition, identifying the critical information that needs to be updated is important because the personalization parameters may have different weights for different users.

After describing different types of learning path personalization methods (CG, SPR and CS), we present in table 2.2 the main algorithms/methods/techniques that were used by researchers mainly after 2010. In the same table, we describe the recommendation strategies that were used in these studies, as well as their types. In addition, figure 2.4 presents the proportion of personalization methods used by researchers per year.

2.4 Evaluation methods

Evaluation is always one of the main challenging phases in the learning path personalization methods. Besides offline evaluation, these methods must be evaluated with real users in a live environment. Researchers evaluated these methods with Information Retrieval measures

Table 2.2: Summarizing the algorithms/methods/techniques in the literature (mainly since 2010), as well as their types and their recommendation strategies.

Ref.	**Type**	**Main Techniques/Algorithms/Methods**	**Recommendation strategy**
[BDL14]	CG	Branch and bound algorithm	Shortest path (number of competency)
[LPK16]	CG	Graph search algorithm	Deadline driven, Goal driven, Ranked driven
[AK16]	CG	Educational Concept Map (ECM), ENCODE (linearize the map)	Minimizing the number of LO
[DBL13]	CG	Greedy algorithm	Shortest path (number of competency)
[GFM$^+$13]	CG	PDDL, HTN, Non-HTN	Maximizes the total reward without exceeding a user's time
[YLL14]	CG	Bloom's taxonomy	It is a path generation method that formulates learning activities and their assessment criteria using the Bloom's taxonomy.
[HKYC10]	CG	RFID,repertory grid-oriented technique,Heuristic Algorithm	Minimizing the relevance loss between consecutive LO along a path
[TLF14]	CG	Explicit Semantic Analysis, Evolutionary algorithm+hill climbing, Clustering	Minimizing the sum of the violated distances regarding the reference paths
[KEI14]	CG	ant-colony (ACO-Map), Ausubel Meaningful Learning Theory, Clustering	Shortest path
[DG13]	CG	Case Based Reasoning (CBR), Colored Petri Nets, Classification	Finding similar solution regarding learning style, goal and performance
[YLL10]	CG	Bloom's taxonomy	It is a path generation method that formulates learning activities and their assessment criteria using the Bloom's taxonomy.
[XXVDS16]	CG	Clustering, Multi-armed bandits	Minimizing the graduation time while maximizing the overall GPA
[LYHC13]	CG	Decision tree classifier, feature weighting, feature selection	Maximizing users' creativity
[XZW$^+$17]	CG	Topic graph extraction, group profiling, estimating time boundaries	Minimizing learning time and Maximizing the learning enjoyment
[JBH$^+$10]	CG	Search engine	Finding a path that meets a user criteria (e.g. start point)
[SP15]	CG	Ant Colony Optimization algorithm (ACO)	Optimizing path regarding users' ability, goals and behavior
[YHY13]	CG	Learning and Cognitive styles	Paths that match the cognitive and learning styles of users
[Chi10]	CG	Domain ontology, Problem-solving ontology, Semantic rules	Generating path regarding a given criteria (competencies)
[BDCS10]	CG	Genetic algorithm	Maximizing fitness value (consists of 3 fitness values)
[BBR13]	CG	Greedy algorithm	Path satisfies users limitation (time) and preferences.
[FXP$^+$10]	CG	Topic graph extraction, group profiling, estimating time boundaries	Minimizing learning time while considering users preferences
[YLH10]	CG	Felder learning style, Broad first searching (BFS), 3-dimension semantic map	Path satisfies users learning style.
[CLM10]	CG	Peer-to-peer (P2P) networks	Generating paths that satisfy a user's time and difficulty level
[GMS12]	CG	PDDL, Case-Based Planning (CBP)	Minimizing the number of changes regarding a referenced path while maximizing the total reward
[NMJPL17]	CG	Depth first search (DFS)	Maximizing users score under his/her time restriction
[AJ09]	CG	Weighted Graph, Eliminating and Optimized Selection (EOS)	Shortest path on a weighted graph
[TLF12]	CG	Explicit semantic analysis, Genetic algorithm, Clustering(K-means)	Minimizing the number of precedence rules violated by the path
[DLK11]	CG	Markov Decision Process	Maximizing reward while minimizing the number of LO
[SRR12]	CG	Bayesian network, Clustering	Recommending activity with the highest probability
[EAJ$^+$10]	CG	Personalization strategy	Matching user' characteristics with the LO features
[FVFNN10]	SPR	Sequential pattern mining (invented algorithm)	Successful paths that satisafy the initial and end point of a user
[KMVIB11]	SPR	Clustering, AprioriAll	Maximizing the sequences' rate while reducing the info redundancy
[VMIB13]	SPR	Clustering, Collaborative filtering, Association rule mining	Maximizing success rate of sequences while reducing the info redundancy
[YJT13]	CS	Item Response Theory (IRT)	Recommending LO that fits the user's ability
[LCCT12]	CS	Collaborative Voting, Maximum Likelihood Estimation, Evolutionary algorithms (GA,PSO)	Matching a user's ability while considering his/her time limitation
[GO13]	CS	PDDL, Constraint Satisfaction Problem (CSP)-based scheduling	Minimizing the length of a path while maximizing the learning reward
[UM10]	CS	Learning scenario selection, HTN	Matching preferred learning scenario of a user and his/her goal
[GK$^+$16]	CS	Evolutionary algorithm (PPSO), Clustering, Bloom's taxonomy	Fitting user's need and proficiency level.
[XWCH12]	CS	Bayes formula, Pearson Correlation Coefficient (KNN), feature weighting	Recommending a LO with the highest probability
[CDSG14]	CS	Functions to estimate the closeness of each LO to a user's profile, Ontology	Matching users' characteristics with the relative parameters of LO
[SÖY13]	CS	Item Response Theory (IRT), Law of Total Probability (LPT)	Matching the understanding degree of users with the difficulty level of LO
[YLL12]	CS	Association Link Network (ALN), TFIDF Direct Document Frequency of Domain (TDDF)	Matching the users knowledge level with the complexity of LO

(*Precision*, *Recall*, etc.) [GK+16], Machine Learning measures (*RSME*, *MAE*, etc.) [NMJPL17, KMVIB11] and Decision Support System (DSS) approaches (e.g. measuring user satisfaction, user loyalty, etc.) [EAJ+10]. Although path personalization methods are often evaluated by IR and ML measures, it is important to monitor the users' transactions and measure their satisfaction. This is important because if users are unsatisfied with the recommendations, they might drop out [MLO+16, LC11], and subsequently the learning goal might not be accomplished.

In this section, we describe the experiments that are applied to evaluate learning path personalization methods and categorize them into four main classes: offline, system performance, online and user study experiments. We start with offline experiments, which do not require live users interactions and are easier to perform. We then describe the system performance experiments, which aim at attaining the highest performance for the method. System performance experiments are followed by online evaluation, which are the most reliable experiments when the users of the path personalization method are not informed about the evaluation. Finally, we describe the user study evaluation, where a path personalization method is tested by users in a controlled environment. The users then report according to their experience.

2.4.1 Offline evaluation

Offline experiments simulate the users' behavior with the path personalization method. The main assumption of these experiments is that the users have similar behavior during the data collection phase and when they are using the method. Being standalone is one of the main advantages of these experiments, which do not require live users' interactions. In addition, this advantage allows us to compare the performance of various algorithms and methods with a small cost [SG11].

Although measures such as *Precision*, *Recall*, *MeanAbsoluteError* (MAE) are required to assess the path personalization methods, they are only used by a few studies. For example, Klašnja-Milićević et al. [KMVIB11] applied the Mean Absolute Error (MAE) [WM05] to measure the deviation of recommendations from their true user-specified values. Similarly, the MAE is used by Nabizadeh et al. [NMJPL17] to evaluate the quality of methods that are proposed to estimate the learning time and score for generated paths.

Despite being easy to use and having a low implementation cost, this type of experiments present some reliability risks, since the users' behaviors might change during the learning process and these experiments do not take these changes into account.

2.4.2 System performance

The main goal of the system performance experiments is to measure the performance of the system. These experiments are used in response to questions such as "How fast is the system?" (response time), "Does the system work with large datasets?" (scalability), or some other questions that address the performance quality of the system. They are often performed after offline experiments, since we initially need to be sure that the system generates acceptable results, and then we attempt to evaluate and improve the performance of the system.

In the learning path personalization methods, some studies such as [LPK16, CLM10, LCCT12, JBH$^+$10] employed system performance evaluations. For instance, in [DBL13], the authors measured the average calculation time for learning paths. Similarly, Li et al. assessed their method by analyzing the execution time [LCCT12]. In this study, the execution time of two evolutionary algorithms (GA [KHUG10] and PSO [Ken11]) were compared regarding different numbers of LO. Another example is measuring the stability, which was performed by Garrido et al. in 2012 [GMS12]. In their study, the path stability was evaluated regarding the number of changes (expressed in terms of number of LO) between the generated path and the referenced one. They also evaluated the system scalability by considering the time that a CPU takes to generate a path.

In this section, we attempted to mention various experiments that were performed on the system performance. A glance at the literature indicates that researchers often used these experiments to analyze the required time for generating paths. Despite having a low evaluation cost when compared with the online evaluation, system performance experiments present some difficulties. First, it is not an easy task to determine the critical parts of the method that need to be improved, and second, evaluating the detected parts to achieve optimum performance is time consuming and not a trivial task.

2.4.3 Online evaluation

The main objective of learning path personalization methods is to enhance the users' knowledge and skills. In order to evaluate how much a method was successful in accomplishing this goal, it is necessary to measure the users' improvement when they are using the method. For this purpose, online experiments can be used. These experiments provide more reliable results than offline experiments since, in these experiments, the method is used by real users performing real tasks.

Online experiments were used in several studies, such as [UM10, VMIB13, YHY13]. In [KMVIB11], the successful completion of a course was used to assess the method. In this study, users were divided into two groups when using the method (control and experimental groups). The results of the experiment show that the users in the experimental group were able to complete a course in less time than the users in the control group. Xu et al. compared the performance of the *C* programs that were written by the users in two different groups (control and experimental groups) [XWCH12]. In the same study, the researchers also compared the grades' differences in two groups. Similarly, [CDSG14] and [FXP$^+$10] compared the grades of the users in two groups to evaluate their methods.

Although online experiments provide the most reliable results about the method and assisting to prevent users' dissatisfaction, these evaluations are time consuming and have a high performance cost.

2.4.4 User study

Another type of evaluation is the user study, which can be used as a complement to online evaluation. In this type of evaluation, a controlled experiment is performed by asking a group of users to perform a set of predefined tasks. This evaluation enables us to analyze the users' interactions with a method. It also allows us to collect both quantitative and qualitative information about the method. In order to collect the qualitative information, we might use the questionnaires and ask the users to answer some questions like "Did you have enough time to follow the path?" or "Do you think the presented task was easy to complete?", etc. These questions can be asked before, during and after completing a task. The quantitative information, such as time to perform a task, can be collected based on the quantities achieved for a task [RRS11, SG11].

As mentioned above, distributing questionnaires among the users in order to report on their experiences, is one of the main methods to conduct the user study. It facilitates collecting information about the users' experiences with the method. For example, in [LCCT12], the researchers designed a questionnaire that is composed of five questions (questions are in a five-point scale) to evaluate the users' satisfaction. Their evaluation was conducted in two stages. In the first stage, the feedback information from 41 users was collected to adjust the difficulty level of LO. In the second stage, after adjusting the difficulty level of LO, the feedback from 62 users, who did not participate in the first stage, was collected. In 2011, Klašnja-Milićević et al. evaluated the users' satisfaction regarding four main features of their system (speed, accuracy, adaptive, convenience) by means of a non-mandatory questionnaire [KMVIB11].

Although user studies might provide information about aspects that are hard to evaluate, such as users' satisfaction, these experiments have several drawbacks. First, user studies are costly to conduct both in terms of time and money. Second, due to the difficulty and high cost, normally user studies are conducted on a small portion of the users and tasks. Therefore, the results of user studies cannot be trustable and generalized for all the users. To overcome this problem, the population size of the experiment should be large enough to represent the users of the method in a real environment (i.e., represent a real situation). Selecting such a population to perform the experiments is not a trivial task.

2.4.5 Evaluation methods summary

After describing the evaluation methods, we have summarized the type of evaluations that were used by researchers. This information is presented in table 2.3. As shown in the table, the system performance experiments were frequently used by researchers, while offline experiments were only used a few times. Table 2.3 also shows that, in nine studies, the researchers did not assess their methods with any type of evaluation. Figure 2.5 shows the proportion of evaluation methods that were used by researchers per year.

Table 2.3: The evaluation methods that were used in studies. Column "Type" indicates the type of personalization methods (CG, SPR, CS) that were used in the studies.

Ref.	Type	Offline	Performance	Online	User study	Ref.	Type	Offline	Performance	Online	User study
[BDL14]	CG					[YLH10]	CG				
[LPK16]	CG		✓			[CLM10]	CG		✓		
[AK16]	CG					[GMS12]	CG		✓		✓
[DBL13]	CG		✓			[NMJPL17]	CG	✓	✓		
[GFM$^+$13]	CG		✓		✓	[AJ09]	CG		✓		
[YLL14]	CG		✓		✓	[TLF12]	CG		✓		
[HKYC10]	CG		✓	✓	✓	[DLK11]	CG				
[TLF14]	CG		✓			[SRR12]	CG				✓
[KEI14]	CG					[EAJ$^+$10]	CG				✓
[DG13]	CG		✓	✓		[FVFNN10]*	SPR				✓
[YLL10]	CG				✓	[KMVIB11]	SPR	✓	✓	✓	
[XXVDS16]	CG		✓	✓		[VMIB13]	SPR			✓	✓
[LYHC13]	CG			✓		[YJT13]	CS				
[XZW$^+$17]	CG			✓	✓	[LCCT12]	CS		✓		✓
[JBH$^+$10]	CG		✓		✓	[GO13]	CS		✓		✓
[SP15]	CG			✓		[UM10]	CS				✓
[YHY13]	CG			✓	✓	[GK$^+$16]	CS		✓	✓	
[Chi10]	CG					[XWCH12]	CS			✓	
[BDCS10]	CG					[CDSG14]	CS			✓	
[BBR13]	CG					[SÖY13]	CS		✓		
[FXP$^+$10]	CG			✓		[YLL12]	CS		✓	✓	

* We consider the proposal that is illustrated in the context of CanadarmTutor.

Finally, all mentioned evaluation methods are not solely enough, and cannot provide trustable results. To this end, researchers often use more than one evaluation method to assess their path personalization methods. Furthermore, the lack of a general evaluation framework that enables us to compare different path personalization methods, is another difficulty in the evaluation.

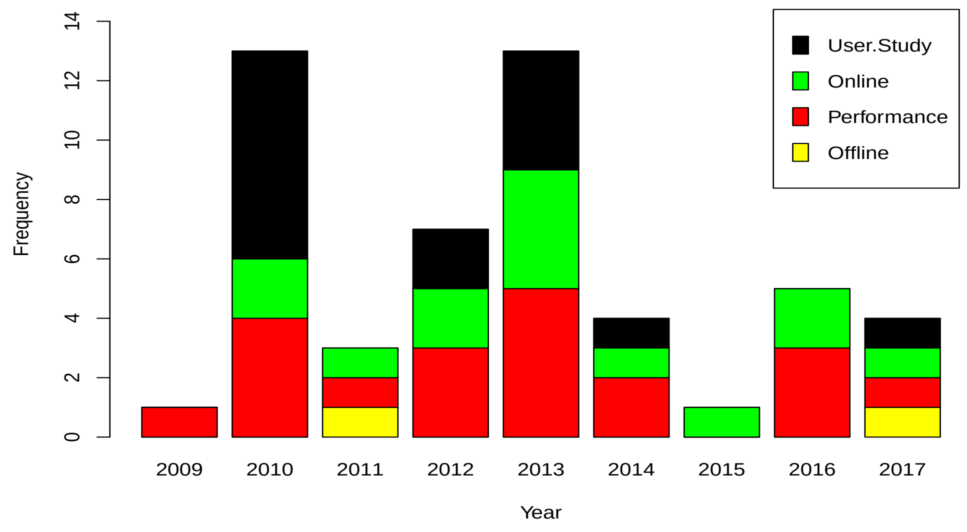


Figure 2.5: Proportion of evaluation methods per year.

2.5 Challenges

Although there are several studies that were conducted on learning path personalization methods, there are still a set of limitations and challenges with regard to these methods. Introducing these challenges can help researchers addressing current drawbacks and give rise to significant results. We previously mentioned several challenges and difficulties regarding path personalization methods, but in this section we present additional challenges, which we consider important for the development of the research on these methods.

2.5.1 Users' time restrictions

One of the main users' requirements is to learn a path that they are able to timely complete. Satisfying this requirement and recommending such paths becomes more challenging when a user is not able to devote enough time to learn a path. A user might not have enough time due to different reasons, such as mismanaging time, multitasking, etc. Therefore, we require a path personalization method that takes into account the users' time constraints.

There are a few studies that consider this limitation, such as [GFM$^+$13, LPK16, XXVDS16]. CourseNavigator [LPK16], which was proposed in 2016, is able to generate deadline driven paths. In this method, given a deadline by a user, the method generates a path that consists of a sequence of semesters, each semester presenting a number of courses that the user needs to take regardless of the time that each course might take in that semester. Similarly, Xu et al. proposed a method to generate a sequence of courses for a user given a time-to-completion [XXVDS16]. In [DLK11], the researchers introduced another system that recommends a path taking into account the time constraint of a user. In this study, the available time of a user is represented in minutes.

Despite addressing the users' time restrictions, in the aforementioned studies the learning time of a LO or a course is often assigned by an expert and is fixed for all users. Hence, estimating a learning time for a LO or a course taking into account the users' responses and abilities can help generate more efficient paths for the limited time of the users.

2.5.2 Scalability

The amount of data used as input to the learning path personalization methods is growing, as more users and LO are added. Subsequently, the size of stored users' interactions data can be large. Despite the large amount of data, the path personalization methods aim to respond quickly in order to keep users engaged. Therefore, a key challenge is designing scalable methods that can cope with the large scale datasets. Although scalability is one of the main problems in learning path personalization methods, it is addressed only by a few studies, such as [DBL13, GMS12, GFM$^+$13].

In path personalization methods, scalability is often measured by checking a method's responses and resources consumption during the scaling up task (i.e., increasing the number of LO and users) [GO13, DBL13]. For example, Garrido et al. evaluated the scalability of their method by generating paths with 1, 2, 4, 8, 16, 32 and 64 users while measuring the execution time [GFM$^+$13]. In [GMS12], the researchers assessed the scalability by estimating the time that a CPU required to obtain the paths. As another example, we can refer to [DBL13], where the researchers measured the average calculation time for learning paths given a set of LO.

Although researchers often measure the technical effects of the scalability on the methods, such as running time of the methods, it is significant to measure the side effects that the scalability imposes on the methods, such as its influence on the accuracy of the recommendations. Such analysis provides valuable information for future research direction.

2.5.3 Updating users' profiles

The users' progresses, abilities and preferences might change during the learning process. In addition, some of the users' characteristics, such as their knowledge level, cannot always be identified precisely in advance, and their actual values can be exposed during the learning process. Therefore, the users' profiles should be updatable, taking into account the users' responses and changes during the learning process. This would enable us to generate paths that fit the users' requirements.

There are methods, such as the CS methods, that update the users' profiles during the learning process (CS methods are explained in section 2.3.2). These methods apply explicit or implicit feedback to update the users' profiles. Implicit feedback is derived from the users' interactions with the path personalizer, such as the time that a user spends on a LO, or which LO are not selected by the user. Although the information that the implicit feedback provides cannot be obtained explicitly, the implicit feedback is more difficult to collect than the explicit one. Explicit feedback is obtained through the users' rates and comments. In spite of being easy to collect, it does not always represent the actual information about a user. Therefore, it is a nontrivial task to determine which type of feedback can be more useful to update the users' profiles.

Besides the ones mentioned, there are other challenges when updating the profiles of the users. Some of these challenges are represented in the questions that are listed below:

1. When does a user profile need to be updated? Should it happen after the same amount of time for all users or does it depend on the user? Determining the updating time is a challenging task since evaluating a user and updating his/her profile frequently is time consuming and might not be necessary, while delaying it might result in recommending improper LO (not fit to that user), which causes misleading the user and wasting his/her time.

2. Which information in a user's profile needs to be updated?

3. Do the updated users' characteristics have the same importance when generating recommendations? Is there a ranking (weight) among them?

4. How can we check the validity of the updated information?

2.5.4 Course graph

In current studies, a course is often designed manually, in a difficult and time consuming task. A course, which is designed by a teacher, is a static graph, not changeable, and it will be the same (regarding graph topology, weights for edges and nodes, etc.) for all users. This means we have a graph that is **teacher-centered** rather than **user-centered** [Ahm13]. The **teacher-centered** design can be problematic, since the way that the users follow the LO/lessons/courses might be different from the paths that the teacher designed. Therefore, it is of interest to design a course taking into account the collected information from the users, including the one collected from the users' interactions with the path personalizer (i.e., similar to the SPR methods where a path is generated considering the users' transactions).

2.5.5 Evaluation

Due to various reasons, the evaluation is always a challenging task in learning path personalization methods. One reason is the lack of a general evaluation framework that would allow us to compare different path personalization methods. Another reason is the unavailability of a benchmark dataset to evaluate the methods. The available datasets are often proprietary and cannot be released due to privacy concerns. Therefore, they cannot be used as benchmark datasets. The availability of such datasets would allow researchers to compare their methods accurately. A third reason is that, to have reliable evaluation results, path personalization methods need to be evaluated with real users in a live environment. Hence, conducting such an evaluation is time consuming and has a high cost of performance.

2.6 Summary

Lack of enough time for users to learn an entire learning path (course) motivates the development of methods that generate learning paths under a limited time of a user. In this thesis, we concentrate on developing a LTRS in E-learning domain, which generates paths that maximize a user's score under a given time restriction. In this chapter, we have made an introduction to learning path recommendation methods. Then, we have presented the terminology that is commonly used in the E-learning domain. In addition, we have described parameters that are used by researchers for personalizing the learning paths, such as learning style, users' knowledge background, etc. We also have provided an overview of the main personalization methods (CG, SPR, CS), and explained their advantages and disadvantages. We then have detailed evaluation methods that are used to assess the quality of learning path personalization methods. Finally, we have described challenges that the learning path personalization methods are facing. All the terms and methods introduced here are fundamental to understand the remainder of this thesis.

Chapter 3

Long Term Goal Recommenders

Various approaches are introduced to generate learning paths based on attributes that characterize learning contents and users' characteristics. A number of these approaches, which are applied to generate a learning path for a user, are described in the previous chapter. A learning path (course or curriculum sequence) includes steps for guiding a user to effectively build up skills and knowledge.

Although existing learning path personalization methods, in particular course generation and sequential pattern recognition methods, provide learning paths to support users during their learning process, most of them do not take into consideration users' feedback, changes and progresses. On the other hand, they often ignore the users available time, and mostly concentrate on what learning content needs to be delivered at each step of the path. Regarding the mentioned limitations, in this thesis we aim to generate learning paths (courses) that maximize users' scores in their available time.

This chapter describes Long Term Goal Recommender Systems (LTRS) and provides motivation for using these systems as well as proposing several use cases for LTRS. We then illustrate a domain that we have selected for implementing the LTRS (E-learning) and provide our main reasons for our selection. Next, the main challenge in this domain is detailed, which is generating recommendations that guide users to obtain a long term goal while satisfying their requirements and constraints. Finally, we divide our main problem into sub-problems and briefly explain the tasks for each of them.

3.1 LTRS - Problem Statement

Recommenders often focus on the immediate value of recommendations and are evaluated as such. This is inadequate for obtaining long term goals, either defined by users or by platform managers. For example, in case of music, current recommenders suggest a track (or a list of tracks) that a user might like. Suppose that we are managing a music company that produces music tracks in a specific genre and due to some reasons we expect to change the music genre of company to another one. As a consequence, we might lose a part of our music market. So, our company looks for solutions to retain its market and keep the same level of selling after changing its music genre. One of the solutions can be diversifying the customers' taste in a way that they also follow the new music genre of the company. Influencing the tastes of customers takes time and need to be done through time since a direct and quick taste's influencing might not be accepted by customers. This task cannot be performed by using the current music recommenders. One solution can be recommending music tracks that are close to the customers' tastes while influencing their tastes toward the target genre of music through time. As another instance, when a student needs to learn a concept, current recommender systems suggest learning materials regarding his/her level of knowledge. Assuming this student intends to learn a concept that with his/her current knowledge and background, he/she would not be able to learn it. Therefore, the main task of the recommender system would be recommending adequate learning materials successively while promoting his/her knowledge in a way that he/she can learn the target concept (long term goal).

To this end, we propose Long Term Goal Recommender Systems (LTRS) that are able to guide users toward a predefined goal while satisfying their immediate requirements and constraints. In such a proposal, users' guidance is achieved by generating a sequence of relevant recommendations through time. Figure 3.1 shows a conceptual view of the LTRS in the case of music where a customer is guided toward a long term goal (target genre) by generating a set of recommendations (seven recommendations). In this figure, we can see how the genre taste of a customer is influenced and expanded through time using a LTRS. By each recommendation, LTRS suggested a set of music to a customer that resulted in slightly broadening his/her genre of music. After generating a sequence of recommendations that drove the customer toward the target genre (highlighted in green), a customer started using the target genre.

LTRS can be applied to different domains. Tourism is one of these domains. Currently, information about travel destinations and their associated resources, such as accommodations, parks, restaurants, bars, museums or events are usually searched by tourists to plan their trips. However, the list of possibilities offered by websites (even specialized traveling websites) can be overwhelming. Visiting all options mentioned in the list might need much time and a huge amount of money while tourists often have a limited budget and time. In this situation, LTRS can be applied to assist tourists in maximizing their experience during their trips (e.g. visiting different places, eating traditional dishes) while satisfying their constraints. Also, LTRS can be used in movie domain, where companies use these systems for guiding the users from a preferred movie genre to a target genre. In this case, these systems gradually influence users' interests through time for improving the profit on selected products (e.g. new

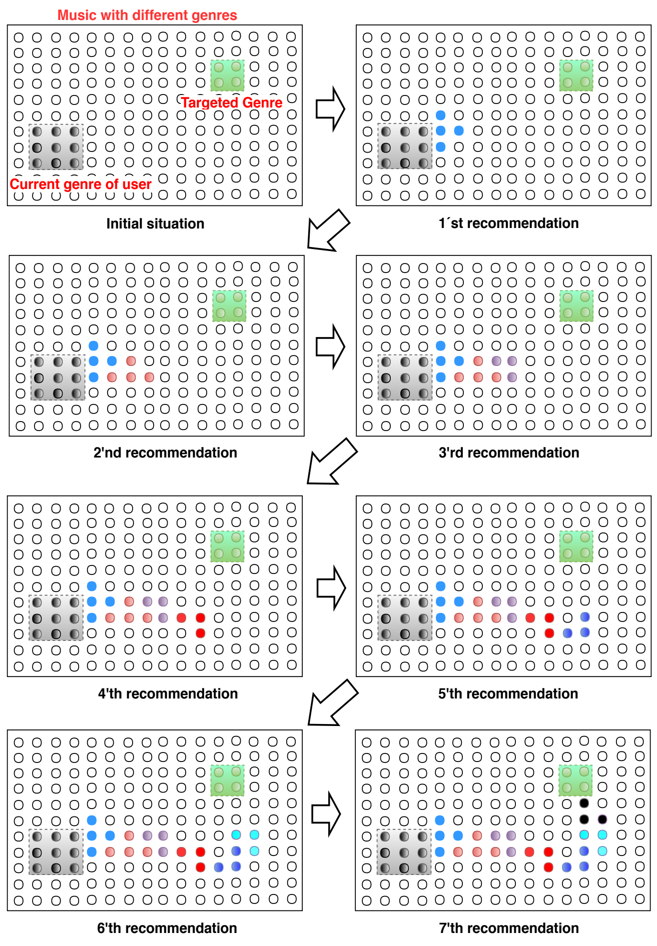


Figure 3.1: Conceptual view of LTRS in music domain.

products or products of a new segment). Another example for the potential application of LTRS is in E-learning. In this example, learning materials can be recommended to students with a higher level objective in view. By applying a LTRS system in the scope of learning, the activities (i.e., teaching and learning) can be more productive and less time consuming for students and teachers.

