Electronic Books in Digital Libraries

Gultekin Ozsoyoglu, N. Hurkan Balkir, Graham Cormode, Z. Meral Ozsoyoglu
Case Western Reserve University
Cleveland, Ohio 44106
(tekin, balkir, grc3, ozsoy)@eecs.cwru.edu

Abstract[1]

Electronic book is an application with a multimedia database of instructional resources, which include hyperlinked text, instructor's audio/video clips, slides, animation, still images, etc. as well as content-based information about these data, and metadata such as annotations, tags, and cross-referencing information. Electronic books in the Internet or on CDs today are not easy to learn from. We propose the use of a multimedia database of instructional resources in constructing and delivering multimedia lessons about topics in an electronic book. We introduce an electronic book data model containing (a) topic objects and (b) instructional resources, called instruction module objects, which are multimedia presentations possibly capturing real-life lectures of instructors. We use the notion of topic prerequisites for topics at different detail levels, to allow electronic book users to request/compose multimedia lessons about topics in the electronic book. We present automated construction of the "best" user-tailored lesson (as a multimedia presentation.

1. Introduction

Presently, a large number of user manuals and books are made available in electronic form over the Internet or in CD-ROMS. These electronic books are typically large, usually contain hyper-linked table of contents, indexed search facilities on keywords, and occasionally have multimedia data such as images, maps, and audio/video streams. Most of the time, the sheer size of these electronic books and their static and black box nature impede the user in effectively learning from such books. One commonly hears the frustration of electronic book readers in trying to learn topics from electronic books or in finding an answer to a specific question that they have. We think that new techniques and tools are needed for modeling, querying, "*teaching*" and "*learning*" from electronic books. We use the term *electronic book* as an application with a multimedia database of instructional resources containing, among others, pre-captured multimedia presentations about topics in the book. Our goal is designing techniques for the automated and/or query-based assembly of *lessons* from electronic books. In doing so, we borrow from the Computer-Assisted Learning (CAL) literature, and use some of the existing CAL techniques such as "over-the-shoulder" *guidance* of users, and *interactivity* in our environment. We propose electronic books as learning environments. We assume that multimedia presentations of instructors about topics in a book are captured and enhanced with content-based information, tags, annotations, etc. We call each such unit of data an *instruction module*, or simply a *module*[2], and maintain it in the database of an electronic book. This way, a content-based model of instruction modules is provided. The DBMS maintains user profiles (such as users' knowledge levels about topics covered in the electronic book), and allows automated or query-based construction of user-tailored multimedia lessons for a given topic. In short, an electronic book is an application with a multimedia database, and has the ability to

a) model its data,
b) maintain users' knowledge about topics covered in the electronic book,
c) keep a (possibly, growing) list of multimedia instruction modules of instructors, e.g., about the electronic book's original author's teaching sessions as well as the teaching sessions of other instructors,
d) automatically or manually create multimedia lessons as multimedia presentations, and
e) enable users to learn about ad hoc topics or to obtain answers to specific questions using a mixture of searching, browsing and querying.

In essence, an electronic book application, together with its database, automated lesson construction techniques and a query language, provides controlled, interactive and over-the-shoulder-guided learning environments. Clearly, such applications can be used for independent, remote and distance learning as well.

This paper

[1] This research is supported by the National Science Foundation grants IRI96-31214 and CDA95-29513.

[2] As an example, an instruction module can be an enhanced, tagged, annotated, and catalogued version of a course lecture, a tutorial, or a seminar. It may contain instructor's audio/video clips, student's audio/video clips (asking questions and interacting with the instructor), whiteboards, animated data, slides, text, etc.

1) extends electronic books with a data model containing (a) topic hierarchies, (b) instruction modules of instructors about topics in the book, (c) *prerequisite dependencies* specifying the order of teaching different topics at different knowledge levels, (d) user profiles with explicit knowledge levels on each topic, (e) table of contents (TOC) hierarchies, (f) relationships between various instructional resources such as topic hierarchies, TOC hierarchies, keywords, references, assignments, assignment solutions (as text data or as instruction modules), tests, etc.,
2) introduces the notions of *teaching* and *learning* topics, by using multimedia presentations, called *lessons*, constructed from instruction modules, and
3) discusses automated lesson construction techniques for assembling the "best" lesson from instruction modules in an automated manner, given the user's request for learning a set of topics.

Section 2 lists the main features of a data model for electronic books, to be used as a basis in the rest of the paper. Section 3 discusses prerequisite dependencies. In Section 4, we discuss automated construction of the best lesson for a specific lesson construction request. Section 5 concludes.

2. Electronic book data model

We use an object-oriented data model for electronic book databases with the following object types and properties.

- Electronic book objects of type: text, tables, figures, pictures, pie-charts, histograms, maps, images. Text objects are paginated with page objects containing some of the above-listed objects as well as hyperlinks to different objects.

