Comparing Curriculum Sequencing Algorithms for Intelligent Adaptive (e)-Learning

Carla Limongelli¹, Filippo Sciarrone¹², Marco Temperini³, Giulia Vaste¹

¹Roma Tre University, ²Open Informatica & Roma Tre University, ³Sapienza University of Roma

Key words: curriculum Sequencing, Personalization, Student Modelling

Abstract:
In the context of Web-Based e-Learning, the pedagogical strategy behind a course is crucial, as well as the capability of a system to automatically tailor the course to the needs and interests of each individual student. In fact Personalization and Adaptation are more and more and more sought in educational systems. In this paper we present the extension of the LS-Lab framework, supporting an automated and flexible comparison of the outputs coming from a variety of Curriculum Sequencing algorithm, applied to common student models. Our framework compares the algorithms’ outcomes, obtained from common conditions (student model and aims, repository of learning objects, characteristics of the produced learning paths to be monitored) by presenting the produced sequences and their metrics values.

1 Introduction

Systems for distance learning can roughly be divided in two families: on the one hand are the Learning Management Systems (LMSs), such as Moodle [16], Docebo [9], Atutor [1] or Ilias [12]: they usually guarantee contents reusability and interoperability, adhere to standard “de facto”, and they offer numerous functionalities both for students and teachers. In the second family are comprised the Intelligent Tutoring Systems (ITS) [15, 19] and the Adaptive Educational Hypermedia (AEH) [4]. Despite their attractive characteristics and wide availability, LMSs provide very limited, often null, personalization capabilities, towards learner's needs and traits. On the contrary personalization is more and more sought in all web-based systems, for providing user's with more efficient and more useful services: search engines and e-commerce applications are two examples of this trend. ITS and AEH are the research answers to this need of personalization applied in educational systems. In these systems one of the main adaptation techniques is Curriculum Sequencing.

Curriculum Sequencing means to "help the student to find an optimal path through the learning material" [5]. Research in this field aims to automatically produce a personalized sequence of didactic materials or activities, on the basis of each student's needs, by dynamically selecting the most appropriate didactic materials at any moment [6]. Several approaches and techniques for Curriculum Sequencing have been proposed in the literature, such as: with rule-based sequencing, as in the AHA! system [8]; with planning-based sequencing, as in the LS-Plan system [13] and in the work of Baldoni et al. [2]; with graph-based sequencing, as in the Intelligent Web Teacher system [17], in the KBS-Hyperbook system [11], in the Lecomps system [18] and in the DCG system [20]; and by bayesian network-based approach, as in the BITS system [7].
However each solution has its own strength and weakness points. The question: 'what is the best sequencing algorithm to use in a particular learning environment?' is a hard question: there are many variables that can affect this choice, and it is difficult, on the one hand to check different solutions working on a uniform input, and on the other hand to compare their output. It would probably be too strong a demand, to seek for the definition of a completely automated and formal environment, in which to conduct 'objective evaluation' of the appropriateness, completeness and suitability of the sequence of learning materials produced by different algorithms: fully objective criteria, and their automated application, may be a chimera. Support to the 'subjective' evaluation, provided by the teachers and domain experts, seems to be unavoidable, so it could be quite useful the use of a framework, providing the specification of uniform input for the algorithms, and allowing the administration of a bouquet of formally defined metrics on the output sequences, can be useful.

In educational literature there is not a framework for comparing curriculum sequences produced by different algorithms running on the same didactic material. In [3] a framework for comparing curriculum sequencing is proposed but the curriculum sequencing algorithms comparison is approached only from a qualitative point of view. In [14] we proposed the LS-Lab system, a self-contained and homogeneous environment, for comparing different sequencing algorithms, belonging to different adaptive educational environments.

Here we extend the approach presented in [14], proposing metrics that can highlight some aspects of the sequences produced by the algorithms, providing support for analyzing the didactic strategies that come with different learning paths. The algorithms, through suitable software interfaces, e.g. parsers, run in the same environment, taking in input the same educational material, the same student model, and the same goal for the student's course. The different generated courses are presented in output by the system, and some measures are provided for supporting evaluation.

The rest of the paper is organized as it follows: Sec.2 shows the LS-Lab system design, where some definitions are carried out in order to correctly setup the framework; Sec.3 discusses the sequencing algorithms currently compared by the system, some proposed metrics and some first evaluations; then, some conclusions are drawn in Sec.4.

