Journal of Computing and Information Technology - CIT 17, 2009, 4, 335–346
doi:10.2498/cit.1001320

Toward a Better Syllabus: Entropy-driven Introspection for Alternative Lesson Plans

Anestis A. Toptsis

Department of Computer Science and Engineering, York University, Toronto, Canada

Developing a lesson plan (or syllabus) for a course can be a very time consuming process. Once the instructor has a pool of topics to be taught in the course, the organization of those topics in a form that facilitates understandability by the students as well as efficiency with respect to the order of presentation can be an elusive task – many times considered to be an art that distinguishes good from exceptional teachers. In this paper we present a method that is helpful in the process of choosing a better lesson plan, among several alternative lesson plans. We do not imply that the presented method is a panacea for the formation of a best lesson plan, however, the method is useful in situations when several alternative complicated and lengthy lesson plans may be contemplated for the purpose of selecting the most viable option. Our method is based on the notion of entropy, borrowed from information theory. It provides a fairly simple and quick way to decide which one – among several alternative lesson plans, is a better choice for adoption.

Keywords: education, syllabus, entropy, information theory

1. Introduction

University level teaching and learning is a complex intellectual and social process, influenced by a variety of factors such as teacher-student degree of rapport, teacher's level of knowledge and understanding of the material, students' motivation, students' background, and of course, the degree of appropriateness of the taught material for the target audience and level of understanding of the taught material. There is a plethora of literature on the subject of teaching-learning in higher education, including [1], [2], [3], [5], [6], [8], [9], [10], [11], [13], [16], [17], [18], [19], [22], [23]. Notably, the reports cover issues and establish standards on learning objects design, representation of course contents in some standard language such as XML, and preferred teaching practices.

In this paper we focus on an aspect of one of the most vital components, the syllabus of a course and present a method that aids in selecting a good lesson plan among several alternatives. By its nature, a course syllabus (or, lesson plan) plays a key role in determining the quality of a course. This is because the syllabus is a major guide of what is presented to the students during the entire duration of the course. As such, there is considerable effort expended in devising guidelines and constructive suggestions for implementing a good syllabus. Examples of such endeavors can be found in [15], [24], [4], [14], [21]. It is not uncommon that educational institutions expend significant resources in devising such guidelines, as well as in organizing seminars for all interested parties involved in course development and delivery, including faculty members and teaching assistants. The work presented here introduces a simple metric of complexity for a syllabus. The metric can be used to gauge the complexity of a syllabus and provide a simple way to compare the complexities of several alternative syllabi during the process of implementing a final lesson plan for a course. By metric we mean a number which can be used to determine the relative complexity of two or more alternative syllabi for the same course. By complexity we mean the degree of difficulty, or confusion (for the student and the teacher), or redundancy that may be embedded in a syllabus. Although the work presented here is applicable to any course of an educational institution, the experience of the author as a computer science professor points to a rather strong belief that it applies especially in computer science courses, due to their high degree of structure and modularity.

The presented method employs the notion of entropy from information theory and uses it as an indicator to guide a teacher's or course designer's decision of which among several alternatives, could be a better lesson plan for adoption.

In this paper we focus on the syllabus part of a course. Its purpose is to provide a metric that can be used to make an educated guess about the complexity of a syllabus.

The rest of the paper is organized as follows. In Section 2 we describe the details of setting up a typical syllabus which is conducive to using the proposed metric. Then we describe the metric and examples of its use. In Section 3 we provide examples from a real course typically taught in the junior/senior level of an undergraduate computer science degree program. Section 4 provides a discussion of some features of the proposed method. Section 5 summarizes our work and discusses further research ideas.

2. Proposed Method

2.1. The Syllabus

A course syllabus is typically a collection of topics that will be taught in that course. This list of topics is usually presented by the instructor during the beginning of the course and serves as a guideline of the material that the students are expected to learn in that course. A syllabus can be of varying degrees of detail. Figure 1 shows parts of a syllabus for an introductory course in computer science.

Introduction to Computer Science

- History of Computing
- My first Java Program
- Variables
- Branching
- Loops
- Etc

Figure 1. A syllabus with very little detail.

Note, a syllabus for the same course may be expressed in various degrees of detail. The objective of this paper is neither to recommend any level of detail over any other level of detail, nor to speculate on the amount of knowledge that should be included in any one course. Although these are important topics, they are outside the scope and purpose of this paper. Our assumption is that the instructor has decided on a fixed set of topics to teach in the course and his/her concern is what could be a "better way" of presenting those topics, as opposed to what topics he will present. Here, we provide a methodology of how an instructor that developed two or more versions of a syllabus for the same course can evaluate which one of those syllabi might be better to follow for teaching the course.

2.2. Instructor-site Syllabus

An *instructor-site syllabus* (ISS) is a version of the syllabus such as the one that the instructor presents to the students, but with the additional characteristic that the *dependency graph* has been created for that syllabus.

Definition 1. An instructor-site syllabus (ISS) is a set T of topics $T_1, T_2, ..., T_k$ selected to be taught in a course.