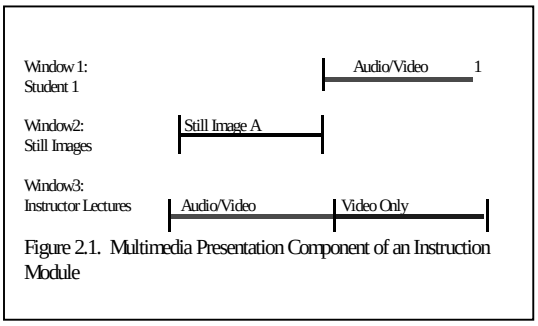

Figure 2.1. Multimedia Presentation Component of an Instruction Module

- *Topic Hierarchies*. In hardcopy books, *topics***,** in their simplest form as keywords, appear as part of a book index at the end of each book. We allow topics to be defined as phrases or keywords. Topics form a main component in our electronic book data model. Topics are organized into a "topic hierarchy" (TP hierarchy). In hardcopy books, the TP hierarchy itself is not used. For each topic, we assume that there are a number of integer-valued *topic detail levels* describing how advance the level of the topic is. To illustrate topic detail levels, for example, the knowledge of a user on the topic "relational calculus" can be at a beginner (i.e., detail level 1) level, e.g., only "relational calculus with propositional calculus formulas". Or, it may be at an advanced (say, detail level n) level, e.g., "relational calculus and its safety", etc. Topic x at detail level i is more advanced (i.e., more detailed) than topic x at detail level j when i > j.
- Text objects that are classified into a *Table of Contents* (TOC) *hierarchy* with the book object at the top level, chapter objects at the next level followed by section objects, subsection objects, and so on. This is the traditional classification hierarchy used in hardcopy textbooks, and also available in (some) electronic books today.
- *User profiles* contain the knowledge levels of users about topics as well as users' preferences. Other information kept in user profiles may be sections (chapters, examples, pictures, etc.) of electronic books that are read/viewed by users, the number of times each section is taught, and the time spent on sections.
  If the user has learned the topic x at level n then we say that the user's *knowledge level* of topic x is at level n. For a given user and a topic, the knowledge level of the user on the topic (zero, originally) is kept in the user profile.
- Multimedia *instruction modules* on topics. The main component of an instruction module, or simply module, is a synchronized multimedia presentation which contains audio/video segments of instructors/students/teaching assistants as well as images, text, animation, whiteboards used by the instructor/students, etc. In addition to the multimedia presentation, an instruction module also contains a content-based model of the multimedia data in the presentation. As an example, the content-based model captures information like "Instructor John Doe explains with an example the (topic) Relational Algebra Divison operator in the audio/video segment #3". In Figure 2.1, using the horizontal x-axis as a timeline, (the multimedia presentation component of) an instruction module is illustrated. Possibly, an instruction module is captured in real-time from a live lecture/teaching session in an automated manner. It is then analyzed and enhanced by a domain expert (instructor or someone highly trained in the subject matter) by identifying important and relevant parts and content information, tagging different media, cross-referencing, etc. Finally, we assume that an electronic book administrator (EBA), a computer expert, modifies the database by entering the structure of the presentation (in the instruction

module), defining the content-based model and entering content data, etc. into the database. We expect that the process of instruction module creation will be a labor-intensive, but one-time, task involving the joint expertise of the domain expert and the EBA.

- *Prerequisite Dependencies.* Some hardcopy textbooks provide "*dependency diagrams"* in an attempt to help instructors/students choose the order of topic coverage. For example, the prerequisite to discussing the topic *relational algebra* in a database course is the coverage of the topic *relational data model.* We formalize this concept into the concept of "*prerequisite dependencies"* among topics, and use it and the existing instruction modules in the database, for automated lesson (i.e., multimedia presentation) construction. For example, we may have the prerequisite dependency "the topic *relational algebra* (*ra*) should be taught after teaching the topic *relational data model* (*rm*)". That is, the teaching dependency *ra*→*rm* holds. In a given course, if topic y is a prerequisite to another topic x (i.e., x→y holds), for the cohesiveness of the course, the instructor makes sure that topic y is covered first, and topic x is covered next. We require that, when a student requests a lesson (a multimedia presentation) on x, and has not yet been rendered those instruction modules that correspond to y then the constructed lesson should also have the instruction modules that correspond to y. Please note that we actually use prerequisite dependencies among topics at different detail levels, e.g., the prerequisite dependency $ra^4 \rightarrow rm^1$ states that "the prerequisite to teaching relational algebra at the detail level 4 is teaching relational data model at the detail level 1 *or higher*".
- Users request multimedia *lessons* from the electronic book. Each lesson is a sequence of (multimedia presentation components of) instruction modules, as defined above. Thus, a lesson is also a multimedia presentation, constructed from multiple modules, and refined by the system. One of the main responsibilities of the electronic book application is to construct "semantically coherent" lessons (that satisfy the prerequisite dependencies and other user-defined constraints) from instruction modules.

We say that lesson S containing a set of instruction modules *covers* topic t at level i if S contains all the instruction modules in the mapping from the topic t at level i to the set of instruction modules.

- For the sake of simplicity, in this paper, we assume that there is a total ordering of all the instruction modules in the database so that, for a given lesson L containing a number of instruction modules to be rendered (i.e., played out), modules in L are ordered into a sequence. Thus, any chosen sets of modules are always ordered by this total ordering in order to form a lesson.

We now define the notions of teaching and learning.

Definition (*Teaching*)**:** Topic t is said to be *taught at level i* if a lesson that covers the topic t at level i and all the prerequisite topics of t at level i are rendered to the user at least once. Such a lesson is said to be a *teaching lesson* for topic t at level i.

For each user, we keep in the database those topics that are taught, the dates and the number of times they are taught, instruction modules played out, etc.