2 LS-lab Design

To allow the comparison and measure of the behaviour of several sequencing algorithms, we have to accommodate various requirements in a common framework. Firstly each algorithm has to be applicable to the respective appropriate data structures, namely Learning Objects (LO); so we devise a basic LO definition called Learning Node (LN). It is based on the use of concept identifiers: Knowledge Item (KI). KIs are used to specify the requirements and the knowledge acquisition related to a Learning Material (LM) into a LN. Secondly we assume that all the algorithms share a common goal driven attitude: each experiment applies an algorithm and produces a course with a determined Target Knowledge (TK) expressed through KIs. Then we devised an initial set of metrics to present the teacher, after each experiment, with measures apt to support the evaluating activity. Some definitions for the common framework follow.

Definition 1 (Knowledge Item). KIs are atomic elements of knowledge (names for concepts).
Definition 2 Learning Node. A LN is a 4-tuple: LN=<LM, AK, RK, E>, where LM is any instructional digital resource; AK is the Acquired Knowledge, a set of KIs representing the concepts acquired after taking the LM; RK is the Required Knowledge, the KIs necessary for studying the LM, i.e. the cognitive prerequisites for the AK associated to the node; E is a measure of the effort needed to study LM, supposing that the requirements in RK are met.
A LN is to be a learning object compliant with the IEEE-LOM specifications\(^1\). Since the data structure we devised for LN, specifies the RK and AK through KIs listed under appropriate IEEE-LOM metadata \([14]\), the compliance is provided.

**Definition 3 (Learning Domain and Knowledge Domain).** The learning domain LD is the repository where LNs, related to a given subject matter, are stored in order to allow the construction of courses about that subject matter. The knowledge domain about the subject, KD, is the set of KIs enclosed in the AK parts of the LNs in LD.

**Definition 4 (Starting Knowledge).** The SK is a subset of the KD, representing the knowledge that the student already possess, prior of the course.

**Definition 5 (Target Knowledge).** It is the subset TK of KD, representing the goal of a course, i.e. the knowledge to be possessed by the student after the course.

**Definition 6 (Learning Object Sequence).** A LOS is a sequence of LNs, \(\{LN_1, ..., LN_n\}\), created by a sequencing algorithm.

So a course (the actual student's learning activity) is a LOS defined by selecting LNs, through a sequencing algorithm, taking care of its goal (TK) and of the individual student's personal traits (such as SK). Other personalization aspects, managed by the different algorithms, can be included in LS-Lab (see Def.7 below). Each algorithm made available into LS-Lab ought to satisfy the following minimal requirements, being 1) goal driven (i.e. building a course to cover the TK), 2) able to manage KIs, 3) coherent with IEEE-LOM compliant LNs, and 4) able to manage the LNs' RK and AK.

**Definition 7 (Super Student Model).** A SSM gradually accumulates in LS-lab the student models managed by the system. From here a student model, or a combination of student models, can be applied when a given algorithm is run. SSM is initially empty. Let SSM be the overall student model while the \(A_1, ..., A_i\) sequencing algorithms are available in the system, each one characterized by its student model \(SM_k\) with \(k \in (1..i)\): then, when algorithm \(A_{i+1}\) is added, which is based on the student model \(SM_{i+1}\), it is \(SSM_{i+1} = SSM_i \cup SM_{i+1}\).

The design of LS-Lab and its functional schema have been presented in \([14]\). Here we give a synthetic description of the system. Once an algorithm has been added to the system, the GUI allows performing experiments. An experiment consists in selecting i) an algorithm (or more, if available), ii) a learning domain, iii) a TK, iv) a student model, and then activating the selected algorithms, so to produce, accordingly, comparable learning sequences for the student (model). The main components of the systems are: i) the Student Model Generator (SMG), that “filters” from the SSM the SM(s) associated with the algorithm; ii) the Knowledge Domain Generator (KDG), that “adapts” the knowledge domain so to make the selected sequencing algorithm applicable; iii) the Target Knowledge Generator (TKG), that “adapts” the TK to be managed by the applied algorithm; iv) the Sequencing Engine (SE), that contains the available sequencing algorithms, \(A_1, ..., A_n\). Each algorithm, \(A_i\), takes as input the information processed by the three previous modules: SMG produces SM, KDG produces the adapted knowledge domain KD, and TKG produces the accommodated TK. The execution of \(A_i\) on input \(<TK, KD, SM_i>\) produces the learning object sequence \(LOS_i\).

### 3 Comparison and Evaluation through LS-Lab

In this section we briefly describe the algorithms that are presently integrated into LS-Lab. Then we describe the metrics devised so far, together with an application experiment.