3.2 LTRS - Our Goal and Domain

In an abstract way, in LTRS the main research question is: how can we generate recommendation sequences that successfully guide users to a goal (a long term one), while satisfying their requirements and constraints? To be successful a long term recommender must act strategically and not merely tactically. This way, a user is guided towards the accomplishment of the goal (defined by a user or a platform manager) instead of merely satisfying immediate preferences.

As mentioned previously, LTRS can be used for different purposes and in various domains. In this thesis, we concentrate on the E-learning area and implement our LTRS recommender for this domain. Education and training is poised as one of the largest domains in the world economy [GMR07], and the expansion of E-learning products and the provision of E-learning opportunities is one of the most rapidly evolving domains of education and training, in both education and industry [Ime02].

In E-learning, users come from diverse academic backgrounds while having different ages, professions, social persuasions and abilities. Therefore, an E-learning system must allow all kinds of users to access information that they need in their available time. In spite of the necessity of such a system, current E-learning systems mostly offer a course which requires a fixed amount of learning time for all users. Such a course offering disregards the available time of the users while it is expected that they all do not have the same amount of available time for learning a course. Besides having different available time, the users all intend to enhance their knowledge (can be measured using the obtained scores on the course) as much as possible in their available time. Hence, in this thesis we propose a LTRS method for generating learning paths that maximize users' scores while satisfying their time constraints. According to our main objective, we formalize our learning path recommendation problem in the following form:

$$Maximize\ Score\ of\ P\ \ where\ \ T_u \geqslant T_p \tag{3.1}$$

In equation 3.1, P refers to a path that is generated for a user, T_u indicates the available time of the user while T_p is the expected time (estimated time) to complete the path. To accommodate uncertainty in time and score, we will also take into account the probability of not completing the learning path in the limited time of the user (R_t), and also the probability of not completing the path with the expected score (estimated score) for the path (R_s).

In order to generate learning paths, we use a structure that defines the prerequisite relations among different parts of a course, such as the relations among the LO or the relations among the lessons (Course, LO, and lessons are explained in chapter 2). This avoids the generation of paths that do not present a sensible sequence. In addition, this structure covers information about the type of different parts. This information is used to recommend appropriate learning materials to a user since during the learning process a user performs a certain task at each moment, such as learning a concept or answering a question. So, lack of such information might result in recommending a learning material which is not appropriate for a certain task, such as recommending a video instead of providing a question for a user. One of the main parts that its type is specified is the LO. A LO can be presented in different forms such as pdf, video, audio, or powerpoint but all of them are categorized in two main classes : expository LO and the evaluative ones. Expository LO are the ones that are learned (watch/read) by a user to answer the evaluative LO. A user is graded considering his/her answers to the evaluative LO. So, such a structure is useful for our learning path recommendation method.

This structure can be presented in the form of a directed graph (figure 3.2), which vertices present different parts of a course (e.g. LO or lessons) and edges show the prerequisite relations among them. This graph, which is known as a course graph, also contains metadata about each part, such as its type, title, and learning resources.

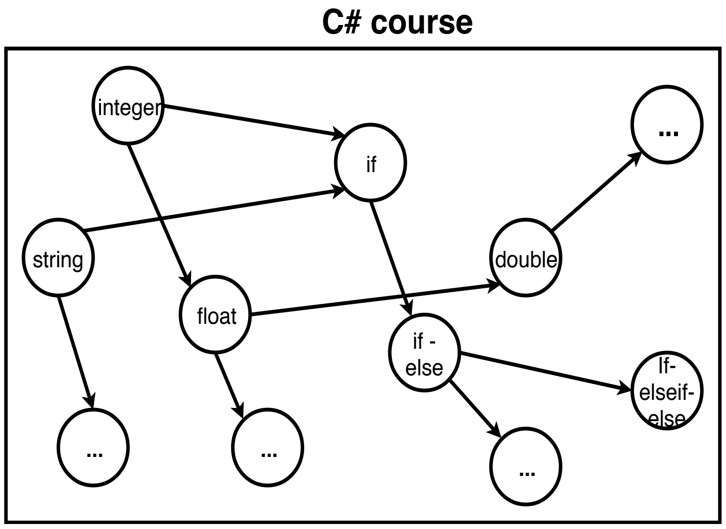


Figure 3.2: A conceptual example of a course graph for C#. In this course graph, vertices are the LO.

This graph can be arranged in different ways. It can be organized in one level as all parts and their relations are presented in a single level (figure 3.2), or can be structured in a more complex way (i.e., having a hierarchy structure like figure 3.3). For instance, all lessons and their relations can be presented in one level while their LO are presented in another level, and these two levels are connected considering the relevancy among lessons and LO.

Having the course graph, we now define a method (or methods) to generate learning paths. Moreover, the generated paths must satisfy the time constraints provided by the user. Therefore, we will have to be able to estimate the time that a user takes to complete the recommended paths. We also use approaches to estimate their learning score because we are interested in maximizing the learning score of a user. In addition, we use approaches for estimating the probability of error for time and score for the learning paths (i.e., R_t and R_s). These components result in an algorithm to recommend a generated path to a user. After recommending a path, how we can be sure that a user is able to follow it. Therefore, we require a strategy to keep tracking users after recommending a path. This strategy is for

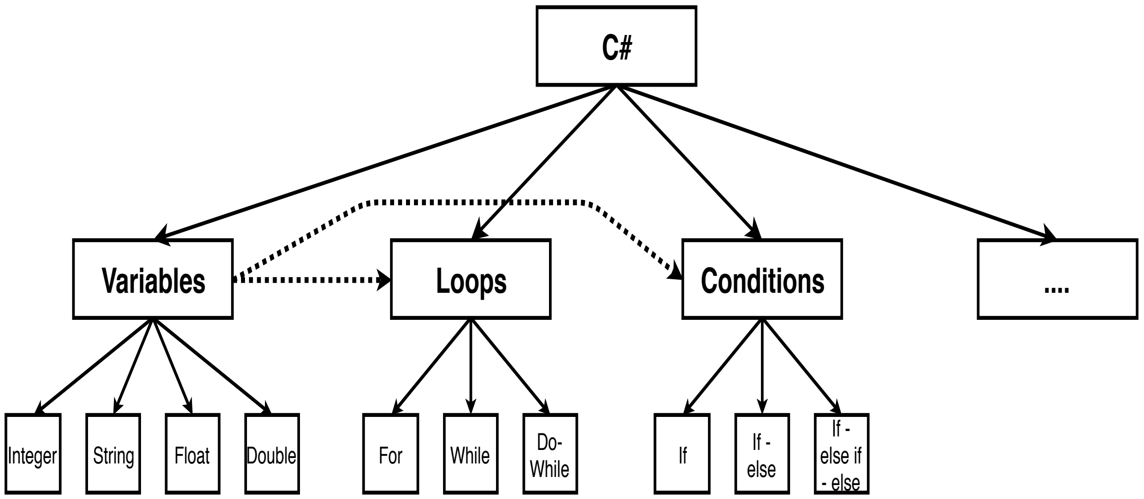


Figure 3.3: An example of a complex course graph for C#. In this graph, dash lines present the prerequisite relations among the lessons while the small squares present the LO for each lesson.

early detection of users disability in following a path and to adapt the path regarding their progress. In the next section, we detail the main problem of generating personalized learning paths for users and divide it into sub-problems and describe each of them shortly.

3.2.1 Sub-Problems Statement

We previously stated our problem of generating personalized learning paths under the time constraints of the users. In order to tackle the problem, we will first answer the following questions:

- What is the structure of our course graph for generating the paths?
- How can paths be generated?
- How do we estimate learning time and score for the paths?
- How do we estimate the probability of error for the estimated time and score (i.e., R_t and R_s)?
- How do we recommend a generated path to a user?
- If a user could not follow the path,
  - ∗ How can we know it?
  - ∗ What would be our solution for it?

- How do we evaluate the quality of our recommender?

In order to answer the questions, we break the main problem into sub-problems and tackle them one by one. The sub-problems are listed below. In addition, we provide a brief explanation about each of them. All these sub-problems are tackled in the following chapters.

1. **Course graph construction**:

   As explained already in this chapter, we can have different structures for a course graph. It can be designed in one layer where all LO and their precedence relations are shown in a single level (figure 3.2). Although this type of course graph has a simple structure and is easy to build, it has some deficiencies. For instance, with this type of course graph, it is not a trivial task to control what are the concepts that a user is learning in a path, or whether all LO of a path cover the same concepts or not. For example, in figure 3.2, "integer, float, double" is a path that covers a single concept (data type) , while "integer, if" is a path that covers two different concepts (data type and condition). Course graphs also can be designed in a more complex way to reflect different relations among LO. In spite of having a complex structure, these graphs have some benefits, such as allowing to have a controlled environment for the path recommendation. Such a structure can also ensure that different LO of a path cover various concepts rather than a single concept (a user learns more concepts rather than a signle one). For instance, a graph for a C# programming course can be structured in two layers, one layer presents the lessons and their relations (e.g. "loop" is a lesson) while the other layer shows the associated LO with the lessons, such as "For loop", "Do-While loop" and "While loop" (figures 3.3 and 3.4). In this thesis, we propose two methods that use the mentioned course graphs to generate paths (explained in the next chapter).

2. **Path generation**: Generating all paths (P) from the course graph (G) for a user u. These paths need to be generated considering the available time (T_u) and the knowledge background (sp) of the user. One possible solution to generate learning paths is using graph search algorithms (e.g. Breadth First Search, Depth First search) which traverse a course graph to find paths.

3. **Learning score estimation**: Since the generated paths should maximize the learning score of the users, we need to know what is the score for each path. So, we need to estimate the learning score for paths (S_p). In order to estimate score for a path, we initially estimate the learning score for each LO in that path. Then, the score for the path is obtained by accumulating the learning score of all LO in that path (explained in chapter 4).

4. **Learning time estimation**: Our paths should be generated under the time constraint of the user. So, we need to estimate how much time is going to be taken for learning a

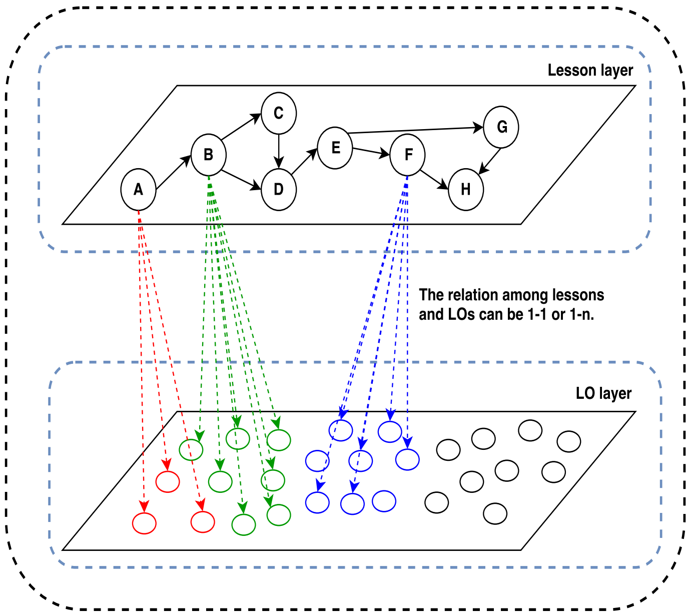


Figure 3.4: Two-layered course graph.

path by a user (T_p). Similar to the score estimation, for estimating time of a path we need to estimate the time of each LO in that path and then accumulate them (described in chapter 4).

5. **Estimating probability of error for learning score**: Estimating the probability of underachieving the estimated learning score for a path by a user (R_s). Our method to estimate R_s is detailed in chapter 4.

6. **Estimating probability of error for learning time**: Estimating the probability of not completing a path in the available time of a user (R_t). For that, we apply the same method as R_s for estimating R_t (presented in chapter 4).

7. **Recommending algorithm**: Designing an algorithm to recommend a generated path. This algorithm allows us to have a control on our recommendations while enabling us to collect information about user's progress during the learning process. This information will be used to adapt the path for the user (described in chapter 4).

8. **Selecting auxiliary LO**: These LO (L_{aux}) are not in the initial generated path and they will be generated and recommended to a user whenever he/she is not able to learn a lesson properly (underachieving the estimated score for a lesson). These LO are used in the approach that uses a two-layered course graph. L_{aux} generation is detailed in chapter 4).

9. **Recommender evaluation**: Our evaluation is twofold. We initially evaluate the quality of methods that are introduced to estimate learning time and score for paths (these

methods are explained in chapter 4). Next, we assess the quality of our recommender approach, which works based on a two-layered course graph, in a live environment. This approach is evaluated since it is an enhanced version of the other approach. For this purpose, we implement and embed our recommender in an E-learning system, and then perform an experiment to compare the performance of two groups. One group uses our recommender while another group uses an E-learning system that does not use recommendation. Our evaluation is explained in chapters 5 and 6.

3.3 Summary

In chapter 1, we described our motivation, research objectives and questions for this thesis. We also detailed the approaches and algorithms that are introduced by researchers to generate learning paths and the methods that are used to evaluate them in chapter 2. In the same chapter, we also highlighted the main challenges in path personalization methods. In this chapter, we have focused on describing our main goal and defined several questions that should be answered in order to achieve the goal. In addition, we detailed the main problem that we intend to tackle in this thesis and divided it into sub-problems.

Chapter 4

Path Generation and Recommendation

In chapter 2, we have described a number of methods and algorithms, which are proposed for generating and recommending learning paths to users. As mentioned in chapter 1, our main goal is to generate learning paths that maximize a user's learning score under a given time constraint. To this end, we propose two methods that generate learning paths regarding a user's time restriction and his/her knowledge background. These methods are explained in this chapter. This chapter consists of two main sections. In section 4.1, we explain a method to generate and recommend learning paths using a one-layer course graph. We initially present how we generate paths from the course graph that is designed by a course expert and describe nine different approaches, which are applied to estimate learning time and score for paths. We also introduce two methods for measuring the probability of error for the estimated score and time for paths. Finally, we highlight the drawbacks of this method.

In section 4.2, we present an enhanced version of the previous method for covering its problems. The second method is a lesson-based one which generates paths from a two-layered course graph. It uses the same methods as the previous method to estimate time, score, and the probability of error for the paths. For this method, we first explain how paths are generated from a two-layered course graph (each path is a sequence of lessons). We then explain how the initial LO are selected for each lesson of a path. We also describe our recommendation algorithm, which is designed for recommending a generated path to a user lesson by lesson. This recommendation algorithm enables the recommender to collect data about a user's progress for adapting the path for him/her. Finally, we describe a method that is used for generating auxiliary LO for a lesson. These LO are not in the initially generated path for a user, and they will be generated and recommended when a user could not learn a lesson properly (not obtaining the estimated score from a lesson).

In the following sections, we describe two methods for recommending a path that maximizes a user's score under a given time constraint. We first describe a method that uses a one-layer course graph to generate paths and explain its drawbacks. We then explain the second method which uses a two-layered course graph to cover the problems of the first method.

4.1 Method using a one-layer course graph

In this section, we explain the method that uses a one-layer course graph (Algorithms 1 and 2), describe how it generates the paths from the course graph and estimates the learning score and time for them. Furthermore, we describe our approaches to estimate the probability of error for the estimated score and time for a learning path. Figure 4.1 shows a general view of our method along with the steps that our method takes to personalize learning paths.

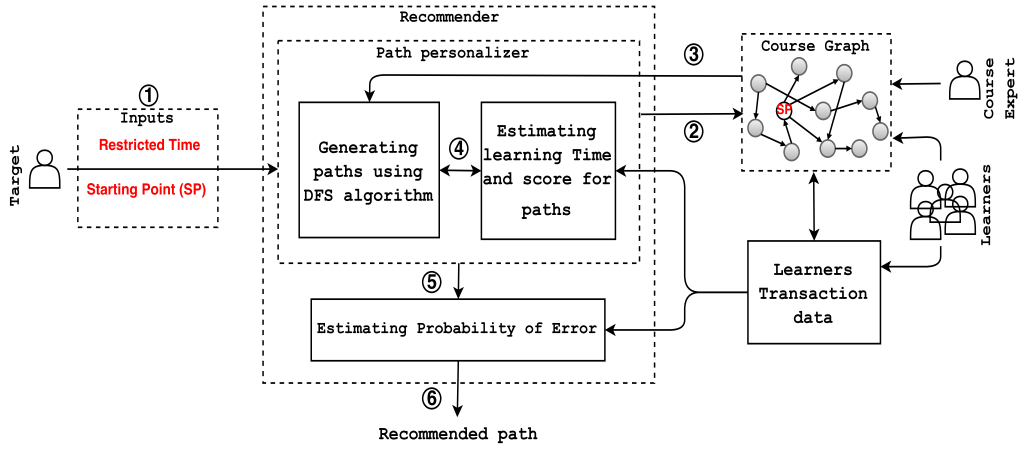

Recommender
Path personalizer
①
Inputs
Restricted Time
Starting Point (SP)
Target
Generating paths using DFS algorithm
④
Estimating learning Time and score for paths
③
②
Course Graph
SP
Course Expert
Learners
Learners Transaction data
⑤
Estimating Probability of Error
⑥
Recommended path

Figure 4.1: General view of our method using one-layer course graph. In the course graph, each LO has two attributes: time and score.

4.1.1 Learning path generation

In our method, after identifying the initial point *sp* by a user *u* (line 2 in algorithm 2), the depth-first-search (DFS) is applied to generate all learning paths (a sequence of LO) that start by *sp* from the course graph *G* (algorithm 1). Selecting the *sp* implicitly defines the knowledge background of a user since there are prerequisite relations among the LO of a course graph, and when a user specifies a LO as the starting point for paths (the LO that he/she wants to learn) we can conclude that he/she already knew the prior LO. In our approach, the DFS algorithm is selected to generate paths since as we exhaustively search the graph and enumerate all possible paths, DFS tends to consume less memory in comparison to the Breadth First Search (BFS) algorithm [BW84]. Dijkstra's algorithm [Joh73] is also conceptually a BFS that takes into account edge costs. Dijkstra is not selected since for using it we need to estimate the time and score for all edges of the course graph which might not be necessary and results in a high cost both in terms of time and computation. Therefore, for scalability reasons (the BFS may need too much memory and impractical to use) and avoiding a time consuming and costly computation the DFS is chosen. Steps 1 to 3 in figure 4.1 refer to the path generation phase.

Concurrent with traversing the course graph for generating paths, our method estimates time and score (algorithm 1) for them by means of our time and score estimation approaches (explained in sections 4.1.2 and 4.1.3). After generating paths and estimating time and score for them, our method estimates probability of not completing a path in the limited time of the user (R_t), and also the probability of not completing a path with the expected score (estimated score) for that path (R_s). Algorithms 1 and 2 represent our method, which uses a one-layer course graph. In the following sections, we explain in detail the different components of these algorithms.

Algorithm 1: DFS Algorithm for one-layer course graph (DFS_{one}).

```
Input: $node$, $T_u$, $G$, $D$, $S_P$, $T_P$, $R_t$, $R_s$, $P$, $i$.
Output: Generate all paths under $T_u$.
1  if (edgelist of $node$ = empty) then
2      Recom[i] ← ($P, T_P, S_P$, $R_t(P)$, $R_s(P)$);        ▷ Recom is a list to contain the paths.
3      i++ ;
4  else
5      foreach ($Newnode$ in edgelist of $node$) do
6          if ($T_P + T_{newnode} <= T_u$) then
7              $T_P += $ Estimating $T$ for $Newnode$;
8              $S_P += $ Estimating $S$ for $Newnode$;
9              $P \leftarrow P + Newnode$;
10             $DFS_{one}$ ($Newnode, T_u, G, D, i, P, T_P, S_P$);
11         else
12             $R_t(P) \leftarrow$ Estimating $R_t$ for $P$;
13             $R_s(P) \leftarrow$ Estimating $R_s$ for $P$;
14             Recom[i] ← ($P, T_P, S_P$, $R_t(P)$, $R_s(P)$);
15             i++ ;
16 Return **Recom**;
```

4.1.2 Time estimation approaches

The time for a path is computed by estimating the time of each LO, given the collection of previous interactions. For this purpose, we have considered nine different approaches. **Median** and **Mean** are two simple approaches, which are applied to estimate the time of each LO. In these approaches, after identifying the users that have visited a target LO, their time's median and mean are estimated and assigned as the learning time of a target LO for a target user.

Algorithm 2: Path generation algorithm using the one-layer course graph.

```
Input: $u$, $sp$, $T_u$, $G$, $D$.
Output: The path with the max score.
1  node ← sp;                                              ▷ $sp$:starting point.
2  P ← [sp];                                               ▷ P is a list.
3  i ← 1;
4  $T_P$ ← Estimating $T$ for $sp$;                        ▷ $T$:time.
5  $S_P$ ← Estimating $S$ for $sp$;                        ▷ $S$:score.
6  $R_t(P)$ ← Estimating $R_t$ for $sp$;                   ▷ $R_t$:probability of error for time.
7  $R_s(P)$ ← Estimating $R_s$ for $sp$;                   ▷ $R_s$:probability of error for score.
8  AllPaths ← $DFS_{one}$ ($node$, $T_u$, $G$, $D$, $S_P$, $T_P$, $R_t$, $R_s$, $P$, $i$);   ▷ Algorithm 1.
9  $P_{max}$ ← Select the path with the max score from **AllPaths**;
10 Return $\boldsymbol{P_{max}}$;
```

4.1.2.1 User Adjusted.Median and User Adjusted.Mean

In addition to the **Median** and **Mean** approaches, we proposed another approach based on equations 4.1 and 4.2. This approach is called **User Adjusted.Mean** (UA.Mean). The main motivation for presenting the **UA.Mean** for time estimation is to determine how much time a target user takes in comparison to the rest of the users in learning a LO.

In equation 4.1, the numerator (t_{uLOi}) indicates the time that a user u has spent for LO i, and the denominator ($mean(t_{.LOi})$) shows the average time that users have spent (excluding a target user) to learn the LO i. In this equation, n shows the total number of LO that are visited by u.

$$R_u = \frac{1}{n}\sum_{i=1}^{n} \frac{t_{uLOi}}{mean(t_{.LOi})} \tag{4.1}$$

After estimating the R_u, the time of the target LO_{tgt} (unseen by u) is estimated using equation 4.2. Based on this equation, the time of LO_{tgt} is obtained by multiplying R_u and the average time of users (seen the LO_{tgt}) on the LO_{tgt}.

$$T_{tgt} = R_u \times mean(t_{.LO_{tgt}}) \tag{4.2}$$

Another time estimation approach is **User Adjusted.Median** (**UA.Median**), which is the same as **UA.Mean** but using the median instead of mean.

4.1.2.2 Clust.Mean and Clust.Median approaches

Another approach used to estimate the learning time employs a clustering algorithm. In this approach, which is called **Clust.Mean**, we first identify all LO that are seen by a target user u. Then, we identify users that have seen those LO. Next, the identified users are divided into three groups with respect to their learning time on LO. Our idea for dividing the users into 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 generate three clusters (such as estimating score for a user having users' binary scores for only one LO), we generate two (slow and quick users). For clustering, we have used the k-means algorithm [HW79]. Finally, to estimate the learning time of a target LO, we estimate the average time of the users (who are in the same cluster as u and have seen the target LO) 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 $LO_{tgt}$.
1 $Seen_u \leftarrow$ Determing all LO seen by $u$.
2 $ALL_{seen} \leftarrow$ Determing users that have visited $Seen_u$.
3 $Clust \leftarrow$ Clustering $(ALL_{seen} + Seen_u)$ into 3 (or 2) clusters using K.Means and their time/score.
4 $Clust_{tgt} \leftarrow$ Find the target cluster (includes $u$) from $Clust$ and then drop $u$ from that cluster.
5 $T_{tgt} \leftarrow$ Average time/score of users in $Clust_{tgt}$ on $LO_{tgt}$.
6 Return $T_{tgt}$
```

Although other clustering algorithms could have been used, we have selected K-means, due to its simple implementation and efficiency[RM05].

Clust.Median is another approach for estimating learning time. This approach is similar to **Clust.Mean** but it uses the median instead of mean.

4.1.2.3 MF.Predict approach

We have also used a Matrix Factorization (MF) approach [KBV09] to estimate learning time for a path. This approach is selected since it has been successful in dealing with *Scalability* and *Sparsity* problems [Gil12]. *Sparsity* occurs when a user selected a few LO (a small portion of the LO) while dealing with high volumes of data (users and LO) causes the *Scalability* problem [Bur02].

MF discovers latent relations between LO and users [NJTY16, Kor08]. Assume that T is a matrix that contains n users (as rows) and m LO (as columns), while each entry presents the learning time of a user for a LO. This matrix will be decomposed into two matrices by applying a MF technique (figure 4.2):

$$T \approx \hat{T} = A \cdot B^T \tag{4.3}$$

In equation 4.3, A indicates a user matrix with n rows (as users) and f columns (as latent factors), while B is a LO matrix, which is composed of m LO (as rows) and f columns (as latent factors). f presents the total number of latent factors that are learned from past responses of users. Finally, we use a dot product (as shown in equation 4.4) to predict the learning time of a LO i for a user u.

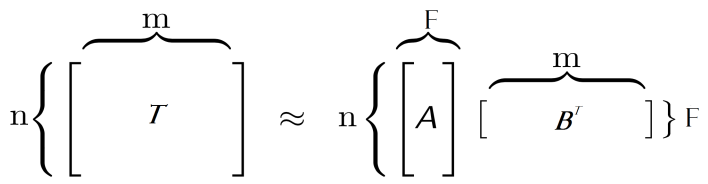


Figure 4.2: Decomposing a matrix into matrices A and B using MF technique.

$$\hat{T}_{ui} = A_u \cdot B_i^T \tag{4.4}$$

In order to generate matrices A and B, the regularized squared error of known entries in T is minimized using equation 4.5.

$$min_{A.,B.} \sum_{(u,i)\in T} (T_{ui} - A_u \cdot B_i^T) + \lambda(||A_u||^2 + ||B_i||^2) \tag{4.5}$$

In equation 4.5, λ presents a regularization parameter while $\lambda(||A_u||^2 + ||B_i||^2)$ is applied to prevent overfitting. Overfitting is avoided by penalizing high values for each parameter.

4.1.2.4 IRT.Predict method

Up to now, the approaches that we have proposed to estimate learning time do not take into account the users' ability and the difficulty level of the LO. To this end, we have adopted a time estimation approach which is based on Item Response Theory (IRT) [AY14, JMW06]. Item response theory (IRT) models are a class of statistical models that are able to explain the response behaviors of users to a set of questions by taking into account the users ability and difficulty level of questions [Joh04].

IRT models can be categorized in two main classes: unidimensional models, such as Rasch model [Joh04, AY14], and multidimensional models, such as 2 and 3 parameters logistic models (2PL, 3PL). The former models are generated based on one parameter (users' ability), and their assumption is : the discrimination level of LO is similar. Contrary to the former models, the latter ones consider more than one parameter to build a model. In these models, the 2PL uses two parameters (discrimination and difficulty), while the 3PL uses three parameters (discrimination, difficulty, and guessing) to build a model. The discrimination parameter (a) shows the differential capability of a LO while the difficulty parameter (b) presents the probability of a correct response to a LO. The guess parameter (c) shows that when a user does not know the correct answer in a multiple choice test, he/she guesses the answer.

The discrimination parameter (a) ranges from 0.5 to about 2.5. A value of a equal to about 1.0 is typical of many test items (LO), while values below 0.5 are insufficiently discriminating for most testing purposes, and values above 2.0 are infrequently found. The theoretical range of ability (difficulty parameter b) is from negative infinity to positive infinity, but practical considerations usually limit the range of values from -3 to +3. As explained previously, the guessing parameter (c) is generally interpreted as the probability of selecting the correct item-option by chance alone. Most test items (LO) have c parameters greater than 0.0 and less than or equal to 0.30 [Ree79, Bak01].

The S-shaped curve in figure 4.3 shows the relationship between the probability of correct response to a LO and the ability scale. In this figure, the horizontal axis is scaled in units of ability and the vertical axis is the probability of answering the item correctly. In IRT, this curve is known as the item characteristic curve (ICC). In figure 4.3, solid curved line shows an ICC for a LO. In this figure, we also have shown how three parameters are estimated for a LO.

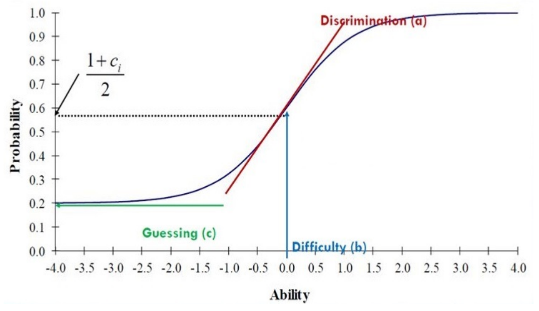


Figure 4.3: How to estimate 3 parameters (a: discrimination, b: difficulty, and c: guessing) for a LO. In this example, a=1.0, b=0.0, and c=0.2.

In this thesis, our assumption is : the LO do not have similar discrimination level. Hence, we select one of the multidimensional IRT approaches (2PL or 3PL) to generate the models. Regarding the results of our analysis, which are shown in section 5.3.1.2, the 2PL builds the most adequate models for our datasets. Therefore, we utilized the 2PL on the transaction data for estimating the discrimination and difficulty parameters of all LO (lines 1 and 2 in algorithm 4). Then, the estimated parameters and a target user's transaction data are used for estimating his/her ability ($\hat{\theta}_u$). The ability of the user is estimated using the Marginal Maximum Likelihood Estimation (MMLE) [J$^+$07] (equation 4.6, line 4 in algorithm 4). In equation 4.6, r_i shows the user's learning time (or learning score) for LO_i, k refers to the number of LO that are seen by u, and L addresses the marginal likelihood.

$$\hat{\theta}_u = argmax_{\theta_u} L(\theta_u | \{r_i\}, \{a_i\}, \{b_i\}) \quad where \quad 1 \leqslant i \leqslant k. \tag{4.6}$$

Finally, the obtained user's ability along with parameters a and b from a LO (LO_{tgt}) are used for estimating the learning time of the LO_{tgt} (line 5 in algorithm 4). For this purpose, we utilized the basic equation of 2PL (equation 4.7). This equation estimates the learning time that a user u with ability $\hat{\theta}_u$ requires for learning the LO_{tgt} correctly. In equation 4.7, the a_{tgt} and b_{tgt} indicate the parameters a and b for the LO_{tgt}.

$$T_{tgt_{(\theta_u, a_{tgt}, b_{tgt})}} = \frac{1}{1 + exp^{(a_{tgt} * (b_{tgt} - \theta_u))}} \tag{4.7}$$

Algorithm 4: IRT.Predict algorithm.