- For each topic and its level, there is a timed test in the database that evaluates the users' knowledge on the topic at that level. The test is *passed* when the user obtains a score above a pre-defined threshold.

As in a traditional classroom environment, testing is not always sufficient by itself to make sure that topics are "learned" by electronic book users. Developing a deeper learning behavior for electronic book users is a research topic for education specialists. In this paper, we will make the, perhaps insufficient, assumption that, for electronic book users, given a topic, passing the associated test constitutes "learning" the topic at that level.

Definition (*Learning*)**:** Topic t is said to be *learned* at level i by a user if the user passes the test for t at level i. A *learning lesson* for topic t at level i is a lesson that includes in it the test for t at level i.

3. Prerequisite dependencies

Prerequisite dependencies can be defined in different ways. One approach is for the electronic book administrator to enter the dependencies directly into the system. Another approach is the automated creation of prerequisite dependencies: the electronic book DBMS can be instructed to create a prerequisite dependency x→z from topic x to topic z when the number of references to topic z in (the instruction modules of) topic x exceed a predetermined threshold, say, K. Similarly, DBMS can create a prerequisite dependency xy→z when the number of references to topic z in (the instruction modules of) topics x and y exceed K, even though the number of references to topic z in x or y alone may be less than K. "References to topic z" can be (i) references to z itself or to its descendant topics in the topic hierarchy of z, or (ii) the occurrences of "keywords" of z.

3.1. Consistency of prerequisite dependencies

We expect that, in different electronic book environments, different and possibly domain-dependent consistency requirements will exist for

prerequisite dependencies. As domain-independent consistency example, consider the following rules.

Consistency Rule #1: For a dependency $x^a \rightarrow x^b$, the property a>b always holds.

That is, for a given topic, if there is a prerequisite dependency within its levels, the dependency is always from a higher (i.e., more detailed; more advanced) level to a lower (i.e., less detailed) level. In order to teach a topic at a given level, it may be necessary to teach it first at a simpler level, but never at a more advanced level.

Consistency Rule #2: For two dependencies $x^a \rightarrow y^b$ and $x^c \rightarrow y^d$, if a<c then b≤d.

This is due to the fact that, since c>a, x at level c is a more detailed (i.e., more advanced) topic than x at level a. Thus, teaching x at level c should necessitate teaching y at a level at least as advanced as b, i.e., y at level d where d is at least as high as b.

The following consistency requirement may apply to some, but not necessarily all, electronic book environments.

Consistency Rule #3: If there is a dependency from x (at any level) to y (at any level) then, for each level i of x, there is a dependency from x at level i to y at some level.

3.2. Computing topic and prerequisite dependency closures

Prerequisite dependencies may be

a) *cyclic* (e.g., x→x forms a trivial cycle; x→y and y→x form a non-trivial cycle). The alternative is to allow only *acyclic* prerequisite dependencies (e.g., trivial or non-trivial cycles are not allowed),
b) (left-hand-side) *decomposable* (e.g., xy→z is equivalent to x→z and y→z) or *nondecomposable* (e.g., xy→z is not equivalent to x→z and y→z).

We first define what it means for a set of dependencies to be acyclic.

Def'n: A set of dependencies is *strongly cyclic* if, applying the rule of transitivity, it is possible to deduce that a topic depends on itself. For example, the set F = {X → Y, Y → Z, Z → X} is strongly cyclic. This still holds if X represents a set of topics.

Def'n: A set of dependencies is *weakly cyclic* if, treating the set of dependencies as decomposable and applying the rule of transitivity it is possible to deduce that a topic depends on itself. For example, the set F = {WX →Y, YZ →V, V →W} is weakly cyclic.

A set of dependencies is considered to be acyclic if it is neither weakly cyclic nor strongly cyclic. Absence of weak cycles implies absence of strong cycles.

The simplest prerequisite dependency model that is commonly used in hardcopy textbooks allows only acyclic and decomposable prerequisite dependencies. However, one can also have electronic book environments in which prerequisite dependencies are cyclic and/or nondecomposable. Consider the case of a cycle of three prerequisite dependencies, namely, x→y, y→z, z→x among topics x, y, and z. We interpret the existence of this cycle as "in any lesson request having one of topics x, y, or z, the instruction modules that cover all three topics must be included into the constructed lesson". Clearly, this attaches a separate semantics to a cycle of prerequisite dependencies, which overrides the semantics of each individual prerequisite dependency in the cycle.

As for decomposability, consider the prerequisite dependency ab→c which states that " a and b together in a presentation request have c as the prerequisite" or "the prerequisite of a and b is c". We say that ab→c is nondecomposable if ab→c does not imply that a→c and b→c. (Note that the reverse is always true, i.e., the prerequisite dependencies a→c and b→c always imply the prerequisite dependency ab→c)[3]. Below we illustrate a case in which prerequisite dependencies are nondecomposable.

Example 3.1. Assume s, q, and r represent the topics SQL, Query-by-Example, and Relational Calculus, respectively. When a lesson about both SQL and Query-by-Example, both at level 2, is requested, it may make sense to include Relational Calculus at level 1 into the lesson for completeness (thus, the prerequisite dependency $s^2q^2 \rightarrow r^1$). However, we may not require Relational Calculus to be included into the lesson if only one of SQL or Query-by-Example is requested (e.g., $s^2q^2 \rightarrow r^1$ is not equal to $s^2 \rightarrow r^1$ and $q^2 \rightarrow r^1$).