Definition 2. The *dependency graph* of an ISS is a directed graph with a set T of vertices, and a set of edges E such that

$$T = \{T_1, T_2, ..., T_k\}\ ,\ T_i \in T,$$

where T_i is a topic and

$$E = \left\{ \begin{array}{l} e|e : T_i \rightarrow T_j \\ \Leftrightarrow T_j\, depends\ on\ T_i, \\ where\ T_i, T_j \in T \end{array} \right\}.$$

Intuitively, the dependency graph of an ISS T is a directed graph in which all topics $T_1, T_2, ..., T_k$ of T appear as vertices and in which there is an edge $T_i \rightarrow T_j$ if and only if the learning of topic T_j requires prior knowledge of topicT_i.

Example 1.

Consider the following ISS (based on Figure 1):

Topic	Content of topic
T1	History of Computing
T2	My First Java Program
T3	Variables
T4	Branching
T5	Loops

The dependency graph DG(ISS) of the above ISS is shown in Figure 2.

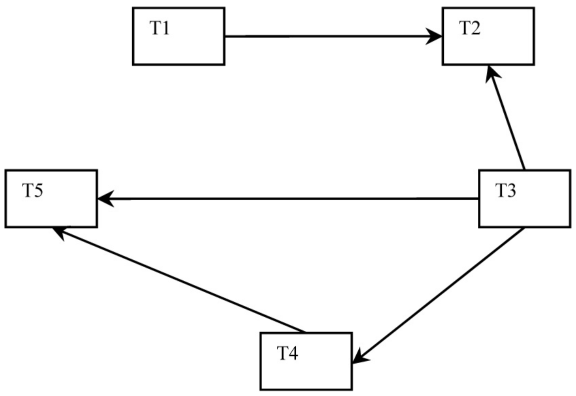

Figure 2. Dependency graph of Example 1.

Example 2.

Consider the following ISS:

Topic	Content of topic
T1	History of Computing
T2	Hardware
T3	Software
T4	Variables
T5	Integer variables
T6	Non-integer variables
T7	Branching
T8	If statement
T9	Switch statement
Loops topics omitted since they would clutter the diagram too much	

The dependency graph DG(ISS) of the above ISS is shown in Figure 3.

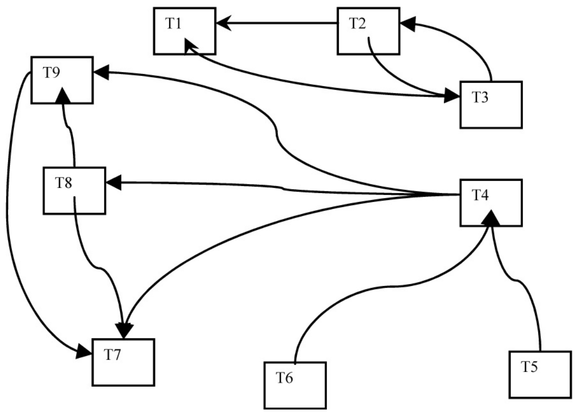

Figure 3. Dependency graph of Example 2.

As we see in the above examples, the longer the syllabus is, the more complex its DG becomes. The chosen granularity of topics affects the complexity of the DG. Example 1 is obviously a collection of topics which is unlikely to be part of any university-level course. However, it should be understood as well that each of the nodes of the DGs of Examples 1 and 2 above could be further broken down to subtopics and that such a breakdown could potentially lead to nodes of the cognitive and learning level of the nodes that appear in the Table of Example 1.

2.3. The Complexity Metric

After describing how to construct an ISS, we now describe a method that can be used to determine the degree of complexity of the given syllabus. The proposed metric relies on the notion of entropy as described by Shannon [20]. The general idea is that as a system becomes more complex, its entropy increases. The fundamental expression of entropy is

$$H = -\sum_{i=1}^{N} P_i \cdot \lg\left(P_i\right) \tag{1}$$

where H is the entropy, P_i is the probability of occurrence of an event X_i among a series of events $X_1, \ldots, X_N$, and lg is logarithm base 2. Note that the right-hand-side of equality (1) is always non-negative, since always $\sum_{i=1}^{N} P_i \cdot \lg\left(P_i\right) \leq 0$. According to the underlying theory of entropy, the higher the value of H (i.e., the higher the entropy), the more complex (or chaotic, or undesirable) a system is.

Note, the notion of complexity used here should not be confused with the complexity which is sometimes used in the context of evolution theory. In that context, higher complexity typically refers to systems with higher degrees of both modularity and integration which, in general, is considered a desirable condition [7]. Entropy, in the sense that is presented here, was originally described by Shannon in his seminal paper that accounts for the birth of information theory, in 1948. In the information theoretic context of entropy that we use here, higher degrees of entropy imply higher complexity, and in this context, higher complexity is considered an undesirable condition. Since its introduction the information theoretic notion of entropy has been used in numerous disciplines, including linguistics and software engineering. Inspired by the use of entropy-based metrics for software construction [12] we present a method of how to develop an entropic complexity metric for a syllabus. Our method assumes that there is a syllabus developed for a course and also that the DG (Dependency graph, as defined in the previous subsection) of that syllabus is available.

Definition.

Given a directed graph G, the *indegree* (ID) of a node v of G is the number of directed edges incoming to node v.

Definition.

Given a directed graph G, the *outdegree* (OD) of a node v of G is the number of directed edges outgoing from node v to some other node(s).

Definition.

Given a dependency graph DG for an instructor-site syllabus ISS (in the sense presented in the previous subsection), a *complexity equivalence class* K in DG is a set of nodes $\{v_1, v_2, ..., v_m\} \subseteq V$, in which all nodes have the same indegree, and all nodes have the same outdegree.