\(^1\)http://ltsc.ieee.org/wg12/
3.1 Sequencing algorithms for LS-lab

In [14] two sequencing algorithms have been included in LS-Lab. The first algorithm is used by the LS-Plan system. It is based on the fact that personalization problems can easily be seen as planning problems [13]. LS-Plan uses both student's previous knowledge and Learning Styles (LS) according to the Felder and Silverman's model (FS) [10]. The second algorithm is implemented into the KBS-Hyperbook system [11], it is based on a topological sort algorithm, and it does not deal with LS.

Here, we augmented the system with the algorithm used in the IWT system [17]. The algorithm starts from three ontology relations: HasPart, RequiredBy, and SuggestedOrder. The HasPart nodes do not actually coincide with actual learning material; they are used to express a higher level of concepts in the ontology. These graph of relations is firstly extended by the explicit relations RequiredBy, then a topological sort algorithm is applied. Since IWT is proprietary, we implemented the algorithm basing on [17]. IWT uses LS under the model FS, as well as LS-Plan does, with a different strategy to state the suitable learning style.

3.2 LS-Lab at work

To enclose a new algorithm, LS-Lab applies a three-tiered process, as it follows.

**Domain.** Basically, what is needed here is 1) a domain parser able to re-interpret the LN definitions to make them usable by the algorithm, and 2) the provision for necessary additional features of the LN, when the algorithm intends to manage them. In the case at hand, the LN are specified compliant with IEEE-LOM, including appropriate metadata for the RK and AK components; and this is enough information to let the KBS work: the parser will create a graph of LN's, and the algorithm will manage to select the appropriate LN's and sequence them according with students' knowledge. Instead, LS-Plan and IWT deal with LS, so the LN have to be equipped with LS weights and with resource types (that the framework allows to store in LOM metadata).

**Student model.** The SSM starts from an empty set and accumulates the student models managed by the algorithms in the framework. For KBS the student's SK is enough: SSM₁ = SSM₀ ∪ SK. To add LS-Plan and IWT algorithms we have to let the overall SSM grow by the definition of the appropriate interpretation of LS: SSM₂ = SSM₁ ∪ {LS}. In general, with the uploading of a new algorithm the SSM grows indeed.

**Target knowledge.** TK is a set of KIs, and it can be directly used by the KBS and IWT algorithms. For the LS-Plan algorithm TK is compiled as a PDDL problem specification.

3.3 Criteria and metrics for comparison

Two basic attitudes could be considered for the teacher's assessment of a LOS. In a subjective comparison attitude, the teacher is left (alone) at liberty to judge appropriateness, completeness and suitability of the sequence. In order to concentrate on a more objective comparison attitude, we use metrics and heuristics to measure certain characteristics and qualities of the LOS and offer the computed results to support teacher's LOS evaluation.

In the following we show three options for the mentioned metrics; such set of metrics is by no means exhaustive, yet they revealed to be very useful to experiment with the present implementation of our framework. Throughout the subsections, we assume that L = {LN₁,..., LNₙ} is the LOS to measure, and M(L) is the metric function applied to L.

3.3.1 Overall_Effort metrics

One possible way to measure a LOS is by computing the cognitive effort implied by the LN's of the sequence. We have defined the effort (cf. Def.2) as a value associated to a LN, and we did not state a univocal meaning for such a characteristic: it might represent the time expected to go through the learning content of the node, or the complexity of such contents. Moreover, presently, we do not consider that the effort endured on a LN might influence the
effort to be endured on another LN in the sequence (i.e. the effort of a LN is the same if taken alone than it is in a sequence). So, the metrics $M_E$ allows comparing LOSes basing on the overall effort required by their respective set of LNs ($L_{N_i},E$ is the effort associated to $L_{N_i}$):

$$M_E(L) = \sum_{i=1}^{n} L_{N_i},E$$

The less the effort, the simpler (or, may be, shorter) is the course.

3.3.2 Overall_Acquired_Knowledge metrics

This metrics allows to compare LOSes by measuring how redundantly a LOS does actually cover the gap SK-TK ($L_{N_i},AK$ is the acquired knowledge associated to $L_{N_i}$):

$$M_{AK}(L) = \cup_{i=1}^{n} L_{N_i},AK$$

The smaller $M_{AK}(L)$ is, the more directly the course goes to the point (i.e.: the TK). Of course a “more direct course” is not necessarily “simpler” in terms of $M_E(L)$.

![Figure 1](image.png)

Figure 1: A graph of learning nodes.