In the rest of this section, we discuss how to compute topic and/or prerequisite dependency closures when prerequisite dependencies are cyclic/acyclic and decomposable/nondecomposable.

3.2.1. Cyclic and nondecomposable prerequisite dependencies.

If prerequisite dependencies are nondecomposable and allowed to be cyclic then their semantics is equivalent to the semantics of functional dependencies. That is, prerequisite dependencies can be axiomatized using Armstrong's axioms, which are sound and complete [3]. One can then compute P^+, the

[3] When prerequisite dependencies are automatically created by the DBMS as discussed in the first paragraph of section 3, their semantics implies nondecomposable prerequisite dependencies.

closure (i.e., the set of implied prerequisite dependencies) of a set P of prerequisite dependencies. More interestingly, one can find the closure (i.e., all the prerequisite topics) X^+ of a set X of topics by using the O(N.L) closure algorithm for a set of attributes [3] where N is the number of prerequisite dependencies, and L is the length of the encoding for a prerequisite dependency.

Assume that there is a nondecomposable prerequisite dependency xy→z in the database. First, the user u asks for a lesson which includes x, but does not include y or z. Later, the user u asks for another lesson which includes y, but does not include x or z. As a result of these two lessons, user u will be taught x and y, but not z, thus violating the prerequisite dependency xy→z. One possible solution to this problem is to utilize the user profiles. Since user profiles contain users' knowledge about all instruction modules that are taught to the user, topic z coverage can be added to the second lesson request when topic y is requested (it is known in the user profile that x is taught to the user before).

3.2.2. Acyclic and decomposable prerequisite dependencies. If prerequisite dependencies are acyclic and decomposable then a given topic cannot be a prerequisite to itself. This means that the reflexivity axiom for functional dependencies does not apply to prerequisite dependencies of this model. Similarly, augmentation axiom of functional dependencies does not apply either[4]. Also, this model allows prerequisite dependencies of the form xy→z to be equivalent to x→z and y→z, which is not true for functional dependencies. For this case, to find the closure P^+ of a set P of prerequisite dependencies, we can first "fully" decompose all prerequisite dependencies into P' so as to have only one topic in the left-hand-side and the right-hand-side of each dependency. Then, we can create a dependency graph $G_P(V,E)$, where V is the set of topics, and the set E of edges contains the edge from node a to node b iff P' contains the prerequisite dependency a→b. The closure P^+ of P can then be found by finding the transitive closure of G_P. And, the closure X^+ of a set of topics X can be found by finding all topics that contain nodes in G_P reachable from each of the nodes in X. Also note that we can check the acyclicity of a set of prerequisite dependencies in this model by simply checking the existence of a cycle in its precedence graph in linear time.

[4] Given x→y and z, zx→zy is valid for functional dependencies. However, for prerequisite dependencies, when z is replaced by x, we have xx→xy, which creates a trivial cycle and is not allowed.

3.2.3. Cyclic and decomposable prerequisite dependencies. If prerequisite dependencies are cyclic and nondecomposable then finding the closure P^+ of a set P of prerequisite dependencies is identical to the solution of section 3.2.2 above. We first "fully" decompose all prerequisite dependencies in P into P' so as to have only one topic in the left-hand-side and the right-hand-side of each dependency. Then, we create the dependency graph $G_P(V,E)$, where V is the set of topics, and the set E of edges contains the edge from node a to node b iff P' contains the prerequisite dependency a→b. The closure P^+ of P can be found by finding the transitive closure of G_P. And, the closure X^+ of a set of topics X can be found by finding all nodes in G_P reachable from each of the nodes in X.

3.2.4. Acyclic and nondecomposable prerequisite dependencies. If prerequisite dependencies are acyclic and nondecomposable then the left-hand-side of a prerequisite dependency may contain multiple topics. In this case, one may think of using a dependency graph where the node from which an edge emanates contains a set of topics. Such a graph leads to a hypergraph as a dependency graph. However, unlike the solutions in sections 3.2.2 and 3.2.3, the transitive closure of such a graph would not capture all the dependencies. Consider, for example, the set of dependencies {x→a, ab→c}, and the request for the closure of the set {x, b} of topics. The transitive closure of the dependency graph returns {x, a, b} as the answer whereas the correct answer should be {x, a, b, c}. Thus, transitivity itself is not sufficient for topic closure. Below we give a sound and complete axiomatization for this case, and describe a topic closure algorithm.

We observe that Armstrong's Axioms, used to axiomatize standard functional dependencies, are not appropriate when acyclicity is demanded. The axiom of reflexivity generates trivial (weak) cycles, as does the axiom of augmentation.

Def'n: Pseudo-transitivity axiom: If X→Y and WY→Z then WX→Z.

Def'n: Split/join axiom: if X→AB then X→A and X→B, and vice-versa.

Theorem 1: The pseudo-transitivity and split/join axioms are sound and complete.

Proof: Omitted due to space requirements. Please see [1] for details.

The following algorithm computes the closure of a set of topics X.

Algorithm:

1. $X^{(0)}$ is set to empty.

2. $X^{(i+1)}$ is $X^{(i)}$ U {y} such that there is a dependency in F of the form $X_i \rightarrow y$, where $X_i \subseteq X$ U $X^{(i)}$ and $y \notin X$.