Notice, in the above definition, the indegree is not necessarily the same as the outdegree. Figure 4 shows two nodes a and b from a graph DG, which according to the definition above belong to the same equivalence class.

The idea is that if two topics in a syllabus (nodes in DG) are "prerequisites" for the same number of topics and also those two topics each need the knowledge of the same number of other topics before they can be learned, then those two topics are considered to be of equivalent cognitive complexity and therefore are placed in the same equivalence class.

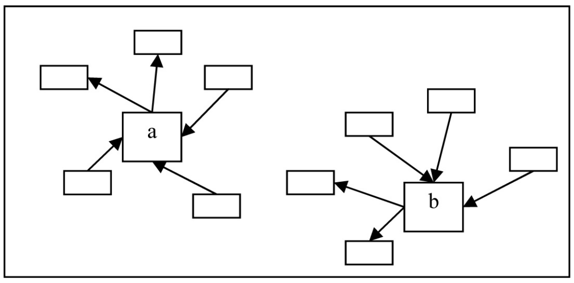

Figure 4. Two nodes a and b in the same equivalence class.

The method of how to calculate the complexity metric is described in algorithm A, below.

Algorithm A.

1. Traverse the DG of the syllabus and determine its complexity equivalent classes. Let $C_1, C_2, ..., C_k$ be the complexity equivalent classes.
2. The overall complexity H of the syllabus is then calculated as

$$H = -\sum_{i=1}^{k} P(C_i) \cdot \lg(P(C_i)).$$

The probabilities $P(C_i)$ in step 2 of algorithm A represent the *frequencies of appearance* of the different equivalence classes. For example, if $P(C_i) = 15\%$ for some class C_i, it means that topics whose complexity is the complexity represented by class C_i in the syllabus, occupy 15% of the syllabus.

2.4. Example of Algorithm A

We present an example of the workings of algorithm A to determine the entropy of a syllabus.

A dependency graph and a table with the related values for Algorithm A are shown in Figure 5 and Table 1, respectively.

Entropy:

$$H = -\sum_{i=1}^{5} P(C_i) \cdot \lg(P(C_i)) = 2.252.$$

Proposed Method

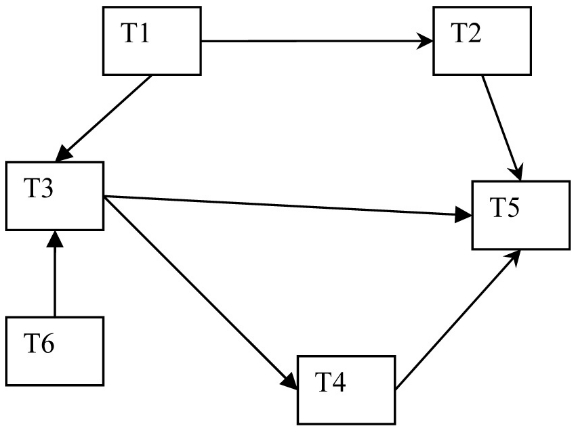

Figure 5. A dependency graph.

Topic	ID	OD	Equivalency class C_i	$P(C_i)$
T1	-	2	C1 <0,2> = {T1}	1/6
T2	1	1	C2 <1,1> = {T2, T4}	2/6
T3	2	2	C3 <2,2> = {T3}	1/6
T4	1	1	C2 <1,1> = {T2, T4}	2/6
T5	3	-	C4 <3,0> = {T5}	1/6
T6	-	1	C5 <0,1> = {T6}	1/6

Table 1. Values related to Algorithm A, for the DG of Figure 5.

3. Experimenting with a Real Syllabus

In this section we present an example of how the described complexity metric can serve to simplify a syllabus for a real course in computer science. Our testbed is an undergraduate course on Introduction to Database Systems. Such a course is taught at least once per year at the author's department and also is common course at the 3rd year of most computer science departments in universities throughout North America. It is considered to be one of the central subjects of knowledge that should be contained in a computer science curriculum [5]. Typical topics covered in this course are the ER model, SQL, relational algebra, database design in terms of normalization, and topics from transaction processing. To illustrate our complexity metric, we choose the topic of normalization. In presenting the topic of normalization, one of the central subtopics is the 3NF Synthesis process (3NFS). This process involves decomposing a set of given attributes into subsets such that the relational database schema corresponding to those subsets forms a database schema that is in third normal form (3NF), preserves dependencies, and has the lossless join property. Although it could be helpful, it is not necessary for the reader to be familiar with the workings of this process. We have identified that the following topics, as shown in Table 2, are necessary bodies of knowledge in order for a student to understand the 3NFS process.

Topic	Description
T.1.2 **3NFS**	3NFS: 3NF Synthesis
T.1.2.1 **F+**	F+: Closure of a set of FDs
T.1.2.2 **X+**	X+: Closure of a set of attributes
T.1.2.3 **LJP**	LJP: Lossless join property
T.1.2.4 **PD**	PD: Preservation of dependencies
T.1.2.5 **MC**	MC: Minimal cover
T.1.2.3.1 **FDD**	FDD: FD definition

Table 2. Topics in the 3NFS process.

A dependency graph DG of a syllabus corresponding to the above topics is shown in Figure 6.