```
Input: User $u$, Transaction data $D$, Target $LO_{tgt}$ unvisited by $u$.
Output: Estimated learning time/score of $LO_{tgt}$ for a learner $u$.
1 for ( $i = 1$ to $n$ (number of $LO$ in $D$)) do
2     $(a_i, b_i) \leftarrow$ Estimating parameters $a$ and $b$ for $LO_i$ using $D$ and 2PL.
3 k $\leftarrow$ number of LO in $D$ that are seen by $u$                         ▷ k<n.
4 $\hat{\theta}_u \leftarrow$ Use MMLE given $\{(r_1, a_1, b_1), ..., (r_k, a_k, b_k)\}$      ▷ Equation 4.6.$r_i$:user's time (or score) for
   $LO_i$. $1 \leqslant i \leqslant k$.
5 $E_{tgt} \leftarrow T_{tgt_{(\hat{\theta}_u, a_{tgt}, b_{tgt})}}$                     ▷ Equation 4.7.  $LO_{tgt}$ is in D.
6 Return $E_{tgt}$.
```

In this thesis, to develop the **IRT.Predict** for estimating learning time, we have used the package *EstCRM* [Zop12], which is available in *R* programming language. This package is designed for continuous variables.

4.1.2.5 Johns Method (IRT 3PL)

As another possibility for estimating time and score for the LO, we have used a method proposed by Johns *et al.* [JMW06]. This method is based on IRT and uses a three parameters logistic model for the estimation. In this method, first, 3PL model is used to estimate three LO parameters (a: discrimination, b: difficulty, and c: guessing) using learners' transaction data. Then, these parameters together with a target learner's transaction data are used to estimate his/her ability (θ_u). Finally, the fundamental equation of 3PL (equation 4.8) is used to predict the learning score of a LO. After predicting the score, the following rule is applied in order to obtain the final score of the LO.

$$S_{tgt_{(\theta_u, a_{tgt}, b_{tgt}, c_{tgt})}} = c_{tgt} + \frac{1 - c_{tgt}}{1 + exp^{(a_{tgt} * (b_{tgt} - \theta_u))}} \tag{4.8}$$

$$Score\ estimation\ rule : \begin{cases} \textbf{if} Prediction \geqslant 0.5 \rightarrow score & = 1 \\ \textbf{if} Prediction < 0.5 \rightarrow score & = 0 \end{cases}$$

Although Johns *et al.* only presented their method to estimate score, we have used it without applying the aforementioned rule to estimate learning time of LO. To estimate time, **Johns** method is implemented using the EstCRM [Zop12] package in R while for score estimation *ltm* package (designed for binary variables) is used.

4.1.3 Score estimation approaches

We applied similar approaches as learning time estimation for estimating learning score while using the users' score instead of time. In our datasets, score is presented as a binary variable (1: Success, 0: Failure). Therefore, a learning score indicates the ability of a user to correctly complete a LO. So, for estimating score, **UA.Median** and **UA.Mean** approaches imply the ability of a user for completing a LO.

In order to estimate the learning score using the **IRT.Predict** method, we used the same approach that we have used for estimating time. In score estimation, the equation 4.7 and 4.8 results in a user's score for a LO. Note that the **IRT.Predict** method for score estimation is implemented using the *ltm* package in *R*, which is available for binary variables.

4.1.4 Time and score for a path

By estimating the learning time and score of each LO in a path using the mentioned approaches, the learning time and score of the path are obtained by accumulating the learning time and score of all LO in that path. Our path generation approach, which is detailed in algorithms 1 and 2, keeps adding LO to a path as long as the estimated learning time of the path (T_P) satisfies the learning time condition ($T_u \geqslant T_P$).

4.1.5 Estimating Probability of error

In the previous sections, we have proposed different approaches to estimate time and score for a path, but a user might underachieve the estimated score or not be able to complete a path in his/her available time [NJL18]. Therefore, in this section we estimate the probability of error for the estimated learning time and score for learning paths. The probability of error for score (R_s) is the probability of underachieving the estimated score for a learning path by a target user, while the probability of error for time (R_t) indicates the probability of not completing a learning path in a user's limited time.

4.1.5.1 Aggregating probability distributions of LO

In order to estimate the probability of error for a path, we first estimate the probability distribution of time (or score) of each LO in the path. Next, we aggregate the probability distributions of LO. For the aggregation of distributions, we use an efficient sampling approach (algorithm 5) where we randomly select several samples from each distribution to aggregate. Obviously, selecting more samples provides more accurate result. By aggregating the probability distribution of LO in a path, we obtain the users' time (or score) distribution for the entire path.

Algorithm 5: Aggregation of distributions.

```
Input: Path $P$, Transaction data $D$, Number of samples $x$.
Output: Aggregated probability distribution of LO of a path ($f_{agg}$).
1 for ( $i = 1$ to $n$ (number of $LOs$ in $P$)) do
2     $Pd_i \leftarrow$ Estimating the probability distribution of $LO_i$ using D;      ▷ using density function in $R$.
3     $S_i \leftarrow$ Randomly select $x$ samples from $Pd_i$;
4 for ( $j = 1$ to $x$) do
5     $T_j \leftarrow$ sum ($S_{(1,j)}, ..., S_{(n,j)}$);      ▷ $j$:samples' indices.  $n$:number of LO in $P$.
6 $f_{agg} \leftarrow$ Probability distribution of $T$;
7 Return $f_{agg}$;
```

4.1.5.2 Probability of error for time (R_t)

One solution to estimate the probability of error regarding learning time is to compute the total percentage of users that could not complete a target path in the available time of a target user (T_u). To this end, after estimating the aggregated distribution of time for a path, we compute the total distribution area that is located between the user's time constraint (T_u) and $+\infty$. This area, which presents the probability of not completing a path in a user's time constraint (figure 4.4-a), is given in equation 4.9.

$$R_t = \int_{T_u}^{+\infty} f_{agg}(x)dx \tag{4.9}$$

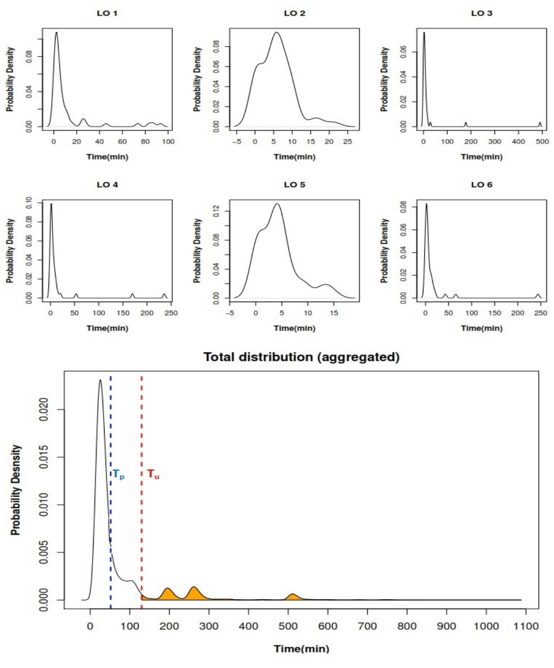
Figure 4.4-a shows an example of estimating probability of error for time for a path. In our example, the path comprises 6 LO that the probability distribution of time of each LO is presented in a separated graph. The total time distribution is obtained using the sampling method (algorithm 5). In the total distribution graph, T_P indicates the estimated time for a target user for the path by means of our estimation methods, while the T_u refers to the user's time constraint. As shown in this figure, the area of the orange highlighted parts corresponds to the probability of error for time for the path, which will be presented in percentile. It is provided in percentile to present the percentage of the users that already took the path but could not complete it in the time T_u. In figure 4.4-a, the probability of error for time could be close to zero if the T_u was more than $\approx$ 500 minutes.

4.1.5.3 Probability of error for score (R_s)

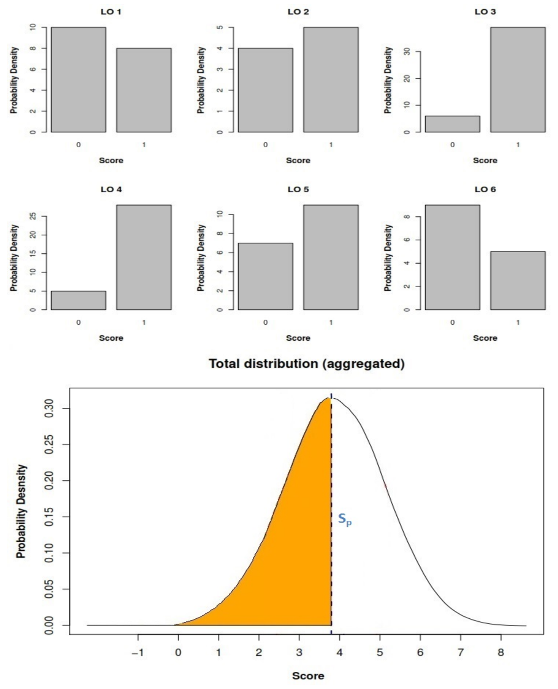
To estimate the probability of error for the score, we proceed in a similar manner as with time. We initially estimate the aggregated distribution of score for a path using the sampling method. After obtaining the aggregated distribution, the distribution area that is located between $-\infty$ and the estimated score for the path (S_P) indicates the probability of error for the score for that path (equation 4.10). This area presents the percentage of users that completed a target path while obtaining a score lower than the estimated score for that path for the target user (S_p). Like probability of error for time, the probability of error for score will be presented in percentile to show the percentage of the users that completed a target path by underachieving the estimated score for that path.

$$R_s = \int_{-\infty}^{S_p} f_{agg}(x)dx \tag{4.10}$$

In figure 4.4-b, the path is composed of 6 LO. The probability distribution of score of each LO is presented in a separated bar chart (in our datasets score is a binary variable). The total score distribution is obtained using the sampling method. As shown in this figure, the orange highlighted part indicates the probability of error for score for the path, which is the area between $-\infty$ and the estimated score for the path (S_P).



(a) For time. A path with 6 LO. T_P: estimated time for the path, T_u: user's time constraint, orange-colored part:R_t.



(b) For score. A path with 6 LO. S_P: estimated score for the path. orange-colored part: R_s.

Figure 4.4: Example of estimating probability of error.

After estimating the probability of error for time and score for a path, a path as well as the errors that are estimated for that path (R_t and R_s) will be provided to a target user.

This information assists a user to make an informed decision about the generated path. For instance, a path might provide a high score for a target user but the estimated R_t and R_s for that path are also high. So, it can be risky for the user to take that path since he/she might spend time for a path that can not learn properly while is not able to complete it on time. While another path which is generated for the same user provides a lower score in comparison with the previous path but it has lower R_t and R_s. Therefore, it would be useful to provide such information for the user that can help to make a knowledgeable decision. Figure 4.5 shows an example of how the estimated probability of error for time and score are provided for a user.

4.1.6 Illustrative example

Figure 4.5 presents an example of our method to generate paths, estimate time, score and probability of error for the estimated time and score for paths. In this example, a target user already knows the LO 1, 3, 4, 5, and 9 (Known LO), his/her available time is 30 minutes, and he/she selects the LO 13 as his/her starting point for the paths. As shown in figure 4.5, our method generates 9 paths using the DFS algorithm and estimates time and score for them. The estimated time of each LO is presented on top of each LO (figure 4.5). The time for a path, as mentioned in section 4.1.4, is computed by summing the time of all LO in that path. For example, in the case of **Path 1**, the total time to complete the entire path (i.e., LO 13, 14, 15, 22, 21, and 26) is equal to 39 minutes, which is more than the user's available time. Therefore, the LO 21 and 26 are dropped. The red-dash line shows the LO that need to be ignored in order to not exceed the limited time of the user. Like time, the score for a path is obtained by accumulating the score of LO in that path. Finally, the estimated score for a path as well as the estimated probability of errors for time and score for the path (R_t and R_s) are presented in front of that path. The R_t presents the percentage of users that already took the path but could not complete it in the available time T_u, while R_s highlights the percentage of users that completed a target path by obtaining a score lower than the estimated score for that path. The estimated R_t and R_s helps the user to make an informed decision about a course registration. Finally, our method recommends a path with the maximum score to the user.

4.1.7 Drawbacks of our method using one-layer course graph

Although the presented method has some advantages such as estimating time, score and probability of errors for the estimated time and score for the paths, it has a few drawbacks which need to be covered. The main shortcomings are listed below:

- **Time consuming and computationally expensive**. In this method, adding more LO to the course graph can increase the number of paths exponentially, which makes the

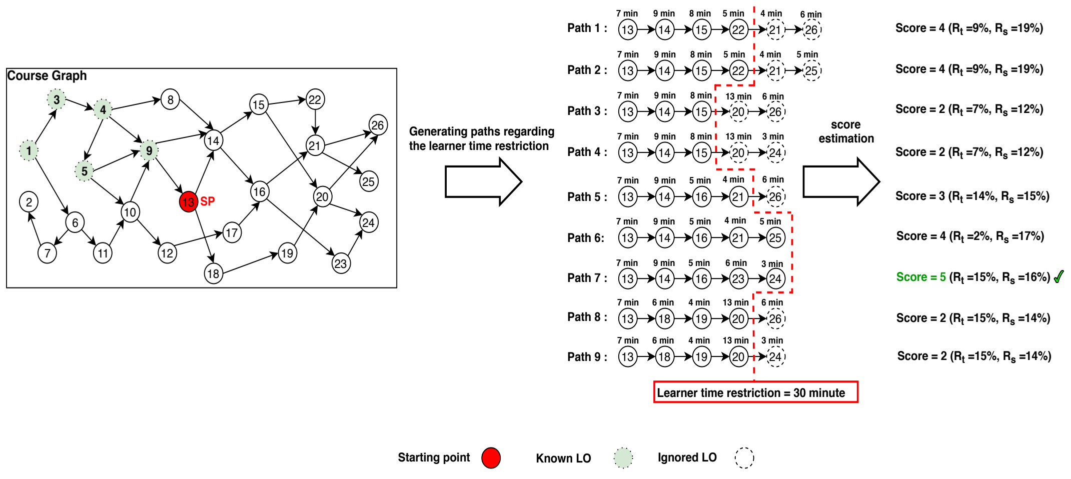

Course Graph
SP
Generating paths regarding the learner time restriction
score estimation
Path 1 : 13 → 14 → 15 → 22 → 21 → 26 (7 min, 9 min, 8 min, 5 min, 4 min, 6 min)
Path 2 : 13 → 14 → 15 → 22 → 21 → 25 (7 min, 9 min, 8 min, 5 min, 4 min, 5 min)
Path 3 : 13 → 14 → 15 → 20 → 26 (7 min, 9 min, 8 min, 13 min, 6 min)
Path 4 : 13 → 14 → 15 → 20 → 24 (7 min, 9 min, 8 min, 13 min, 3 min)
Path 5 : 13 → 14 → 16 → 21 → 26 (7 min, 9 min, 5 min, 4 min, 6 min)
Path 6: 13 → 14 → 16 → 21 → 25 (7 min, 9 min, 5 min, 4 min, 5 min)
Path 7 : 13 → 14 → 16 → 23 → 24 (7 min, 9 min, 5 min, 6 min, 3 min)
Path 8 : 13 → 18 → 19 → 20 → 26 (7 min, 6 min, 4 min, 13 min, 6 min)
Path 9 : 13 → 18 → 19 → 20 → 24 (7 min, 6 min, 4 min, 13 min, 3 min)
Score = 4 (R_t =9%, R_s =19%)
Score = 4 (R_t =9%, R_s =19%)
Score = 2 (R_t =7%, R_s =12%)
Score = 2 (R_t =7%, R_s =12%)
Score = 3 (R_t =14%, R_s =15%)
Score = 4 (R_t =2%, R_s =17%)
Score = 5 (R_t =15%, R_s =16%) ✓
Score = 2 (R_t =15%, R_s =14%)
Score = 2 (R_t =15%, R_s =14%)
Learner time restriction = 30 minute
Starting point
Known LO
Ignored LO

Figure 4.5: Example of method using one-layer course graph.

process of path generation time consuming and computationally expensive (searching graph and estimating time and score for the paths). For instance, the time complexity of searching a graph using DFS with branch factor (out-going degree) of b and the maximum depth of d is equal to b^d, and increasing the maximum depth of the graph (d) exponentially increase the time complexity. This problem can become a non-trivial one when we are dealing with a large-scale graph and the number of paths can be large. In addition, since the time and score for all paths need to be estimated, it makes the problem even harder. In the following section, we will propose a solution for keeping the search space restricted.

- **Proceeding on a single concept (lesson) rather than the whole course.** In this method, since there is not any control on recommending LO of different concepts (lessons), all the recommended LO in a path could cover the same concept (covering a single lesson). Therefore, the recommended path results in maximizing a user's score on a single lesson rather than a course. For instance, in the case of a programming language course, all the recommended LO of a path could cover the "loop" concept (for loop, while loop, do-while loop) while other important lessons for that course are ignored, such as "conditions" and "arrays".

To tackle the mentioned problems, we propose a method based on a two-layered course graph which is presented in the following section.

4.2 Method using a two-layered course graph

As mentioned in section 4.1.7, the previous method has a few deficiencies. To overcome the mentioned problems, we propose a lesson-based method that generates learning paths through the lessons rather than the LO. The course graph for this method is designed in two layers, a lesson layer and a LO layer. Lessons and their precedence relations are presented in the lesson layer while the LO layer includes the LO that are associated with the lessons (figure 4.6). In this method, since a path is made of lessons, therefore, learning a path ensures that a user is learning different concepts (not sticking to a single lesson). Furthermore, in this method the lesson layer is searched to find paths, which has a more restricted space than the LO graph (used in the previous method). Therefore, it results in reducing the cost of path generation both in terms of time and computation (searching paths and estimating time and score for them). In this method, lessons are sets of LO and a few of LO will be recommended to a user per lesson.

This lesson-based method uses the DFS algorithm to find the paths and estimate time and score for them (time and score estimation methods are explained in sections 4.1.2 and 4.1.3). In addition, it estimates the probability of error for the estimated time and score for the paths using the presented methods for them (section 4.1.5). In the following sections, we explain how this method selects LO for each lesson of a path. Furthermore, we introduce a method to update a path whenever a user is not able to follow it.

4.2.1 Learning path generation

We initially describe how we generate paths from a two-layered course graph and compute learning time and score for them. To generate learning paths using our approach, first, a target user selects a lesson as a starting point (*sp*) for the paths (line 2 in Algorithm 7). Selecting the *sp* implicitly defines the knowledge background of a user since there are prerequisite relations among the lessons in the lesson layer, and when a user specifies a lesson as the starting point for paths (the lessons that he/she wants to learn) we can conclude that he/she already knew the prior lessons. After specifying the *sp*, our approach applies the DFS algorithm [Tar71] to extract all lesson sequences (paths) from the lesson layer of a course graph (Algorithm 6). These extracted paths must satisfy the available time of the user (time estimation is explained in section 4.1.2). In figure 4.6, we present a general view of path generation approach that uses a two-layered course graph as well as the steps taken for generating personalized learning paths for a user. In this figure, the first three steps address the learning path generation phase.

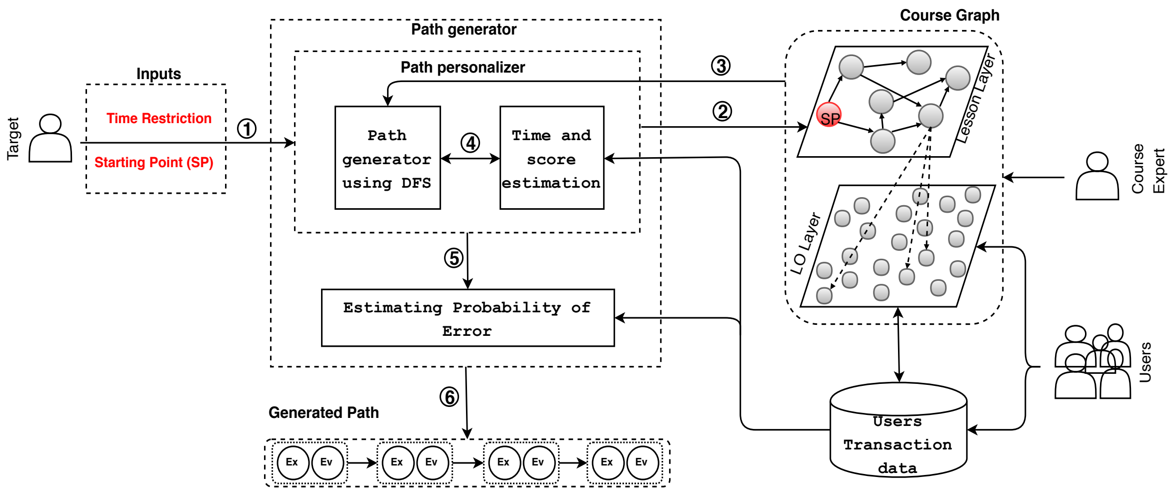


Figure 4.6: General view of path generation method using a two-layered course graph. Each block in the path indicates a lesson. *Ex* and *Ev* refer to the expository and evaluative LO.

4.2.1.1 LO selection for each lesson

The DFS algorithm is used to generate all possible lesson sequences (paths) which start with the *sp*. For each lesson of each path, there is a set of LO that can be selected and learned. Regarding the main goal of this thesis, the selected LO for each lesson should maximize a user's score while their accumulated time with the other LO of the path (LO that are already added to a path) needs to satisfy the time restriction of the user. In this thesis, two LO (LO which are related to a lesson) are assigned for each lesson of the path, one expository and one evaluative LO. Two LO are selected since the concepts that various LO of a lesson are delivering are related, such as explaining different kinds of "Loop", while knowing one type of loop is sufficient to answer all practical questions about the loop (number of selected LO can be modified regarding the course). These LO are selected regarding our main objective, which is recommending a path that satisfies a user's time constraint while maximizing the expected score. So, among a set of LO for a lesson those will be selected that a user most likely is able to learn them successfully in his/her available time (T_u). Therefore, we formalize our LO selection for a lesson as follows:

$$Maximize\left(S(LO_{ex\in\mathbb{L}}) * W(LO_{ex\in\mathbb{L}}) + S(LO_{ev\in\mathbb{L}}) * W(LO_{ev\in\mathbb{L}})\right) where \underbrace{T_P}_{\text{Path}} + \underbrace{(T_{LO_{ex}} + T_{LO_{ev}})}_{T_{newnode} \text{ in alg 6}} \leqslant T_u \tag{4.11}$$

In equation 4.11, $\mathbb{L}$ indicates a lesson of a course, S refers to the estimated score for a LO, LO_{ex} and LO_{ev} are the expository and evaluative LO, and W addresses the weight of LO. In our method, the weights of all expository and evaluative LO are considered equal, and also expository LO 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 time restriction of the user.

To select LO, since there would be a possibility of ties in the learning score of LO (having different LO with the same score), the learning time is used to break the ties (minimizing time). If the time of LO are also tied, the completion rate of LO and later on (if needed) the visited rate of LO are used to break the tie (maximizing these two rates). The completion rate shows how many times other users were successful in completing a LO while visited rate indicates how many times a LO is visited by the users.

Algorithm 6: DFS algorithm for two-layered course graph (DFS_{two}).

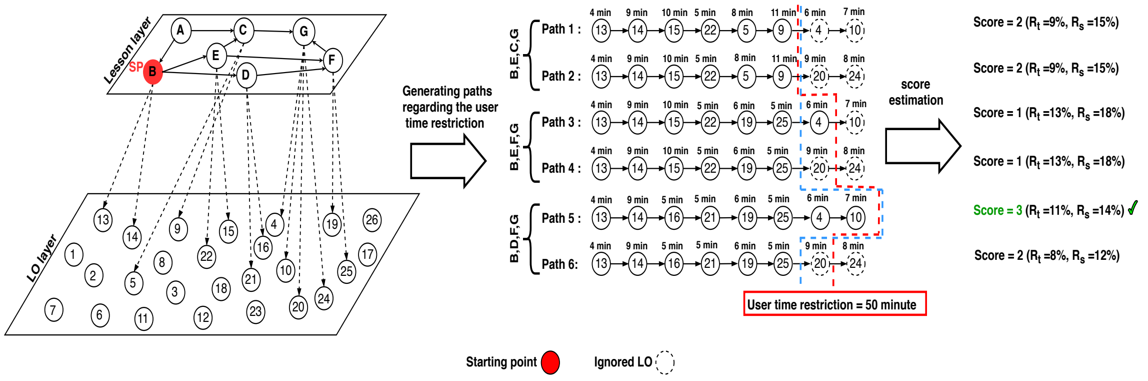
```
Input: $node$, $T_u$, $G$, $D$, $S_P$, $T_P$, $R_t$, $R_s$, $P$, $i$.
Output: Generate all paths under $T_u$.
1  if (edgelist of node = empty) then
2      Recom[i] ← $(P, T_P, S_P, R_t(P), R_s(P))$;              ▷ Recom is a list to contain the paths.
3      i++ ;
4  else
5      foreach (Newnode in edgelist of node) do
6          $LO_{all}$ ← Estimate time and score for $LO$ of $Newnode$;      ▷ $Newnode$ is a lesson.
7          $LO_{selected}$ ← Select $LO_{ex}$ and $LO_{ev}$ from $LO_{all}$;   ▷ LO selection : Section 4.2.1.1.
8          $S_{newnode}$ ← Accumulating the score of $LO_{selected}$;
9          $T_{newnode}$ ← Accumulating the time of $LO_{selected}$;
10         if $(T_P + T_{newnode} <= T_u)$ then
11             Assign $LO_{selected}$ to $Newnode$;
12             $T_P += T_{newnode}$;
13             $S_P += S_{newnode}$;
14             $P \leftarrow P + Newnode$;
15             $DFS_{two}$ $(Newnode, T_u, G, D, P, T_P, S_P, i)$;
16         else
17             $R_t(P)$ ← Estimating $R_t$ for $P$;
18             $R_s(P)$ ← Estimating $R_s$ for $P$;
19             Recom[i] ← $(P, T_P, S_P, R_t(P), R_s(P))$;
20             i++ ;
21 Return Recom;
```

4.2.1.2 Illustrative example of path generation

Figure 4.7 presents an example of the method that uses a two-layered course graph to generate paths, estimate time, score and probability of error for them. In this example, the available time of the target user is 50 minutes, and he/she selects the lesson B as his/her starting point for the paths. As shown in figure 4.7, our method generates 6 paths using the DFS algorithm and estimates learning time and score for them. The estimated time of each LO is presented on top of each LO (figure 4.7). The time for each path, as mentioned in section 4.1.4, is computed by summing the time of all LO in that path. For example, in the case of **Path 1**, the total time to complete the entire path (i.e., LO 13, 14, 15, 22, 5, 9, 4, and

Algorithm 7: Path generation algorithm using the a two-layered course graph.