The algorithm terminates when $X^{(j)}=X^{(j+1)}$ (when no dependency can be invoked), and the output X^+ is $X^{(j)}$. Clearly it will always terminate.

Lemma 1: Algorithm 1 correctly computes $X^{(+)}$.
Proof: Omitted due to space requirements. Please see [1].

This algorithm can be implemented naively to check through the set of dependencies at each iteration to see whether any new topics can be added. A more efficient implementation is described in [3], which runs in time linear in the size of the dependencies (counting one for each topic which appears in each dependency).

Finally, we show that our system does not break the condition of acyclicity.

Lemma 2: Computation of the closure of a set of topics X under a set F of acyclic nondecomposable dependencies does not violate acyclicity. That is, $X \Rightarrow X^+$ will not imply any cycles.
Proof: Omitted due to space requirements. Please see [1].

4. Automated lesson construction

When users request a lesson from an electronic book in an automated manner, what types of constraints would they attach to their requests? We list some possibilities:

(a) Lessons about topics. An example request is "prepare a lesson on topics x at level i and y at level j".
(b) an upper bound t_{UB} on the time length of the lesson. An example is "prepare a lesson on topic x which is at most 30 minutes long".
(c) Lessons constructed around tests, assignments, quizzes, chapters, etc. An example is "Prepare a lesson on (the topics covered in) the current assignment". Since we assume that there are mappings from tests, quizzes, assignments, chapters, etc., into topics, these requests reduce to requests of type (a) above, and we will not deal with such requests.
(d) A quantifiable increase, say integer *k*, on the user's knowledge level(s) on a given topic. An example is "prepare a learning lesson (i.e., one with tests) on topic x that, if I pass the tests in the lesson, increases my current knowledge on topic x by *k* units (e.g., from "beginner" to "intermediate")".

In this section, we characterize and classify "typical" automated lesson construction requests, and discuss how they can be evaluated.

4.1. Automated lesson construction requests

The lesson construction requests described in this section have different solutions for each prerequisite dependency case (1-4) described in Section 3.2. The differences between the solutions are in the calculation of topic closures and in the handling of cycles. Topic closure calculation is included in deciding the complexity of the algorithms: topic closure can be calculated in O(N) for all four cases where N is the number of topics in the database and the length of a dependency encoding is one.

Lesson Request 1. Given (a) the user's knowledge levels for topics, (b) the set X of topics, and (c) prerequisite dependencies in the electronic book, produce a lesson that teaches topics X, in the order given, at the highest levels.

Request 1 can be evaluated by a polynomial-time algorithm. First, we calculate the topic closure X^+ of X using the highest detail level. Then we eliminate the topics known by the user from X^+. The last step is to find the instruction modules that map to the topics that are left in X^+, and to order them (using their total ordering) to obtain a lesson. Steps 1 has complexity O(N) where N is the number of topics in the database, and steps 2 and 3, each, have O(M) complexity, where M is the number of topics in X^+.

Lesson Request 2. Given (a) the user's knowledge levels for topics, (b) prerequisite dependencies in the electronic book, (c) the set X of topics, and (d) an upper bound t_{UB} on the lesson timelength, produce within the time bound t_{UB} a lesson that teaches all the topics in X, in the order given in X, at the highest equal possible levels.

Request 2 can also be evaluated by a polynomial-time algorithm [2].

Lesson Request 3. Given (a) the user's knowledge levels for topics, (b) the set X of topics and priorities attached to topics in X, (c) prerequisite dependencies in the electronic book, and (d) an upper bound t_{UB} on the lesson timelength, produce a lesson of duration t_{UB} or less that has the highest total priority.

Theorem 2. Request 3 is a NP-Complete problem.

Proof: Omitted due to space requirements. Please see [2].

The following request asks for a lesson that maximizes the number of topics taught from the user's list of chosen topics.

Lesson Request 4. Given (a) the user's knowledge levels for topics, (b) the set X of topics, (c)

prerequisite dependencies in the electronic book, and (d) an upper bound t_{UB} on the lesson timelength, produce a lesson of duration t_{UB} or less that teaches as many of the topics in X as possible.

Theorem 3: Request 4 is NP-Complete.

Proof: Omitted due to space requirements. Please see [1].

In the next three sections, we propose four heuristics to evaluate Requests 3 and 4, and evaluate their expected and worst-case behavior. The algorithm below uses these four heuristics in evaluating Request 4.

Request 4 Heuristic Algorithm:

```
begin
time:=0;
results:={};
repeat
 begin
   Pick topic x from X using one of the heuristics in
     section 4.2;
   Find the topic closure x+ of x at the highest detail
     level;
   Eliminate from x+ the topics that are already known
     by the user, to obtain y;
   results := results UNION y;
   time := time + time of y;
 end
until time > tUB;
```

The complexity of the above algorithm is O(N) where N is the number of topics.

Lesson construction requests above dealt with constructing teaching lessons, i.e., lessons with no tests. The requests below are for construction learning lessons, i.e., lessons with tests, where the user's knowledge levels about topics are evaluated.

Lesson Request 5. Given (a) the user's knowledge levels for topics, (b) prerequisite dependencies in the electronic book, and (c) an upper bound t_{UB} on the lesson time length, produce a learning lesson of duration t_{UB} or less for topics X such that, if the tests in the lesson are passed, the sum of the level increases on topics in X is maximized.

Theorem 4: Request 5 is NP-Complete.

Proof: Omitted due to space requirements. Please see [2].