3.3.3 Overall_p-effort metrics

According to Def.2, the LNs repository $LD$ can be seen as a graph of *propaedeutical* relations, where two nodes (e.g. the A and B in Fig. 1) are connected by an arc when there exist a relation of direct derivation between them, namely (some of) the knowledge acquired through the predecessor is part of the knowledge required by the successor.

We suppose that studying two subsequent LNs that are in direct derivation, such as (A, B) in Fig.1, does impose a lesser *propaedeutical-related effort* (*p-effort*) on the learner, than taking two subsequent propaedeutical-independent LNs, such as (A, C) in Fig.1. Hence, the third metrics we propose deals with *p-effort*: the motivating idea for the measure is in that after having studied A, it is indeed different studying B rather than C (i.e. there are different levels of *p-effort* to meet):

- In the former case, B, the learner uses concepts just learned in A to build some more knowledge (in B).
- In the latter case, the learner is supposed to manage with her *Cognitive State* (the set of knowledge items possessed at the present time of the course), in two manners:
  - temporarily “dismissing” some concepts, just acquired in A (cf. funct. $fwd$ below), and
  - recollecting/use the already known concepts that are required by C (cf. $up$ below).

We label each pair of adjacent nodes ($L_{N_i}, L_{N_{i+1}}$) in $L$, as *p-dependent* if there is an arc in the LD graph between them. A *p-dependent-chain* is a subsequence of L, $\{L_{N_{i_1}},...,L_{N_{i_m}}\}$ such that each pair ($L_{N_{i_j}}, L_{N_{i_{j+1}}}$) is p-dependent. Given ($L_{N_{i_j}}, L_{N_{i_{j+1}}}$) as above, the *p-effort* related to studying $L_{N_{i+1}}$ after $L_{N_i}$ is null when the pair is p-dependent, otherwise it is computed basing on the path existing between them in the $LD$, namely on the length of the minimal path between the nodes – $cca(LD,L,L_{N_i}, L_{N_{i+1}})$ is a closest common ancestor of the nodes in the repository and in L, e.g.: for B and C it is node D in Fig. 1):

$$up(LD, L, L_{N_i}, L_{N_{i+1}}) = n. \text{ of arcs to travel back from } L_{N_i} \text{ to } cca(LD,L,L_{N_i}, L_{N_{i+1}})$$

$$fwd(LD, L, L_{N_i}, L_{N_{i+1}}) = n. \text{ of arcs to travel forward from } cca(LD,L,L_{N_i}, L_{N_{i+1}}) \text{ to } L_{N_{i+1}}$$

5(8)
We then compute the p-effort of the pair, depending on its being p-dependent (A) or p-independent (B) as

\[
p_{-\text{eff}}(SM, L, i, i + 1) = \begin{cases} 
0 & \text{if } \text{Alice}\text{-}LN_{i+1} \text{ is p-dependent} \\
\frac{1}{2} \cdot \text{up}(L, LN_i, LN_{i+1}) + \frac{1}{4} \cdot \text{fwd}(L, LN_i, LN_{i+1}) & \text{if } \text{Alice}\text{-}LN_{i+1} \text{ is p-independent}
\end{cases} \quad (A)
\]

then the overall p-effort of the sequence for the student model is the sum of all the pairs

\[
M_{p-\text{eff}}(L) = \sum_{i=1,\ldots,n-1} p_{-\text{eff}}(SM, L, i, i + 1)
\]

### 3.4 An example of comparison

In Fig. 2 we show the system interface in the preliminary phase of the process. Among the learner's data, her LS (learning style) profile is specified, in terms of the FS model: for each one of the four dimensions, an integer between $-11$ and $+11$ is given (for instance, in the dimension of perception - with orientation given as sensing or intuitive - a value of $-11$ means fully sensing and a value of $+11$ means fully intuitive).

![Figure 2: Experimental use of LS-Lab data setting phase.](image)

The lower part of the interface (just close to the submit button) set the selection of the available algorithms and the metrics. The domain related to the example is presented in Fig.3.

In this context, we input the following student model:

- $SK = \{\text{rec_runtimestack}_k, \text{rec_runtimestack}_a\}$;
- $TK = \{\text{rec_exercises}\}$;
- $LS = -10; 8; 8; 10$ (Active-Intuitive-Verbal-Global, cf. discussion on Fig.2 - first item).