```
Input: $u$, $sp$, $T_u$, $G$, $D$.
Output: The path with the max score.
1  node ← sp;                                                        ▷ $sp$:starting point.
2  P ← [sp];                                                         ▷ P is a list.
3  Select initial LO for P;                                          ▷ LO selection : Section 4.2.1.1.
4  i ← 1;
5  $T_P$ ← Estimating $T$ for $sp$;                                   ▷ $T$:time.
6  $S_P$ ← Estimating $S$ for $sp$;                                   ▷ $S$:score.
7  $R_t(P)$ ← Estimating $R_t$ for $sp$;                              ▷ $R_t$:probability of error for time.
8  $R_s(P)$ ← Estimating $R_s$ for $sp$;                              ▷ $R_s$:probability of error for score.
9  AllPaths ← $DFS_{two}(node, T_u, G, D, S_P, T_P, R_t, R_s, P, i)$;  ▷ Algorithm 6.
10 $P_{max}$ ← Select the path with the max score from the **AllPaths**;
11 Return $\boldsymbol{P_{max}}$;
```



Lesson layer
SP
A
B
C
D
E
F
G
LO layer
Generating paths regarding the user time restriction
B,E,C,G
B,E,F,G
B,D,F,G
Path 1 : 13 → 14 → 15 → 22 → 5 → 9 → 4 → 10
Path 2 : 13 → 14 → 15 → 22 → 5 → 9 → 20 → 24
Path 3 : 13 → 14 → 15 → 22 → 19 → 25 → 4 → 10
Path 4 : 13 → 14 → 15 → 22 → 19 → 25 → 20 → 24
Path 5 : 13 → 14 → 16 → 21 → 19 → 25 → 4 → 10
Path 6: 13 → 14 → 16 → 21 → 19 → 25 → 20 → 24
4 min
9 min
10 min
5 min
8 min
11 min
6 min
7 min
User time restriction = 50 minute
score estimation
Score = 2 (R_t =9%, R_s =15%)
Score = 2 (R_t =9%, R_s =15%)
Score = 1 (R_t =13%, R_s =18%)
Score = 1 (R_t =13%, R_s =18%)
Score = 3 (R_t =11%, R_s =14%) ✓
Score = 2 (R_t =8%, R_s =12%)
Starting point ●
Ignored LO

Figure 4.7: Example of our path generation method. Red dash-line shows the dropped LO while the blue dash-line shows the ignored lessons.

10) is equal to 60 minutes, which is more than the user's available time. Therefore, the last lesson is dropped (includes LO 4 and 10). The red dash-line shows the LO that need to be ignored in order to not exceed the limited time of the user (blue dash-line shows the ignored lessons). Like time, the score for a path is obtained by accumulating the score of LO in that path. Finally, the estimated score for a path as well as the estimated probability of errors for time and score for the path (R_t and R_s) are presented in front of that path.

4.2.2 Path recommendation

After assigning LO to each lesson of a path, estimating learning time, score and their probabilities of errors (using the methods in sections 4.1.2, 4.1.3 and 4.1.5), a path will be recommended to a user lesson by lesson (algorithm 8). This algorithm allows the recommender to monitor a user's progress while collecting information about his/her interactions with a course, such as his/her learning score and time for each lesson, which has two main benefits:

1. Monitoring users constantly enables the recommender to detect a user's failure in early stages and avoid wasting time of a user by adjusting the recommended path.
2. Collected users information helps to recommend a path that fits a user's competency level. It results in enhancing the users' satisfaction and keeping them engaged with the system.

As mentioned previously, in this method (uses a two-layered course graph) a user receives a path lesson by lesson. By completing a lesson, a user's obtained score for the lesson (U_s) is compared with the estimated (expected) score (E_s) for that lesson. If the user could accomplish the lesson with the estimated score for that lesson ($U_s \geqslant E_s$), he/she will receive LO for the next lesson, otherwise, the obtained score (U_s) needs to be analyzed. To this end, a score threshold (δ_s) is considered in our recommender. This threshold 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 maximum possible score). If δ_s is higher than the obtained score for a lesson ($U_s < \delta_s$), it means that the user could not learn the lesson. Therefore, the recommender reshows the LO that are visited by the user for that lesson. In a situation that $\delta_s \leqslant U_s < E_s$, auxiliary LO will be recommended to help the user on a target lesson. Auxiliary LO as well as our approach to estimate them are explained in section 4.2.2.1.

$$For\ each\ Lesson = \begin{cases} \textbf{if}\ \ U_s \geqslant E_s \rightarrow Next\ lesson \\ \textbf{if}\ \ \delta_s \leqslant U_s < E_s \rightarrow Auxiliary\ LO \\ \textbf{if}\ \ U_s < \delta_s \rightarrow Reshow\ the\ LO \end{cases}$$

4.2.2.1 Auxiliary LO

As explained in the previous chapter, auxiliary LO are the ones that are not in the initial generated path for a user, and they will be recommended to a user when he/she could not learn a lesson properly and did not accomplish it with an estimated (expected) score (E_s). To generate auxiliary LO for a lesson, our approach considers a score threshold (δ_s). If an obtained user's score for a lesson (U_s) is more than the δ_s and less than the expected score for that lesson ($\delta_s \leqslant U_s < E_s$), our approach generates auxiliary LO for that lesson.

To generate auxiliary LO, our recommender uses a similar method that is used to identify the initial LO for a Lesson (explained in section 4.2.1.1). For this purpose, whenever auxiliary LO are needed our recommender generates two ranking lists for LO of a lesson using the method in section 4.2.1.1. Two lists are generated since there are two types of LO for each lesson, expository and evaluative. To recommend auxiliary LO for a lesson, the recommender goes through each ranking list and recommends LO with the highest rank which are not recommended to a target user. Whenever that recommending auxiliary LO result in exceeding the available time of the user (T_u), our recommender algorithm ignores the lessons from the end of the path that exceed the time restriction of the user. Algorithm 9 details our method to recommend auxiliary LO for a lesson.

Algorithm 8: Path Recommendation algorithm.

```
Input: Path $P$, User available time $T_u$, transaction data $D$.
for ($i = 1$ to number of lesson in $P$) do
    Recommend LO of $L_i$ to $u$;                         ▷ $L$ indicates a lesson.
    $U_s \leftarrow$ Obtained score of $u$ on $L_i$;
    $\delta_s \leftarrow$ Minimum score to pass a lesson;
    if ($U_s \geqslant E_s$) then
        $T_u \leftarrow T_u - T_{L_i}$;       ▷ $E_s$ Estimated score for a lesson.  $T_{L_i}$:Time that U spent on $L_i$.
        if ($T_u < T_{L_{i+1}}$) then
            Terminate;                    ▷ $T_{L_{i+1}}$:Estimated (expected) time for the next lesson.
    else if ($\delta_s \leqslant U_s < E_s$) then
        $L_{aux} \leftarrow$ Estimating auxiliary LO using algorithm 9;
        $T_{aux} \leftarrow$ Estimating time for $L_{aux}$;
        if ($T_{aux} \leqslant T_u$) then
            Recommend $L_{aux}$;
            $T_u \leftarrow T_u - T_{aux}$;
            $U_{s_{aux}} \leftarrow$ Obtained score of $u$ on $L_{aux}$;
            if ($U_{s_{aux}} \geqslant E_s$) then
                Go to Line 1;
            else if ($\delta_s \leqslant U_{s_{aux}} < E_s$) then
                Go to Line 9;
            else
                Go to Line 24;
        else
            Terminate;
    else
        $T_{reshow} \leftarrow$ Time to re-read the same LO;
        if ($T_{reshow} \leqslant T_u$) then
            Reshow the same LO;
            $T_u \leftarrow T_u - T_{reshow}$;
            $U_{s_{re}} \leftarrow$ Obtained score of $u$ on same LO;
            if ($U_{s_{re}} \geqslant E_s$) then
                Go to Line 1;
            else if ($\delta_s \leqslant U_{s_{re}} < E_s$) then
                Go to Line 9;
            else
                Go to Line 24;
        else
            Terminate;
```

Algorithm 9: Generating auxiliary LO for a lesson.