An approximate algorithm similar to the one in Figure 4.2 can also be used for evaluating Request 5.

Lesson Request 6. Given (a) the user's knowledge levels for topics, (b) prerequisite dependencies in the electronic book, (c) the set X of topics, (d) an upper bound t_{UB} on the lesson time length, and (e) integers t_p and k, produce a learning lesson of duration less than t_{UB} that, if the tests in the lesson are passed, increases the user's knowledge levels on at least t_p topics in X by at least k levels.

4.2. Heuristics for expensive lesson requests

Best Base Heuristic (BB): Find the topic x in X which is a prerequisite to the largest number of topics in X; and add the corresponding instruction modules into the lesson being constructed.

The motivation for heuristic BB is that if a topic x is included in a lesson, it will satisfy, as much as possible, the prerequisite requirement of other topics in X. To find x, we find the prerequisites of each topic in X. Next, we calculate the number of times a topic appears in the prerequisite lists of other topics in X. The topic with the highest prerequisite count is chosen.

Example 4.1. Assume that the knowledge level of the user is zero on all topics; X = $\{a^4, b^6, c^5, d^6, e^5, f^4\}$; the instruction modules of all topics at all levels take the same amount of time, say t, to present (e.g. a^4 takes 4t time to present); total time allowed for the presentation is 20t; and the prerequisite dependencies are $a^4 \rightarrow b^5$, $c^4 \rightarrow a^4$, $d^6 \rightarrow b^3$, and $e^3 \rightarrow f^2$. We calculate the prerequisite count (the number of times a topic appears in the second column) for a as 1, b as 3, c as 0, d as 0, e as 0, and f as 1. Using heuristic BB, b will be the first topic included in the result. A solution set of topics using BB would be $\{b^6, a^4, f^4, c^5\}$ with duration 19t. Any other solution set with four or more topics which does not include b will have a duration longer than 20t. Clearly, for this example, including b as the first topic into the solution by heuristic BB is a good choice.

Lowest Detail Level Heuristic (LDL): Find the topic with the lowest detail level, which is not known by the user, and add the corresponding instruction modules into the lesson being constructed.

The motivation for heuristic LDL is that lower detail levels of topics are more likely to be prerequisites to other topics. Then, it is easier to include a topic in a lesson if the prerequisite of the topic is already included in a lesson. Hence, adding the topic with the lowest detail level into the lesson being constructed increases the chances of other topics in X being included.

Example 4.2. Assume that the knowledge level of user is zero on all topics; X = $\{a^4, b^6, c^3, d^6, e^4, f^4\}$; all topics at all levels take the same amount of time (t) to present (e.g. a^4 takes 4t to present); total time allowed for the presentation is 15t; and the prerequisite dependencies are $a^4 \rightarrow c^2$, $b^4 \rightarrow c^1$, $d^6 \rightarrow b^3$, and $e^3 \rightarrow f^2$. Using heuristic LDL, c will be the first topic included in the result as it has the lowest detail level unknown to the user. A solution set of topics using LDL would be $\{c^3, a^4, f^4, e^4\}$ with duration 15t. Any other solution set

with four or more topics will have duration longer than 15t, which is not acceptable. Including c as the first topic into the solution by the heuristic LDL allows us to include other topics that depend on c, and is clearly a good choice.

Highest Number of Detail Levels Heuristic (HNDL): Find the topic with the highest number of detail levels that is not known by the user, and add the corresponding instruction modules into the lesson being constructed.
The motivation for heuristic HNDL is that a topic with high number of detail levels has a higher chance of being a prerequisite to other topics than a topic with a low number of detail level. Similar to LDL, including more prerequisites in a lesson increases the chances of other topics in X to be included into the lesson.

Example 4.3. Assume that the knowledge level of user is zero on all topics; X = {a^4, b^6, c^4, d^4, e^4, f^5}; a topic at level x takes x*t time to present (e.g. a^4 takes 4t time to present); total time allowed for the presentation is 15t; and the prerequisite dependencies are $a^4 \rightarrow b^6$, $c^4 \rightarrow b^6$, $d^4 \rightarrow b^6$, $b^6 \rightarrow f^5$ and $e^4 \rightarrow f^5$. Using heuristic HNDL, b will be the first topic included in the result as it has the highest number of detail levels unknown to the user. A solution set of topics using HNDL would be {b^6, f^5, a^4} with duration 15t. Any other solution set with three or more topics will have a duration of at least 15t, which is not any better than the solution found by HNDL heuristic. Including b as the first topic into the solution by the heuristic HNDL allows us to include other topics that depend on b, and is clearly a good choice.

Lowest Number of Prerequisites Heuristic (LNP): Find the topic with the lowest number of prerequisites (that are not known by the user), and add the corresponding instruction modules into the lesson being constructed.
The motivation for heuristic LNP is that we expect to include more topics by choosing topics with few prerequisites.