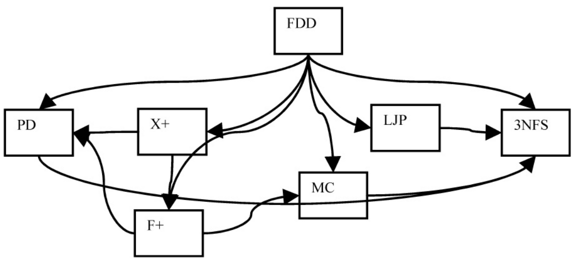

Figure 6. Dependency graph for 3NFS syllabus.

Table 3 shows the related values needed for algorithm A, from the graph of Figure 6.

Entropy calculated: H = 2.81

The complexity of this syllabus is therefore 2.81.

Topic	ID	OD	Equivalence class C_i	$P(C_i)$
3NFS	4	-	C1 <4,0> = {3NFS}	1/7
X+	1	2	C2 <1,2> = {X+}	1/7
F+	2	2	C3 <2,2> = {F+}	1/7
LJP	1	1	C4 <1,1> = {LJP}	1/7
PD	3	1	C5 <3,1> = {PD}	1/7
MC	2	1	C6 <2,1> = {MC}	1/7
FDD	-	6	C7 <0,6> = {FDD}	1/7

Table 3. Related values for Algorithm A.

3.1. Transitive Dependencies

Can we devise a better syllabus? As we observe in Figure 6, the topic FDD is a prerequisite for many other topics: MC, F+, X+, LJP, PD, 3NFS. The legitimacy of the dependencies as listed in Figure 6 is not disputed, however, we observe that there are several *transitive dependencies* in Figure 6.

One such transitive dependency is shown in Figure 7. This transitive dependency means that knowledge of topic F+ requires knowledge of topics X+ and FDD; and knowledge of topic X+ requires knowledge of topic FDD. So in devising a lesson plan an instructor might opt to teach the topic FDD first, followed by the topic X+, followed by F+. In such a case, Figure 7 would become as shown in Figure 8.

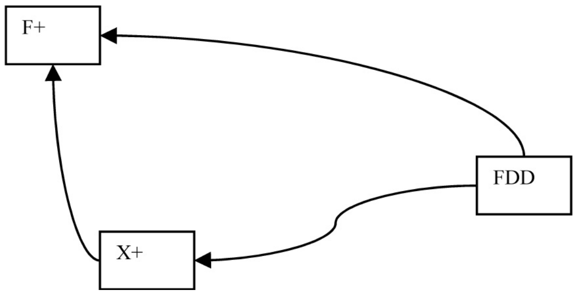

Figure 7. Transitive dependency in topics.

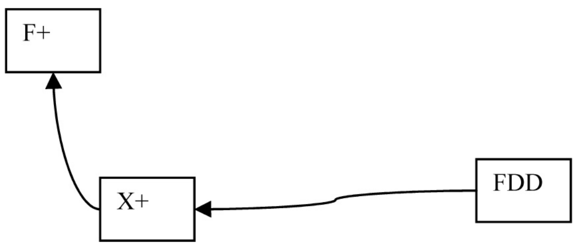

Figure 8. Transitive dependency removed.

Figure 9 shows what the DG of Figure 6 becomes after some (but not all) of the transitive dependencies have been removed.

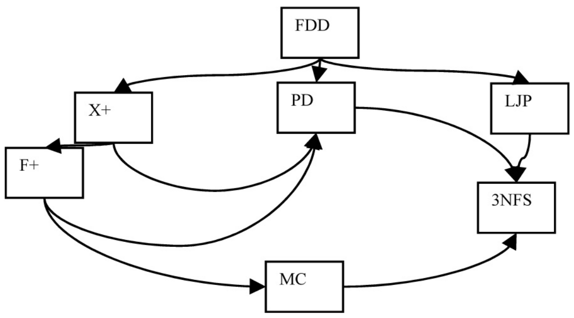

Figure 9. Some transitive dependencies have been removed.

Table 4 shows the related values needed for algorithm A, from the graph of Figure 9.

Topic/Node	ID	OD	Equivalence class C_i	$P(C_i)$
3NFS	3	–	C1$< 3,0 >$={3NFS}	1/7
X+	1	2	C2$< 1,2 >$={X+,F+}	2/7
F+	1	2	C2$< 1,2 >$={F+,X+}	2/7
LJP	1	1	C3$< 1,1 >$={LJP,MC}	2/7
PD	3	1	C4$< 3,1 >$={PD}	1/7
MC	1	1	C3$< 1,1 >$={LJP,MC}	2/7
FDD	-	3	C5$< 0,3 >$={FDD}	1/7

Table 4. Values related to Algorithm A, for the DG of Figure 9.

Entropy: H = 2.236.

As we see, the entropy of the second syllabus is decreased from 2.81 in Figure 6 to 2.236 in Figure 9. This indicates that following a lesson plan based on the second syllabus rather than the first one, could make the overall topic of 3NFS more understandable by the students as well as easier for the instructor to deliver such a plan. Note, the edge from X+ to PD is not removed although we have a transitive dependency as shown in Figure 10.

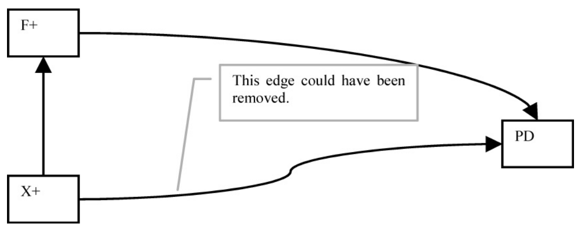

Figure 10. Transitive dependency.