```
Input: User $u$, Lesson $L$, Score threshold $\delta_s$, Transaction data $D$, Estimated score for $L$ ($E_s$), Obtained score
    for $L$ ($u_s$).
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 $D$;
6          $V_i \leftarrow$ Estimating visited rate for $Ex_i$ using $D$ ;    ▷ For expository LO, score and completion rate
             are ignored since they do not have such information.
7      $Aux_{Ex} \leftarrow$ Selecting a LO using $T$ and $V$ which is unvisited by $u$ ;       ▷ LO selection is explained in
         4.2.1.1.
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 a LO using $S$, $T$, $C$ and $V$ which is unvisited by $u$ ;  ▷ LO selection is explained in
         4.2.1.1.
14 Return ($Aux_{Ex}, Aux_{Ev}$);
```

4.3 Summary

In this chapter, we present two methods for generating paths that maximize a user's score under a given time constraint. We first detail a method that works based on a one-layer course graph. For that, we initially explain how it generates all paths from a course graph. Next, we explain the methods to estimate the time and score for the generated paths. We then describe two methods that are proposed for estimating the probability of error for the estimated time and score for the paths. The probability of error for time (R_t) addresses the probability of not completing a path in a user's available time, while the probability of error for the score (R_s) presents the probability of underachieving the estimated score for a path. The idea by providing the R_t and R_s is to assist a user in making an informed decision about selecting a learning path. Finally, we highlight the main drawbacks of this method which uses a one-layer course graph.

We then detail the second method, which is an enhanced version of the first method. The second method is a lesson-based one which generates paths from a two-layered course graph. It uses the same methods as the previous method to estimate time, score, and the probability of error for the paths. For this method, we first present how it generates paths (lesson sequences) from the course graph and selects LO for each lesson of a path. We then explain an algorithm that is used by this method to recommend a path to a user lesson by lesson. This algorithm allows the recommender to update a path regarding a user's progress. Finally, we explain our approach to estimate auxiliary LO for a lesson when a user could not learn a lesson correctly.

Chapter 5

Evaluating Estimation Methods

Up to this chapter, we have explained our main objective and research questions. In addition, we have introduced two methods to obtain our main goal. In this chapter, we assess the quality of learning time and score estimation methods, which are introduced in chapter 4. For that, we initially describe the methodology that we have used for our evaluation. We then describe the two datasets that we have utilized for the evaluation. In order to have a better understanding of the datasets, we also analyze them in detail. We then compare the performance of the estimation methods using the Mean Absolute Error (MAE) and Average MAE measures. Furthermore, we evaluate our methods in the case of overestimating and underestimating learning time and score for the paths.

5.1 Evaluation Methodology

In order to assess the performance of our estimation approaches for a particular user, we determine a sequence of LO (path) that the user already selected and visited. We then ignore (hide) the learning time and score of some of the LO from the sequence (always last LO in the sequence) and consider them as unobserved LO (figure 5.1), and attempt to estimate the learning time and score of unobserved LO (unobserved LO: test set, observed LO: train set). The sequence of unobserved LO are set as a window. Experimentally, the window's size ranges from 1 to 10. For each window, we estimate time and score for unobserved LO. We then use MAE to assess the performance of the estimation. The reason for considering different window sizes is to compare estimation approaches under varying estimation horizon conditions.

In detail, we perform the following tasks in our evaluation methodology:

1. Find users having enough transactions. For example, when the window size is set to 5, we need to find users that already visited more than 5 LO, so, when the time and score of 5 LO are ignored, they still have visited LO.
2. Identify a sequence of LO that each of these users have already selected.
3. For each user, we hide the score (or time) of LO from the end of the sequence considering the size of the window (unobserved LO). For instance, if the window size is 5, we hide the learning score (or time) of 5 LO from the end of a sequence for a user.

4. For each unobserved LO for a user, we use the learning score (or time) of other users that have already visited that LO, and estimate score (or time) for it by means of our estimation methods.

5. Calculate the error between two vectors, the one that contains the actual score (or time) of unobserved LO and the one that contains the estimated ones. Each element in the estimated vector is an estimation for the corresponding element in the actual vector. In this thesis, since we use the MAE, the error computation is estimating the average error for a user for a window size.

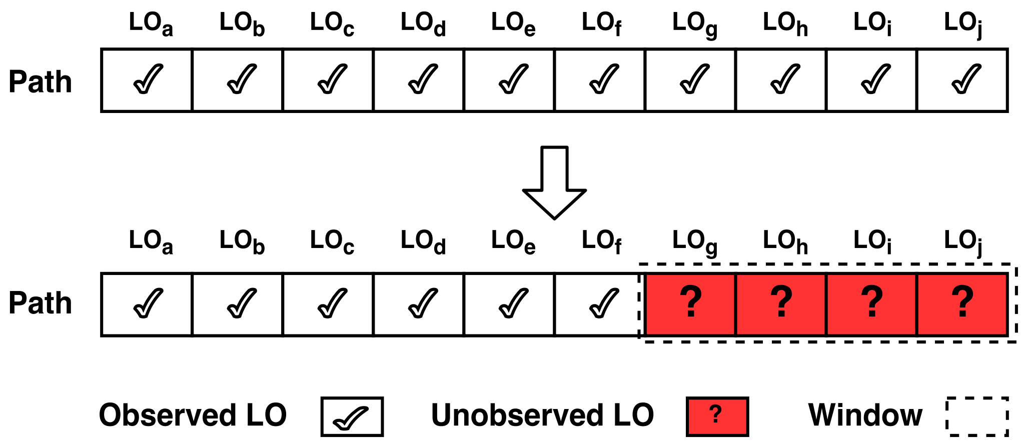


Figure 5.1: Example of our evaluation method. For a sequence, learning time and score of observed LO are used to estimate the time and score for unobserved LO. Window size=4.

5.2 Dataset Description

In order to conduct the evaluation and assess the quality of learning time and score estimation methods, we use two datasets. One of the datasets is taken from Enki, which is a web-based learning environment [PLQ16]. This data is for an open course on C# that was organized by the Polytechnic Institute of Porto for a week in 2015-2016. The Enki dataset includes two kinds of data, usage data and a course graph.

The second dataset is from Mooshak, which is a system for managing programming contests on the web and also can be used as a pedagogical tool [LS03]. This data is for a regular database course which was organized by the Faculty of Science of the University of Porto in the second semester of 2016-2017. The users are the second year undergraduate students in computer science field.

Table 5.1 summarizes the information about the Mooshak and Enki datasets that we have used in this thesis.

	Attributes	**Mooshak**	**Enki**
Transaction data	# Users	144	61
	# LO	31	59
	# Transactions	2646	917
	LO type	Ev	Ex - Ev
	Time range	(1,93)	(1,490)
	Time scale	Minute	Minute
	Score scale	Binary	Binary
	Sparsity degree	40.72%	74.52%
Course graph	# Nodes (LO)	31	59
	# Lessons	-	5
	# Links	-	9

Table 5.1: Datasets description. *Ex* and *Ev* refer to the expository and evaluative LO. Links indicate the precedence relation among the lessons. In score, 0 means fail to learn a LO while 1 means successful to correctly learn a LO by a user.

5.2.1 Data analysis

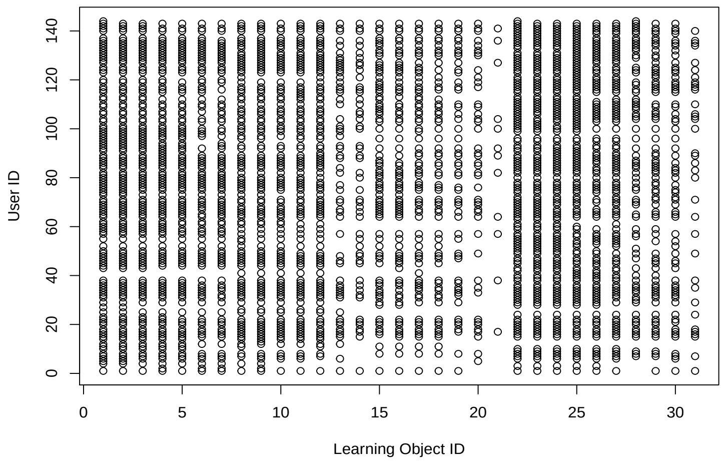
In order to have a more clear view of the datasets and assess how the users interacted with the Mooshak and Enki courses, we analyze them in this section. This analysis also assists us to have a better understanding of the evaluation results that we are going to get later on for the time and score estimation methods. In this analysis, we aim to answer the following questions:

1. How users interacted with the LO of each system?

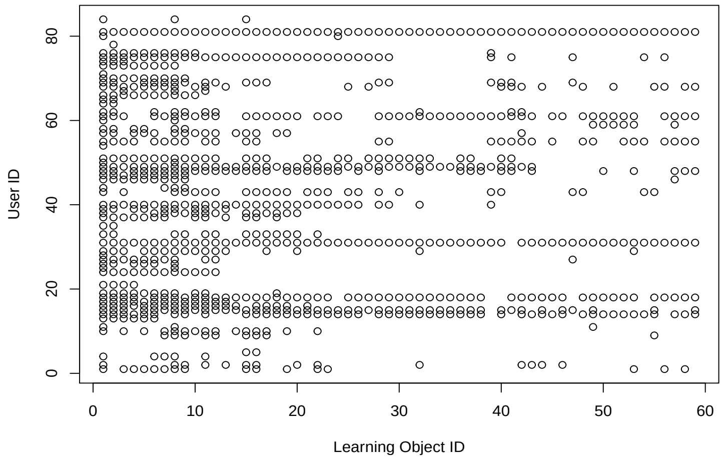
2. How much time did users spend on each LO?

3. What was the success rate of users for each LO?

For that, we investigate each dataset and visualize the results in the form of different graphs. Figure 5.2 presents how users interacted with LO in each system and how dense are the datasets. In this figure, the horizontal axis presents the ID of each LO, the vertical axis is the users' ID, and each point (circle) in the figure shows the selection of a LO by a user. As shown in this figure, users attempted almost all available LO in Mooshak, while in Enki users mainly selected and learned a few first LO which are presented to them at the beginning of the learning process, and then they quitted the system.



(a) Mooshak.

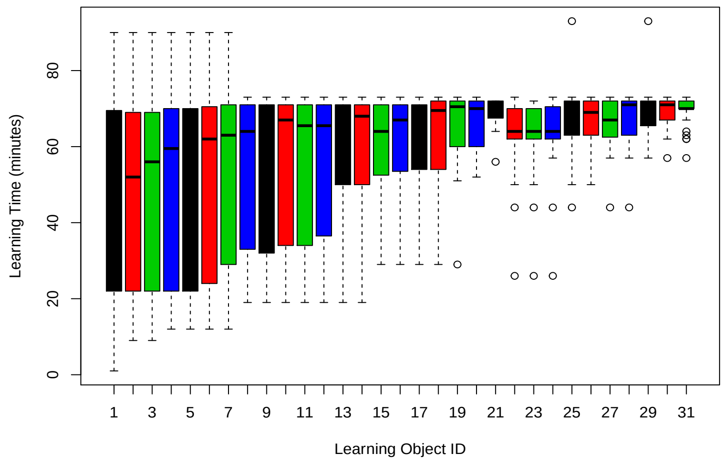


(b) Enki.

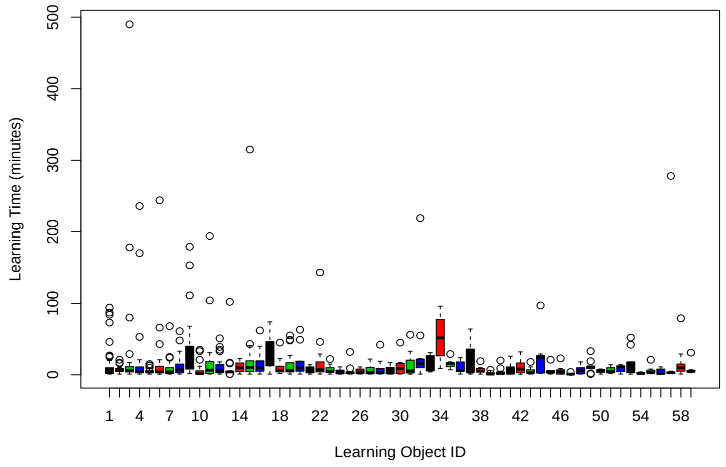
Figure 5.2: Users interactions with LO in Mooshak and Enki systems.

Figure 5.3 shows the time variation of users on different LO for both Enki and Mooshak. In this figure, the horizontal axis shows the ID of each LO while the vertical axis presents the time variation of users on each LO. In Mooshak, since users have access to LO one after another and all LO are available for users until a specified date (all LO have almost similar availability deadline), the learning time of LO gets less variation over time, while in Enki LO are available for users without any restriction.

Figure 5.4 shows the success rate of users on different LO in Enki and Mooshak. As in figure 5.3, the horizontal axis indicates the ID of each LO while the vertical axis shows the success rate for LO, which means how many times a LO is answered correctly by users. In Enki, since there are two types of LO, expository and evaluative, the expository type has no score, therefore the success rate of this



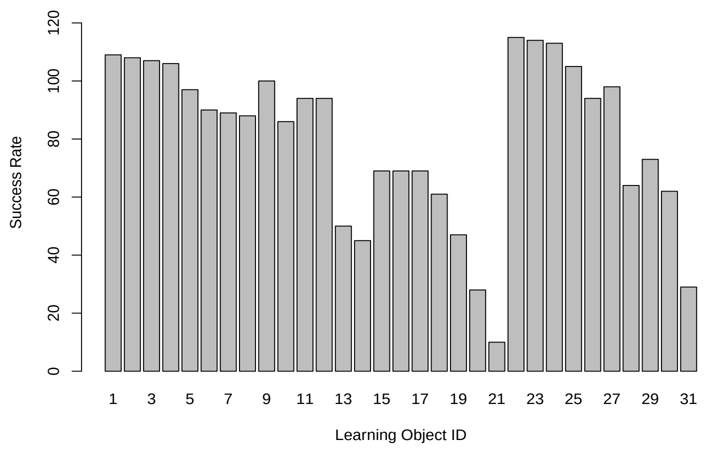
(a) Mooshak.



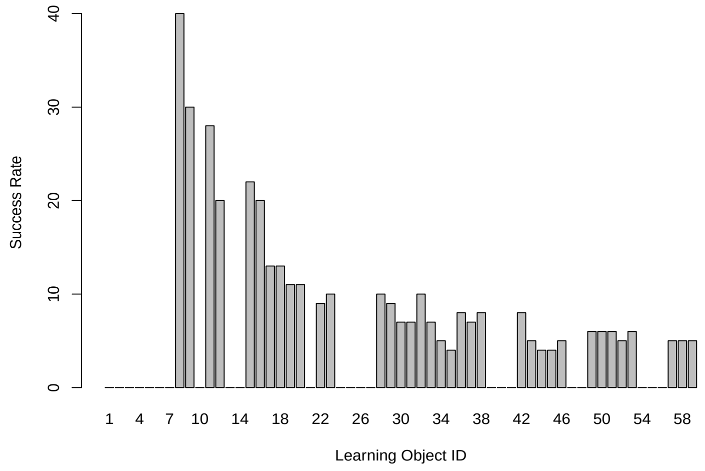
(b) Enki.

Figure 5.3: Learning time variation of users on different LO in Mooshak and Enki systems. To have a more clear view, the boxplots are presented in different colors.

type is equal to zero, while in Mooshak all LO need to be answered and they have score.



(a) Mooshak.



(b) Enki.

Figure 5.4: Success rate of users on different LO in Mooshak and Enki systems.

5.3 Results and Discussions

We describe and discuss our experiments in this section. In brief, the evaluations that are presented in this chapter are as follows:

1. To generate a model for the MF.Predict approach (MF.Predict is based on MF method), we need to determine the optimum values for its parameters, which are **regularization**, **learning**

rate, **number of factors**, and **number of iterations**. For this purpose, we have used the cross-validation technique which is explained in section 5.3.1.1.

2. Comparing the goodness of fit of the generated 2PL and 3PL IRT models for each dataset (analysis of variance).
3. Comparing the performance of nine methods for estimating time and score.
4. Evaluating our best estimation methods (**Clust.Mean** and **Clust.Median**) in the case of underestimation and overestimation of time and score.

5.3.1 Generating models for MF.Predict and IRT.Predict

Before evaluating the performance of the introduced approaches for estimating time and score, we aim to generate a model for the **MF.Predict**, and also choose an IRT model for the **IRT.Predict** approach.

5.3.1.1 Generating MF.Predict model

In order to generate a model for the MF.Predict approach we initially need to determine optimum values for the regularization (λ), learning rate (η), number of factors (f), and number of iteration ($iter$) parameters. The role of the regularization parameter is to avoid overfitting to training data while the η is useful, especially when the training data is large. It is typically set to lower values to ensure that the algorithm does not miss a local optimum. As a consequence, the algorithm may take several iterations (i.e., take much time) to converge. Therefore, the selection of η value is important to keep the balance between the recommendation accuracy and convergence rate [LXZ13].

To determine the optimum values for MF parameters, we have used the cross-validation technique [NJTY16]. For that, we have changed one parameter at the time while other parameters were fixed. Every time the MF model was trained using a training set while the test set was used to assess how well the MF model was trained. The MF.Predict models were generated using the MyMediaLite framework [GRFST11]. Table 5.2 shows the optimum values to generate models for both datasets. For each dataset, we have used the same parameters to generate models for time and score.

Table 5.2: Optimum MF parameters for Mooshak and Enki.

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

5.3.1.2 Evaluating 3PL and 2PL IRT models

As explained in section 4.1.2.4, we assume that all LO have different discrimination level, therefore, we use a multidimensional IRT to generate our models. There are two multidimensional IRT models: 2PL and 3PL. In order to select one of these models (2PL or 3PL), we use the analysis of variance (ANOVA) [Gui00] to compare the goodness of fit of models for each dataset. The results of ANOVA are presented in table 5.3. In this table, the Akaike information criterion (AIC) is a measure that given a set of models for a dataset, selects the best-fitting model that uses the fewest parameters. The Bayesian information criterion (BIC), which is also a criterion for model selection among a set of models, penalizes complex models more than the AIC and selects a simpler model (less complex). In both AIC and BIC, a model with the lowest value will be selected [Sta17, BA04]. Therefore, according to our results, we select the 2PL to model our datasets.

Table 5.3: ANOVA results for comparing the generated models using 2PL and 3PL.

Datasets	Models	AIC	BIC