Example 4.4. Assume that the knowledge level of user is zero on all topics; X = {a^5, b^6, c^5, d^6, e^3, f^4}; a topic at level x takes t*x time to present (e.g. a^5 takes 5t time to present); total time allowed for the presentation is 20t; and the prerequisite dependencies are $a^5 \rightarrow b^6$, $c^5 \rightarrow a^5$, $d^6 \rightarrow b^6$, $b^6 \rightarrow f^4$, and $e^3 \rightarrow f^4$. Then the number of prerequisites for a is 2 (i.e., b^6 and f^4), b is 1 (i.e., f^4), c is 3 (i.e., a^5, b^6 and f^4), d is 2 (i.e., b^6 and f^4), e is 1 (i.e., f^4), and f is 0. Using heuristic LNP, f will be the first topic included in the result. A solution set of topics using LNP would be {f^4, e^3, b^6, a^5} with duration 18t. Any other solution set with four or more topics, which does not include f, will have a duration of at least 22t. Clearly including f as the first topic into the solution by the heuristic LNP is a good choice.

4.3. Evaluating the expected case behavior of the heuristics for lesson request 4

We now briefly summarize the experiments conducted to evaluate the expected performances of the four heuristics described above for only the lesson request 4. To evaluate the heuristics, we simulated an electronic classroom. Electronic classroom is an education environment where students decide on the length and the content of a presentation about a lecture using various constraints. We used four components (users, topics, dependencies, and requests) to model the electronic classroom environment. Please see the details at [2].

To observe the effects of changing the number of prerequisite dependencies, we kept the following parameters constant: the number of topics 1000, topic depth 12, length of a topic detail level 10 minutes, presentation length 60minutes, and length of requests 10 topics. We observed that, as the number of prerequisite dependencies increases, the number of presented topics decreases. This result is expected as increasing the number of prerequisite dependencies increases the length of the presentation of topics, and hence decreases the chances of topics being included into the resulting lesson. All heuristics performed within 7% of the theoretical maximum. Among the heuristics, LNP performed the best while HNDL performed the poorest.

To observe the effect of the topic depth (i.e., the number of detail levels) on the percentage of requested topics presented, we kept the following parameters constant: number of topics 1000, length of a topic detail level 10 minutes, presentation length 60 minutes, and length of requests 10 topics. The number of prerequisite dependencies (400-4000) was changed proportional to the change in the number of detail levels (2-20). The results were similar to prerequisite dependency results. As the topic depth increases, topics at higher detail levels are included in the requests. Topics at higher detail levels have longer durations than topics at lower detail levels; and this decreases the chances of a topic being included into the resulting lesson. Obviously, topic depth and the percentage of the requested topics presented are inversely proportional. As the topic depth increases, the percentage of the requested topics presented decreases.

As for the effect of increasing the time upper bound t_{UB} on the presentation, we kept the following parameters constant: number of topics 1000, topic depth 12, length of a topic detail level 10 minutes, number of

prerequisite dependencies 2400, and length of requests 10 topics. Clearly, increasing the time limit increases the chances of a topic being included into the resulting lesson. Similar to the previous results, the behaviors of all heuristics closely resemble the theoretically possible best result. Time upper bound and the percentage of the requested topics presented are directly proportional. As the time upper bound increases, the percentage of the requested topics presented increases.

As expected, changing the number of topics in the simulation had no effect on the performance of the heuristics. To observe this, we kept the following parameters constant: topic depth 12, length of a topic detail level 10 minutes, presentation length 60 minutes, and length of requests 10 topics.

Changing the request length (i.e., the number of topics in X) had an effect similar to changing the prerequisite dependencies or changing the topic depth. In this experiment, we kept the following parameters constant: number of topics 1000, topic depth 12, length of a topic detail level 10 minutes, presentation length 60 minutes, and the number of prerequisite dependencies 2400. Since the time limit on the lesson does not change, the percentage of requested topics that are presented decreases. Request length and the percentage of the requested topics presented are inversely proportional. As the request length increases, the percentage of the requested topics presented decreases.

And, for the effectiveness of the heuristics compared to calculating the best lesson by enumeration to the lesson request 4 of Section 4.1, we observed that, as the length of the lesson increases, the time to calculate the best solution increases exponentially. When the lesson includes 18 topics, *all* heuristic algorithms produce a solution under 5 mseconds while it takes over 167 seconds for the solution by enumeration. Clearly, as the request length increases, heuristic solutions become a must for an efficient implementation. Thus, all of the four heuristics performed well with results that are within 7% of the best solution. Relatively, LNP performed the best, BB and LDL performed very close to LNP, and HNDL performed the poorest. Similar results can be shown for lesson requests 3, 5 and 6.

4.4. Worst-case performances of lesson requests

From the previous section, we have observed that the expected performances of lesson construction requests are shown to have acceptable performance on randomly generated test data. However, in the worst case their performance can be dramatically poorer, on data contrived to elicit this performance. Next, we illustrate this for the lesson construction request 4 and two of the heuristics.

4.3.1. Request 4 and Best-Base heuristic. This heuristic picks the topic x from X which is a prerequisite to most other topics from X, and adds $(x)^+$ to the output, then iterates. We consider the case where t_{UB} has some specified value, k, and all topics have unit cost. We set X to be the topics x', x", and x_i for all $0 \le i \le k$. We choose F to consist of $x" \rightarrow x'$, $x' \rightarrow y_i$, for all $1 \le i \le k$. There are no dependencies of the form $x_i \rightarrow z$, and so we could teach all k topics x_i. However, the Best Base Heuristic leads us to choose to teach x_0, as it is the base of the most topics in X. Since x_0 depends on k topics which are not in X, these must be taught first, meaning that applying this heuristic results in none of the topics in X being taught.