In Table 1 we show the LOSes produced by the three algorithms supported so far by LS-Plan. From the results shown in the table, we observe that

- LS-Plan has one additional LN (id14) and, consequently a bigger effort;
- all the three sequences present the same LNs, proposed in different order;
- the node Recursive Function Intro is proposed in LS-Plan with a different LS than in the other two algorithms. The choice of LS for LS-Plan is motivated from the fact that the LS associated into the pool to the node id3 is $[-11; -5; -9; -2]$, while it is $[8; -1; 11; 1]$ for id4. Basing on the Euclidean distance computed by LS-Plan, node id4 is the closest node to the student’s LS (the Euclidean norm is 24.55 for id3 and 22.24 for id4). In the case of IWT the way of computing the closest node is different: IWT associates LS to the node basing on resources type, according to the IEEE-LOM LearningResourceType tag.
- measures of p-effort reveal that: 1) KBS and LS-Plan have a similar behaviour: the difference of 0.75 between the measures of the related LOSes is due to node id14, that is
only in the second LOS; 2) IWT provides the lowest value, because it uses earlier the knowledge already possessed by the student, suggesting to take node id9 before dealing with the functional approach to recursion, given in nodes id3, id5 and id6.

On these bases the teacher has some elements to judge/compare algorithms’ behaviour.

<table>
<thead>
<tr>
<th>KBS</th>
<th>LS-Plan</th>
<th>IWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_E = 13$</td>
<td>$M_E = 16$</td>
<td>$M_E = 13$</td>
</tr>
<tr>
<td>($\text{effort}$)</td>
<td>($\text{effort}$)</td>
<td>($\text{effort}$)</td>
</tr>
<tr>
<td>$M_{p\text{-eff}} = 2.25$</td>
<td>$M_{p\text{-eff}} = 3.00$</td>
<td>$M_{p\text{-eff}} = 1.75$</td>
</tr>
<tr>
<td>($\text{distance}$)</td>
<td>($\text{distance}$)</td>
<td>($\text{distance}$)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id1:Unit description</th>
<th>id1:Unit description</th>
<th>id1:Unit description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id2:Recursive programs</td>
<td>id2:Recursive programs</td>
<td>id2:Recursive programs</td>
</tr>
<tr>
<td>id3:Rec. Function intro</td>
<td>id4:Rec. Function intro</td>
<td>id9:Rec. r/t stack examples</td>
</tr>
<tr>
<td>id5:Rec. Function StrgReverse</td>
<td>id5:Rec. Function StrgReverse</td>
<td>id3:Rec. Function intro</td>
</tr>
<tr>
<td>id6:Rec. Function examples</td>
<td>id5:Rec. Function StrgReverse</td>
<td>id3:Rec. Function intro</td>
</tr>
<tr>
<td>id9:Rec. r/t stack examples</td>
<td>id6:Rec. Function examples</td>
<td>id6:Rec. Function examples</td>
</tr>
<tr>
<td>id10:Recursion exercises</td>
<td>id14:Recursive list</td>
<td>id10:Recursion exercises</td>
</tr>
<tr>
<td>id10:Recursion exercises</td>
<td>id10:Recursion exercises</td>
<td>id10:Recursion exercises</td>
</tr>
</tbody>
</table>

Table 1: A comparison of the course sequences produced by the algorithms available in LS-Lab.

Figure 3: The graph of LNs in the Recursion Domain. Multiple indexes point out nodes given in different LS-aware versions: id3_4, id11_12_13_14, id16_17 (indexes correspond to actual learning material).

4 Conclusions and Future Work

We presented the LS-Lab framework, a developing system to support comparison and evaluation of course sequencing algorithms for teachers and, in general, e-learning experts. Besides presenting the current state of the framework, we added here considerations about the support to the evaluation of the course sequences produced by the algorithms included in the system. In particular we discussed about possible metrics to adopt, in order to provide the assessing teacher with a range of values on whose ground to state opinions and evaluations on the algorithms and their outputs. We also have shown a limited example of application for such metrics. The work is in progress and we plan to let other algorithms join the pioneers presently included in the framework. Future work for making the system more effective is to
provide instruments that automatically take in a set of different student models and generate statistic about the behaviour of the learning paths.

References:


[16] moodle.org open source e-learning tool. As of April 2010, information available through http://moodle.org/about/


Author(s):

Carla Limongelli, Giulia Vaste, Roma Tre University, Dept. of Computer Science and Automation, Roma, Italy, email {limongel, vaste}@dia.uniroma3.it

Filippo Sciarrone, Open Informatica srl, E-learning Division, Pomezia, Italy, email f.sciarrone@openinformatica.org

Marco Temperini, Sapienza University of Roma, Dept. of Computer and System Sciences, Roma, Italy, email marte@dis.uniroma1.it