Mooshak	3PL	1124.04	1400.23
	2PL	**1072.47**	**1256.6**
Enki	3PL	461.85	688.04
	2PL	**405.67**	**556.47**

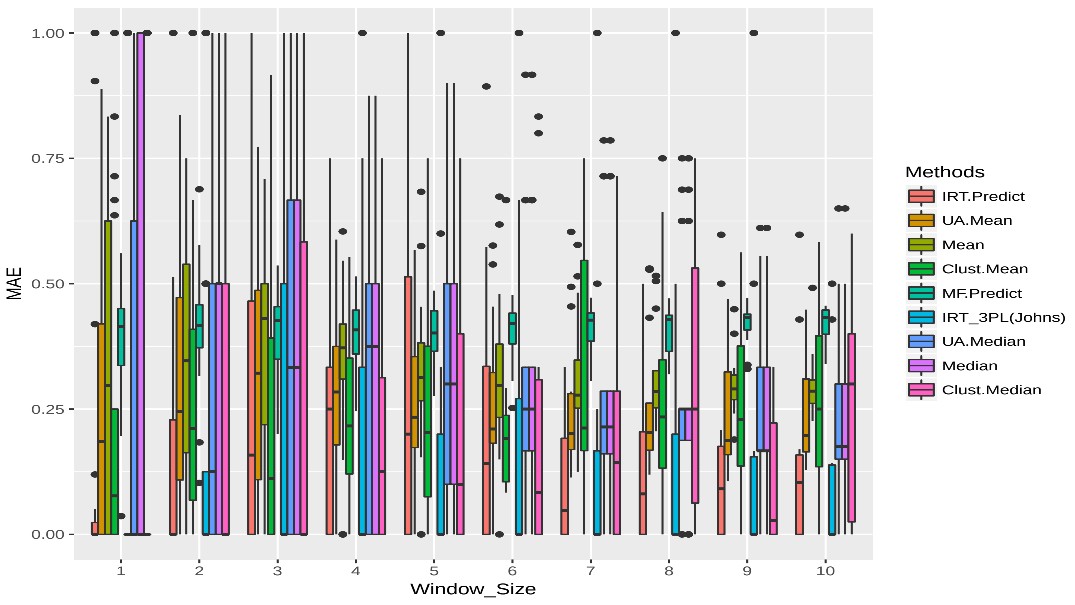
5.3.2 Evaluating estimation approaches

In this section, we evaluate the quality of our estimation approaches. For this purpose, we use the method that we have introduced in section 5.1. Regarding the evaluation results that are shown in figures 5.5a and 5.5b, **MF.Predict** approach performs worse than the others in score estimation. It can have two reasons: MF-based methods require enough data to train; or MF-based methods often perform well for sparse data.

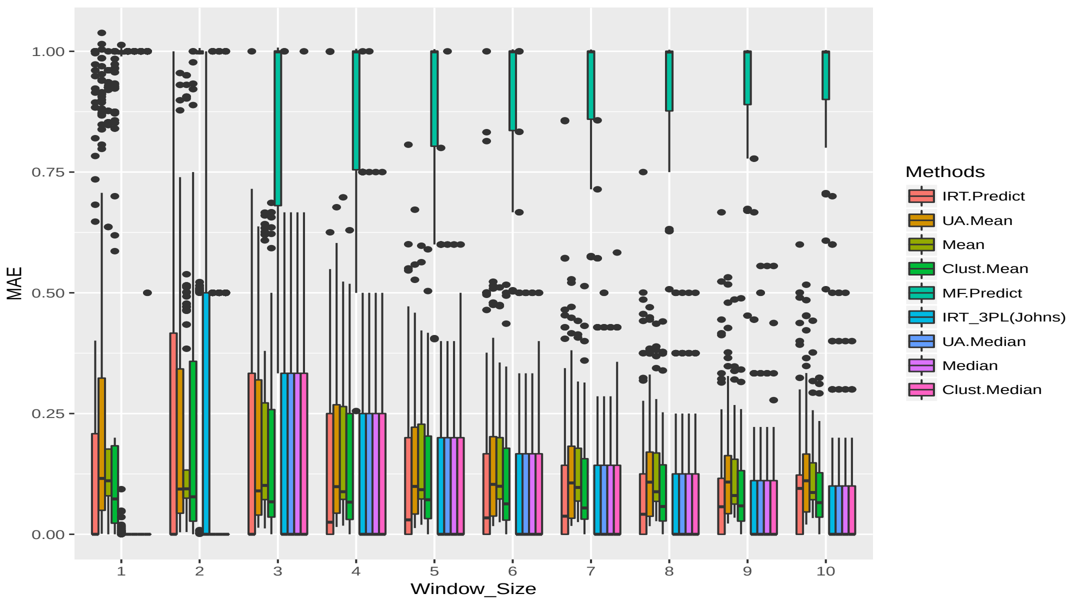
In score estimation, determining an approach which outperforms the other ones is not clear. To this end, we estimated the average of each boxplot (boxplots of figure 5.5) and presented in figure 5.6. As presented in this figure, a few approaches perform better than the others for each dataset. Among these approaches, the **Clust.Median** performs well for both datasets.

In time estimation as in score estimation, we show MAE results in the form of boxplots for the two datasets (figure 5.7). According to our results, the performance of a few approaches are competitive in both datasets. To have a more clear view of the results, we estimated the average of MAE for each boxplot and presented in the form of a line graph (figure 5.8). Regarding the results in figure 5.8, **Median**, **Mean**, **Clust.Median** and **Clust.Mean** outperform the rest of approaches in Mooshak data, while in Enki it is not clear which methods perform better. Among the mentioned methods, **Clust.Mean** and **Clust.Median** are the ones that performed well in estimating score for both datasets. Although **IRT-3PL** also performed well in estimating score for both datasets, it mainly generated more outliers than the **Clust.Mean** and **Clust.Median** (figure 5.5). So, to estimate the learning time and score, we need to select one of the two approaches, **Clust.Mean** or **Clust.Median**. For that, we conduct another evaluation, which is explained in section 5.3.3.

To obtain the results that are shown in figures 5.5 to 5.8, we have used the approach presented in section 5.1. In this method, increasing the size of window results in decreasing the number of observed LO. Subsequently, the number of training cases decreases. Having a larger training set often results in more accurate results but in this case this is not clear because the amount of data is not sufficient to monitor how the size of the window affects the evaluation results.

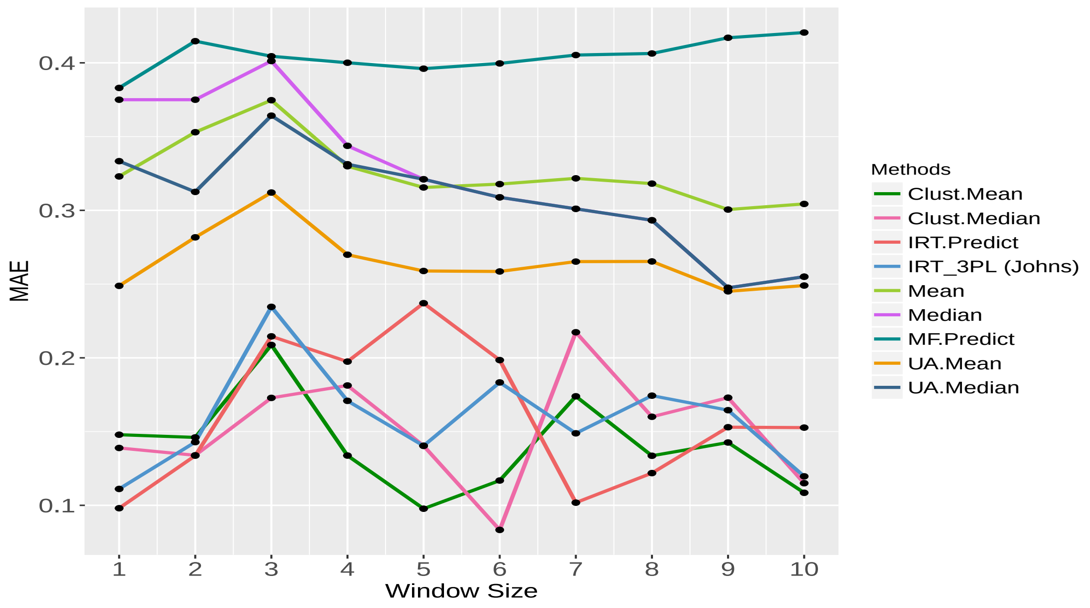


(a) Enki.

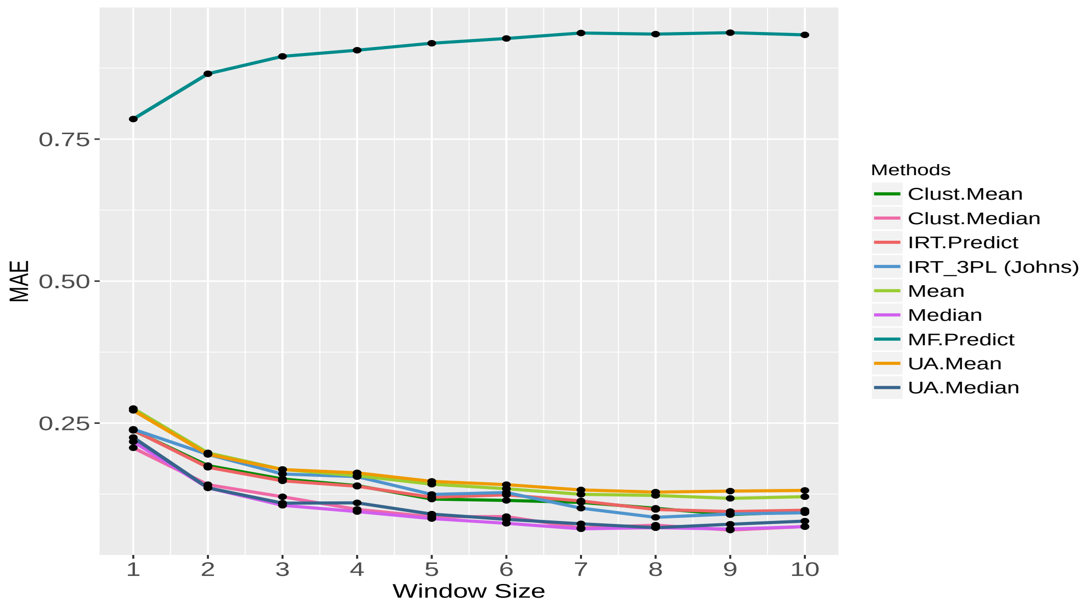


(b) Mooshak.

Figure 5.5: Mean Absolute Error results for score estimation approaches.

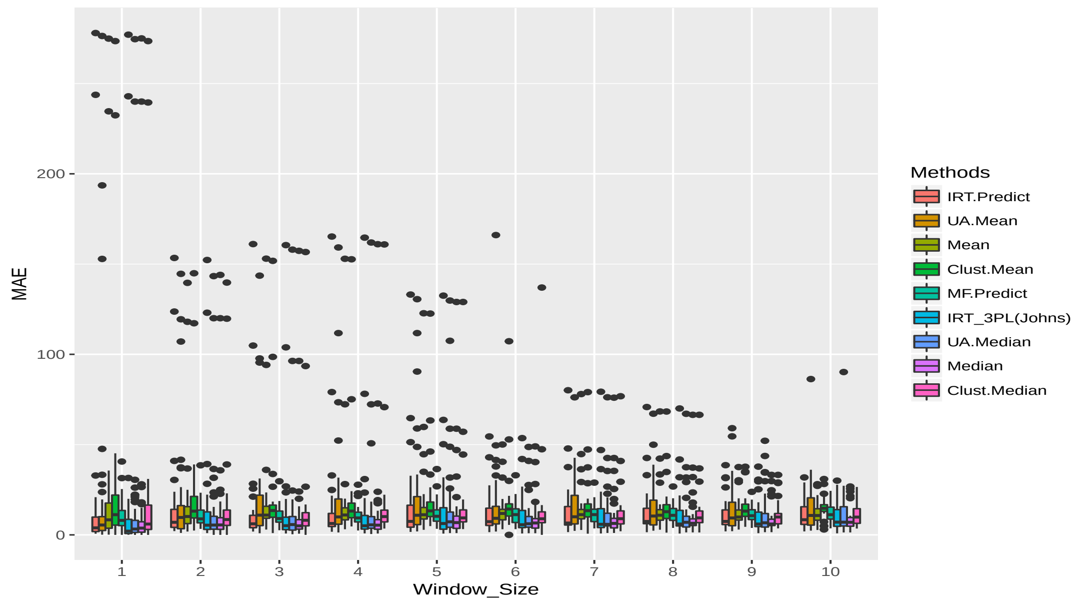


(a) Enki.

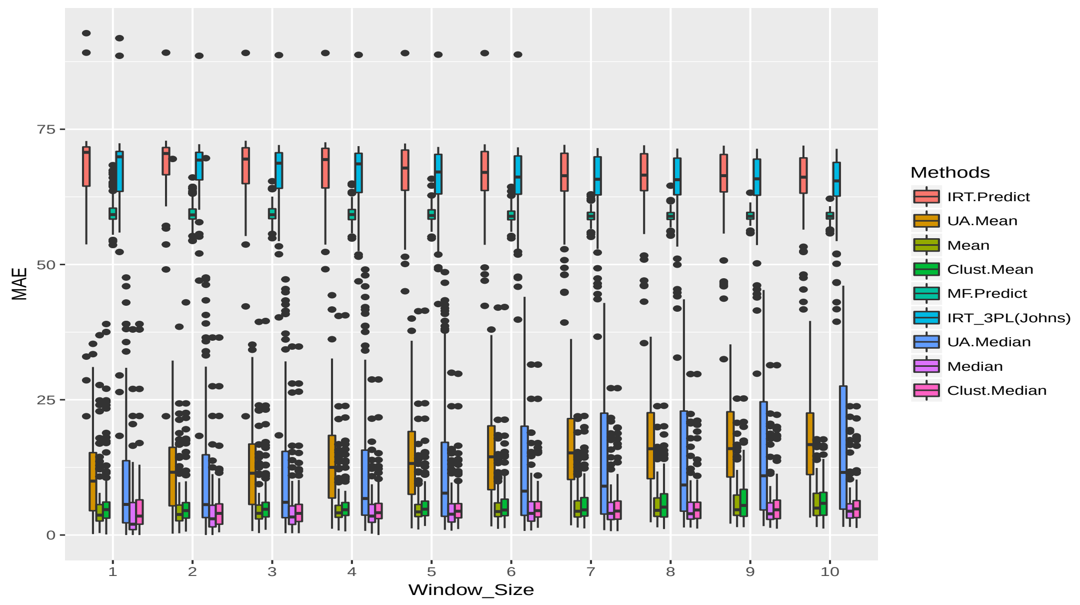


(b) Mooshak.

Figure 5.6: Average MAE for score estimation.

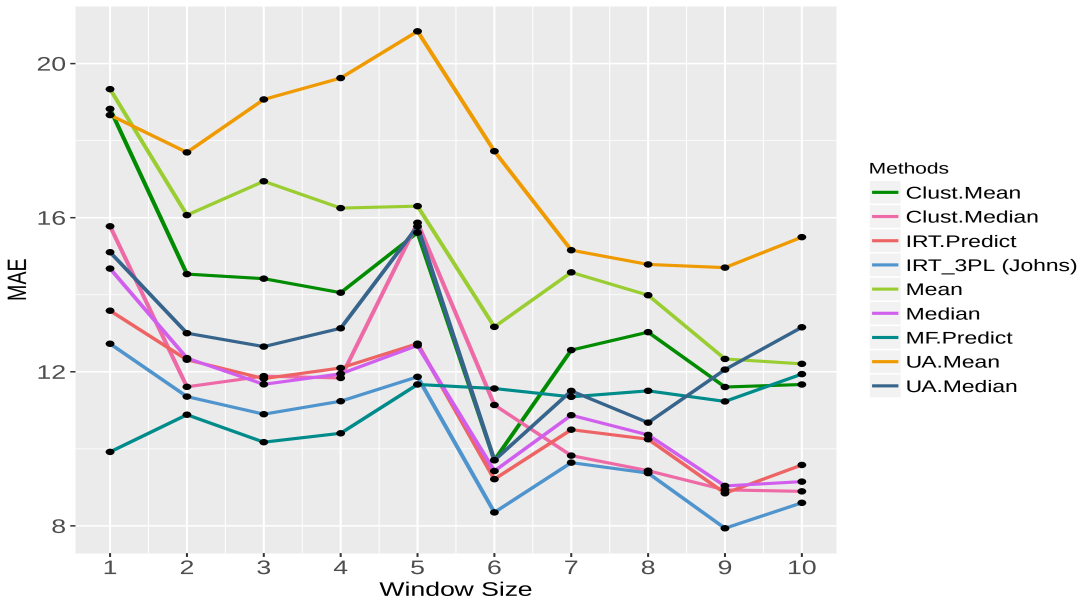


(a) Enki.

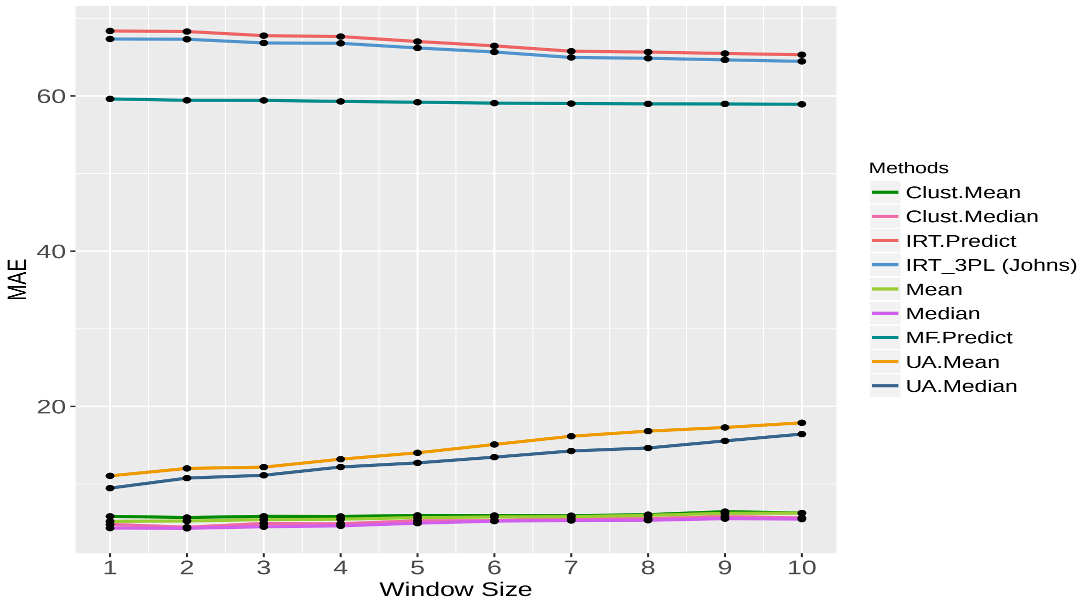


(b) Mooshak.

Figure 5.7: Mean Absolute Error (MAE) for estimating time.



(a) Enki.



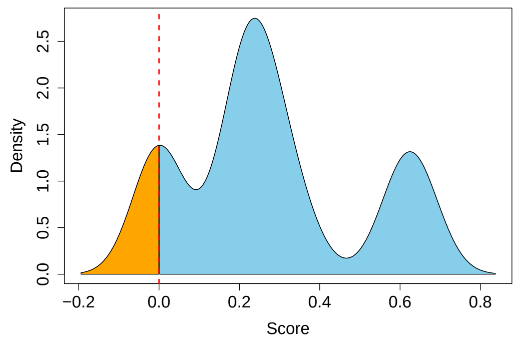
(b) Mooshak.

Figure 5.8: Average MAE for time estimation.

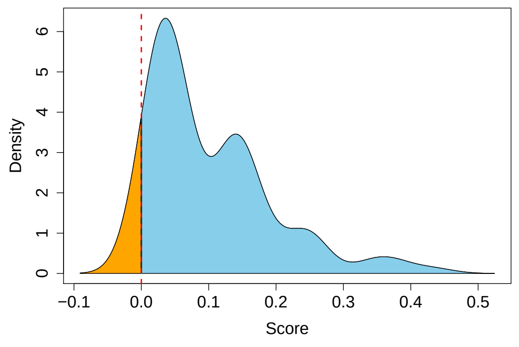
5.3.3 Underestimation and overestimation assessment

As mentioned in the previous section, both **Clust.Median** and **Clust.Mean** are performing well in time and score estimation for both datasets. In order to select one of them, we evaluate them in the case of overestimation and underestimation learning time and score. The main reason of this evaluation is, for time, underestimation implies higher risk than overestimation because a user might not be able to complete a learning path in the estimated time. Contrary to time, score overestimation implies higher risk since a user might not attain the estimated score for a generated path.

The results are shown in the form of probability density for the two datasets (figures 5.9, 5.10, 5.11, and 5.12) to show the percentage of overestimation (estimated values> real values) and underestimation (estimated values < real values) of the selected methods.

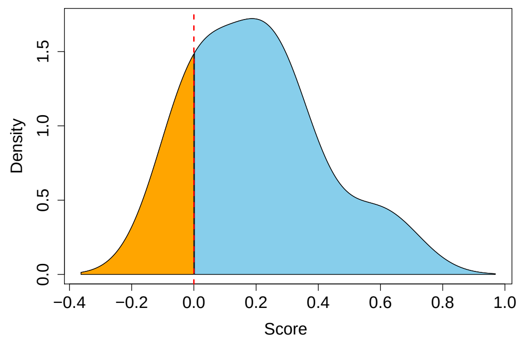


(a) Enki.Overestimation:88%, Underestimation:12%

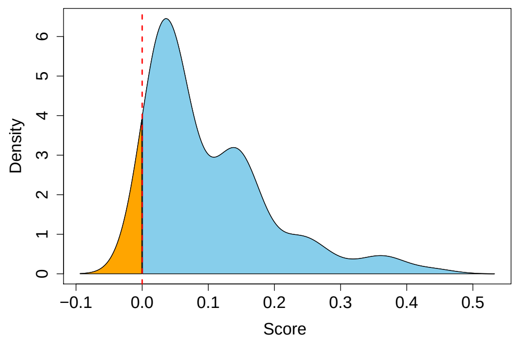


(b) Mooshak.Overestimation:90%, Underestimation:10%

Figure 5.9: Score overestimating and underestimating using **Clust.Median** approach.



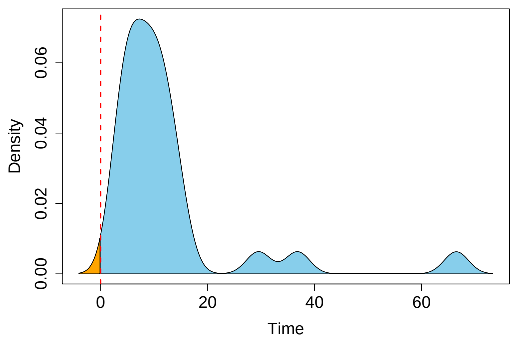
(a) Enki.Overestimation:79.8%, Underestimation:20.2%



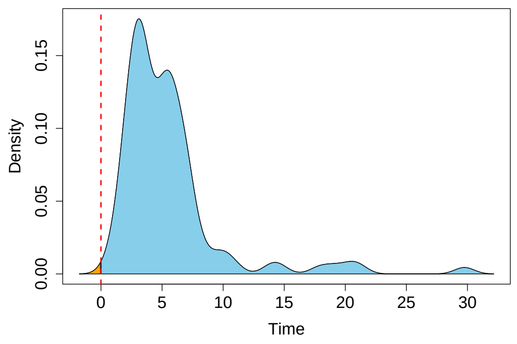
(b) Mooshak.Overestimation:90.9%, Underestimation:9.1%

Figure 5.10: Score overestimating and underestimating using **Clust.Mean** approach.

In these figures, the blue highlighted part presents the overestimation region while the orange-colored part shows the underestimation region. Regarding the results, although both methods perform well in time estimation, **Clust.Mean** performs better than **Clust.Median** in score estimation (Underestimation error in Enki: Clust.Median:12 %, Clust.Mean: 20.2%, in Mooshak: Clust.Median: 10%, Clust.Mean: 9.1%). Therefore, according to the results which are presented in this and previous sections, we select **Clust.Mean** to estimate learning time and score for users. Complementary overestimation and underestimation results are mentioned in appendix B.

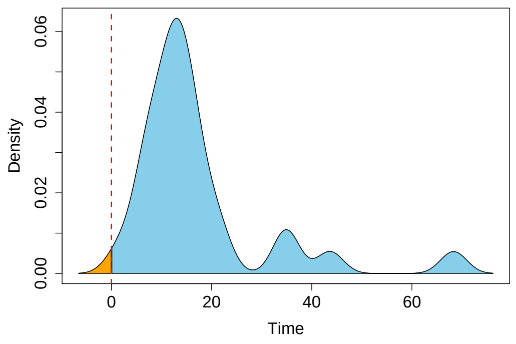


(a) Enki.Overestimation:98.8%,Underestimation:1.2%

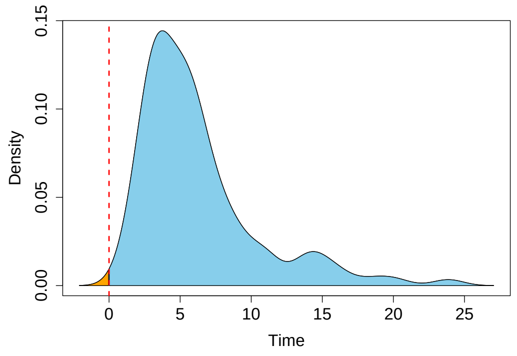


(b) Mooshak.Overestimation:99.7%, Underestimation:0.3%

Figure 5.11: Time overestimating and underestimating using **Clust.Median** approach.



(a) Enki.Overestimation:98.8%,Underestimation:1.2%



(b) Mooshak.Overestimation:99.6%, Underestimation:0.4%

Figure 5.12: Time overestimating and underestimating using **Clust.Mean** approach.

5.4 Summary

In this chapter, we evaluate the quality of nine different methods that are introduced for estimating the learning time and score for the learning paths (explained in chapter 4). To this end, we first detailed our evaluation methodology that we have used to assess our estimation methods. Next, we explain two datasets that are used for the evaluation. One of these datasets is from a C# course which is taken from Enki system, and the second one is for a database course that is taken from Mooshak. These datasets are also statistically analyzed for providing a better understanding of them. Finally, we compare the performance of the estimation methods using MAE and Average MAE, and also assess them in terms of overestimating and underestimating time and score for the learning paths.

Chapter 6

Recommender Evaluation

Our goal is to generate learning paths that maximize a user's score under a given time restriction. For this purpose, we introduced two approaches to generate learning paths considering the knowledge background and time restriction of a user. The first approach uses a one-layer course graph to generate paths while the second one is based on a two-layered course graph. The second approach is introduced to cover the problems of the first approach. In the previous chapter, we assessed the quality of different methods for estimating learning time and score for the paths. Based on experimental results, **Clust.Mean** was selected for time and score estimation.

In this chapter, we aim to assess the quality of the recommender approach, which is based on a two-layered course graph. This approach is evaluated since it is an enhanced version of the other approach (using one-layer course graph). For that, we developed our approach and integrated it with an E-learning system, called Enki, to generate learning paths for target users. Therefore, we initially describe the architecture of our system, detail its design and implementation. We then explain an experiment that we have conducted to assess the quality of the recommender. For this purpose, we selected 32 participants and divided them in two groups (a control group and an experimental group) to attend a short course. Both groups used two different versions of the same system (Enki). The experimental group used the version that guided participants using recommendations while control group used the version of the system that delivered LO without using recommendations. Finally, we compared the performance of the two groups.

6.1 E-Learning System Integration

As explained previously, our approach (using a two-layered course graph) is integrated with a system (Enki) in order to generate and recommend paths that maximize users' score while satisfying their time constraints. Our system blends learning and assessment, and is able to present different content formats (e.g. hypertext, video, etc.) and exercises. It is also an adaptive system, which adjusts a generated path according to the users' feedback. In addition, this system includes interfaces for teachers to author and manage both exercises and content, as well as to browse assessment results and users' profiles. The next subsections present the architecture of our system and its main components, and describe its implementation.

6.1.1 Architecture

Our approach is embedded in the Enki system. Enki is a part of Mooshak 2.0 [LS03], a web environment for automated assessment in computer science. Mooshak has interoperability features that enable it to interact with other E-learning tools such as Learning Management Systems (LMS). Enki takes advantage of Mooshak 2.0 to have a pivotal role in a network of E-learning systems, coordinating the communication with all external components as depicted by the Unified Modeling Language (UML) components diagram in figure 6.1.

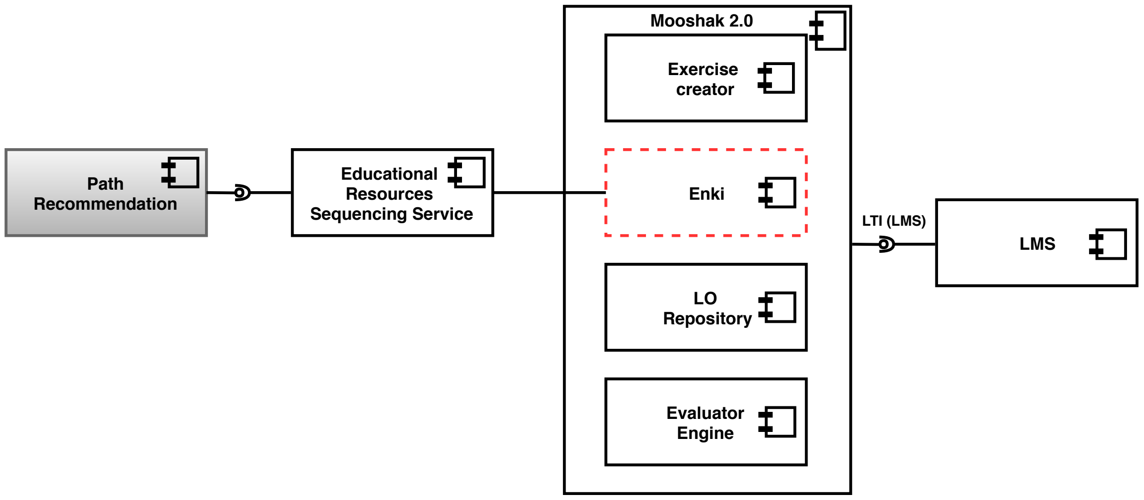


Figure 6.1: Components diagram of the network of Enki where Mooshak 2.0 acts as a tool provider for an LMS. The gray component (recommender) is implemented and added to Enki.

The next sub-subsections explain in detail the types of systems and tools that build the network presented in figure 6.1.

6.1.1.1 Learning Management System (LMS)

An LMS is a software application for administration, documentation, tracking and reporting; used in training programs and classrooms [Ell09]. Typically it is used by two types of users: learners and teachers. The learners apply the LMS to plan their learning experience and to collaborate with their colleagues, while teachers deliver educational content and track, analyze and report the learners' evolution within an institute/organization.

An LMS often plays a central role in an E-learning architecture, but it still cannot be isolated from other systems in an educational institution. Therefore, the potential for interoperability is an important aspect of an LMS [LQ11]. To this end, we integrate our E-learning recommender based on an LMS (e.g. Moodle). For this reason, our recommender benefits from the interoperability mechanisms inherited from Mooshak 2.0 to provide authentication directly from the LMS and to submit grades of exercises to the LMS, using the Learning Tools Interoperability (LTI) specification.

6.1.1.2 Path Recommender (PR)

The Path Recommendation (PR) generates paths regarding knowledge background (i.e., starting point) and time restriction of a user. It performs the following tasks:

1. Receive time constraint and the starting point (sp) of a user.
2. Generate a path that maximizes a user's score under his/her available time.
3. Estimate time, score, and the probability of error for the estimated time and score for a path.
4. Generate auxiliary LO for a lesson whenever it is required.

6.1.1.3 Educational Resources Sequencing Service (ERSS)

The role of ERSS is to present a sequence of concepts that matches the learning goal and then select learning resources for each concept of that sequence. The selected ERSS is Seqins, which contains a simple and flexible sequencing model that fosters users to learn at different rhythms [QLC14]. Precedence among units of a course (LO and lessons), users' progress and assessment results are delivered to the Seqins by Enki, and Seqins generates an XML file which includes the representation of the resources for a target user. This XML file is generated using the results of the Path Recommendation (explained in section 6.1.1.2).

6.1.1.4 Evaluator Engine (EE)

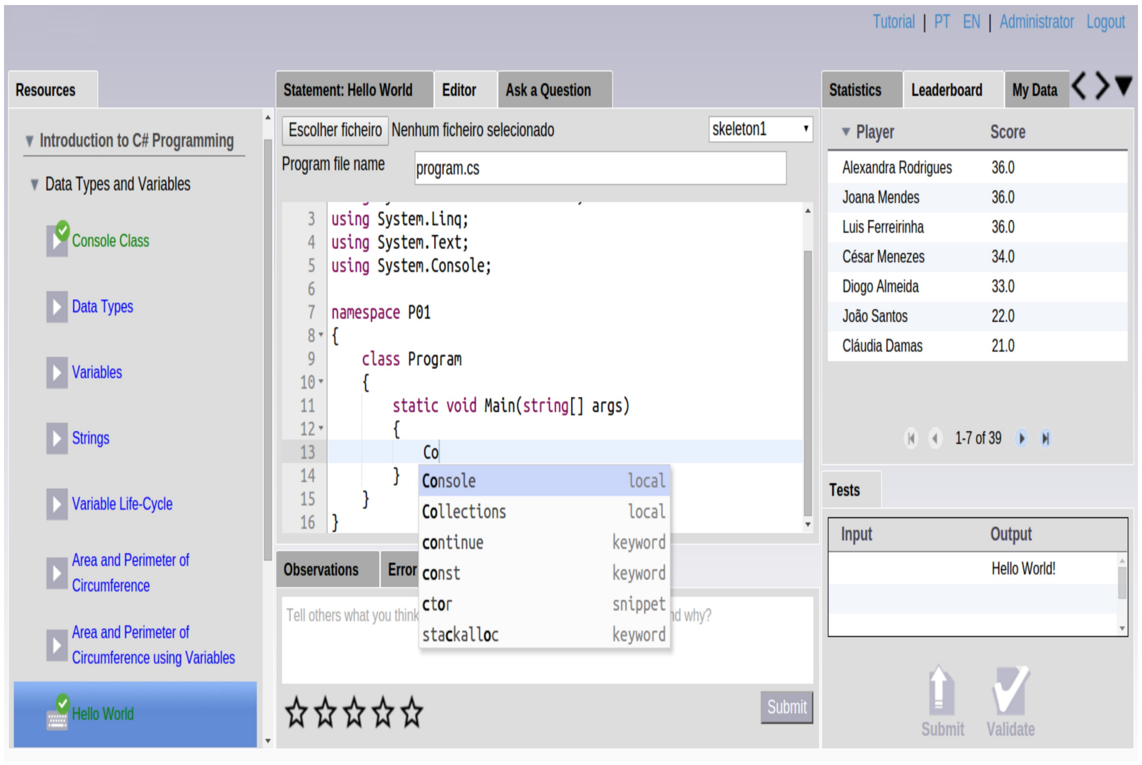
The main goal of EE is to mark and grade exercises (i.e., evaluative LO). It is provided by Mooshak 2.0 and performs four tasks:

1. Receive a reference of an exercise (evaluative LO) and a reference of a user that submits an answer (e.g. a code) as well as the answer of the user.
2. Load the exercise from the LOR (LOR is explained in section 6.1.1.6) using the given reference.
3. Compile the solution and run the tests, related to the exercise, against the answer of the user.
4. Generate an evaluation report with feedback and, possibly, corrections.

The main feature of EE is the automatic evaluation of exercises, providing better feedback and support for different exercise types.

6.1.1.5 Exercise Creator (EC)

An EC enables teachers to generate a complete exercise package, including a statement, a solution, tests, skeletons, and a manifest file describing the contents of a package. The generated package needs to follow the same package characteristics as the LOR (LOR is explained in section 6.1.1.6). Similar to EE, the EC is also provided by Mooshak 2.0.