There are two transitive dependencies left in Figure 9.

Transitive dependency (I):

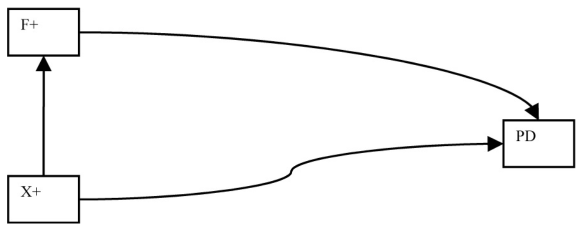

Figure 11. Transitive dependency I.

Transitive dependency (II):

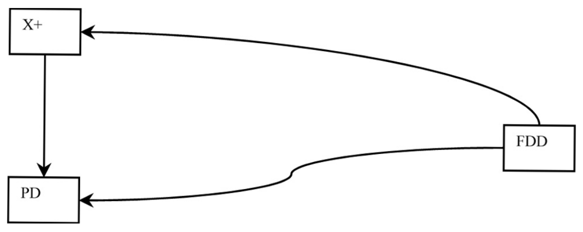

Figure 12. Transitive dependency II.

If we remove the dependency shown in Figure 11, then Figure 9 becomes as shown in Figure 13.

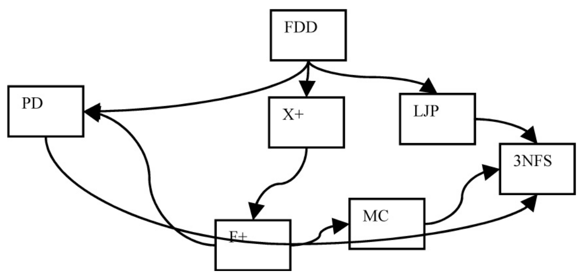

Figure 13. Graph of Figure 9 with transitive dependency (I) removed.

Table 5 shows the related values needed for algorithm A, from the graph of Figure 13.

Topic/Node	**ID**	**OD**	**Equivalence class C_i**	$P(C_i)$
3NFS	3	–	C1$< 3, 0 >$={3NFS}	1/7
X+	1	1	C2$< 1, 1 >$= {X+,LJP,MC}	3/7
F+	1	2	C3$< 1, 2 >$={F+}	1/7
LJP	1	1	C2$< 1, 1 >$= {X+,LJP,MC}	3/7
PD	2	1	C4$< 2, 1 >$={PD}	1/7
MC	1	1	C2$< 1, 1 >$= {X+,LJP,MC}	3/7
FDD	–	3	C5$< 0, 3 >$={FDD}	1/7

Table 5. Values related to Algorithm A, for the DG of Figure 13.

Entropy: H = 2.128.

Note that H dropped from H=2.236 in the graph of Figure 9, to H=2.128 in Figure 13. Note, removing the dependency shown in Figure 11 caused the dependency shown in Figure 12 to disappear! So there is no other dependency to remove from the graph of Figure 9. Further examination of the graph of Figure 13 reveals that there are no more dependencies.

3.2. Producing a Final Syllabus

Given the graph shown in Figure 13, how do we proceed to devise a syllabus? To achieve this, we perform topological sort on the graph of Figure 13.

(Topological sort)

1. *Identify a node with 0 (zero) indegree.*
2. *Output this node*
3. *Remove all the edges that connect this node to its immediate successor nodes.*
4. *Repeat (1) (2) (3), until all nodes have been output.*

The output of this topological sort is a syllabus, i.e., a sequence of topics to be taught in that sequence. In the case of the graph of Figure 13 the output of a topological sort is the sequence FDD, LJP, X+, F+, MC, PD, 3NFS.

Note, however, that a topological sort is meaningful only if the DG is acyclic. In case that the DG contains a (directed) cycle, the topological sort leaves out the vertices (topics) that participate in that cycle and as a result it produces a syllabus that does not include all the topics that need to be taught.

Definition. A *reasonable* syllabus is a syllabus in which no two topics are mutual prerequisites. That is, if T1 and T2 are two topics in a reasonable syllabus, and T1 is prerequisite knowledge for T2, then T2 cannot be prerequisite knowledge for T1.

Proposition.

If a DG G corresponds to a reasonable syllabus, then G is acyclic.

Proof.

Let G have a cycle T1 → T2 →...→ Tk → T1. Then the path T1 → T2 →...→ Tk implies that

T1 is prerequisite knowledge for Tk (1)

Also, the edge Tk → T1 implies that

Tk is prerequisite knowledge for T1 (2)

(1) and (2) are contradictory in a reasonable syllabus. Therefore, G is acyclic.

According to the above proposition, a final syllabus can be produced by performing topological sort on its DG, as long as the syllabus corresponding to that DG is *reasonable*, i.e., it does not contain any two topics which are mutual prerequisites.

4. Discussion

As we saw in the above example, modifying the DG of a syllabus can lead to lower entropy and a simpler lesson plan. We have identified a few DG configurations that produce extremely low entropies.

DG Configuration A: Chain

In this configuration the learning topics are arranged in a linear chain, as shown in Figure 14.