Although this example is contrived, it could feasibly occur. Suppose X consisted of two distinct kinds of topic: a set of basic topics which have no prerequisites, but also are not prerequisite to any other topics in X; and a few very advanced topics, which have a common prerequisite, which in turn depends on many other (unrequested) topics. The most topics from X would be achieved by teaching the simple topics, but the Best Base Heuristic will cause the system to try to teach the advanced topic that has many prerequisites.

4.3.2. Request 4 and the Lowest-Number-of Prerequisites heuristic. In tests, the Lowest Number of prerequisites (LNP) Heuristic performed the best out of the heuristics tested, but again we can force it to give bad results. We consider t_{UB} to be set to a constant value, 2k, and all topics have unit cost. We set X to be the topics x_i for $0 \le i \le k$, and create F with the following dependencies: $x_0 \rightarrow y_i$ for $1 \le i \le k-1$; and $\forall i$, $1 \le j \le k$, $x_i \rightarrow z_j$. Since x_0 has k-1 prerequisites, and all other topics in X have k prerequisites, LNP will lead us to choose to teach x_0, at total cost k. To teach any further topics from X, we require all the k z_j's, but by the time these have been taught, the time bound of 2k has been reached. In total, LNP allows one topic from X to be taught. However, the optimal solution is to teach all k topics z_j and then all k dependent topics x_i, $i>0$, resulting in k topics from X being taught within the time-bound. Again, this situation could feasibly occur, if X consisted on a large set of similar topics, which have a large common set of prerequisites, and one unrelated topic which has a lesser number of prerequisites. Although teaching the unrelated topic has lower initial cost, this cost does not `buy' anything useful.

4.5. Worst-case performance guarantees of lesson requests

For worst-case performance guarantees of lesson request algorithms, we will consider the simplest case,

i.e., decomposable prerequisite dependencies, and the lesson request 4. We first transform the problem into the bipartite graph problem, and state it as a mathematical integer programming problem, which are known to be difficult to approximate.

So far we have often considered the case where the hierarchy of dependencies is shallow: the topics are partitioned into two sets, with dependencies from one set to the other. We shall now show that this situation is not unrepresentative: any set of decomposable dependencies can be rewritten as a two-level hierarchy. Each topic is represented by a node, x, on the left side of the bipartite graph. The cost of this topic is set to zero. We also create a topic, x', on the right hand side of the bipartite graph whose cost is that of the topic. We initialize F, the new set of dependencies, to be $x \rightarrow x'$. We then add dependencies to F such that $x \rightarrow y'$ for each $y \in (x)+$. This problem is identical to the original problem instance.

In the case that we are trying to answer a request of the form of Request 4, we can reduce the problem further. Our observation is that we are only interested in the requested topics in X. Where we have that some y not in X has closure (y)+ such that no member of (y)+ is in X, then we can replace the whole of (y)+ with a single topic whose length is the sum of the lengths of the component topics. We can also merge any topics which form a cycle into a single topic, whose prerequisites are the union of the prerequisites of the component topics. The intuition here is that if any topic in a cycle is chosen, then all topics in that cycle must be included. This leads to a canonical form for representing such requests as a bipartite graph problem. The goal is to 'collect' as many nodes on the left side as possible within the time limit. To collect such a node, we must 'buy' all the nodes on the right to which it is connected, each of which has a certain cost. We have a total budget of t_{UB}. This problem can also be stated as a mathematical integer-programming problem:

Maximize f(X) subject to: $C.X \leq t_{UB}$ $X_i = 0,1 \ \forall i$
where f(X) is defined as $\Sigma_{x \in X} \Pi_{y \in x+} y$.

Unfortunately, problems of this type are hard to approximate. Results from Mathematical Programming Theory [4] show that there is effectively no approximation for the general nonlinear programming problem. Even considering the extreme restriction that each topic can depend on at most one topic, (that is, for a topic x then (x)+ contains at most one other item), then the problem is still hard. This restricted problem forms an instance of quadratic programming, for which no general approximation algorithms are known [4]. This leads us to conclude that for requests like Request 4, there are unlikely to be approximation algorithms that can guarantee that their results are within any factor of the optimal, and so we should be content with using ad hoc heuristics to solve real instances of the problems.

5. Conclusions

In this paper, we have studied the use of a multimedia database, and database techniques for electronic books containing pre-captured multimedia presentations about topics in an electronic book. We have designed an electronic book environment for the automated assembly of multimedia lessons, and discussed possible heuristics for lesson construction and their expected-case and worst-case time complexities.

6. Acknowledgments

We would like to thank Dr. G. Phillip Cartwright for providing us references on CAL and CAI literature, and for his insightful comments on this work. G. Cormode thanks Mike Paterson for a discussion regarding the hardness of Request 4 under decomposable dependencies.

7. References

[1] Cormode, G., "Topic Dependencies for Electronic Books", Unpublished manuscript, 1999. Available at http://erciyes.ces.cwru.edu/tekin/eb2.ps

[2] Ozsoyoglu, G,. Balkir, N.H., Cormode, G., Ozsoyoglu, Z.M., "Electronic Books as Multimedia Databases", unpublished manuscript, available at http://erciyes.ces.cwru.edu/tekin/eb1.ps

[3] Ullman, J.D., "Principles of Database and Knowledge-Base Systems", Vol. 1, 1988.

[4] Bellare, M., and Rogaway, P., "The Complexity of Approximating a Nonlinear Problem", *Math. Programming*, 69: 429-442, 1995