```
using System.Linq;
using System.Text;
using System.Console;

namespace P01
{
    class Program
    {
        static void Main(string[] args)
        {
            Co
        }
    }
}
```

```
Console      local
Collections  local
continue     keyword
const        keyword
ctor         snippet
stackalloc   keyword
```

Player	Score
Alexandra Rodrigues	36.0
Joana Mendes	36.0
Luis Ferreirinha	36.0
César Menezes	34.0
Diogo Almeida	33.0
João Santos	22.0
Cláudia Damas	21.0

Input	Output
	Hello World!

Figure 6.2: Interface of our recommender for users.

6.1.1.6 Learning Objects Repository (LOR)

A Learning Objects Repository (LOR) stores educational resources (LO) and enables users to share, manage and use them. These resources (or LO) are small, self-contained and reusable learning units which, typically, have additional metadata to catalog and search them. Similar to EE and EC, LOR is also provided by Mooshak 2.0.

6.1.2 Graphical User Interface

Our recommender is implemented using Google Web Toolkit (GWT), an open source software development framework that allows a fast and easy development of AJAX applications in Java [HT07]. One of the special components of our recommender is the user interface, shown in figure 6.2, which emulates an integrated development environment (IDE). Through this interface, users can rearrange panels and tabs to their requirements, by drag-and drop and resizing features, which are provided by the Enki.

In figure 6.2, every level of the Resources tab, such as Data Types and Variables, presents a lesson that includes several LO. These LO can have various types, such as text (HTML or PDF), multimedia (video), and exercises (evaluative LO). A recommended LO is initially colored with yellow and has a star over its icon. Once a user visits a recommended LO, the yellow color turns into green.

6.2 Experimental Methodology

After implementing our recommender and embedding it in Enki, we designed a short course on the C# programming language and offered it to participants using two different versions of the same system. One of the versions used our recommender approach and the other version delivered the course to participants without any recommendation.

Learners use a learning system when it makes the learning process more efficient, effective and attractive. These three are the common measures in educational researches [VMIB13]. Effectiveness indicates the number of correctly completed LO and lessons by participants during the learning process. Efficiency implies the time that learners spend to obtain their goals (study time). Finally, the last measure (attractiveness) shows the learners' satisfaction of an E-learning system. In the following, we use these measures to assess the performance of both groups on the course and the final exam and to validate the following hypotheses:

1. Our recommender promotes a higher lesson coverage than the baseline (assessing the effectiveness and efficiency on the course).
2. Users of our recommender get better scores on the final exam than the ones without recommendation (assessing the effectiveness on the final exam).
3. Considering the time of the course, users of our recommender are more satisfied with their scores for the exam than the ones without recommendation (assessing the attractiveness).

The level of the C# course was basic and it included 5 lessons, which were Data types and variables, Conditions, Loops, Arrays, and Strings, and contained 59 LO. The experiment was conducted for 32 participants in June 2018. In our experiment, participants were separated in two groups: 16 participants in a control group and 16 participants in an experimental group. In the control group, participants followed a predefined order (designed by a course expert) for lessons and LO and did not receive any recommendation or personalized guidance through the course. In the experimental group, participants used our recommender. To assign the participants to the experimental and control groups, the standard practice is to assign them randomly. In our experiment, we had a small number of participants while they were very diverse and they could not attend at the same time (32 participants attended in 5 different times). Therefore, in order to be sure that both groups were similar or had negligible differences, we selected and assigned participants to groups manually. For this purpose, we have considered three criteria for assigning participants to the groups: (1) level of programming skill, (2) gender, and (3) familiarity with the Portuguese language. The last criteria is considered (familarity with portugesse language) since the course was in Portuguese, therefore, we assumed knowing Portuguese can affect the performance of the participants. Statistics of both groups are presented in table 6.1.

Since the experiment was conducted in a controlled environment, participants in both groups did not take any other parallel courses or activities. Also, they were asked not to use any extra learning materials or help except using google translator, since some of the participants were foreigners living in Portugal and were not familiar with the Portuguese language for understanding the questions (questions were in Portuguese).

The experiment consisted of three main phases. In the first phase, participants from both groups were

Table 6.1: Statistics of experimental and control groups. In this table, we also test the homogeneity of two groups. For that, we estimated the p-value using the Fisher exact test [RR95, McD09]. The estimated p-values do not reject the null hypothesis, which is "two groups are similar".

	Gender		Portugese Level		Coding Level		
	M	F	Know	Not Knowing	Not Know coding	Know coding	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		

asked to follow the course for 2 hours. In the second phase, after taking the course, all participants were asked to attend a small final test (equal for both groups) to assess their knowledge on the lessons that they learned during the course. The duration of the test was 1 hour and it included five practical questions that the participants were required to provide correct compilable solutions for them. In the last phase, participants in both groups answered two different short questionnaires to provide their opinions about the baseline (a version of the system that did not use the resommendations) and the recommender (2 questions for the control group, 5 questions for the experimental group). The final test and the questionnaires were integrated into the system. Figure 6.3 shows a graphical view of our experimental procedure.

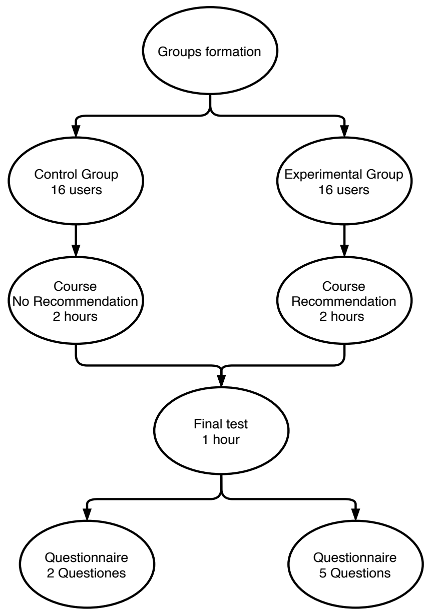
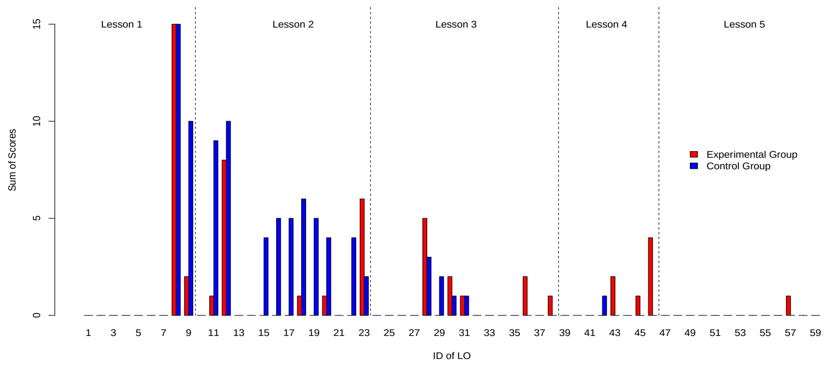

Groups formation
Control Group 16 users
Experimental Group 16 users
Course No Recommendation 2 hours
Course Recommendation 2 hours
Final test 1 hour
Questionnaire 2 Questiones
Questionnaire 5 Questions

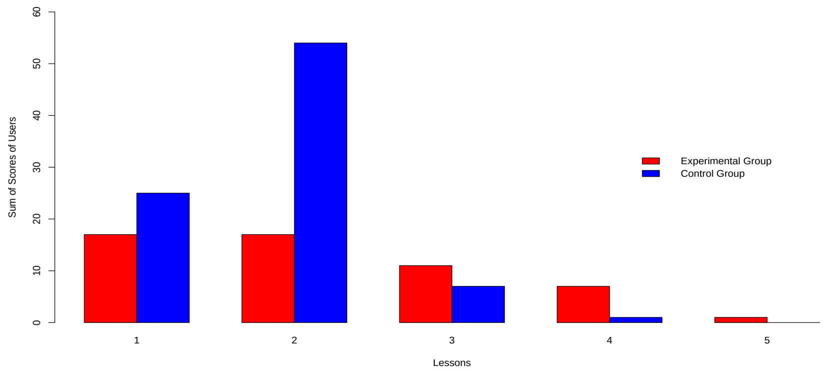
Figure 6.3: Graphic depictions of the experimental procedures.

6.3 Course Performance

To assess which system (baseline or our recommender) promotes a higher lesson coverage, we monitored and compared the completion of LO and lessons by the participants in both groups (assessing the effectiveness). For that, the grades for LO and lessons were made after completing each of them. The experiment results shown in figure 6.4 indicate that within the available time for the course, participants in the control group could complete more LO from the first two lessons while the participants in the experimental group were able to learn and complete more lessons than the participants in the control group. One possible reason is, in the control group a participant received the LO sequentially regardless of the fact that he/she might not need to learn all LO to learn a lesson. In this group, a participant learned and answered the LO as he/she received them while a participant in the experimental group received and completed only those LO that were necessary for him/her to learn a lesson in the available time. Therefore, the participants in the experimental group were able to complete more lessons than the participants in the control group.



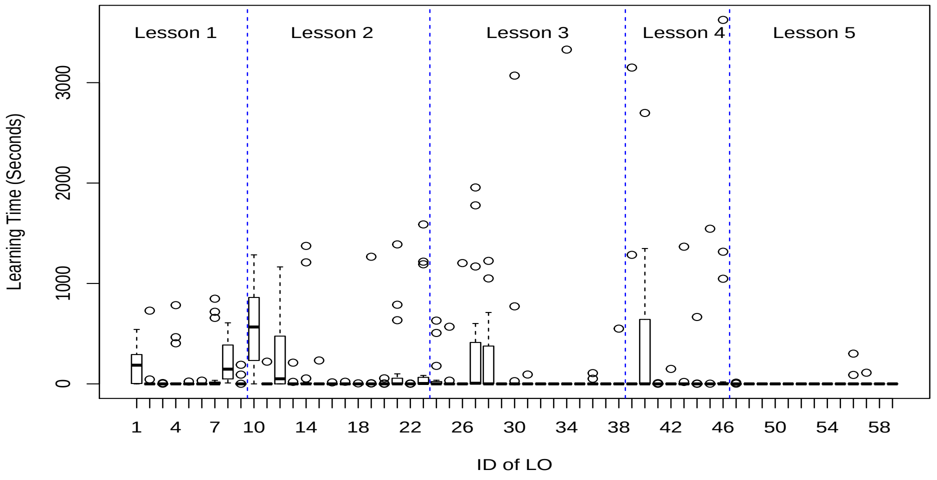
(a) Sum of participants' scores on each LO.



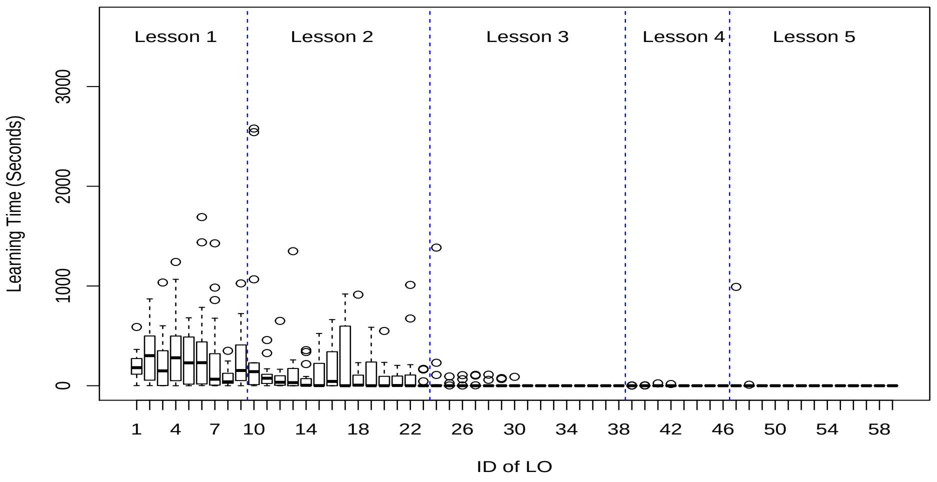
(b) Sum of participants' scores on each lesson.

Figure 6.4: Comparing the effectiveness of both groups on different LO and lessons.

In addition, we assessed both groups in terms of efficiency for the course coverage. For this purpose, we calculated the time that the participants of both groups have spent on each LO and lesson. The results are presented in figures 6.5 and 6.6 . Regarding the results, the participants in the control group could mostly focus on the first two lessons while the participants in the experimental group were able to learn four lessons within the same amount of time. Therefore, according to the results presented in figures 6.4 to 6.6, the efficiency and the effectiveness of participants in the experimental group are highly enhanced in compare to the participants in the control group.



(a) Experimental group.



(b) Control group.

Figure 6.5: Comparing the efficiency of both groups on different LO and lessons.

In order to assess how significant is the difference between the course coverage of two groups (first hypothesis mentioned in section 6.2), we estimated the p-value [WL+16] using the learning time of the participants (figures 6.5 and 6.6 shows the groups' learning time on each lesson and LO).

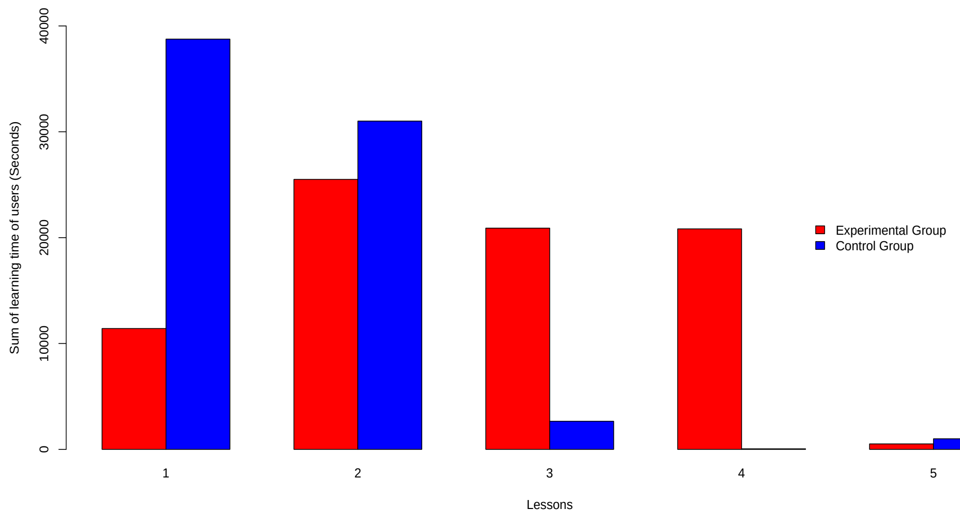


Figure 6.6: Sum of time that participants spent on each lesson.

To this end, we counted the lessons that a participant could spend more than 5 minutes for them. The threshold was set to 5 minutes since for each lesson a participant needed to learn two LO, an expository one (watching a video around 3 to 4 minutes) and answering an evaluative LO (at least 1 minute). Then, we made two samples of 16 values, each obtained by counting the lessons accessed by each participant. Finally, we compared two samples using the Wilcoxon-Mann-Whitney [Neu11] and Kruskal-Wallis tests [MN10]. These tests can be used if the two samples are independent and the variables are continuous or at least ordinal. The obtained p-value **0.001024** (same results for both tests) strongly rejects the null hypothesis for a significance level of 0.1, which is "our recommender and the baseline systems promote similar course coverage".

For the first hypothesis, we also did the same test using the learning score of participants (in figure 6.4 the groups' scores are compared). For that, we counted, for each participant, how many lessons had at least one LO graded. We then generated two samples of 16 values, each obtained by counting the lessons graded by each participant. The estimated p-value **0.07727** (same results for both tests) rejects the null hypothesis. Therefore, according to the obtained p-values we can conclude that the efficiency and the effectiveness of our recommender are significantly higher than the baseline for a significance level of 0.1.

6.4 Final Exam Performance

Similar to the course performance assessment, in the final exam, we also analyzed and compared the effectiveness of the participants in both groups. In the final exam, the time that participants spent on each question or the whole exam was not of our interest since in all educational exams participants have a similar amount of time and they are allowed to allocate their time for the questions as they want.

The exam was similar for both groups and it included 5 coding questions (one question per lesson). The participants were requested to provide compilable solutions that gave the correct results. The exam duration was 1 hour and the participants had no restrictions on how to devote their time to different questions. In order to control confounding variables related to the participants' performance, no other C# lessons were taught during the exam.

After the final exam, we assessed if the participants in the experimental group obtained a higher score than the participants in the control group. This comparison of the results allowed us to assess not only the difference between the gained knowledge but also to determine how much the recommender was successful in achieving the main goal of this thesis.

Regarding the results presented in figure 6.7, all participants in the experimental group could answer the first question correctly. They also had a better performance than the participants in the control group in answering all questions except the second question. The reason is, each question was associated with a lesson of the course and the participants in the control group could learn the second lesson better than the participants in the experimental group (figure 6.4).

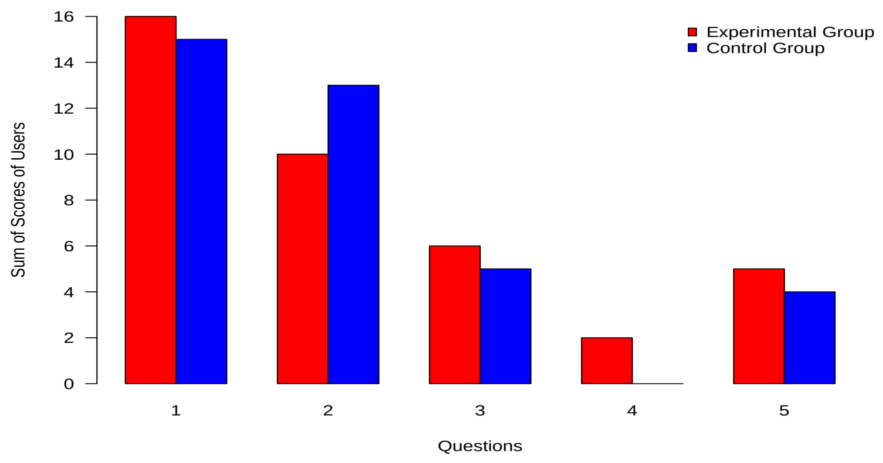


Figure 6.7: Sum of the participants' scores on each question.Correct answer=1, otherwise 0.

The minimum and the maximum possible scores for the exam (for a participant) were 0 and 5, since there were five questions in the final exam, and a participant gained 1 score by providing a correct answer for a question. Figure 6.8 shows the frequency of the final scores that the participants of both groups could obtain from the final exam. The worst result was made by a participant from the control group, while a participant from the experimental group obtained the best possible score. Apart from that, more participants in the experimental group got "4" than the participants in the control group but the number of the participants that their final scores were equal or less than 2 (9 participants for both groups), and more than 2 (7 participants for both groups) is equal for both groups. Therefore, although the participants in the experimental group could complete the exam with better scores in 4 and 5, if we consider 3 as the minimum score to pass the exam, the participants in both groups had almost a similar performance.

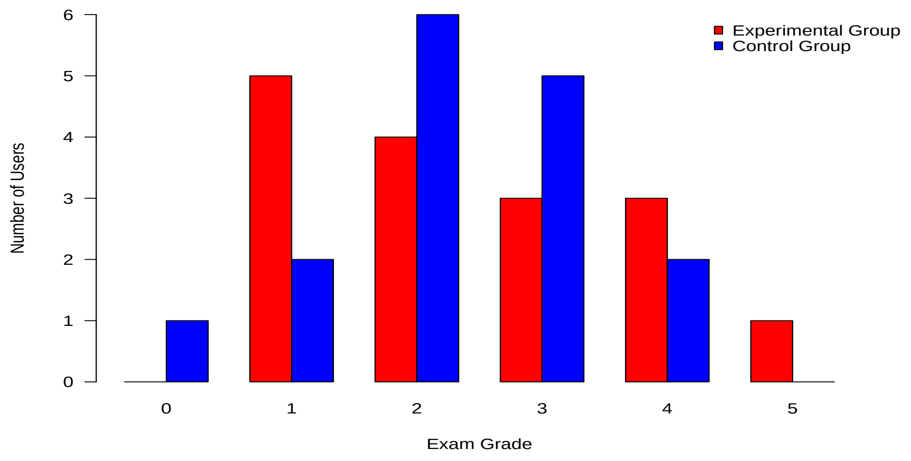


Figure 6.8: Frequency of final scores for participants in both groups.

In order to evaluate how significant is the difference between the obtained final scores of the participants in two groups (second hypothesis in section 6.2), we compared two groups using their final grades on the exam. For that, we applied a similar methodology as we used for comparing the course coverage of two groups (using score's data) since the samples had the same structures and nature. The obtained p-values using the Wilcoxon-Mann-Whitney and Kruskal-Wallis tests are equal (**0.9571**). A brief look at the results presented in figure 6.8 shows that although participants of the recommender obtained higher scores than the participants in the control group, this difference is not statistically significant. We believe that the relatively small size of the samples may partly explain this lack of observable significance.

6.5 Participants' Satisfaction Assessment

The participants' satisfaction was evaluated in two steps. In the first step, after completing the final exam, we provided a short questionnaire to collect the participants' opinions about the quality of our recommender and the generated recommendations. This questionnaire was only for the participants in the experimental group since they were the ones that used the recommender. Participants were asked to rate the statements using a five-point Likert scale. This questionnaire had the following statements:

1. Recommendations were generated quickly.
2. Recommendations were helpful for completing the course.
3. The additional generated recommendations for a lesson were helpful to understand the lesson and answer the questions.

We limited our questionnaire to these statements since any other statement could be related to the Enki and not to the recommender (our recommender is integrated with Enki). Therefore, in order to not having misleading data, we only focused on the quality of the recommender and distributed the questionnaire among the participants of the experimental group.

Figure 6.9 shows the participants' opinions about the quality of the recommender and recommendations. As presented in this figure, almost all participants had positive opinions (agreed or strongly agreed) about the quickness of the recommender. In addition, 50% of the participants were satisfied with the relevancy and usefulness of the recommended LO (expository and evaluative) in completing the course (question 2). Also, 75% of the participants "agreed" or "strongly agreed" with the usefulness of the generated auxiliary LO for each lesson (question 3).

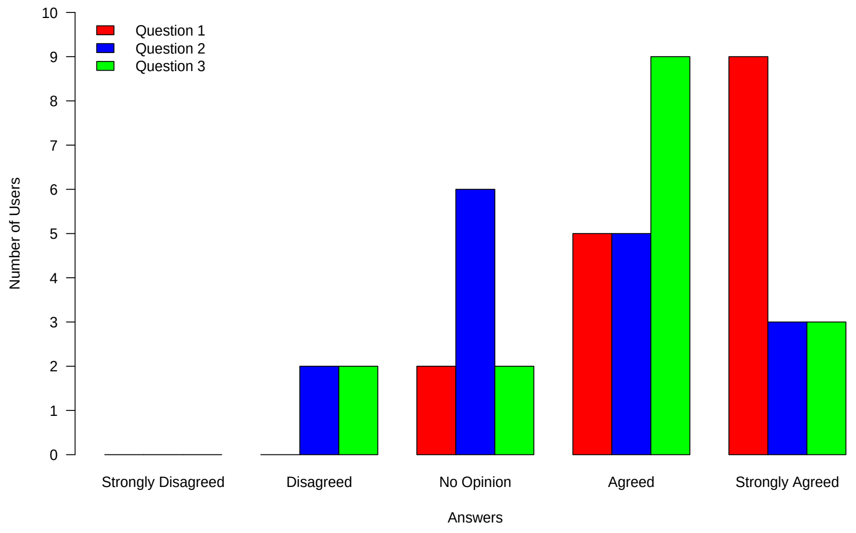


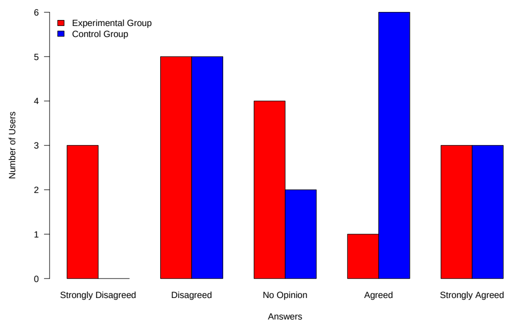
Figure 6.9: Participants' opinions about the quality of the recommender.

The main goal of our recommender is to maximize the expected score of a user in a restricted learning time. So, in the second step, we collected the participants' opinions about the success of the recommender and the baseline (not using the recommendation) in achieving the mentioned goal. To this end, we designed two statements and asked the participants from both groups to rate the statements using a five-point Likert scale. The two statements were:

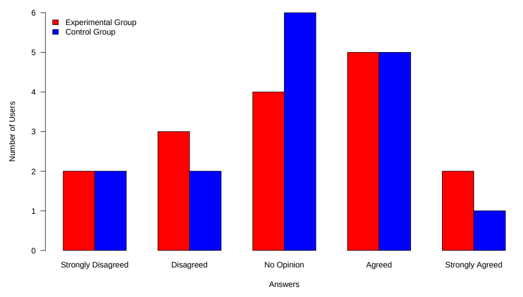
1. I could understand most of the course within time.
2. Regarding the time that I have spent on the course, I am satisfied with my final score.

Figure 6.10 shows the participants' opinions about the mentioned statements. The results of the first statement show that the participants in the control group were more satisfied with the amount of the course that they could learn (course coverage) than the participants in the experimental group. It is the opposite of the results that are presented in figures 6.4 to 6.6, which show that the participants in the experimental group had a higher course coverage than the ones in the control group. One reason can be that the participants were not careful enough in providing their opinions for the first statement and they answered the questionnaire mechanically.

The results for the second statement are almost similar for both groups. These results comply with the results that are presented in figure 6.8. Figure 6.8 also shows similar results for both groups in the final exam. Also, as a reason for having such results for the second statement 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 might give a high rate to this statement. Therefore, the third hypothesis (mentioned in section 6.2) could not be tested because of having the mentioned problem in the data collection process that compromised the conclusions.



(a) First statement.



(b) Second statement.

Figure 6.10: Participants' opinions about the success of the recommender and the baseline in achieving the goal.

6.6 Experiment Observation

Up to now, we have explained how we performed an experiment to evaluate the quality of our recommender. In addition to assessing our recommender, this experiment allowed us to study participants' behaviors (in the experimental group) and monitor how they interacted with our recommender. During the observation, we could recognize four different types of participants in the experimental group. These groups were:

1. Advanced participants: These participants knew how to code, and asked few questions from us during the course and exam. They were able to follow the course and answer the questions with minimal guidance. They almost completed all the lessons of the course and got the best scores for the exam (i.e., 4 and 5).

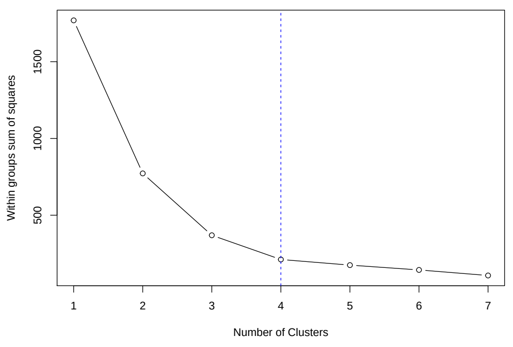
2. Good participants: These participants had some experience in coding but they still had some difficulties. These users asked a few questions through the course but they managed to complete most of the lessons. They mainly got good scores for the final exam (i.e., more than 3).

3. Keen participants: They knew the basics of coding. During the course, they frequently asked questions (needed much help), and they were keen to answer the questions of the course and the final exam. In spite of their attempts, they were mostly not able to complete most of the lessons and their scores for the final exam were not good (i.e., less than 3).

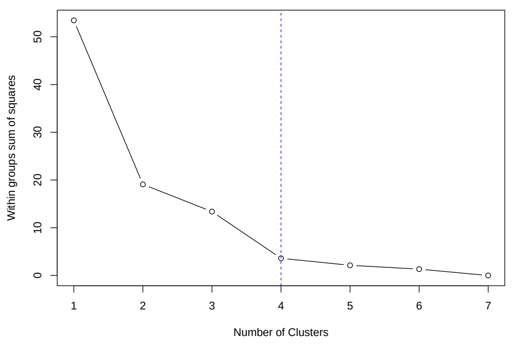
4. Novice participants: They were not familiar with coding. During the course, these participants usually did not ask any question and mostly spent their time on the first lesson. They often got the minimum scores for the final exam (i.e., 0 and 1).

In order to confirm the validity of this observation, we used a clustering technique on the obtained scores by the participants through the course and the final exam (scores of the experimental group). To this end, we followed a data preparation method that was used in [BGJG14]. Hence, we initially accumulated the participants' scores over the LO (for the course and final exam). We then proceeded to cluster analysis, by using accumulated scores on LO as attributes, to group participants by similarities of score acquisition. K-means [HW79] was the selected clustering algorithm since it has a linear complexity, easy implementation and interpretation [RM05].

Our selected method to validate the number of clusters was the elbow method. The idea of the elbow method is to run K-means clustering on the data for a range of K values (e.g. K from 1 to 7), and for each value of K calculate the sum of squared errors (SSE). The results for finding the optimal number of the cluster among the participants in the experimental group is presented in figure 6.11.



(a) Number of clusters on course.

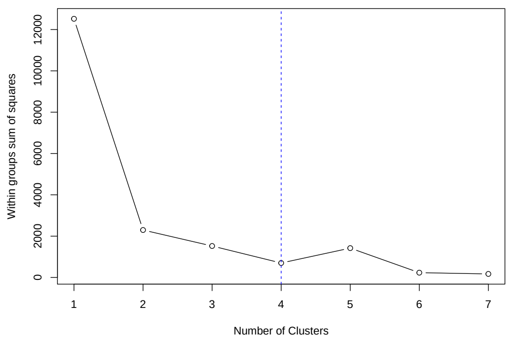


(b) Number of clusters on exam.

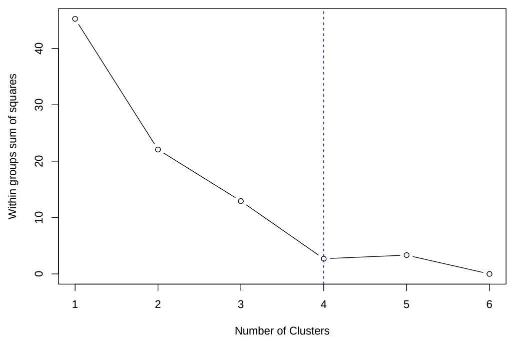
Figure 6.11: Optimal number of clusters for the participants in the experimental group.

According to the results presented in figure 6.11, "4" is one of the most promising values for the number of the clusters. Interestingly, our results are compatible with the results that are presented in [BGJG14]. In this paper, considering the performance of the participants on a course, they are classified into four different groups. The first group of participants are "Achievers", which their performance matches the performance of the participants in our advanced group. "Regular Students" are the second best group of participants that they performed well on the course but not as good as the "Achievers". This group can be compared with our "Good participants" group, which had some difficulties to answer the questions. Next group is called "Halfhearted Students" that their scores are less than the scores of "Achievers" and "Regular Students". Their performance is compatible with the participants in the "keen group". Finally, the participants in the novice group have similar performance to the "Underachievers" group presented in the paper.

In order to assess if these groups were evenly distributed among the experimental and control groups, we performed the same clustering technique on the obtained scores of the control group for the course



(a) Number of clusters on course.



(b) Number of clusters on exam.

Figure 6.12: Optimal number of clusters for the participants in the control group.

and the exam. Similar to the experimental group, "4" is one of the most promising values for the number of the clusters (figure 6.12). In figure 6.12, the maximum number of clusters for the exam is set to 6 since there was no more than 6 clusters (clusters' centers) in the exam data.

6.7 Summary

In this chapter, we initially explain how we implemented our recommender approach (uses a two-layered course graph) and embedded it in an E-learning system called Enki. We then describe an experiment that was performed to assess the quality of the recommender. For that, we organized a short course on C# programing language. Then, 32 participants were selected and divided in two groups, a control group and an experimental group. These groups used two different versions of the same system (Enki). The experimental group used the version that utilized our recommender method while control group used the version of the system that delivered LO without using recommendations. After completing the course, all participants attended a short exam for one hour, which was equal for both groups. Finally, we compared the performance of the participants in both groups on the course and on the exam. In addition, we collected participants' comments about the recommender using a short questionnaire.

Chapter 7

Conclusion

In this thesis, we introduce Long Term Goal Recommender Systems (LTRS) that in addition to satisfying prompt needs of users, guide them toward a pre-determined long term objective by generating a set of relevant recommendations in successive moments. LTRS can be applied in various domains, such as tourism, music, E-learning. In this thesis, we have concentrated on the E-learning domain. In this domain, one of the significant challenges is recommending learning materials that a user is able to complete timely. This challenge becomes more difficult when users cannot devote sufficient time to learn an entire course (path).

Therefore, in this thesis, we present two approaches, that are examples of LTRS, for generating and recommending paths (courses) when users have specific time constraints. In our approaches, we recommend paths regarding knowledge background and available time of users. In these methods, paths are generated from different types of course graphs. The first approach uses a one-layer course graph to generate paths while the second one is based on a two-layered course graph to cover the problems of the first approach.

In the first approach, a user first needs to specify his/her knowledge background by identifying a LO in the course graph. This method then finds all paths (LO sequences) using a DFS algorithm, and estimates time and score for them. It also estimates the probability of underachieving the estimated score for a path (probability of error for score) as well as the probability of not completing a path in the available time of a user (probability of error for time). Finally, it recommends a path that satisfies the time constraint of the user while maximizing the expected score.

In the second approach, we have tried to cover the problems of the first approach. After specifying the knowledge background of a user (selecting a lesson), this approach applies a DFS to find all lesson-sequences that start by the selected lesson. It then uses the same methods as the previous approach to estimate time, score, and the probability of error for the paths. Finally, a path that satisfies the time constraint of a user while maximizing the expected score is recommended. To recommend the path, it applies an algorithm, which allows it to iterate over each lesson of the path and recommend the associated LO.

To evaluate the recommender approaches, we first evaluate the quality of time and score estimation methods using offline approaches. Then, we assess the quality of the recommender approach, which works based on a two-layered course graph, in a live environment. This approach is evaluated since it is an enhanced version of the other one (first recommender approach).

7.1 Research Contributions

In this thesis, we aim at generating learning paths for users that satisfy their available learning time while maximizing their learning score. Here, we summarize and discuss the main contributions of this thesis, addressing the research goal. These contributions are organized according to their significance level.

1. We present two approaches for generating learning paths considering a user's time restriction and knowledge background. The approach, which uses a two-layered course graph, is able to adapt a path using a user's transactions data. Furthermore, it applies an algorithm to recommend a path lesson by lesson, which results in early detection of users' disability in learning a path. In a situation that a user could not follow the recommended LO for a lesson, it recommends auxiliary LO. These LO are the ones that are not in the initial path, and they are generated to assist a user to learn a lesson correctly. They help that instead of generating a new path, which is time-consuming and computationally expensive, we guide users by providing more LO on a lesson that they could not learn.

2. We implement and evaluate the quality of nine different methods for estimating learning time and score for learning paths. These methods are implemented using various techniques and algorithms, such as **MF.Predict** that is developed using machine learning techniques, or **IRT.Predict** which is implemented using statistical models. Researchers often use static learning time values, which are specified by a course expert and mentioned in the metadata of LO, for estimating the time for paths.

3. We also present a comprehensive overview of the learning path personalization methods as well as their advantages and disadvantages. The main parameters for personalizing paths are also described. In addition, we present approaches that are used to evaluate path personalization methods. Finally, we highlight the most significant challenges of these methods, which need to be tackled to enhance the quality of the personalization.

4. We have developed our recommender using *R* programming language. For that, we have used SQLite, which is a free library that implements a self-contained SQL database engine. In addition, we have developed a version of our recommender approach in Java and embedding it in a system called Enki. We plan to use our recommender at the Departement of computer science of the University of Porto in order to assist students in learning programming languages.

7.2 Research Limitations

Although we were able to generate learning paths for users and maximize their learning score under their given time, our approaches suffer from a few drawbacks. In this section, we highlight the main limitations of this thesis.

7.2.1 Availability of datasets

One of the main difficulties that we had during this research was the scarcity of publicly available data in E-learning domain, which contains the learning time and score of users per LO or lesson. Although there were a vast amount of data relevant to this domain, these datasets were often proprietary and could not be released because of privacy concerns, and therefore, they were unavailable to us. Hence, due to lack of such datasets, we were not able to conduct extensive experiments (i.e., offline evaluation) and assess our methods in the case of scalability problem. So, we used two relatively small datasets that we obtained from our own systems for the offline evaluation. Apart from that, in this thesis, the analysis of the Mooshak dataset, as presented in section 5.2.1, reveals a strange pattern in learning time of LO (the maximum learning time of all LO is almost the same), which might mislead us in determining our best estimation method.

7.2.2 Size of experiments

In an experiment involving human subjects, selecting a proper sample size is a pivotal issue. An under-sized experiment might not challenge the methods enough in order to show their deficiencies while an over-sized experiment in addition to being expensive, both in terms of time and money, might not be necessary. Although in this thesis we have conducted both offline and controlled experiments, the work would benefit from experiments with more users and with longer duration.

7.2.3 Estimation methods

As presented in chapter 4, we have introduced nine different approaches for estimating score and time for learning paths. Although the evaluation results, which are presented in chapter 6, shows that the selected method (i.e., **Clust.Mean**) is able to estimate time and score, it still generates some outliers (bad estimation) that need to be considered. In addition, regarding the evaluation results for the overestimation and underestimation of time and score, the selected method performs noticeably well in estimating time (overestimating more than underestimating) while the performance of this method is not as expected in score estimation (it overestimates rather than underestimating).

7.2.4 Demonstrating only in a single domain

As mentioned in chapter 1, the LTRS can be applied in different domains, such as music where a LTRS can be applied to diversify users' tastes or influencing their tastes to follow and buy a specific genre of music, or in tourism where these systems are capable of maximizing tourists experiences (e.g. visiting places, eating traditional dishes, etc.) in their available time and budget. In spite of the applicability of LTRS in different domains, due to lack of time and datasets we were not able to develop our approach in other domains.

7.3 Future work

The contributions proposed in this thesis and the obtained results as well as the mentioned limitations suggest additional research directions. The most promising ones are listed as follow:

7.3.1 Cold start problem

We selected the **Clust.Mean** method to estimate learning time and score for the generated learning paths. This method, as presented in chapter 4, requires sufficient data from a target user and LO in order to estimate learning time and score. Hence, in a situation that there is not enough data for estimation, this method suffers from Cold-start problem. In our recommender approaches, this problem describes the difficulty of learning time and score estimation when users or LO are new or there is not sufficient data about them to estimate their time and score.

7.3.2 Take advantage of users and LO metadata