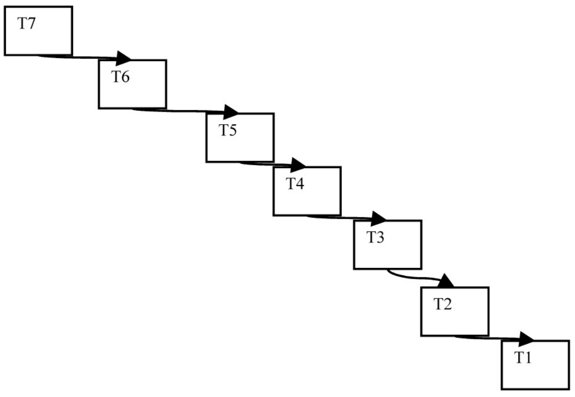

Figure 14. Chain configuration.

Topic/Node	**ID**	**OD**	**Equivalence class C_i**	$P(C_i)$
T1	1	-	C1 <1,0> = {T1}	1/7
T2	1	1	C2 <1,1> = {T2, T3, T4, T5, T6}	5/7
T3	1	1	C2 <1,1> = {T2, T3, T4, T5, T6}	5/7
T4	1	1	C2 <1,1> = {T2, T3, T4, T5, T6}	5/7
T5	1	1	C2 <1,1> = {T2, T3, T4, T5, T6}	5/7
T6	1	1	C2 <1,1> = {T2, T3, T4, T5, T6}	5/7
T7	-	1	C3 <0,1> = {T7}	1/7

Table 6. Values related to Algorithm A, for the DG of Figure 14.

Entropy: H = 1.1488 (very good entropy)

DG configuration B:

1. **All-Into-One**
2. **One-To-All**

In these configurations the learning topics are arranged in such a way that all but one topics solely contribute to the learning of the one remaining topic, as shown in Figure 15, or just one topic is needed for the learning of several other topics, as shown in Figure 16.

The DG of Figure 15 has two classes, C1 = {T1, T2, ..., T6}, C2 = {T7}, and an entropy H = 0.5917. The DG of Figure 16 has also two classes, C1= {T1}, C2 = {T2, T3, ..., T7}, and the same H = 0.5917. The low entropies indicate that if some topics can be taught in this manner, then it could be more

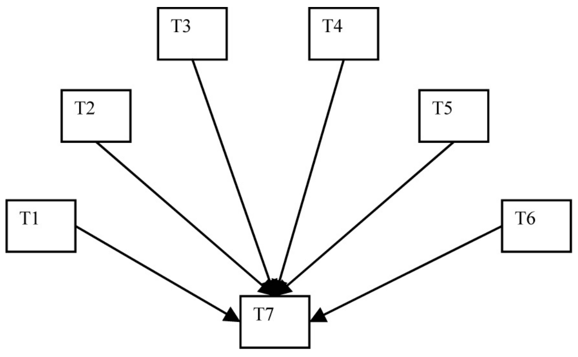

Figure 15. All-into-one configuration.

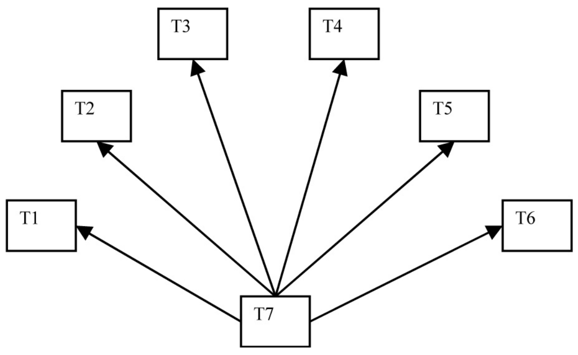

Figure 16. One-to-all configuration.

beneficial for the student's understanding. Either of these configurations produces very low entropy. Of course, not all syllabi exhibit properties that can be configured in ways shown in the above configurations. So these configurations are presented here as "extreme" cases and with the understanding that even if an instructor identifies such a pattern in the list of topics to be taught, he/she may still opt not to follow a lesson plan as indicated by the above figures. The reason is that such configurations may have low entropy in terms of structural complexity, but they tend to produce rather boring teaching plans. Examples of topics that could subscribe to the above configurations are learning basic terminology entries of some area of study.

A best, albeit unrealistic case: note that if the DG of a syllabus is fully connected as shown in Figure 17, then its entropy is zero, which is the best possible value of entropy.

Since in a fully connected DG all nodes have the same indegree and the same outdegree, there is only one equivalency class. Because of that, $P(C_i) = 1$ and therefore $\lg(P(C_i)) = 0$, which gives H $= 0$, the best possible entropy. However, it should be obvious that such a DG is not really possible, since such a DG means that every topic prerequisites knowledge of every other topic before it can be learned. Recall, according to our definition at the end of Section 3, this is *not* a reasonable syllabus. On the other hand, the low entropy of such a structure may indicate that *repetition* in presenting topics during a class, may be extremely beneficial for the students' understanding and absorption of all topics. This is something that most teachers of course know. At the same time, the degree of repetition that is required by the diagram is most likely practically unattainable in any classroom environment.

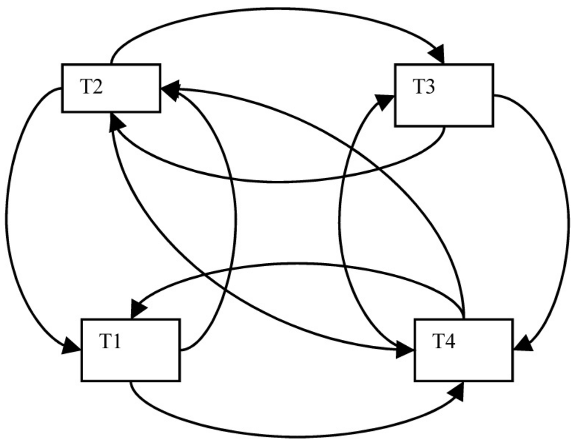

Figure 17. Unrealistic case for a syllabus.

5. Conclusion and Further Research Directions

We presented a method for selecting a good lesson plan among several alternatives. The presented method employs the notion of entropy from information theory and uses it as an indicator to guide a teacher's or course designer's decision of which among several alternatives could be a better lesson plan for adoption. To the best of our knowledge the use of entropy for this purpose is novel. Although the notion of entropy has been utilized in the past as a metric in software and linguistics contexts, there is no reported research of entropy utilization for educational purposes and especially for measuring syllabus complexity.

Although the presented method is meant to serve as an indicator that helps choose among alternatives rather than as a process of devising a better lesson plan in the absence of alternatives, we also identify some occasions when certain optimizations of an existing plan can produce a better plan. These are the cases of removal of transitive dependencies in the lesson plan, as well as the lesson plan patterns presented in Section 4.

In the future, based on the same notion of entropy that we use here, we are interested in developing mechanisms that aid in the development of a better lesson plan, instead of only help selecting one plan among existing alternatives. Also, it would be of interest to investigate the extent to which the optimizations presented in Section 4 are beneficial. For example, is it always beneficial to the overall lesson plan if *part* of the plan is converted to the chain model of Section 4? Note, this is not obvious, since removal and/or addition of edges in the syllabus graph is typically accompanied by side-effects to the entropy metric due to the modifications in the overall structure of the syllabus graph. It is also of interest to investigate how *non-transitive repetitions* in a lesson plan impact its complexity.

Another interesting issue arises from the definition of *alternative* lesson plans. In this paper we consider two lesson plans La and Lb to be alternative if they *contain the same topics*. Based on this definition, plan La is preferred over plan Lb if the entropy of La is smaller than the entropy of Lb. However, on further thought, several interesting issues arise. Denote by

- Sa and Sb: the sets of topics contained in La and Lb,
- |Sa| and |Sb|: the number of topics in La and Lb,
- Ha and Hb: the entropies of La and Lb.

According to the discussion in this paper, if $|\mathrm{Sa}| = |\mathrm{Sb}|$ and $\mathrm{Ha} < \mathrm{Hb}$, then La is preferable to Lb. That is, a lesson plan with smaller entropy is preferable to a lesson plan with bigger entropy, provided that both plans are derived from the *same* set of topics. However, if two plans La and Lb are derived from two different sets of topics Sa and Sb, then there are three possible cases for Sa and Sb.

(A) $Sa \subset Sb$, or

(B) $Sb \subset Sa$, or

(C) $Sb \cap Sa = Sc$, with $Sc \neq Sa$ and $Sc \neq Sb$

Cases (A) and (B) above are similar and they mean that one syllabus contains more amount of knowledge than the other syllabus. For example, in (A), syllabus Sb contains more knowledge than syllabus Sa. As such, comparing the entropies of La and Lb is not an applicable (or, for that matter, meaningful) measure of comparison of the complexities of La and Lb. This is because the entropy metric described in this paper assumes that the alternative comparable lesson plans are derived from the same set of topics i.e., that for two alternative lesson plans La and Lb the corresponding sets of topics are identical. (i.e., Sa = Sb). If $Sa \subset Sb$, then Sb contains more knowledge than Sa and thus comparing the entropies of the corresponding lesson plans is irrelevant. The same comments apply for cases (B) and (C). In the last case, (C), the corresponding sets of topics Sa and Sb contain not only different amounts of knowledge, but also different *kinds* of knowledge. It is interesting to investigate how (and if) lesson plans that derive from cases such as (A), (B) and (C), compare. However, such research would require the introduction of metrics beyond the entropy metric that is used here to compare structural complexities of plans deriving from the same set of topics. An idea would be to quantify the amount of knowledge by the size of the data stream conveyed by any given plan and then employ error correcting codes, with the assumption that more corrective effort means a higher degree of cognitive effort from the receiving agent (the student, in our case). At this point, unfortunately, we do not have any specifics for formulating this problem.

Finally, it is of interest to devise some way to evaluate the presented method. As it stands the evaluation is done by the syllabus designer, and it is thus prone to a human expert's subjective opinion. It may be argued that such an evaluation could be very subjective, and we do not disagree. At the same time, it is pointed out that the teacher's evaluation of the lesson plan is probably the only criterion used during the long history of lesson planning. Moreover, it is pointed out that it is also impossible to test the method in real-life situations; this would amount to executing several alternative lesson plans on a real audience and then deciding if the one favored by the live participants is also the one selected by our method! Such an experiment is simply impractical (if not counter-pedagogical as well) given the time required to perform such tests, as well as other factors involved in the teaching process (presentation ability of the instructor, audience background, receptive mood, etc).