A minimal information on users (i.e., the time and score of users) is needed to estimate their learning time and score for paths. The minimal information is used since users often do not want to share additional information, such as their feedbacks on the difficulty and the quality of LO. It is of interest to analyze the influence of applying different kind of users and LO information (e.g. number of users' attempts to complete a LO, or users' rates on the difficulty level of LO) in improving the accuracy of estimation results.

7.3.3 Big data and Scalability

One of the main features of learning path recommenders is to react rapidly in order to keep users engaged with the system. This feature can be influenced by large scale datasets. Hence, it is significant to design a scalable learning path recommender method that handles this kind of data. Although our estimation method, which is based on clustering technique, theoretically should be able to cope with the large scale datasets, due to unavailability of large datasets (for offline evaluation) and not having a large number of users in our controlled experiment, we could not confirm the scalability of our recommender approaches.

7.3.4 Update scheduling

In the lesson-based approach, we update users' time and score after completion of each lesson without considering any precedence to score or time. It is of interest to find a proper time to update each user's profile since users have different progress speed, which a repeated updating procedure might not be necessary meanwhile it is computationally costly while postponing it might mislead the user. In addition, different information that is used to generate a path can have different priorities for users, hence, it can be important to generate learning paths and update users' profiles regarding these priorities.

7.3.5 User-centered course structure

In our recommender approaches, two course graphs, which are constructed by a course expert, are used to generate learning paths for all users. Therefore, the structure of these courses that are used for recommendation will be the same for all users. It is important to generate paths taking into consideration the users' preferences for following a course since users might not always follow the same structure as a course expert to learn a course. For example, for each lesson in our lesson-based approach, expository LO will be recommended to users first while users might prefer a different way (e.g. Socratic order/method [Lam11]) to receive the recommendations for learning a course.

7.3.6 Evaluation framework

One of the most important shortcomings in these recommenders is the lack of a general framework that researchers can apply to evaluate and compare their learning path recommender methods. This framework can include the key factors that need to be evaluated (e.g. grade), the required information as well as the methods that need to be used for the evaluation. It can be significantly useful in promoting the research in this domain.

References

[ACBT04] Rachid Anane, S Crowther, J Beadle, and G Theodoropoulos. elearning content provision. In *Database and Expert Systems Applications, 2004. Proceedings. 15th International Workshop on*, pages 420–425. IEEE, 2004.

[Ahm13] Ahmed Khaled Ahmed. Teacher-centered versus learner-centered teaching style. *Journal of Global Business Management*, 9(1):22, 2013.

[AJ09] Marwah Alian and Riad Jabri. A shortest adaptive learning path in elearning systems: Mathematical view. *Journal of American Science*, 5(6):32–42, 2009.

[AK15] Giovanni Adorni and Frosina 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.

[AK16] Giovanni Adorni and Frosina Koceva. Educational concept maps for personalized learning path generation. In *AI* IA 2016 Advances in Artificial Intelligence*, pages 135–148. Springer, 2016.

[ANRR01] Albert Angehrn, Thierry Nabeth, Liana Razmerita, and Claudia Roda. K-inca: using artificial agents to help people learn and adopt new behaviours. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on*, pages 225–226. IEEE, 2001.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[AY14] Xinming An and Yiu-Fai Yung. Item response theory: what it is and how you can use the irt procedure to apply it. *SAS Institute Inc. SAS364-2014*, 2014.

[BA04] Kenneth P Burnham and David R Anderson. Multimodel inference: understanding AIC and BIC in model selection. *Sociological methods & research*, 33(2):261–304, 2004.

[Bak01] Frank B Baker. *The basics of item response theory.* Education Resources Information Center - ERIC, 2001.

[BBR13] Prasenjit Basu, Suman Bhattacharya, and Samir Roy. Online recommendation of learning path for an e-learner under virtual university. In *International Conference on Distributed Computing and Internet Technology*, pages 126–136. Springer, 2013.

[BDCS10] Manju Bhaskar, Minu M Das, T Chithralekha, and S Sivasatya. Genetic algorithm based adaptive learning scheme generation for context aware e-learning. *International Journal on Computer Science and Engineering*, 2(4):1271–1279, 2010.

[BDL14] Nabil Belacel, Guillaume Durand, and François Laplante. A binary integer programming model for global optimization of learning path discovery. In *Proceedings of the The 7th International Conference on Educational Data Mining*, 2014.

[BGJG14] Gabriel Barata, Sandra Gama, Joaquim AP Jorge, and Daniel JV Gonçalves. Relating gaming habits with student performance in a gamified learning experience. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*, pages 17–25. ACM, 2014.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[BW84] Alan Bundy and Lincoln Wallen. Breadth-first search. In *Catalogue of Artificial Intelligence Tools*, pages 13–13. Springer, 1984.

[CDSG14] Francesco Colace, Massimo De Santo, and Luca Greco. E-learning and personalized learning path: A proposal based on the adaptive educational hypermedia system. *International Journal of Emerging Technologies in Learning*, 9(2), 2014.

[Chi10] Y Chi. Developing curriculum sequencing for managing multiple texts in e-learning system. In *Proceedings of international conference on engineering education*, pages 1–8, 2010.

[CLA$^+$03] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592. ACM, 2003.

[CLM10] Vincenza Carchiolo, Alessandro Longheu, and Michele Malgeri. 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, 2010.

[DBL13] Guillaume Durand, Nabil Belacel, and François LaPlante. Graph theory based model for learning path recommendation. *Information Sciences*, 251:10–21, 2013.

[DG13] B Dharani and TV Geetha. Adaptive learning path generation using colored petri nets based on behavioral aspects. In *Recent Trends in Information Technology (ICRTIT), 2013 International Conference on*, pages 459–465. IEEE, 2013.

[DH03] Erik Duval and Wayne Hodgins. A lom research agenda. In *WWW (Alternate Paper Tracks)*, 2003.

[DHC17] Yuli Deng, Dijiang Huang, and Chun-Jen Chung. Thoth lab: A personalized learning framework for cs hands-on projects. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 706–706. ACM, 2017.

[DHK08] Hendrik Drachsler, Hans GK Hummel, and Rob Koper. Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model. *International Journal of Learning Technology*, 3(4):404–423, 2008.

[DLK11] G Durand, F Laplante, and R Kop. A learning design recommendation system based on markov decision processes. In *KDD-2011: 17th ACM SIGKDD conference on knowledge discovery and data mining*, 2011.

[DOS$^+$10] Pavla Drážďilová, Gamila Obadi, Kateřina Slaninová, Shawki Al-Dubaee, Jan Martinovič, and Václav Snášel. Computational intelligence methods for data analysis and mining of elearning activities. In *Computational intelligence for technology enhanced learning*, pages 195–224. Springer, 2010.

[DS10] Marco Dorigo and Thomas Stützle. Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics*, pages 227–263. Springer, 2010.

[DSM12] Joumana Dargham, Dana Saeed, and Hamid 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.

[EAJ$^+$10] Fathi Essalmi, Leila Jemni Ben Ayed, Mohamed Jemni, Sabine Graf, et al. A fully personalization strategy of e-learning scenarios. *Computers in Human Behavior*, 26(4):581–591, 2010.

[EAJ$^+$15] Fathi Essalmi, Leila Jemni Ben Ayed, Mohamed Jemni, Sabine Graf, et al. Generalized metrics for the analysis of e-learning personalization strategies. *Computers in Human Behavior*, 48:310–322, 2015.

[Ell09] Ryann K Ellis. Field guide to learning management systems. *ASTD Learning Circuits*, pages 1–8, 2009.

[FVFNN10] Philippe Fournier-Viger, Usef Faghihi, Roger Nkambou, and Engelbert Mephu Nguifo. Exploiting sequential patterns found in users' solutions and virtual tutor behavior to improve assistance in its. *Journal of Educational Technology & Society*, 13(1):13–24, 2010.

[FVNN08] Philippe Fournier-Viger, Roger Nkambou, and Engelbert Mephu Nguifo. A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Mexican International Conference on Artificial Intelligence*, pages 765–778. Springer, 2008.

[FXP$^+$10] Xiuzhen Feng, Haoran Xie, Yang Peng, Wei Chen, and Huamei Sun. Groupized learning path discovery based on member profile. In *ICWL Workshops*, pages 301–310. Springer, 2010.

[GFM$^+$13] Antonio Garrido, Susana Fernández, Lluvia Morales, Eva Onaindía, Daniel Borrajo, and Luis Castillo. On the automatic compilation of e-learning models to planning. *The Knowledge Engineering Review*, 28(02):121–136, 2013.

[GGW11] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[Gil12] Nicolas Gillis. Sparse and unique nonnegative matrix factorization through data preprocessing. *The Journal of Machine Learning Research*, 13(1):3349–3386, 2012.

[GK+16] Kannan Govindarajan, Vivekanandan Suresh 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.

[GMR07] Jennifer Gilbert, Susan Morton, and Jennifer Rowley. e-learning: The student experience. *British Journal of Educational Technology*, 38(4):560–573, 2007.

[GMS12] Antonio Garrido, Lluvia Morales, and Ivan Serina. Using AI planning to enhance e-learning processes. In *ICAPS*, 2012.

[GO13] Antonio Garrido and Eva Onaindia. Assembling learning objects for personalized learning: An ai planning perspective. *IEEE Intelligent Systems*, 28(2):64–73, 2013.

[GRFST11] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: a free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[GRVDC09] Enrique García, Cristóbal Romero, Sebastián Ventura, and Carlos De Castro. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2):99–132, 2009.

[Gui00] Minitab User's Guide. 2: Data analysis and quality tools. *Minitab Inc*, 2000.

[HKYC10] Gwo-Jen Hwang, Fan-Ray Kuo, Peng-Yeng Yin, and Kuo-Hsien Chuang. A heuristic algorithm for planning personalized learning paths for context-aware ubiquitous learning. *Computers & Education*, 54(2):404–415, 2010.

[Hod06] H Wayne Hodgins. The future of learning objects. *Educational Technology*, pages 49–54, 2006.

[HT07] Robert Hanson and Adam Tacy. *GWT in action: easy Ajax with the Google Web toolkit.* Dreamtech Press, 2007.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[IALS12] Jose Antonio Iglesias, Plamen Angelov, Agapito Ledezma, and Araceli Sanchis. Creating evolving user behavior profiles automatically. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):854–867, 2012.

[Ime02] Susan Imel. E-learning. trends and issues alert. pages 1–4, 2002.

[J+07] Matthew S Johnson et al. Marginal maximum likelihood estimation of item response models in r. *Journal of Statistical Software*, 20(10):1–24, 2007.

[Jai10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[JBH+10] José Janssen, Adriana J Berlanga, Stef Heyenrath, Harry Martens, Hubert Vogten, Anton Finders, Eelco Herder, Henry Hermans, Javier Melero Gallardo, Leon Schaeps, et al. Assessing the learning path specification: A pragmatic quality approach. *J. UCS*, 16(21):3191–3209, 2010.

[Jin11] Qun Jin. *Intelligent Learning Systems and Advancements in Computer-Aided Instruction: Emerging Studies: Emerging Studies*. IGI Global, 2011.

[JK13] Aastha Joshi and Rajneet Kaur. A review: Comparative study of various clustering techniques in data mining. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(3), 2013.

[JMW06] Jeff Johns, Sridhar Mahadevan, and Beverly Woolf. Estimating student proficiency using an item response theory model. In *International Conference on Intelligent Tutoring Systems*, pages 473–480. Springer, 2006.

[Joh73] Donald B Johnson. A note on dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20(3):385–388, 1973.

[Joh04] Matthew S Johnson. Item response models and their use in measuring food insecurity and hunger. In *NAS Committee on National Statistics Workshop on the Measurement of Food Insecurity and Hunger, Washington, DC*, 2004.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[KEI14] Ahmad A Kardan, Molood Ale Ebrahim, and Maryam Bahojb Imani. A new personalized learning path generation method: Aco-map. *Indian Journal of Scientific Research*, 5(1):17, 2014.

[Ken11] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[Khr10] Andrei Khrennikov. On the physical basis of theory of" mental waves". *NeuroQuantology*, 8(4), 2010.

[KHUG10] Manoj Kumar, Mohammad Husian, Naveen Upreti, and Deepti Gupta. Genetic algorithm: Review and application. *International Journal of Information Technology and Knowledge Management*, 2(2):451–454, 2010.

[KMIN15] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 44(4):571–604, 2015.

[KMVIB11] Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, and Zoran Budimac. E-learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education*, 56(3):885–899, 2011.

[KNB05] Froduald Kabanza, Roger Nkambou, and Khaled Belghith. Path-planning for autonomous training on robot manipulators in space. In *IJCAI*, pages 1729–1731, 2005.

[Koc16] Frosina Koceva. Encode - environment for content design and editing. *University of Genoa, PhD thesis*, 2016.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[KS05] Pythagoras Karampiperis and Demetrios Sampson. Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society*, 8(4):128–147, 2005.

[Lam11] Faith Lam. The socratic method as an approach to learning and its benefits. *Carnegie Mellon University, Senior Honors Thesis*, 2011.

[LC11] Youngju Lee and Jaeho Choi. A review of online course dropout research: Implications for practice and future research. *Educational Technology Research and Development*, 59(5):593–618, 2011.

[LCCT12] Jian-Wei Li, Yi-Chun Chang, Chih-Ping Chu, and Cheng-Chang Tsai. A self-adjusting e-course generation process for personalized learning. *Expert Systems with Applications*, 39(3):3223–3232, 2012.

[LOdP11] Miltiadis D Lytras and Patricia Ordóñez de Pablos. Software technologies in knowledge society j. ucs special issue. volume 17, pages 1219–1221, 2011.

[LPK16] Zhan Li, Olga Papaemmanouil, and Georgia 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.

[LQ11] José Paulo Leal and Ricardo Queirós. *A comparative study on LMS interoperability.* IGI-Global, 2011.

[LS03] José Paulo Leal and Fernando Silva. Mooshak: a web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6):567–581, 2003.

[LXZ13] Xin Luo, Yunni Xia, and Qingsheng Zhu. Applying the learning rate adaptation to the matrix factorization based collaborative filtering. *Knowledge-Based Systems*, 37:154–164, 2013.

[LYHC13] Chun Fu Lin, Yu-chu Yeh, Yu Hsin Hung, and Ray I Chang. Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, 68:199–210, 2013.

[McD09] John H McDonald. *Handbook of biological statistics*, volume 2. sparky house publishing Baltimore, MD, 2009.

[MIBW14] William C McGaghie, Saul B Issenberg, Jeffrey H Barsuk, and Diane B Wayne. A critical review of simulation-based mastery learning with translational outcomes. *Medical education*, 48(4):375–385, 2014.

[MKKP10] Urszula Markowska-Kaczmar, Halina Kwasnicka, and Mariusz Paradowski. Intelligent techniques in personalization of learning in e-learning systems. In *Computational Intelligence for Technology Enhanced Learning*, pages 1–23. Springer, 2010.

[MLO$^+$16] Sandro Monteiro, José Alberto Lencastre, António José Osório, Bento Duarte da Silva, Paula De Waal, Sukru Çetin İlin, Yalın Kılıç Türel, and Muhammed 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.

[MN10] Patrick E McKight and Julius Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.

[MS04] Brenda Mallinson and David Sewry. Elearning at rhodes university-a case study. In *Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on*, pages 708–710. IEEE, 2004.

[MZB$^+$16] Alva Muhammad, Qingguo Zhou, Ghassan Beydoun, Dongming Xu, and Jun Shen. Learning path adaptation in online learning systems. In *Computer Supported Cooperative Work in Design (CSCWD), 2016 IEEE 20th International Conference on*, pages 421–426. IEEE, 2016.

[Nai16] Vannie Naidoo. Challenges facing e-learning. *Multiculturalism and Technology-Enhanced Language Learning*, page 271, 2016.

[Neu11] Markus Neuhäuser. Wilcoxon–mann–whitney test. In *International encyclopedia of statistical science*, pages 1656–1658. Springer, 2011.

[NJL15a] Amir Hossein Nabizadeh, Alípio Mário Jorge, and José Paulo Leal. Long term goal oriented recommender systems. In *WEBIST*, pages 552–557, 2015.

[NJL15b] Amir Hossein Nabizadeh, Alípio Mário Jorge, and José Paulo Leal. Long term goal oriented recommender systems. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, page Ph.D. consortium, 2015.

[NJL18] Amir Hossein Nabizadeh, Alípio Mário Jorge, and José Paulo Leal. Estimating time and score uncertainty in generating successful learning paths under time constraints. *Expert Systems, DOI:e12351*, 2018.

[NJTY16] Amir Hossein Nabizadeh, Alípio Mário Jorge, Suhua Tang, and Yi Yu. Predicting user preference based on matrix factorization by exploiting music attributes. In *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*, pages 61–66. ACM, 2016.

[NMJPL17] Amir Hossein Nabizadeh, Alípio Mário Jorge, and José Paulo Leal. Rutico: Recommending successful learning paths under time constraints. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 153–158. ACM, 2017.

[PLQ16] José Carlos Paiva, José Paulo Leal, and Ricardo Alexandre 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.

[PR13] Martha C Polson and J Jeffrey Richardson. *Foundations of intelligent tutoring systems.* Psychology Press, 2013.

[QLC14] Ricardo Queirós, José Paulo Leal, and José Campos. Sequencing educational resources with seqins. 2014.

[Ree79] Malcolm James Ree. Estimating item characteristic curves. *Applied Psychological Measurement*, 3(3):371–385, 1979.

[RM05] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[RR95] Michel Raymond and François Rousset. An exact test for population differentiation. *Evolution*, 49(6):1280–1283, 1995.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[SGS07] Anthony Savidis, Dimitris Grammenos, and Constantine Stephanidis. Developing inclusive e-learning and e-entertainment to effectively accommodate learning difficulties. *Universal Access in the Information Society*, 5(4):401–419, 2007.

[SÖY13] Mehmet Ali Salahli, Muzaffer Özdemir, and Cumali Yasar. Concept based approach for adaptive personalized course learning system. *International Education Studies*, 6(5):92–103, 2013.

[SP15] N Sivakumar and R Praveena. Determining optimized learning path for an e-learning system using ant colony optimization algorithm. *International Journal of Computer Science Engineering Technology*, pages 61–66, 2015.

[SRR12] Irma Gamez Suazo, César Garita Rodriguez, and Mario Chacón Rivas. Generating adaptive learning paths in e-learning environments. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–10. IEEE, 2012.

[Sta17] LLC StataCorp. stata bayesian analysis reference manual. release 15, 2017.

[Tar71] Robert Tarjan. Depth-first search and linear graph algorithms. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*, pages 114–121. IEEE, 1971.

[TJ15] Samir R Thakkar and Hiren D Joshi. E-learning systems: a review. In *Technology for Education (T4E), 2015 IEEE Seventh International Conference on*, pages 37–40. IEEE, 2015.

[TLF12] Vincent Tam, Edmund Y Lam, and ST Fung. Toward a complete e-learning system framework for semantic analysis, concept clustering and learning path optimization. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on*, pages 592–596. IEEE, 2012.

[TLF14] Vincent Tam, Edmund Y Lam, and ST Fung. A new framework of concept clustering and learning path optimization to develop the next-generation e-learning systems. *journal of computers in education*, 1(4):335–352, 2014.

[TM05] Tiffany Tang and Gordon McCalla. Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on E-learning*, 4(1):105–129, 2005.

[UM10] Carsten Ullrich and Erica Melis. Complex course generation adapted to pedagogical scenarios and its evaluation. *Educational Technology & Society*, 13(2):102–115, 2010.

[VA13] T Vander Ark. The future of learning: Personalized adaptive and competency-based. *DreamBox Learning*, pages 1–13, 2013.

[VMIB13] Boban Vesin, Aleksandra Klasnja Milicevic, Mirjana Ivanovic, and Zoran Budimac. Applying recommender systems and adaptive hypermedia for e-learning personalizatio. *Computing and Informatics*, 32(3):629–659, 2013.

[WL$^+$16] Ronald L Wasserstein, Nicole A Lazar, et al. The asa's statement on p-values: context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[WM05] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79, 2005.

[XWCH12] Dihua Xu, Zhijian Wang, Kejia Chen, and Weidong Huang. Personalized learning path recommender based on user profile using social tags. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 1, pages 511–514. IEEE, 2012.

[XXVDS16] Jie Xu, Tianwei Xing, and Mihaela Van Der Schaar. Personalized course sequence recommendations. *IEEE Transactions on Signal Processing*, 64(20):5340–5352, 2016.

[XZW$^+$17] Haoran Xie, Di Zou, Fu Lee Wang, Tak-Lam Wong, Yanghui Rao, and Simon Ho Wang. Discover learning path for group users: A profile-based approach. *Neurocomputing*, 254:59–70, 2017.

[YD16] Fan Yang and Zhenghong Dong. *Learning Path Construction in E-learning: What to Learn, how to Learn, and how to Improve*. Book, Springer, 2016.

[YHY13] Tzu-Chi Yang, Gwo-Jen Hwang, and Stephen Jen-Hwa Yang. Development of an adaptive learning system with multiple perspectives based on students' learning styles and cognitive styles. *Journal of Educational Technology & Society*, 16(4):185, 2013.

[YJT13] Maryam Yarandi, Hossein Jahankhani, and Abdel-Rahman Tawil. A personalized adaptive e-learning approach based on semantic web technology. *Webology*, 10(2):Art–110, 2013.

[YLH10] Juan Yang, Hongtao Liu, and Zhixing Huang. Smap: To generate the personalized learning paths for different learning style learners. In *Edutainment*, pages 13–22. Springer, 2010.

[YLL10] Fan Yang, Frederick Li, and Rynson Lau. An open model for learning path construction. *Advances in Web-Based Learning–ICWL 2010*, pages 318–328, 2010.

[YLL12] Fan Yang, Frederick Li, and Rynson Lau. Learning path construction based on association link network. *Advances in Web-Based Learning-ICWL 2012*, pages 120–131, 2012.

[YLL14] Fan Yang, Frederick WB Li, and Rynson WH Lau. A fine-grained outcome-based learning path model. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(2):235–245, 2014.

[ZL01] Osmar R Zaıane and J Luo. Web usage mining for a better web-based learning environment. In *Proceedings of conference on advanced technology for education*, pages 60–64, 2001.

[Zop12] Cengiz Zopluoglu. Estcrm: An r package for samejima's continuous irt model. *Applied Psychological Measurement*, 36(2):149, 2012.

[ZTPS07] Anna Zoakou, Aimilia Tzanavari, GA Papadopoulos, and Sofoklis Sotiriou. A methodology for elearning scenario development: the unite approach. In *Proceedings of the ECEL2007-European Conference on e-Learning, Copenhagen, Denmark. ACL publications*, pages 683–692. Citeseer, 2007.

Acronym list

LTRS Long Term Goal Oriented Recommender System

RS Recommender System

CF Collaborative Filtering

CB Content-Based Filtering

DFS Depth-First Search

BFS Breadth-First Search

LO Learning Objects

SP Starting Point

CG Course Generation

CS Course Sequence

SPR Sequential Pattern Recognition

ITS Intelligent Tutoring System

IRT Item Response Theory

ICC Item Characteristic Curve

2PL Two parameters logistic model

3PL Three parameters logistic model

MF Matrix Factorization

MAE Mean Absolute Error

RMSE Root Mean Square Error

Appendix A

Summarized studies

In this appendix, we have summarized the papers that we have presented in chapter 2 of this thesis (Table A.1). Take into consideration that this information was collected until the middle of October 2017. In table A.1, column "Country" refers to the country of the first author. Furthermore, in this table, the learning goal (one of the personalization parameters) is ignored since all papers have learning goals, and only the type of goal might be different.

Table A.1: Summarized papers

				Evaluation Methods			
No.	**Ref.**	**Type**	**Main techniques/algorithms/tools**	**Offline**	**Performance**	**Online**	**User study**
1	[BDL14]	CG	Branch and bound algorithm	-	-	-	-
2	[LPK16]	CG	Graph search algorithm	-	✓	-	-
3	[AK16]	CG	Educational Concept Map (ECM), ENCODE (linearize the map)	-	-	-	-
4	[DBL13]	CG	Greedy algorithm	-	✓	-	-
5	[GFM$^+$13]	CG	PDDL, HTN, Non-HTN	-	✓	-	✓
6	[YLL14]	CG	Bloom's taxonomy	-	✓	-	✓
7	[HKYC10]	CG	RFID,repertory grid-oriented technique,Heuristic Algorithm	-	✓	✓	✓
8	[TLF14]	CG	Explicit Semantic Analysis, Evolutionary algorithm+hill climbing, Clustering	-	✓	-	-
9	[KEI14]	CG	ant-colony (ACO-Map), Ausubel Meaningful Learning Theory, Clustering	-	-	-	-
10	[DG13]	CG	Case Based Reasoning (CBR), Colored Petri Nets, Classification	-	✓	✓	-
11	[YLL10]	CG	Bloom's taxonomy	-	-	-	✓
12	[XXVDS16]	CG	Clustering, Multi-armed bandits	-	✓	✓	-
13	[LYHC13]	CG	Decision tree classifier, feature weighting, feature selection	-	-	✓	-
14	[XZW$^+$17]	CG	Topic graph extraction, group profiling, estimating time boundaries	-	-	✓	✓
15	[JBH$^+$10]	CG	Search engine	-	✓	-	✓
16	[SP15]	CG	Ant Colony Optimization algorithm (ACO)	-	-	✓	-
17	[YHY13]	CG	Learning and Cognitive styles	-	-	✓	✓
18	[Chi10]	CG	Domain ontology, Problem-solving ontology, Semantic rules	-	-	-	-
19	[BDCS10]	CG	Genetic algorithm	-	-	-	-
20	[BBR13]	CG	Greedy algorithm	-	-	-	-
21	[FXP$^+$10]	CG	Topic graph extraction, group profiling, estimating time boundaries	-	-	✓	-
22	[YLH10]	CG	Felder learning style, Broad first searching (BFS), 3-dimension semantic map	-	-	-	-
23	[CLM10]	CG	Peer-to-peer (P2P) networks	-	✓	-	-
24	[GMS12]	CG	PDDL, Case-Based Planning (CBP)	-	✓	-	✓
25	[NMJPL17]	CG	Depth first search (DFS)	✓	✓	-	-
26	[AJ09]	CG	Weighted Graph, Eliminating and Optimized Selection (EOS)	-	✓	-	-
27	[TLF12]	CG	Explicit semantic analysis , Genetic algorithm, Clustering(K-means)	-	✓	-	-
28	[DLK11]	CG	Markov Decision Process	-	-	-	-
29	[SRR12]	CG	Bayesian network, Clustering	-	-	-	✓
30	[EAJ$^+$10]	CG	Personalization strategy	-	-	-	✓
31	[FVFNN10]	SPR	Sequential pattern mining (invented algorithm)	-	-	-	✓
32	[KMVIB11]	SPR	Clustering, AprioriAll	✓	✓	✓	-
33	[VMIB13]	SPR	Clustering, Collaborative filtering, Association rule mining	-	-	✓	✓
34	[YJT13]	CS	Item Response Theory (IRT)	-	-	-	-
35	[LCCT12]	CS	Collaborative Voting, Maximum Likelihood Estimation, Evolutionary algorithms (GA,PSO)	-	✓	-	✓
36	[GO13]	CS	PDDL, Constraint Satisfaction Problem (CSP)-based scheduling	-	✓	-	✓
37	[UM10]	CS	Learning scenario selection, HTN	-	-	-	✓
38	[GK$^+$16]	CS	Evolutionary algorithm (PPSO), Clustering, Bloom's taxonomy	-	✓	✓	-
39	[XWCH12]	CS	Bayes formula, Pearson Correlation Coefficient (KNN), feature weighting	-	-	✓	-
40	[CDSG14]	CS	Functions to estimate the closeness of each LO to a user's profile, Ontology	-	-	✓	-
41	[SÖY13]	CS	Item Response Theory (IRT), Law of Total Probability (LPT)	-	✓	-	-
42	[YLL12]	CS	Association Link Network (ALN), TFIDF Direct Document Frequency of Domain (TDDF)	-	✓	✓	-

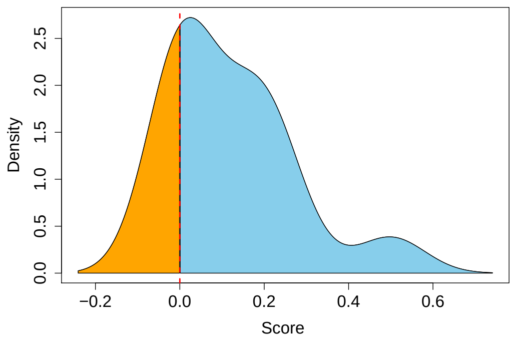
			Personalization Parameters						
No.	**Ref.**	**Recommendation strategy**	**Time**	**Mastery level**	**Style**	**Know. Background**	**Year**	**Citation**	**Country**
1	[BDL14]	Shortest path (number of competency)	-	✓	-	-	2014	6	Canada
2	[LPK16]	Deadline driven, Goal driven, Ranked driven	✓	-	-	-	2016	1	USA
3	[AK16]	Minimizing the number of LO	-	✓	✓	✓	2016	1	Italy
4	[DBL13]	Shortest path (number of competency)	-	✓	-	-	2013	25	Canada
5	[GFM$^+$13]	Maximizes the total reward without exceeding a user's time	✓	-	✓	-	2013	13	Spain
6	[YLL14]	It is a path generation method that formulates learning activities and their assessment criteria using the Bloom's taxonomy.	-	✓	-	-	2014	15	HongKong
7	[HKYC10]	Minimizing the relevance loss between consecutive LO along a path	-	-	-	-	2010	171	Taiwan
8	[TLF14]	Minimizing the sum of the violated distances regarding the reference paths	-	-	-	-	2014	5	HongKong
9	[KEI14]	Shortest path	-	-	-	-	2014	6	Iran
10	[DG13]	Finding similar solution regarding learning style, goal and performance	-	✓	✓	-	2013	5	India
11	[YLL10]	It is a path generation method that formulates learning activities and their assessment criteria using the Bloom's taxonomy.	-	✓	-	-	2010	15	UK
12	[XXVDS16]	Minimizing the graduation time while maximizing the overall GPA	✓	✓	-	✓	2016	9	USA
13	[LYHC13]	Maximizing users' creativity	-	-	-	-	2013	71	Taiwan
14	[XZW$^+$17]	Minimizing learning time and Maximizing the learning enjoyment	✓	-	-	✓	2017	2	HongKong
15	[JBH$^+$10]	Finding a path that meets a user criteria (e.g. start point)	-	✓	-	-	2010	5	Netherlands
16	[SP15]	Optimizing path regarding users' ability, goals and behavior	-	✓	-	-	2015	0	India
17	[YHY13]	Paths that match the cognitive and learning styles of users	-	-	✓	-	2013	102	Taiwan
18	[Chi10]	Generating path regarding a given criteria (competencies)	-	✓	-	-	2010	21	Taiwan
19	[BDCS10]	Maximizing fitness value (consists of 3 fitness values)	-	-	✓	-	2010	24	India
20	[BBR13]	Path satisfies users limitation (time) and preferences.	✓	-	-	-	2013	1	India
21	[FXP$^+$10]	Minimizing learning time while considering users preferences	✓	-	✓	✓	2010	5	China
22	[YLH10]	Path satisfies users learning style.	-	-	✓	-	2010	6	China
23	[CLM10]	Generating paths that satisfy a user's time and difficulty level	✓	✓	-	-	2010	37	Italy
24	[GMS12]	Minimizing the number of changes w.r.t. to a similar path in library	-	-	✓	✓	2012	27	Spain
25	[NMJPL17]	Maximizing users score under his/her time restriction	✓	-	-	✓	2017	0	Portugal
26	[AJ09]	Shortest path on a weighted graph	✓	✓	✓	✓	2009	20	Jordan
27	[TLF12]	Minimizing the number of precedence rules violated by the path	-	-	-	-	2012	12	HongKong
28	[DLK11]	Maximizing reward while minimizing the number of LO	-	✓	✓	-	2011	17	Canada
29	[SRR12]	Recommending activity with the highest probability	-	✓	-	-	2012	2	Costa Rica
30	[EAJ$^+$10]	Matching user' characteristics with the LO features	-	✓	✓	✓	2010	120	Tunisia
31	[FVFNN10]	Successful paths that satisafy the initial and end point of a user	-	✓	-	-	2010	11	Canada
32	[KMVIB11]	Maximizing the sequences' rate while reducing the info redundancy	-	✓	✓	-	2011	315	Serbia
33	[VMIB13]	Maximizing success rate of sequences while reducing the info redundancy	-	✓	✓	✓	2013	24	Serbia
34	[YJT13]	Recommending LO that fits the user's ability	-	✓	✓	-	2013	29	UK
35	[LCCT12]	Matching a user's ability while considering his/her time limitation	✓	✓	-	-	2012	20	Taiwan
36	[GO13]	Minimizing the length of a path while maximizing the learning reward	-	-	-	✓	2013	24	Spain
37	[UM10]	Matching preferred learning scenario of a user and his/her goal	-	✓	-	-	2010	18	China
38	[GK$^+$16]	Fitting user's need and proficiency level.	-	✓	-	-	2016	1	Canada
39	[XWCH12]	Recommending a LO with the highest probability	-	-	-	-	2012	3	China
40	[CDSG14]	Matching users' characteristics with the relative parameters of LO	✓	✓	✓	-	2014	9	Italy
41	[SÖY13]	Matching the understanding degree of users with the difficulty level of LO	-	✓	-	-	2013	0	Turkey
42	[YLL12]	Matching the users knowledge level with the complexity of LO	-	✓	-	-	2012	0	UK

Appendix B

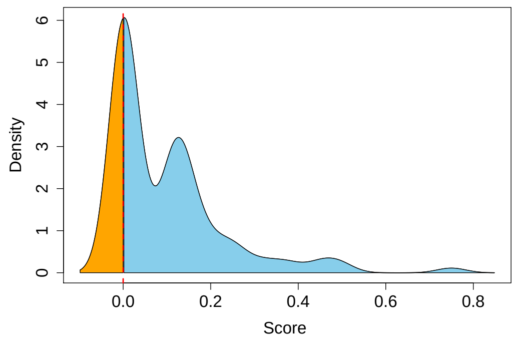
Complementary Overestimation and Underestimation Results

In this appendix, we have presented the overestimation and underestimation results for the competitive approaches (regarding the results in figures 5.5,5.6, 5.7, and 5.8). In estimating learning score, the results of **Clust.Median**, **Clust.Mean**, **IRT.predict** and **IRT-3PL (Johns)** were competitive, while for time estimation **Clust.Median**, **Clust.Mean**, **Mean**, and **Median** approaches had competitive results. Since we have shown the overestimation and underestimation results for the two approaches, **Clust.Median** and **Clust.Mean** in section 5.3.3, here, we present the results of the rest of competitive approaches.

B.1 IRT.Predict and IRT-3PL (Johns) Methods for Score Estimation

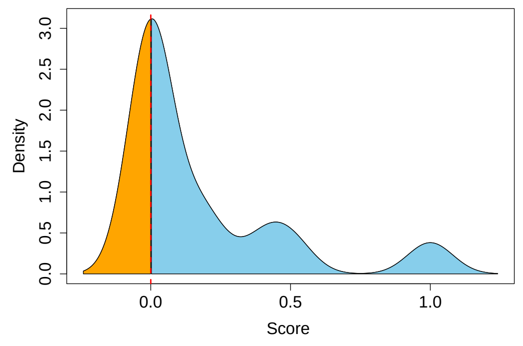


(a) Enki.Overestimation:75.6%,Underestimation: 24.4%

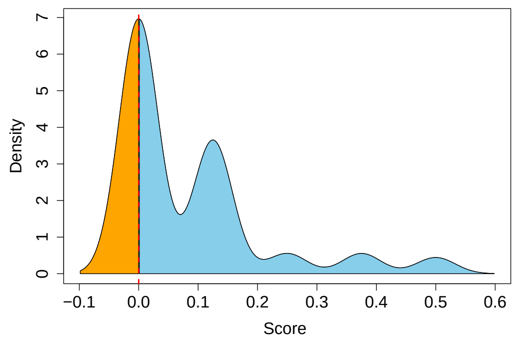


(b) Mooshak.Overestimation:75.4%, Underestimation:24.6%

Figure B.1: Score overestimating and underestimating using **IRT.Predict** approach.



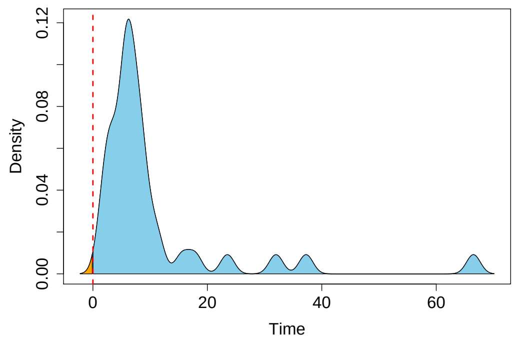
(a) Enki.Overestimation:68.8%,Underestimation: 31.2%



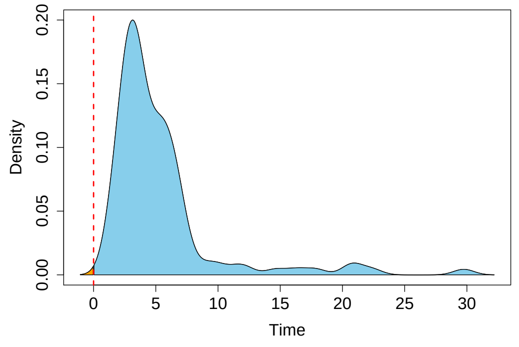
(b) Mooshak.Overestimation:71%, Underestimation:29%

Figure B.2: Score overestimating and underestimating using **IRT-3PL (Johns)** approach.

B.2 Mean and Median Methods for Time Estimation

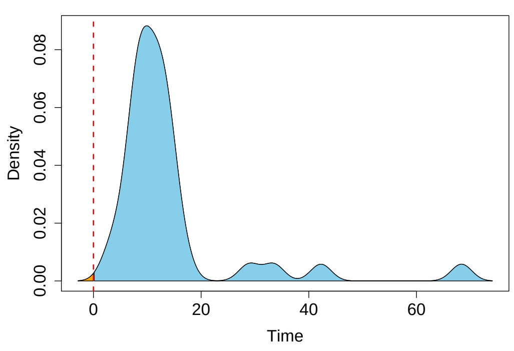


(a) Enki.Overestimation : 99.4%, Underestimation : 0.6%

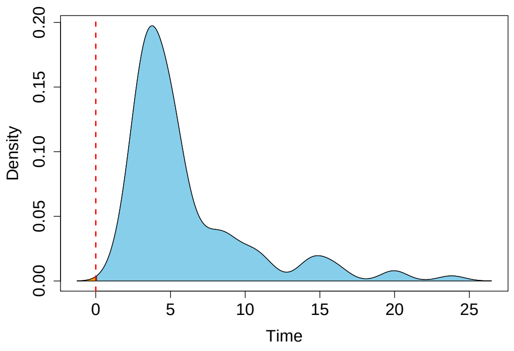


(b) Mooshak.Overestimation : 99.8%, Underestimation : 0.2%

Figure B.3: Time overestimating and underestimating using **Median** approach.



(a) Enki.Overestimation : 99.8%, Underestimation : 0.2%



(b) Mooshak.Overestimation : 99.9%, Underestimation : 0.1%

Figure B.4: Time overestimating and underestimating using **Mean** approach.