Acknowledgment

I would like to thank the anonymous referees whose comments helped improving the paper.

References

[1] W. K. Adams, S. Reid, R. LeMaster, S. B. McKagan, K. K. Perkins, C. E. Wieman, A study of educational simulations part I – engagement and learning. *Journal of Interactive Learning and Research*, *Journal of Interactive Learning Research*, Volume 19, Issue 3, July 2008.

[2] W. K. Adams, S. Reid, R. LeMaster, S. B. McKagan, K. K., Perkins, C. E. Wieman, A study of educational simulations part II – interface design. *Journal of Interactive Learning and Research* (in press).

[3] Bain, Ken, What the Best College Teachers Do. *Harvard University Press*, 2004.

[4] Becta, Packaging and publishing learning objects: Best practice guidelines. January 2005, (last accessed July 7, 2009). `http://www.becta.org.uk/`, `http://www.edtechpost.ca/wordpress/2005/03/09/BECTA's-Packaging-and-Publishing-LOs:-Best-Practice-Guidelines`.

[5] Computing Curricula 2005, The Overview Report, *The ACM, AIS, IEEE-CS Joint Task Force for Computing Curricula 2005*, 30 September 2005. `http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf` (last accessed July 7, 2009).

[6] V. N. Convertini, D. Albanese, A. Marengo, V. Marengo, M. Scalera, The OSEL Taxonomy for the Classification of Learning Objects. *Interdisciplinary Journal of Knowledge and Learning Objects*, Volume 2, 125–138, 2006.

[7] M. Csikszentmihalyi, The Evolving Self. *Harper*, 1994.

[8] Dijkstra, E., Note EWD447 (1974) On the role of scientific thought. Reproduced in *SelectedWritings on Computing: A Personal Perspective*, *Springer-Verlag*, Berlin, 1982, ISBN 0–387–90652–5.

[9] Dijkstra, E., Note EWD1036-0 (1988) On the cruelty of really teaching computer science. Available at *University of Texas at Austin, Dijkstra Archives*.

[10] N. Friesen, Best Practice for Learning Object Metadata, *Metadata Forum*, September 19, 2003, CanCore, `http://www.cancore.ca/`

[11] N. Friesen, Connecting Collections: An overview of approaches. February 15, 2006, `http://www.cancore.ca/protocols_en.html` (last accessed July 7, 2009).

[12] Harrison, W., An Entropy-based Measure of Software Complexity. *IEEE Transactions on Software Engineering*, vol. 18, no. 11, 1992, pp. 1025–1029.

[13] IEEE Draft Standard for Learning Object Metadata. *IEEE Document 1484.12.1-2002*, 15 July 2002, `http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html`, (last accessed July 7, 2009).

[14] MSU Teaching Thought #13, Creating the "Complete" Syllabus, *Michigan State University Teaching Assistant Programs*. `www.tap.msu.edu/PDF/thoughts/tt13.pdf` (last accessed July 7, 2009).

[15] C. C. McGeoch, Research in the Curriculum, and the Web. *ACM Computing Surveys 28(4es)*, December 1996.

[16] K. Perkins, W. Adams, M. Dubson, N. Finkelstein, S. Reid, C. Wieman, PhET: Interactive Simulations for Teaching and Learning Physics. *The Physics Teacher* Vol. 44, January 2006, 18–23.

[17] Rogers, C., Freedom to Learn for the 80's. *Charles Merrill Publishing Company*, 1983.

[18] H. Roumani, Practice What You Preach: Full Separation of Concerns in CS1/CS2. *SIGCSE'06*, March 1–5, Houston, Texas, USA, p. 491, 2006.

[19] J. F. Sanford, L. M. Sztandera, Thoughts on the future of education in information technology. *Int. J. Teaching and Case Studies*, Vol. 1, No. 1/2, 23–32, 2007.

[20] C. E. Shannon, A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol. 27, pp. 379–423, 623–656, 1948.

[21] K. Thompson, F. Yonekura, Practical Guidelines for Learning Object Granularity from One Higher Education Setting. *Interdisciplinary Journal of Knowledge and Learning Objects*, Volume 1, 163–179, 2005.

[22] A. B. Tucker et al, Strategic Directions in Computer Science Education. *ACM Computing Surveys*, Vol. 28, No. 4, December 1996, 836–845.

[23] I. Varlamis, I. Apostolakis, The Present and Future of Standards for E-Learning Technologies. *Interdisciplinary Journal of Knowledge and Learning Objects*, Volume 2, 59–76, 2006.

[24] M. J. V. WOOLCOCK, Constructing a Syllabus: A Handbook for Faculty, Teaching Assistants and Teaching Fellows. *Brown University, HARRIET W. SHERIDAN CENTER for Teaching and Learning*, 2005. www.brown.edu/sheridan_center.

Received: August, 2008
Revised: July, 2009
Accepted: August, 2009

Contact address:
Anestis A. Toptsis
Dept. of Computer Science and Engineering
York University
4700 Keele Street, Toronto
Ontario, M3J1P3, Canada
e-mail: anestis@yorku.ca
URL: http://www.yorku.ca/anestis

ANESTIS A. TOPTSIS is an Associate Professor of Computer Science at York University, Canada. He obtained his Ph.D. in Computer Science from the University of Illinois, Chicago, United States. His research interests include artificial intelligence, affective computing, heuristic search, software and web mining, and computer-aided education.